Testing Document

Version 6.0

McGill University

ECSE 211

Team 20

TESTING DOCUMENT

Project: DPM Final Project

Document Version Number: 6.0
Date: 28/11/2018
Authors: Ashish Paul

Edit History

Document Version	Modification	Date	Editor
1.0	• Creating of Testing document.	20/10/2018	Ashish
1.1	• Added tests 1 and 2	22/10/2018	Ashish
2.0	• Added tests 3 and 4	28/10/2018	Ashish
2.1	• Added tests 5 and 6 • Restructured doc	30/10/2018	Ashish
3.0	• Added test 7,8 and 9	5/11/2018	Ashish
4.0	• Added more tests • Restructure doc again	10/11/2018	Ashish
4.1	• Added Pre-Beta unit tests	11/11/2018	Ashish
4.2	• Added Beta Demo Integration Test	12/11/2018	Ashish
4.3	• More Beta Test	13/11/2018	Ashish
5.0	• Added final integration tests	27/11/2018	
6.0	• Added results from the integration tests • Final restricting of document.	28/11/2018	Ashish

TABLE OF CONTENTS

1.0 Overview

This document constitutes the testing documents which includes procedures, test results, and conclusions of the test results. The test follows the evolution of the design of the robot. Figures have been used to help the illustrate the test and/or results to better help in recreation of the test if required by a third party.

2.0 Requirements

2.1 Project requirements

see Requirements document

2.2 Testing Requirements

- Each test should note: date, testers, author, hardware version, software version, goal, procedure, expected result, test report, conclusion, action and distribution.
- Each test should have at least 10 trials.
- Weak points should be tested exhaustively
- Testers should have a clear expected outcome for each test.

2.3 General Test Procedures

- Test whether the hardware design is stable when moving and turning.
- Test whether the robot can light localize accurately
- Test whether the robot can ultrasonic localize accurately
- Test whether the robot can correctly detect the color of the ring.
- Test whether the robot can correctly turn to any direction
- Test whether the robot can correctly travel to any coordinate after localization

3.0 Testing Plan

3.1 Hardware

- Hardware stability (Complete)
    - The center of gravity needs to be center of the robot so that the robot has equal weight distribution and does not lean on one side, affecting navigation.
- Large EV3 motor performance (Complete)
    - The motors need to be accurate when moving around the field at different speeds. The motors that are selected need to be of similar performance to make sure the robot navigates properly.
- Ultrasonic sensor (Complete)
    - The ultrasonic sensor must be accurate to ensure the requirements of ultrasonic localization and obstacle avoidance are met

- Light sensor (Complete)
  - The light sensor must be accurate to ensure the requirements of light localization and ring color detection are met.

3.2 Software

- Navigation (Completed)
  - The navigation algorithm must accurately move the robot to any given co-ordinate on the field. The accuracy should be such that when the robot re-localizes, it should be at the correct co-ordinate
- Ultrasonic localization (Completed)
  - The ultrasonic localization algorithm must have an accuracy of 10 degrees.
- Light localization (Completed)
  - The light localization algorithm must have an accuracy of less than 2cm and an angle of less than 5 degrees
- Ring Color detection (Completed)
  - The color detection algorithm must accurately determine the color of the ring up to 5cm away from the light sensor.

3.3 Integration Test

- Beta Demo (Completed)
  - The robot must localize within 30 seconds, navigate to the tunnel, pass the tunnel navigate to the ring, collect the ring, back off and stop.
- Final Demo (Complete)
  - The robot must localize within 30 seconds, navigate to the tunnel, pass the tunnel, navigate to the ring, collect the rings, navigate to the tunnel, pass the tunnel, navigate back to the starting corner and unload the rings.

4.0 Tests

<u>Test 1: Ultrasonic localization I (Falling edge)</u>
Date: 20/10/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.1 from the hardware documentation
Software Version: N/A

Goal: Determine if the robot can localize reliably using 2 large EV3 motors and a ultrasonic sensor mounted in the front of the robot.

Procedure:

1. Place the robot facing away from the wall on the left corner tile.
2. Run LocalizationTest.java.

3. Press any button on the robot to start the program
4. Once the robot completes ultrasonic localization, measure the angle and record it on a table.

Expected result: The robot should turn clockwise and detect the wall when it is within 30cm and record it. Then the robot turns anticlockwise till it detects the wall again and record it. The robot will then compute 0 degrees and turn to it.

Test Report: The test was performed 30 times and the results are shown in the table below.

Run	US Angle(degrees)
1	6
2	-2
3	3
4	0
5	-5
6	4
7	-1
8	2
9	4
10	0
Mean	1
Standard Deviation	3

Table 1:Ultrasonic Localization results for test 1

The robot localized within less than 5 degrees 80% of the time.

Conclusion: The robot always localized within 6 degrees which is within the acceptable margin of error. It should be noted that this test was done when the battery was outputting at 8.0V and accurate localization is not guaranteed for anything less than 8.0V.

Action: No further testing is required unless the hardware configuration is changed.

Distribution: Software Development, Project management

<u>Test 2: Light localization I (2 light sensors)</u>
Date: 20/10/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.1 from the hardware documentation
Software Version: N/A

Goal: Determine if the robot can light localize reliably with 2 light sensors.

Procedure:

1. Place the robot facing away from the wall on the left corner tile.
2. Run the LocalizationTest.java
3. Press any button to start the program.
4. After ultrasonic localization finishes measure and record the angle.
5. Press any button again to start light localization.
6. The robot should move in X first, then Y direction correcting the angle as it detects the lines on the board.
7. Once localization is completed measure the Euclidean error and final angle of the robot.

Expected result: The Final is expected to be within 5 degrees and the Euclidean error is expected to be within 2cm

Test Report: the test was performed 10 times and the results are shown in the table below

Run	US Angle (degrees)	Euclidean error (cm)	Final Angle(degrees)
1	6	0.1	2
2	-2	1.3	-3
3	3	0.4	4
4	0	0.6	1
5	-5	1.5	-2
6	4	1.1	6
7	-1	0.1	-4
8	2	1.9	5
9	4	0.3	3
10	0	1	2
Mean	1	0.8	1
Standard Deviation	3	0.6	3

Table 2: Light localization Results for test 2

Conclusion: The robot always localized within 5 degrees and 2cm from (0,0) which is within the acceptable margin of error. It should be noted that this test was done when the battery was outputting at 8.0V and accurate localization is not guaranteed for anything less than 8.0V.

Action: No further testing is required unless the hardware configuration is changed.

Distribution: Software Development, Project management

<u>Test 3: Ultrasonic localization II (Falling edge)</u>

Date: 20/10/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.2 from the hardware documentation
Software Version: N/A

Goal: Determine if the robot can localize reliably using 2 large EV3 motors and an ultrasonic sensor mounted in the front of the robot.

Procedure:

1. Place the robot facing away from the wall on the left corner tile.
2. Run UltrasonicLocalizationTest.java.
3. Press any button on the robot to start the program
4. Once the robot completes ultrasonic localization, measure the angle and record it on a table.

Expected result: The robot should turn clockwise and detect the wall when it is within 30cm and record it. Then the robot turns anticlockwise till it detects the wall again and record it. The robot will then compute 0 degrees and turn to it.

Test Report: the test was performed 10 times and the results are shown in the table below.

Run	US Angle(degrees)
1	10
2	7
3	6
4	8
5	11
6	12
7	3
8	9
9	9
10	5
Mean	8
Standard Deviation	2.64

Table 3: Ultrasonic Localization results for test 3

The angles are mostly greater than 5 degrees. The standard deviation is also too large. The robot did not localize with an acceptable margin of error

Conclusion: Ultrasonic localization is not reliable.

Action: This report will be sent to the hardware and software team to make necessary changes.

Distribution: Software Development, Hardware Development, Project management

Test 4: Light localization II (2 light sensors)
Date: 20/10/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4 from the hardware documentation
Software Version: N/A

Goal: Determine if the robot can light localize reliably.

Procedure:

1. Place the robot facing away from the wall on the left corner tile.
2. Run the LocalizationTest.java
3. Press any button to start the program.
4. After ultrasonic localization finishes measure and record the angle.
5. Press any button again to start light localization.
6. The robot should move in X first, then Y direction correcting the angle as it detects the lines on the board.
7. Once localization is completed measure the Euclidean error and final angle of the robot.

Expected result: The Final is expected to be within 5 degrees and the Euclidean error is expected to be within 2cm

Test Report: the test was performed 10 times and the results are shown in the table below

Run	US Angle (degrees)	Euclidean error (cm)	Final Angle (degrees)
1	10	FAILED	FAILED
2	7	1.0	3
3	6	FAILED	FAILED
4	8	1.6	2
5	11	FAILED	FAILED
6	12	3.2	2
7	3	FAILED	FAILED
8	9	FAILED	FAILED
9	9	2.1	3
10	5	1.9	2
Mean	8	N/A	N/A
Standard Deviation	2.64	N/A	N/A

Table 4: Light localization results for test 4

When testing the light localization, the light sensor failed to detect lines in 5 out of the 10 trials and therefore failed the test.

Conclusion: Light localization is not reliable. The tests need to be done again when the software or/and hardware have been changed.

Action: This report will be sent to the hardware and software team to make necessary changes.

Distribution: Software Development, Hardware Development, Project management

<u>Test 5: Ring color detection</u>
Date: 22/10/2018
Tester: Ashish
Author: Ashish
Hardware Version: N/A
Software Version: N/A

Goal: To record the RGB values of each ring.

Procedure:

1. Place the light sensor 2 cm away from the ring.
2. Run the DataAcquisition.java.
3. Press any button on the robot. The robot will display the RGB values on the console/display.
4. Record the RGB values that are printed on the console.
5. Collect 18 samples of RGB values by pressing any button on the robot 18 times.
6. Repeat same procedure for the other 3 rings.
7. Use the values to plot graphs.

Expected result: The RGB values should give a gaussian distribution.

Test Report: The test was performed 18 times for each of the 4 rings. The mean and standard deviation was calculated using the following formulas:

The tables and graphs below show the results:

	R	G	B
1	11.76471	54.90196	53.92157
2	21.56863	92.15687	61.76471
3	7.843138	33.33334	29.41176
4	16.66667	73.52941	62.74512
5	19.60784	84.31373	54.90196
6	23.52941	106.8628	93.13726
7	27.45098	112.7451	95.09804
8	26.47059	113.7255	85.29412
9	12.74514	66.66667	50.34523
10	21.56863	96.07843	72.54903
11	10.78431	50.98039	39.21569
12	28.43137	110.7843	100.9804
13	30.39216	124.5098	112.7451
14	34.31373	133.3333	112.7451
15	30.39216	116.6667	103.9216
16	23.52941	100.9804	90.19608
17	19.60784	86.27451	72.54903
18	20.58824	93.13726	74.50983
Mean	21.51416	91.72113	75.87146
STD.dev	7.237454	26.37606	26.37606

Table 5: RGB values for the blue ring

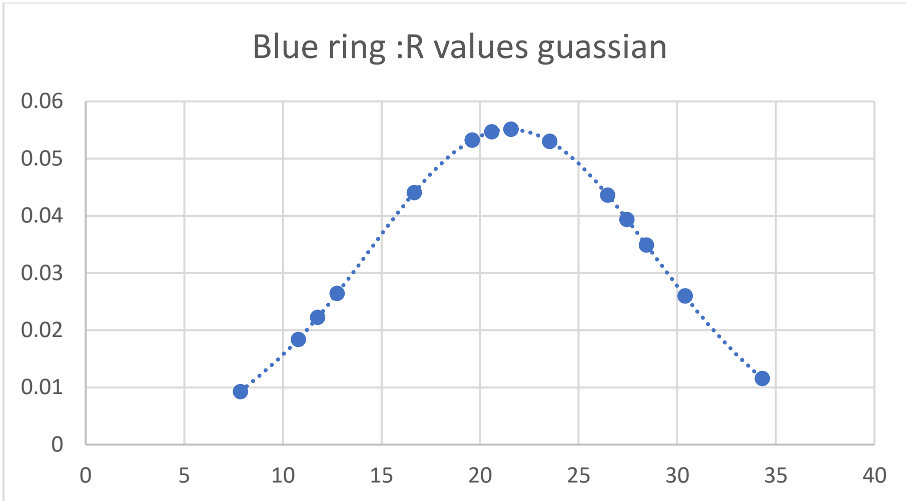

Figure 1: Graph of red values for the Blue ring

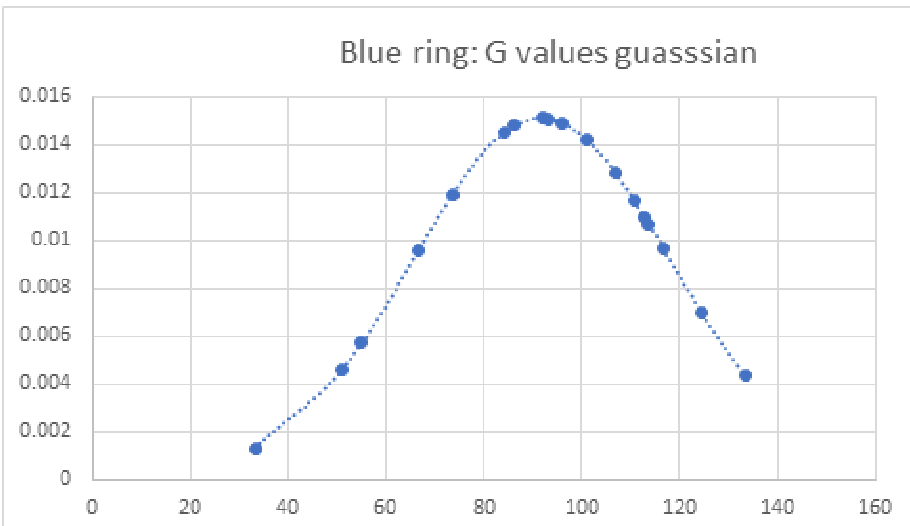

Figure 2: Graph of Green values for the Blue ring

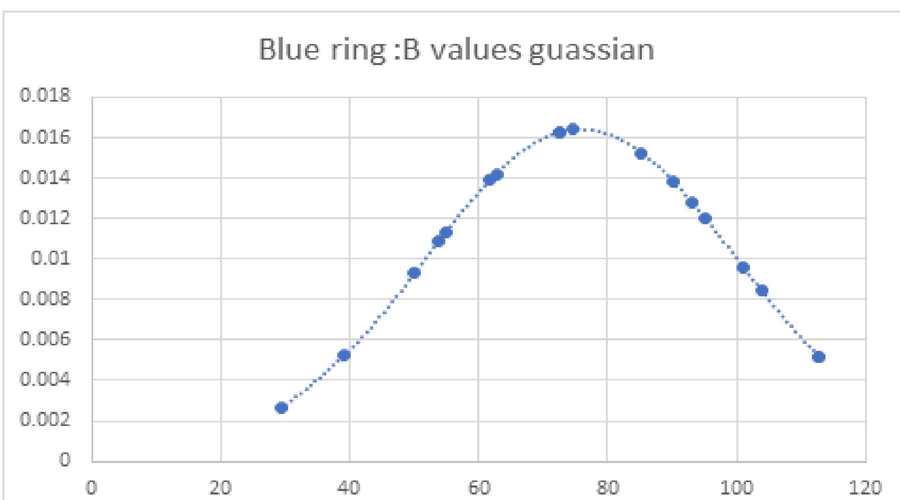

Figure 3: Graph of Blue values for the Blue ring

	R	G	B
1	43.13726	82.35294	14.70588
2	35.29412	66.66667	12.74512
3	61.76471	124.5098	18.62745
4	63.72549	126.4706	24.50983
5	43.13726	85.29412	14.70588
6	11.76471	27.45098	3.921569
7	26.47059	55.88236	6.862745
8	51.96079	105.8824	20.58824
9	33.33334	68.62746	11.76471
10	29.41176	56.86275	10.78431
11	55.88236	117.6471	19.60784
12	22.54902	50.34244	7.843138
13	34.31373	69.60785	11.76471
14	30.39216	59.80392	10.78431
15	47.05882	96.07843	15.68628
16	37.25492	77.45098	12.74514
17	74.50933	148.0392	24.50983
18	26.47059	54.90196	8.823533
19	9.803922	23.52941	1.960784
Mean	38.85449	78.79257	13.31269
STD.dev	16.79279	32.88598	6.104473

Table 6: RGB values of the green ring

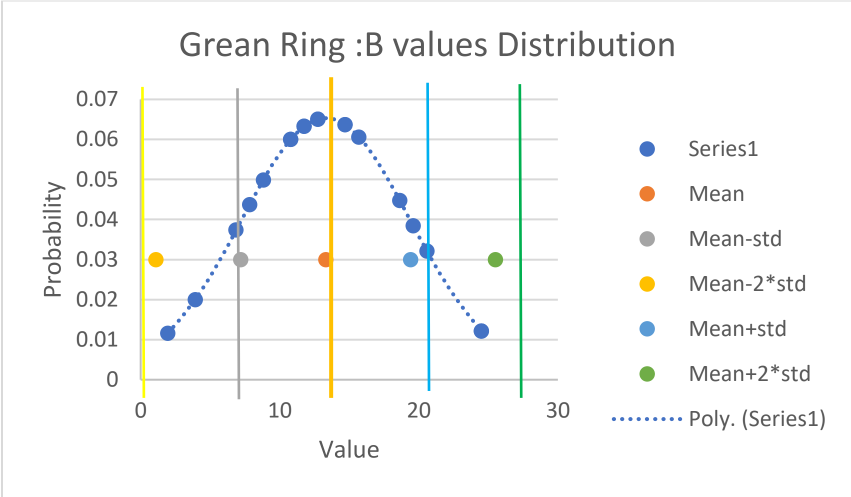

Figure 4: Graph of Blue values for the Green ring

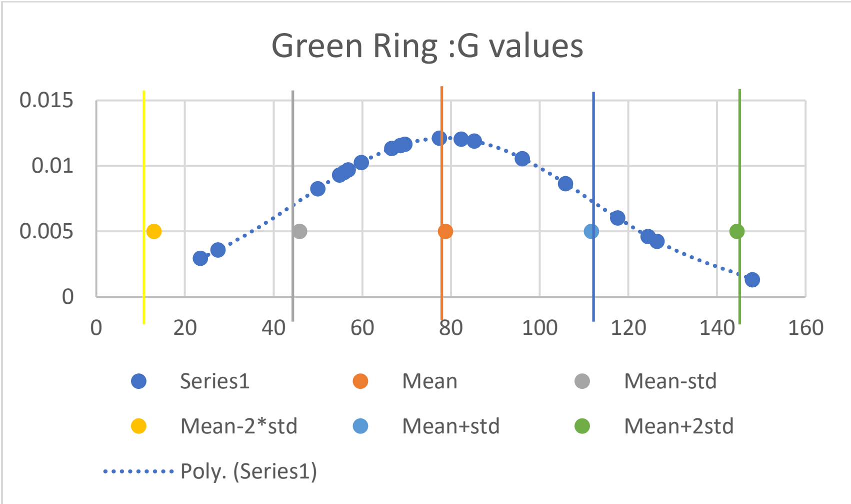

Figure 5: Graph of Green values for the Green ring

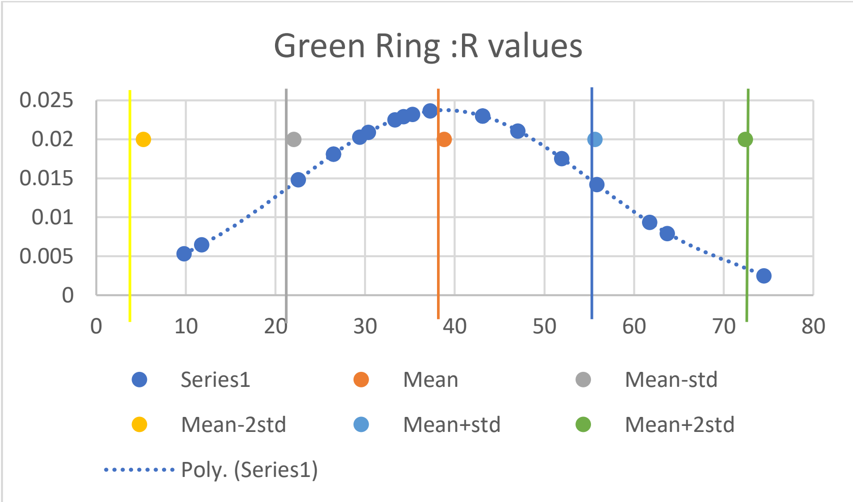

Figure 6: Graph of Red values for the Green ring

	R	G	B
1	117.6471	80.39216	18.62745
2	127.4513	83.33334	20.58824
3	88.23532	59.80392	12.74513
4	56.86275	39.21569	7.843138
5	51.96079	35.29412	6.862745
6	104.9025	66.66667	16.66667
7	108.8235	67.64706	19.60784
8	96.07843	64.70589	14.70588
9	110.7843	76.47059	15.68628
10	156.8628	105.8824	22.54902
11	146.0784	93.13726	24.50985
12	66.66667	40.19608	12.74514
13	77.45098	50.98039	11.76471
14	78.43137	49.01961	11.76471
15	83.33334	50.98039	15.68628
16	89.21569	54.90196	15.68628
17	129.4118	84.31373	19.60784
18	176.4706	118.6275	25.49024
Mean	115.6437	75.70333	17.94544
STD.dev	38.39451	25.98696	5.541675

Table 7: RGB values of the yellow ring

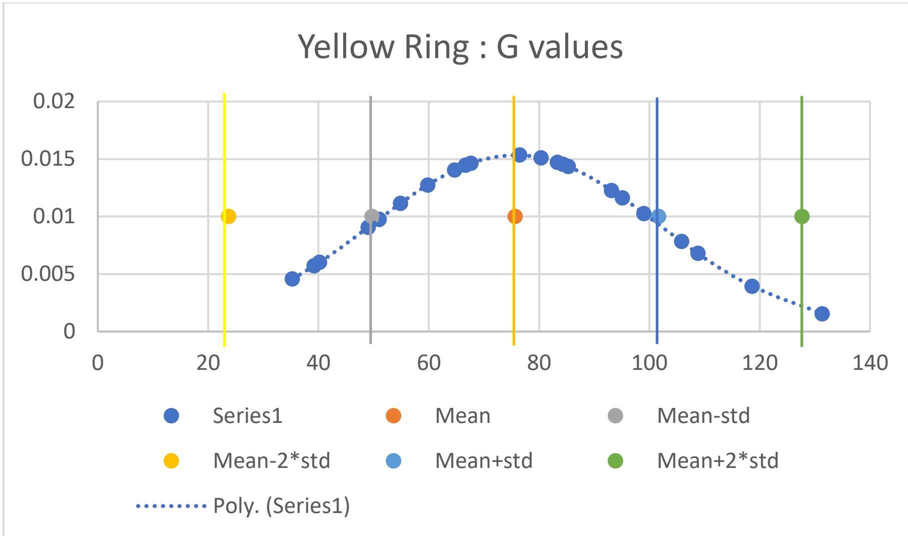

Figure 7: Graph of Green values for the Yellow ring

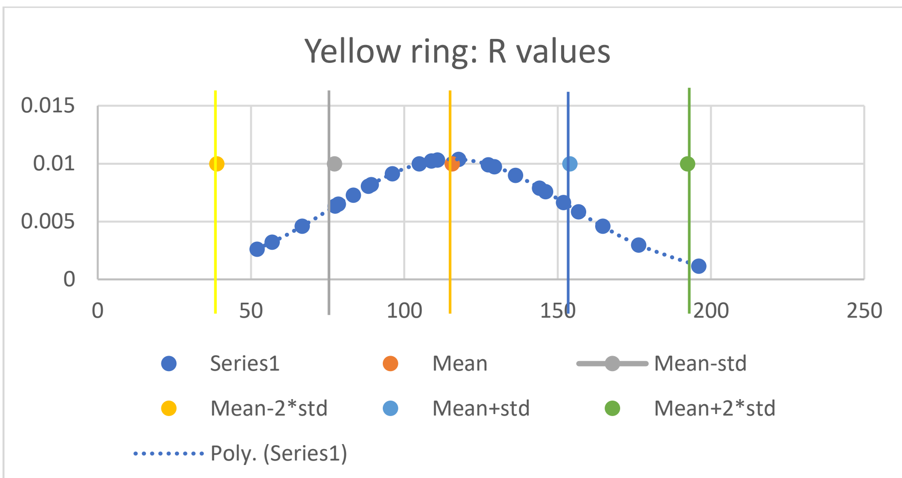

Figure 8: Graph of Red values for the Yellow ring

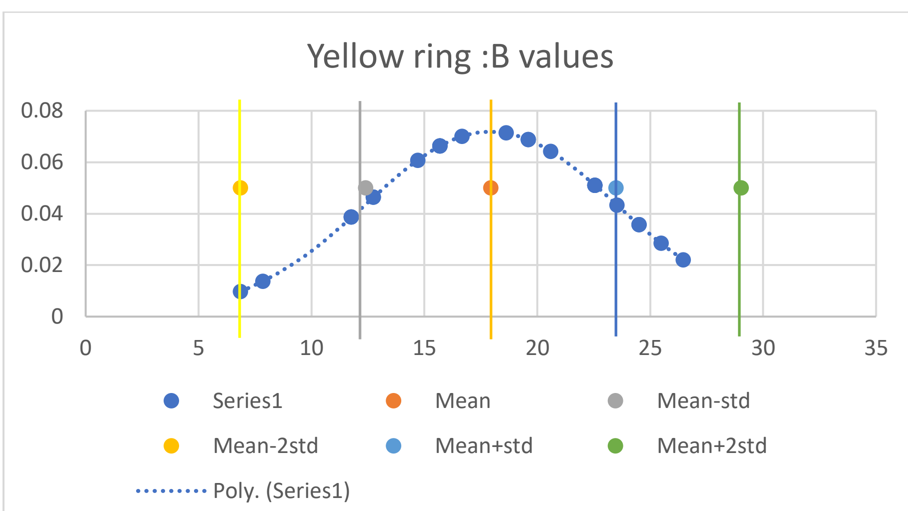

Figure 9: Graph of Blue values for the Yellow ring

	R	G	B
1	178.431	59.832	18.645
2	142.156	45.004	16.667
3	155.882	50.876	15.628
4	164.709	58.853	19.784
5	98.0223	31.355	7.8138
6	121.568	42.186	11.471
7	107.841	36.251	9.8225
8	85.9122	25.402	6.8453
9	147.048	50.345	13.749
10	157.841	45.804	21.563
11	94.165	26.479	11.771
12	144.177	44.115	14.588
13	156.868	44.115	17.606
14	175.492	58.853	22.502
15	125.492	36.251	15.628
16	172.545	62.751	19.684
17	129.418	44.115	10.781
18	150.984	48.922	18.625
Mean	139.326	44.464	15.198
STD.dev	27.8844	10.607	4.4924

Table 8: RGB values of the orange ring

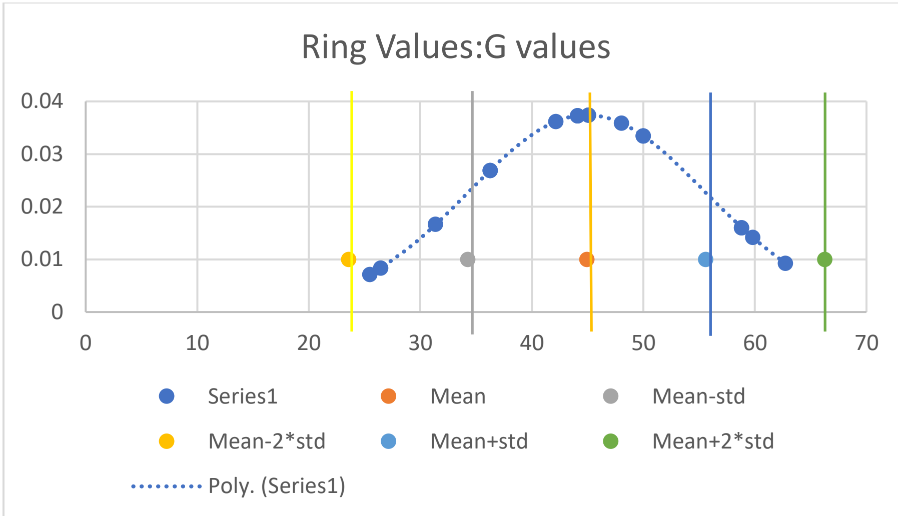

Figure 10: Graph of Green values for the Orange ring

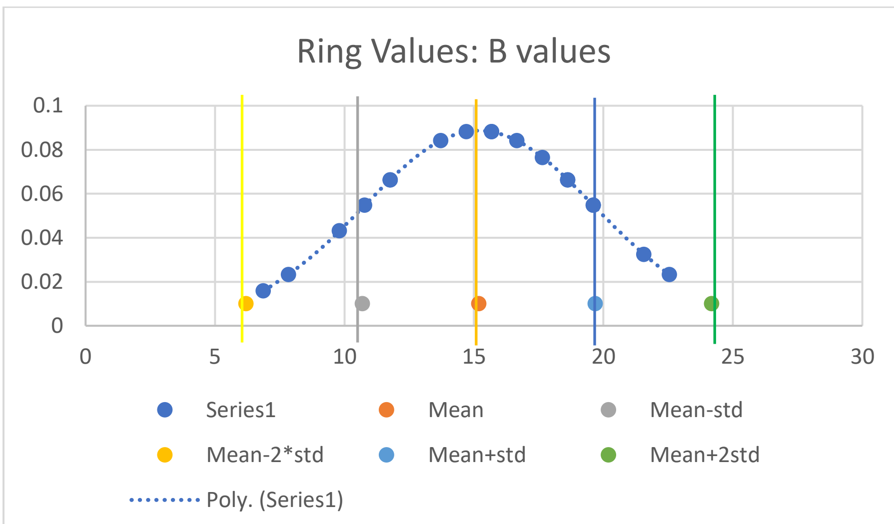

Figure 11: Graph of Blue values for the Orange ring

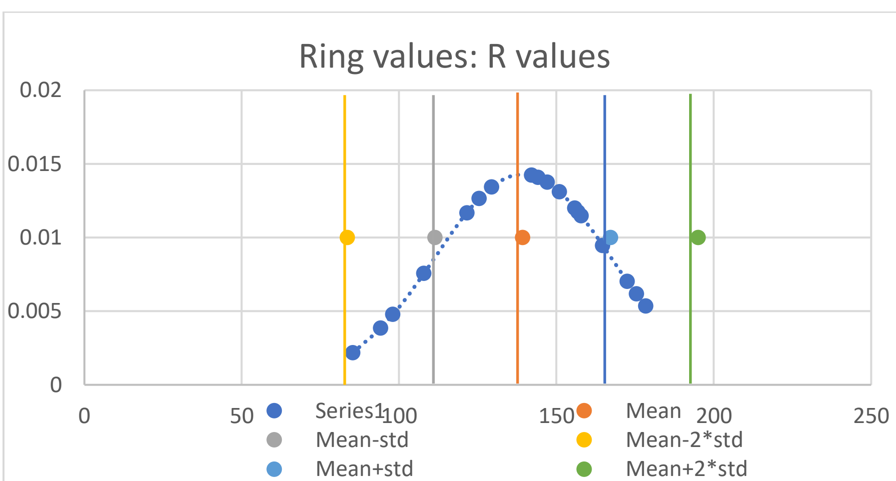

Figure 12: Graph of Red values for the Orange ring

Conclusion: The test results shows that we can use the RGB values to detect the color of the ring accurately.

Action: The test report will be sent to the software team to make an accurate ring detection algorithm using the results.

Distribution: Software Development, Project management

Test 6: Ultrasonic accuracy test

Date: 3/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: N/A
Software Version: N/A

Goal: To determine which ultrasonic sensor is the best for our use case.

Procedure:
1. Place the robot one grid length away from the wall.
2. Run UltrasonicAccuracyTest.java
3. Press any button on the robot. The robot will measure the distance from the wall and display on the console/display. Record the values.
4. Repeat these 30 times and get compute the mean and standard deviation.
5. Repeat step 2 and 3 for 2 grid lengths away from the wall.
6. Repeat steps 2 to 4 for the other 2 ultrasonic sensors.

Expected result: The sensors will read the distance from the wall and display the results on the console. The distances read from the sensors should have an error of less than 2cm.

Test Report: The test was done 30 times for each grid length for each sensor.

Trial	1st Sensor		2nd Sensor		3rd Sensor	
	Distance (30.48cm)	Distance (60.96cm)	Distance (30.48cm)	Distance (60.96cm)	Distance (30.48cm)	Distance (60.96cm)
1	30	60	30	61	32	61
2	31	60	30	61	31	63
3	32	61	29	59	32	63
4	31	61	32	63	31	62
5	31	61	31	61	33	60

6	31	61	31	63	29	60
7	31	61	30	61	29	62
8	31	61	31	63	33	62
9	32	60	29	61	34	60
10	33	60	32	60	29	63
11	31	61	32	62	30	62
12	32	61	31	62	32	61
13	31	61	30	60	33	62
14	31	61	31	62	29	62
15	30	61	31	62	32	62
16	32	61	32	61	32	63
17	31	62	32	62	31	61
18	33	60	29	62	32	62
19	32	61	31	61	30	61
20	32	61	31	63	34	62
21	31	61	32	62	31	59
22	32	61	31	61	33	64
23	31	62	31	62	31	63
24	31	61	31	63	34	61
25	33	61	31	62	34	61
26	31	60	33	62	32	61
27	33	61	32	62	31	63
28	31	61	32	60	34	63
29	31	60	33	63	31	62
30	30	60	32	61	32	60
Mean	31.4	60.8	31.1	61.6	31.7	61.7
STD.dev	0.84063468	0.54160256	1.04403065	1.0198039	1.55241747	1.15902257

Table 9: Results from the Ultrasonic accuracy test

From the data, all the sensors produced values close to the actual distance from the wall therefore we had to look at the standard deviation to decide on what sensor to choose. The first sensor had the smallest standard deviation.

Conclusion: We decided to choose the first sensor for ultrasonic localization as it was the best of the 3.

Action: The report is to be sent to the hardware team to make the changes to the robot. Update Gantt Chart.

Distribution: Hardware development, Project Management.

Test 7: Light Poller

Date: 5/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4 from the hardware documentation
Software Version: N/A

Goal: To determine the light differential required to accurately detect the lines on the board.

Procedure:

1. Place robot in the middle of the tile like in the figure below.
2. Run DataAcquisition.java.
3. The robot moves forward and reads the values from the light sensors and displays it on the console/display.
4. Copy and paste the values on excel and plot a differential graph to show the required differential.

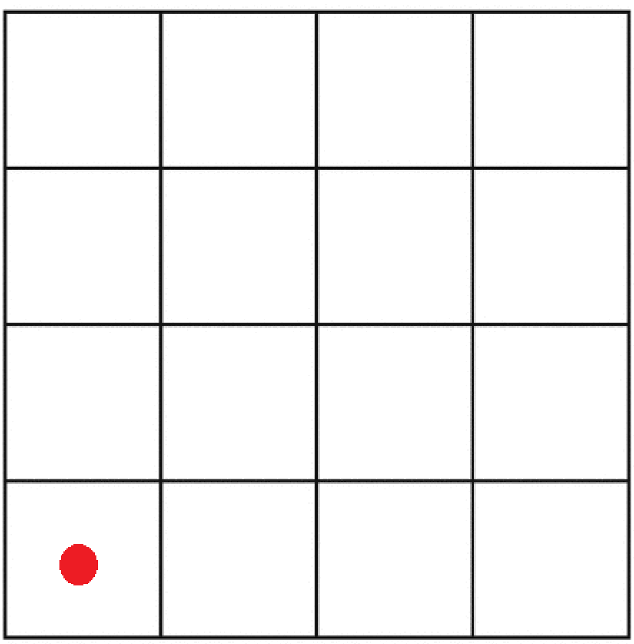

Figure 13: Starting point of the robot

Expected result: The graph should show when the lines are detected by the sensor.

Test Report: The robot went over 3 lines and the graphs below show the results.

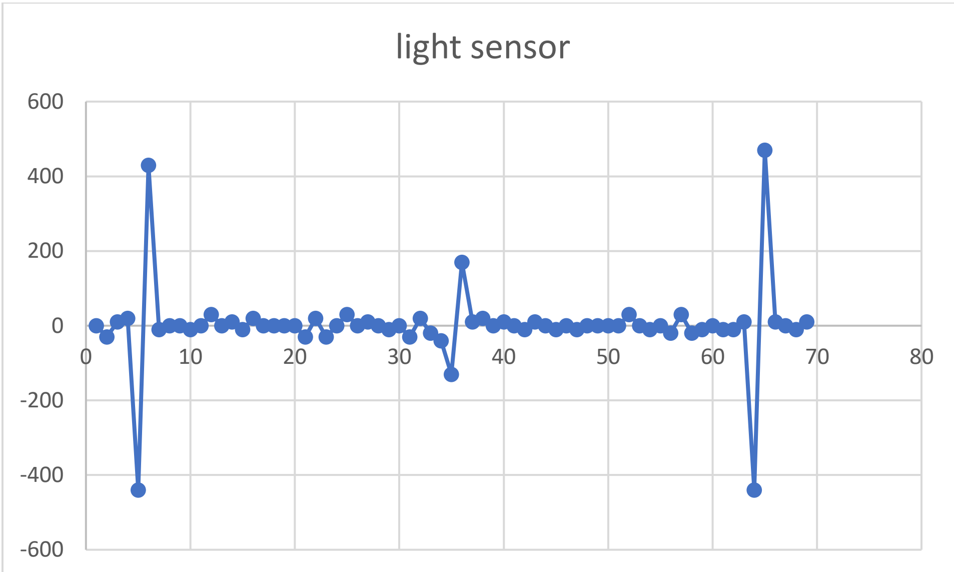

Figure 14: Differential graphs of right light sensor.

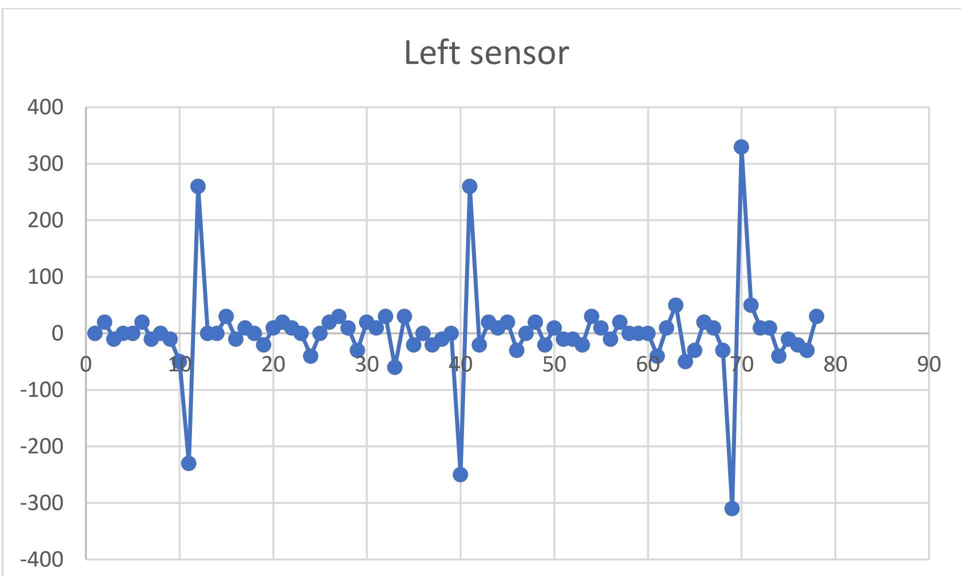

Figure 15: Differential graphs of left light sensor.

The dips in the graph are when the sensor detects the lines.

Conclusion: A differential of -100 would allow the light sensor to accurately detect lines.

Action: This report will be sent to the software team to make necessary changes. Update Gantt Chart

Distribution: Software development, Project Manager

Test 8: Ring detection

Date: 5/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.1 from the hardware documentation
Software Version: N/A

Goal: To determine the reliability of ring color detection.

Procedure:

1. Place the light sensor 2cm from the ring.
2. Run ColorDetectionTest.java
3. Press any button for the robot to detect the color of the ring. The color will be displayed on the console/display.
4. Record the value.
5. Repeat steps 3 and 4, 15 times.
6. Repeat steps 3 to 5 for all the other rings.
7. Repeat steps 3 to 6 for distances 4cm and 6 cm.

Expected result: The rings should accurately detect the color for 2cm and 4cm but may not correctly detect the ring for 6cm because the light sensor does not work accurately at that distance.

Test Report: The tables below show the results:

Distance (2cm)				
	Ring			
	Blue Ring	**Green Ring**	**Yellow Ring**	**Orange Ring**
Trial				
1	Blue	Green	Yellow	Orange
2	Blue	Green	Yellow	Orange
3	Blue	Green	Yellow	Orange
4	Blue	Green	Yellow	Orange
5	Blue	Green	Yellow	Orange
6	Blue	Green	Yellow	Orange
7	Blue	Green	Yellow	Yellow
8	Blue	Green	Yellow	Orange
9	Blue	Green	Orange	Orange
10	Blue	Green	Yellow	Orange
11	Blue	Green	Yellow	Orange

12	Blue	Blue	Yellow	Orange
13	Blue	Green	Yellow	Orange
14	Blue	Green	Yellow	Orange
15	Blue	Green	Yellow	Orange
Accuracy	100%	93.33%	93.33%	93.33%

Table 10: Color detection results from 2cm

Distance (4cm)				
	Ring			
	Blue Ring	**Green Ring**	**Yellow Ring**	**Orange Ring**
Trial				
1	Blue	Green	Orange	Orange
2	Blue	Green	Orange	Orange
3	Blue	Blue	Yellow	Orange
4	Blue	Green	Yellow	Orange
5	Blue	Green	Yellow	Orange
6	Blue	Green	Yellow	Orange
7	Blue	Green	Yellow	Orange
8	Blue	Green	Yellow	Orange
9	Blue	Green	Yellow	Orange
10	Blue	Green	Yellow	Yellow
11	Blue	Green	Yellow	Orange
12	Blue	Green	Yellow	Orange
13	Blue	Green	Yellow	Orange
14	Blue	Green	Yellow	Orange
15	Blue	Green	Yellow	Orange
Accuracy	100%	93.33%	86.67%	93.33%

Table 11: Color detection results from 4cm

Distance (6cm)				
	Ring			
	Blue Ring	**Green Ring**	**Yellow Ring**	**Orange Ring**
Trial				
1	Blue	Green	Orange	Orange
2	Blue	Green	Yellow	Yellow
3	Blue	Green	Yellow	Orange
4	Blue	Green	Yellow	Orange
5	Blue	Green	Yellow	Orange
6	Blue	Green	Orange	Orange
7	Blue	Blue	Yellow	Orange
8	Blue	Blue	Yellow	Yellow
9	Blue	Green	Yellow	Orange
10	Blue	Green	Yellow	Orange

11	Blue	Green	Yellow	Orange
12	Blue	Green	Yellow	Orange
13	Blue	Green	Orange	Orange
14	Blue	Blue	Yellow	Orange
15	Blue	Green	Yellow	Orange
Accuracy	100%	80.00%	80.00%	86.67%

Table 12:Color detection results from 6cm

The accuracy of the detection was always 80% or above. As expected the at 6cm the results are not as accurate as the 2cm.

Conclusion: Ring detection reliable. It is recommended that the light sensor come at least 2cm to the ring before trying to detect for accurate result.

Action: This report will be sent to the software team and hardware team to make necessary changes. Update Gantt Chart.

Distribution: Software development, Project Management

<u>Test 9: Odometer I</u>

Date: 5/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.1 from the hardware documentation
Software Version: N/A

Goal: To determine the reliability of the odometer.

Procedure:

1. Place the robot in the middle of the left tile like in the figure below.
2. Mark the starting position.
3. Run the squaredriver.java from lab 2.
4. The robot will travel in a square and return to the starting position like shown in the diagram below. Measure the resulting Xs and Ys distances with respect to the starting position of the robot and record the X and Y values shown on the odometer.
5. Compute the Euclidean error, mean value and standard deviation.

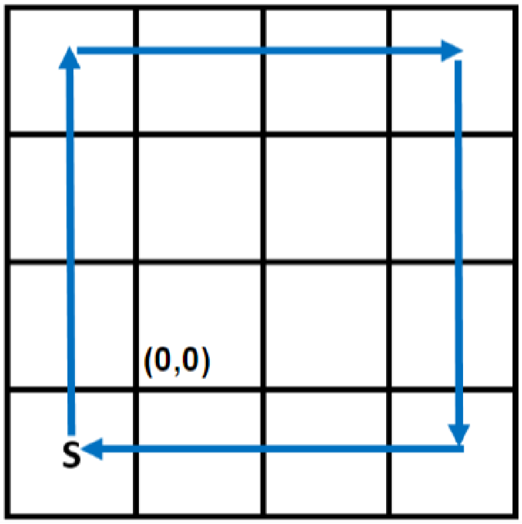

Figure 16: Path the robot follows for odometry

Expected Results: The Euclidean error should be smaller than 2cm.

Test Report: The test was performed 30 times and the results are shown in the table below:

Trial	Xs	Ys	X	Y	Error
1	-0.40	0.10	0.21	0.38	0.64
2	0.30	-1.50	-0.63	-0.72	1.24
3	0.30	-1.20	0.45	-0.37	0.84
4	-0.10	0.20	0.34	0.21	0.42
5	0.40	0.20	0.61	-0.04	0.37
6	-0.20	-0.60	-0.17	0.64	1.20
7	0.30	0.70	0.06	-0.67	1.35
8	0.00	0.10	0.29	0.07	0.27
9	0.30	0.60	0.48	-0.35	0.92
10	0.10	0.80	-0.30	0.06	0.82
11	0.80	0.30	0.24	-0.60	1.07
12	0.20	-0.40	0.40	-0.52	0.20
13	0.60	0.00	0.36	0.29	0.36
14	-0.30	-0.20	-0.10	-0.09	0.20
15	0.10	1.00	0.11	-0.28	1.26
16	-0.10	0.10	-0.42	-0.66	0.88
17	0.60	-0.30	-0.02	-0.15	0.62
18	0.60	-0.80	-0.10	-0.27	0.86

19	-0.30	-0.10	0.30	-0.73	0.86
20	0.00	-0.10	0.07	-0.41	0.30
21	0.00	-0.70	0.44	-0.73	0.46
22	-0.80	-0.50	0.05	-0.51	0.87
23	-0.30	0.30	-0.42	-0.64	0.98
24	-0.30	-0.30	0.00	-0.33	0.31
25	0.10	-0.10	-0.33	-0.77	0.82
26	-0.40	-0.20	-0.20	-0.15	0.19
27	-0.60	-0.10	-0.47	-0.35	0.25
28	-0.70	-0.20	-0.17	0.37	0.79
29	-0.60	-0.30	-0.25	-0.05	0.44
30	-0.30	-0.60	-0.41	-0.44	0.20
Mean			0.01	-0.26	0.67
Standard deviation			0.33	0.37	0.36

Table 13: Results from the odometer test

The mean values and Euclidean error were all below 2 cm.

Conclusion: The odometer is reliable in the current hardware configuration as the error is in the acceptable margin. It should be noted that this test was done when the battery was outputting at 8.0V and accurate localization is not guaranteed for anything less than 8.0V.

Action: No further testing for odometer is required in this current hardware configuration. Update Gantt Chart.

Distribution: Software development, Project management

<u>Test 10: Navigation I</u>
Date: 5/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.1 from the hardware documentation
Software Version: N/A

Goal: Determine if the robot can navigate from one point to another reliably.

Procedure:

1. Place the robot at (1,1) point on the 8x8 grid like in the figure below.
2. The robot must travel to 5 different waypoints. The robot must stop and beep after reaching every waypoint. The waypoints are (3,2), (3,6), (5,7), (4,5), (4,1) like shown in the figure below.

3. Run NavigationTest.java.
4. One the robot finishes measure the X and Y relative to the grid intersection and record it. Also record the odometer values that are displayed on the console/display.
5. Compute the Euclidean error, mean and standard deviation.

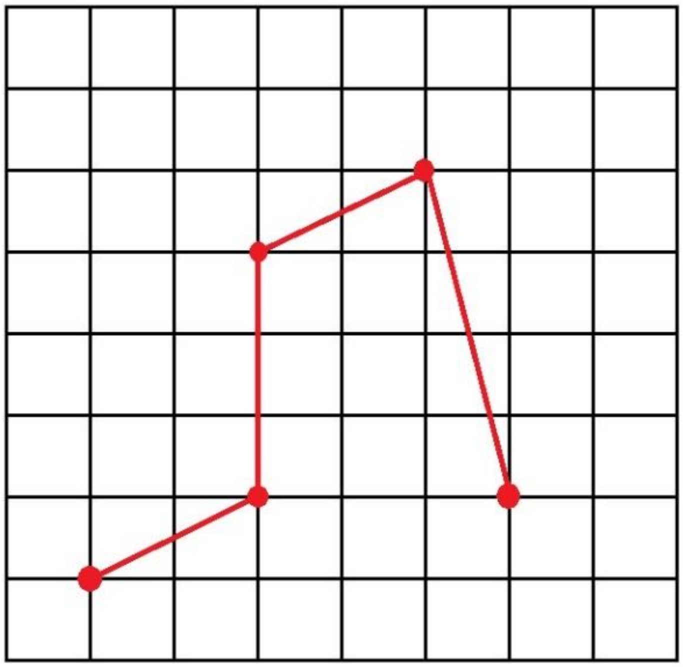

Figure 17: Path the robot follows for navigation

Expected result: The results should be close to the actual x, y and theta which are 182.88cm, 60.96cm and 180 degrees

Test Report: The test was done 30 times and the results are shown in the table below:

Trial	Measured X (cm)	Measured Y (cm)	Odometer X (cm)	Odometer Y (cm)	EU Error
1	180.10	59.49	182.30	62.01	3.35
2	182.30	61.70	180.80	58.01	3.98
3	185.30	60.10	181.80	61.45	3.75
4	183.10	62.65	182.60	59.49	3.20
5	180.80	60.75	178.90	61.68	2.12
6	182.50	62.10	180.80	60.78	2.15
7	182.50	60.61	179.20	62.89	4.01
8	181.90	60.84	183.40	63.65	3.19
9	181.40	62.00	181.80	63.00	1.08
10	184.00	62.90	179.80	62.33	4.24

11	182.30	61.40	180.30	61.72	2.03
12	182.90	61.85	185.40	61.68	2.51
13	183.10	58.82	182.90	62.37	3.56
14	185.10	58.39	181.40	61.11	4.59
15	184.00	62.00	181.10	61.02	3.06
16	180.60	59.16	183.20	61.02	3.20
17	182.80	58.67	181.70	61.41	2.95
18	183.00	62.08	183.10	58.96	3.12
19	184.40	58.24	182.80	62.43	4.49
20	180.30	62.70	184.00	63.00	3.71
21	180.90	60.42	183.90	60.16	3.01
22	181.60	61.01	181.40	62.10	1.11
23	182.20	57.38	180.80	57.21	1.41
24	181.20	60.24	181.10	60.20	0.11
25	184.90	60.64	181.70	62.78	3.85
26	182.30	61.84	182.10	61.41	0.47
27	183.30	62.52	183.50	60.09	2.44
28	182.90	60.15	182.20	62.37	2.33
29	181.40	62.87	181.70	62.55	0.44
30	177.40	62.14	180.30	63.13	3.06
Mean	182.22	60.92	181.87	61.87	3.01
STD.dev	1.833	1.833	1.444	1.713	0.407

Table 14: Results from navigation test

The mean error is 3.01 which is above the acceptable margin of error. The cause of the improper navigation stems from the lack of rigidity in the ring collector part of the robot.

Conclusion: Minor modifications are required on the ring collector section of the robot for accurate navigation

Action: This report will be sent to the hardware team to improve the design of the robot and software to rework the code accordingly. Update Gantt Chart.

Distribution: Hardware Development, Software development, Project management.

Test 11: Hardware stability I
Date: 5/11/2018

Tester: Ashish
Author: Ashish
Hardware Version: 4.4.1 from the hardware documentation
Software Version: N/A

Goal: Determine whether the hardware design is stable enough when the robot is moving and turning. Also check if the robot can move reliably while the rings are loaded.

Procedure:

1. Place the robot on the ground.
2. Run the HardwareStabilityTest.java. The robot moves forward, backwards and turns.
3. Test the robot for each direction at speeds of 50, 100, 200 and 300.
4. Test turn to 90, 180, 270 and 360.
5. Repeat steps 3 and 4 with 0 rings, 1 ring, 2 rings and 4 rings.

Expected result: The robot should remain stable and adding weights should not make it any less stable.

Test Report: Each of the tests were done 5 times for a total of 120 trials. The robot was stable when there were no rings loaded. After adding one ring the turning of the robot was slightly affected because the center of gravity changed. After adding 4 rings the robot stopped turning properly.

Conclusion: The ring collection section of the robot needs to be altered to be more rigid. Counter weights are also required on the back of the robot to balance the robot once the rings have been collected by the robot

Action: This report will be sent to the hardware team to immediately change hardware design, so it supports the rings, moves at the same time and the structure as a whole is more rigid. Update Gantt Chart.

Distribution: Hardware Development, Project management

<u>Test 12: Motor accuracy test(Pre-beta demo)</u>
Date: 12/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.2 form the hardware documentation
Software Version: N/A

Goal: Determine which two of the six motors available reliable enough for the requirements

Procedure:

1. Run MotorWheelAccuracy.java.
2. Spin the wheel one full rotation.
3. The robot will measure the rotation using the odometer and display the angle rotation on the console/display. Record the value.
4. Repeat steps 1 to 3, 20 times and compute the mean and standard deviation.
5. Repeat steps 1 to 4 with the other 6 motors

Expected result: The odometer should display around 360 ± 2.

Test Report: the test was performed 20 times and the results are shown in the table below.

Trial	**1st Motor**	**2nd Motor**	**3rd Motor**	**4th Motor**	**5th Motor**	**6th Motor**
1	362	363	365	360	359	367
2	360	367	363	359	356	366
3	359	355	365	358	366	360
4	358	367	362	360	361	359
5	360	363	362	362	362	356
6	362	366	367	361	364	366
7	361	360	363	358	365	361
8	358	359	359	359	363	362
9	359	356	356	359	367	364
10	359	366	366	362	363	365
11	362	361	356	362	359	362
12	360	362	366	360	356	367
13	359	364	361	359	366	363
14	358	365	362	358	363	359
15	360	363	364	360	367	362
16	362	365	365	362	355	362
17	361	362	363	361	364	360
18	358	362	359	358	365	359
19	359	367	362	359	363	358
20	359	363	360	359	365	360
Mean	359.8	362.8	362.3	359.8	362.4	361.9
Std dev	1.4	3.31	3.03	1.4	3.58	3.06

The 2nd, 3rd,5th and 6th motors have a standard deviation that is too high and will give bad results if used for navigation.

The 1st and 4th motors had a mean closest to the actual result. They also had the lowest standard deviation out of all the other motors

Conclusion: The best motors are the first and fourth.

Action: This report will be sent to the hardware team to change the robot to use motors 1 and 4. Update Gantt Chart.

Distribution: Hardware Development, Project management.

<u>Test 13: Hardware stability II (Pre-beta demo)</u>
Date: 11/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.2 from the hardware documentation
Software Version: Beta Demo

Goal: Determine whether the hardware design is stable enough when the robot is moving and turning. Also check if the robot can move reliably while the rings are loaded, paying close attention to the flaws experienced in Test 11.

Procedure:

1. Place the robot on the ground.
2. Run the HardwareStabilityTest.java. The robot moves forward, backwards and turns.
3. Test the robot for each direction at speeds of 50, 100, 200 and 300.
4. Test turn to 90, 180, 270 and 360.
5. Repeat steps 3 and 4 with 0 rings, 1 ring, 2 rings and 4 rings.

Expected result: The robot should remain stable and adding weights should not make it any less stable.

Test Report: Each of the tests were done 5 times for a total of 120 trials. The robot was stable when there were no rings loaded. After 1 ring or even 4 rings were added to the robot, it could still move properly around the board. The ring collection section of the robot is very rigid and does not flex at all even when 4 rings are on it.

Conclusion: Flaws from the previous hardware stability tests have all been fixed.

Action: No further testing for hardware stability is required for this hardware configuration. Update Gantt Chart.

Distribution: Hardware Development, Project management

<u>Test 14: localization III (Pre-Beta Demo)</u>
Date: 11/11/2018
Tester: Ashish
Author: Ashish

Hardware Version: 4.4.2 from the hardware documentation
Software Version: Beta Demo

Goal: Determine if the robot can first localize with the ultrasonic sensor and then light localize reliably.

Procedure:

1. Place the robot facing away from the wall on the left corner tile.
2. Run the LocalizationTest.java
3. Press any button to start the program.
4. After ultrasonic localization finishes measure and record the angle.
5. Press any button again to start light localization.
6. The robot should move in X first, then Y direction correcting the angle as it detects the lines on the board.
7. Once localization is completed measure the Euclidean error and final angle of the robot.
8. Measure the time taken to complete the whole localization.

Expected result: The Final angle is expected to be within 5 degrees and the Euclidean error is expected to be within 2cm. The time taken for localization is should be less than 30 seconds.

Test Report: the test was performed 40 times and the results are shown in the table below:

Trial	US Angle (deg)	Euclidean Error (cm)	Final Angle (deg)	Time (seconds)
1	4	1.7	3	26
2	5	0.9	2	26
3	3	0.3	3	26
4	4	2.0	2	26
5	3	1.4	1	26
6	5	1.5	1	26
7	6	1.5	2	26
8	2	0.3	1	26
9	5	1.3	2	27
10	5	1.0	0	27
11	0	1.2	2	27
12	0	1.1	4	27
13	7	2.8	2	27

14	4	2.0	1	27
15	4	-0.4	3	28
16	3	1.9	2	28
17	3	3.3	2	28
18	4	0.4	3	28
19	3	3.0	2	28
20	3	-0.1	3	28
21	5	1.9	1	28
22	2	0.9	2	28
23	1	1.6	2	28
24	4	2.0	3	28
25	4	1.5	4	28
26	5	0.4	3	29
27	4	1.0	1	29
28	5	1.2	2	29
29	5	2.1	1	29
30	6	1.1	1	29
31	5	1.4	1	29
32	1	1.5	3	29
33	5	1.9	1	30
34	7	0.1	3	30
35	5	1.6	2	30
36	5	1.1	3	30
37	5	0.5	3	30
38	6	1.1	0	30
39	5	0.6	2	30
40	3	1.7	2	30
Mean	4	1.3	2	28
STD.dev	1.544	0.792	0.942	1.387

Table 15: Results from Localization test

The mean of the final angle is 2 degrees and the mean Euclidean error is 1.3 which is in the acceptable margin of error. The mean time taken for localization is 28 seconds.

Conclusion: The robot always localized within 2 degrees and less than 2cm which is within the acceptable margin of error. The average time for localization was 28 seconds which is below the required 30 seconds. It should be noted that this test was done when the battery was outputting at 7.8V or higher and accurate localization is not guaranteed for anything less than 7.8V.

Action: No further tests are required for localization unless the hardware configuration is changed. Update Gantt Chart.

Distribution: Software Development, Project management

<u>Test 15: Navigation II</u>
Date: 11/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.2 from the hardware documentation
Software Version: Beta demo

Goal: Determine if the robot can navigate from one point to another reliably and to also check if the flaws from the previous navigation tests have been fixed.

Procedure:

1. Place the robot at (1,1) point on the 8x8 grid like in the figure below.
2. The robot must travel to 5 different waypoints. The robot must stop and beep after reaching every waypoint. The waypoints are (3,2), (3,6), (5,7), (4,5), (4,1) like shown in the figure below.
3. Run NavigationTest.java.
4. One the robot finishes measure the X and Y relative to the grid intersection and record it. Also record the odometer values that are displayed on the console/display.
5. Compute the Euclidean error, mean and standard deviation.

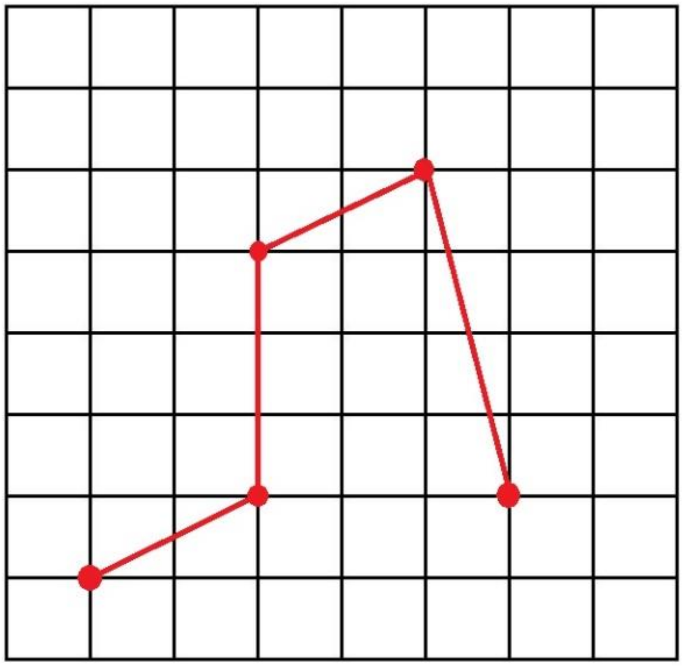

Figure 18: Path the robot follows for navigation test

Expected result: The results should be close to the x, y and theta

Test Report: Test was performed 30 times and the results are shown in the table below:

Trial	Measured X (cm)	Measured Y (cm)	Odometer X (cm)	Odometer Y (cm)	Error
1	178.3	59.4	182.4	61.0	4.36
2	186.7	61.4	185.8	62.5	1.41
3	179.6	60.4	179.6	60.5	0.57
4	181.2	60.6	181.9	61.0	0.85
5	180.5	60.9	184.9	61.1	4.38
6	181.1	60.0	181.4	60.6	0.67
7	180.2	60.2	181.4	59.5	1.44
8	186.7	61.5	181.6	61.0	5.18
9	179.8	61.9	181.9	60.1	2.72
10	181.2	61.6	178.1	61.7	3.14
11	180.7	60.8	182.5	61.9	2.14
12	179.7	62.2	181.6	60.7	2.44
13	181.1	59.8	179.9	62.3	2.81

14	181.3	62.3	182.5	61.4	1.46
15	178.6	60.7	182.4	61.4	3.83
16	183.1	60.8	181.5	59.9	1.79
17	178.2	63.1	182.5	61.6	4.52
18	180.4	60.6	182.1	62.7	2.68
19	179.9	61.4	181.6	62.3	1.90
20	183.5	61.4	180.3	62.1	3.27
21	176.3	60.3	181.9	60.8	5.58
22	181.4	59.3	181.6	60.8	1.50
23	180.1	59.7	182.7	62.5	3.80
24	183.5	61.0	179.7	61.5	3.87
25	179.8	62.8	184.1	61.4	4.55
26	183.2	60.9	179.3	59.6	4.15
27	178	61.0	180.9	59.4	3.34
28	179.9	61.7	178.2	60.5	2.07
29	182.8	61.2	183.4	61.3	0.62
30	180.9	59.3	176.3	61.2	5.02
Mean	180.9	60.9	181.5	61.1	0.58
STD.dev	2.274	0.963	1.949	0.888	0.33

Table 16: Results from the navigation tests

The mean error is 0.58 which is within the acceptable margin of error. There was also no flexing in the ring collection section of the robot.

Conclusion: Flaws from the previous navigation test have been fixed and the errors are also in the acceptable margin of error. It should be noted that this test was done when the battery was outputting at 7.8V or higher and accurate localization is not guaranteed for anything less than 7.8V.

Action: No further testing is required unless the hardware configuration is changed. Update Gantt Chart.

Distribution: Software Development, Project management

<u>Test 16: Ring Grabbing I (Pre-beta demo)</u>
Date: 11/11/2018
Tester: Ashish
Author: Ashish

Hardware Version: 4.4.2 from the hardware documentation
Software Version: N/A

Goal: Determine if the robot can grab the robot successfully.

Procedure:

1. Place the robot on the one grid intersection behind the tree like in the figure below.
2. Run the RingGrabbingTest.java
3. The Robot will go straight towards the tree, detect the ring, collect the ring and back off.
4. Record the number of times the ring is successfully collected by the robot and the color of the ring collected.

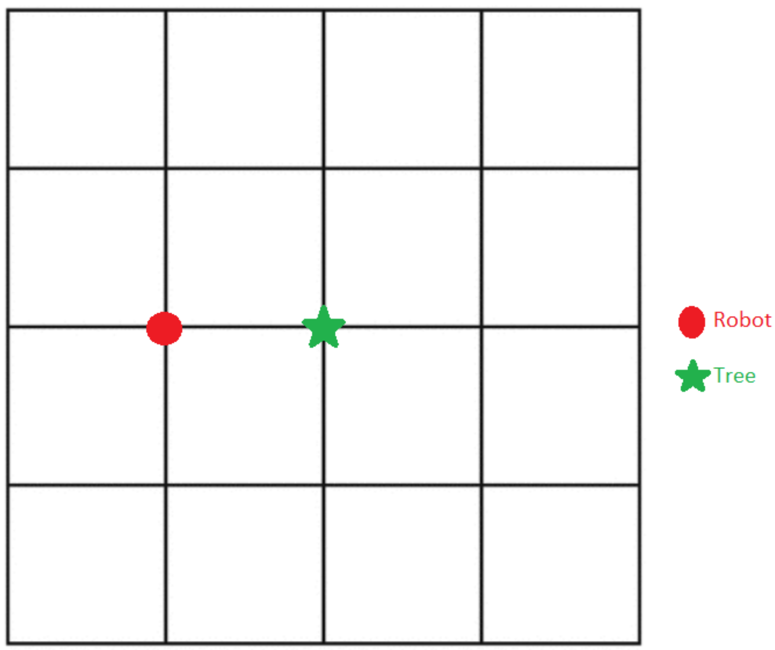

Figure 19: Where the robot is to be placed relative to the tree

Expected result: The robot will successfully collect the ring.

Test Report: The test was run 20 times and the results are shown in the table below:

Trial	Ring grabbed	Ring color
1	Yes	Blue
2	Yes	Orange
3	Yes	Yellow
4	Yes	Green

5	Yes	Blue
6	Yes	Green
7	Yes	Orange
8	Yes	Yellow
9	No	Orange
10	Yes	Yellow
11	Yes	Green
12	Yes	Orange
13	Yes	Blue
14	No	Orange
15	Yes	Green
16	Yes	Orange
17	Yes	Blue
18	Yes	Orange
19	Yes	Yellow
20	Yes	Green

Table 17: Results from the ring grabbing test

The ring collection algorithm has a 90% success rate. The robot only failed when the it was not positioned properly before the program was started. The robot only fails when the smallest ring (orange ring) is on the top part of the tree.

Conclusion: The ring collection algorithm is good enough to be used in the beta demo.

Action: Further testing may be required if the code or hardware configuration is changed to get the ring to be grabbed always. Update Gantt Chart.

Distribution: Software Development, Project management

Test 17: Odometry II (Pre-beta demo)
Date: 11/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.2 from the hardware documentation
Software Version: Beta demo

Goal: Determine if the change in hardware configuration has changed odometry.

Procedure:

1. Place the robot in the middle of the left tile like in the figure below.
2. Mark the starting position.
3. Run the squaredriver.java from lab 2.
4. The robot will travel in a square and return to the starting position like shown in the diagram below. Measure the resulting Xs and Ys distances with respect to the starting position of the robot and record the X and Y values shown on the odometer.
5. Compute the Euclidean error, mean value and standard deviation.

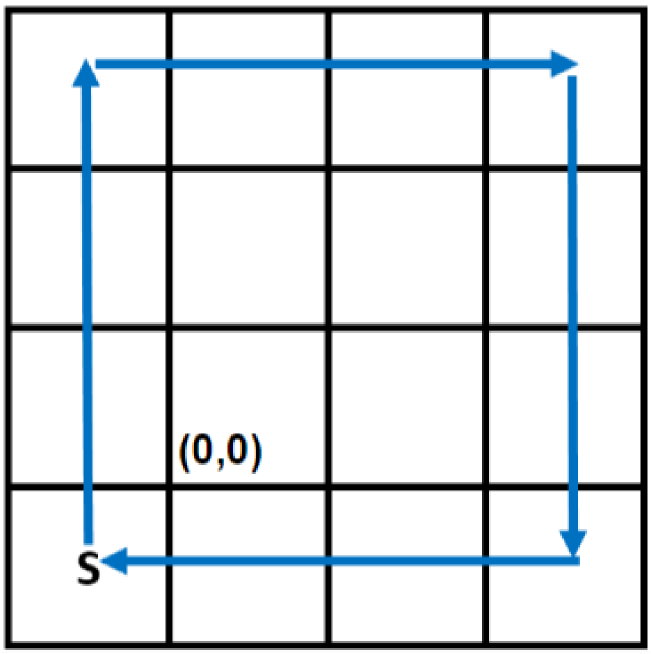

Figure 20: Path the robot follows for the odometry test

Expected result: The error values should be as close to zero as possible.

Test Report: The test was done 30 times and the results are shown in the table below:

Trial	Xs	Ys	X	Y	Error
1	0.87	0.60	-0.01	0.81	0.91
2	0.70	0.34	0.67	1.21	0.87
3	0.36	0.71	0.76	0.08	0.74
4	0.38	0.47	0.35	1.28	0.81
5	0.51	-0.13	0.77	1.33	1.48
6	0.05	1.07	0.65	0.37	0.92
7	0.09	0.26	0.62	0.39	0.54

8	0.94	0.97	0.50	-0.25	1.29
9	0.99	-0.73	0.67	-0.09	0.72
10	0.17	0.31	-0.09	0.03	0.39
11	0.22	0.23	0.44	0.82	0.63
12	1.53	2.03	-0.50	0.73	2.41
13	0.61	-0.08	-0.42	0.20	1.07
14	0.33	1.02	0.17	0.56	0.49
15	0.96	0.37	1.83	0.17	0.89
16	0.63	-0.15	1.05	0.44	0.72
17	0.65	0.40	-0.71	0.75	1.40
18	0.25	0.70	0.75	0.17	0.72
19	0.48	-0.27	0.43	0.13	0.41
20	0.86	-0.65	0.87	0.33	0.98
21	0.43	-0.36	-0.44	0.08	0.98
22	1.84	-0.24	0.95	0.24	1.01
23	1.05	-0.02	0.24	1.29	1.54
24	1.06	0.30	0.51	-0.16	0.72
25	0.12	0.99	0.65	0.26	0.91
26	0.16	0.47	1.17	0.36	1.01
27	0.34	1.25	0.73	0.84	0.56
28	-0.34	1.05	0.52	0.32	1.13
29	0.94	1.47	1.26	0.84	0.70
30	0.29	0.50	0.99	-0.56	1.27
Mean	0.58	0.43	0.51	0.43	0.07
STD.dev	0.455	0.621	0.550	0.467	0.180

Table 18: Results from the odometry Test

Figure 21: Results from the odometry test

The results show that odometry is still in the acceptable margin of error

Conclusion: The change in hardware configuration did not affect the odometry. It should be noted that this test was done when the battery was outputting at 7.8V or higher and accurate odometer values are not guaranteed for any voltage less than 7.8V.

Action: No further testing is required unless the hardware configuration is changed. Update the Gantt chart

Distribution: Software Development, Project management

<u>Test 18: Integration Test I (Beta demo)</u>
Date: 12/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.2 from the hardware documentation
Software Version: Beta Demo Version 1

Goal: Determine if the robot can successfully localize, navigate and collect a ring on an 8x8 board

Procedure:

1. First change the server IP on the BetaDemo.java to your computers IP address.
2. Run DPMServer.jar and insert the coordinates for starting corner, Green_UR, Green_LL, TNG_LL, TNG_UR, Island_LL, Island_UR and TG. Make sure the rest of the fields are left blank. Also insert the team number that is on BetaDemo.java.
3. Place the robot at corner 1 and run BetaDemo.java.
4. Once the robot has connected, (you should see Team 20 connected on DPMServer) click start.
5. The robot should start localizing. Upon finishing localization, the robot should beep 3 times. Measure the angle error, X error and Y error. Also record the time taken to localize.
6. The robot should then navigate to (TNG _LL+ 0.5, TNG_LL - 0.5) or (TNG_UR + 0.5, TNG_UR – 0.5) depending on the orientation of the tunnel. Measure the X error, Y error and the angle error.
7. The robot should then pass through the tunnel and localize to the nearest grid intersection. Record whether the robot successfully passed the tunnel
8. After completing localization, the robot will navigate to the closest grid intersection before the ring set.
9. The robot should localize again.
10. The robot should then move forward slowly while the color sensor sweeps at 45-degree angles to look for a ring.
11. If there is a ring, the robot will beep according to the color of the ring and the ring sweepers with swipe the ring into the ring collector.
12. If there is no ring the robot should go back to the grid intersection, go to the next side of the tree and repeat steps 8 to 11 until a ring has been found.
13. Once a ring has been found, the robot should back off and beep 5 times to signal that the demo is over. Record whether ring was collected.
14. Repeat this procedure with different coordinates of the ring set and tunnel.

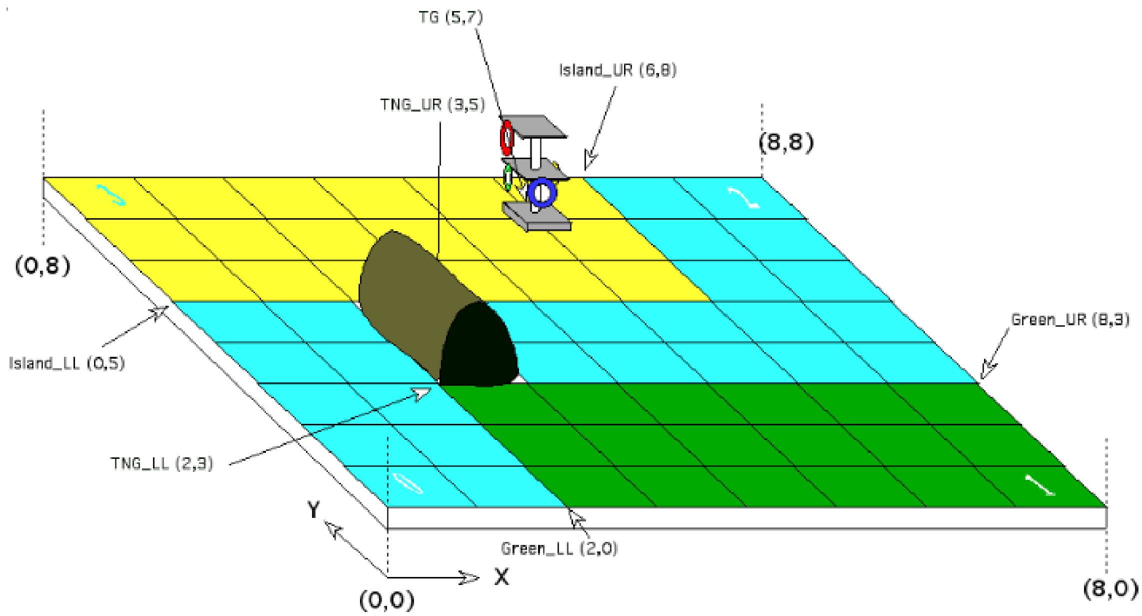

Figure 22: Beta Demo map

Expected result: The robot should successfully collect the ring.

Test Report: The results of the beta demo are shown in the table below:

Conclusion: The results conclude that the robot fails to properly navigate to coordinate (4,7) 60% of the time thereby failing to always to grab the ring.

Action: This report will be sent to the software team to fix navigation of the robot. Update the Gantt chart.

Distribution: Software Development, Project management

Trial	Localization 1	Angle error (deg)	X error (cm)	Y error (cm)	Eu error (cm)	Navigation 1	Angle error (deg)	X error (cm)	Y error (cm)	Tunnel pass	Navigation 2	Angle error (deg)	X error (cm)	Y error (cm)	Ring collected
1		3.0	0.30	0.30	0.42		1.0	1.20	2.10	Yes		2.0	0.20	2.10	Yes
2		2.0	0.10	1.40	1.40		2.0	1.30	1.50	Yes		3.0	0.10	1.10	No
3		3.0	1.20	1.40	1.84		1.0	0.70	1.60	No		4.0	1.10	1.00	No
4		1.0	1.20	2.30	2.59		2.0	0.80	2.70	Yes		1.0	1.00	0.90	Yes
5		2.0	2.10	1.50	2.58		3.0	1.90	3.00	No		3.0	1.20	0.30	No
6		3.0	1.30	0.80	1.53		2.0	2.20	2.90	Yes		2.0	0.70	0.20	Yes
7		4.0	0.60	0.90	1.08		2.0	2.10	2.00	Yes		1.0	0.50	0.10	No
8		2.0	0.70	1.10	1.30		1.0	1.20	3.90	No		1.0	0.90	0.70	No
9		2.0	0.90	1.20	1.50		1.0	3.10	2.20	Yes		3.0	1.30	1.10	Yes
10		1.0	1.00	1.14	1.52		1.0	1.40	2.39	No		1.0	1.10	1.00	No
Mean		2.3	0.94	1.20	1.58		1.6	1.59	2.43			2.1	0.81	0.85	
STD.dev		0.9	0.54	0.50	0.62		0.7	0.70	0.68			1.0	0.40	0.55	

Table 19: Results from the beta demo

<u>Test 19: Integration Test II (Beta demo)</u>
Date: 13/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.2 form the Hardware documentation
Software Version: Beta Demo

Goal: Determine if the robot can successfully localize, navigate and collect a ring on an 8x8 board.

Procedure:

1. First change the server IP on the BetaDemo.java to your computers IP address.
2. Run DPMServer.jar and insert the coordinates for starting corner, Green_UR, Green_LL, TNG_LL, TNG_UR, Island_LL, Island_UR and TG. Make sure the rest of the fields are left blank. Also insert the team number that is on BetaDemo.java.
3. Place the robot at corner 1 and run BetaDemo.java.
4. Once the robot has connected, (you should see Team 20 connected on DPMServer) click start.
5. The robot should start localizing. Upon finishing localization, the robot should beep 3 times. Measure the angle error, X error and Y error.
6. The robot should then navigate to (TNG _LL+ 0.5, TNG_LL - 0.5) or (TNG_UR + 0.5, TNG_UR – 0.5) depending on the orientation of the tunnel. Measure the X error, Y error and the angle error.
7. The robot should then pass through the tunnel and localize to the nearest grid intersection. Record whether the robot successfully passed the tunnel
8. After completing localization, the robot will navigate to the closest grid intersection before the ring set.
9. The robot should localize again.
10. The robot should then move forward slowly while the color sensor sweeps at 45-degree angles to look for a ring.
11. If there is a ring, the robot will beep according to the color of the ring and the ring sweepers with swipe the ring into the ring collector.
12. If there is no ring the robot should go back to the grid intersection, go to the next side of the tree and repeat steps 8 to 11 until a ring has been found.
13. Once a ring has been found, the robot should back off and beep 5 times to signal that the demo is over. Record whether ring was collected.
14. Repeat this procedure with different coordinates of the ring set and tunnel.

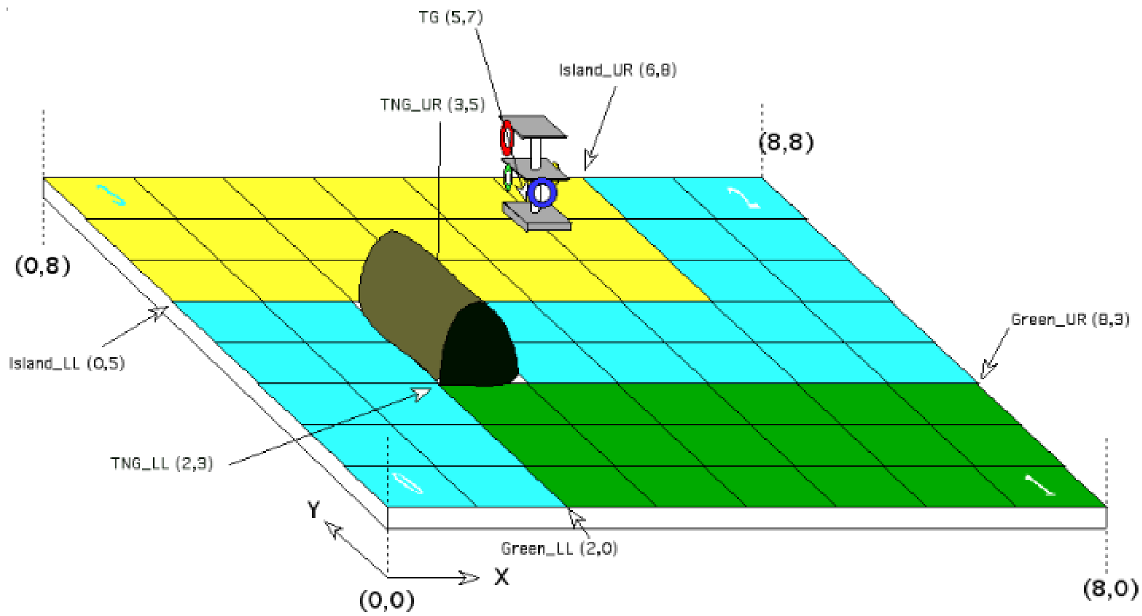

Figure 23: Beta demo map

Expected result: The robot should successfully collect the ring.

Test Report: The results of the beta demo are shown in the table in the next page.

Conclusion: The test conclude that the robot is working as intended and should successfully pass the beta demo

Action: No further testing required for beta demo. Update the Gantt Chart.

Distribution: Software Development, Project management

Trial	Localization 1	angle error (deg)	X error (cm)	Y error (cm)	Eu error (cm)	Time (s)	Navigation 1	angle error (deg)	X error (cm)	Y error (cm)	Tunnel pass	Navigation 2	angle error (deg)	X error (cm)	Y error (cm)	Ring collected
1		1.0	1.10	0.60	1.25	25.00		1.0	1.20	1.20	Yes		3.0	1.10	0.60	Yes
2		2.0	0.30	0.30	0.42	27.00		3.0	0.30	0.60	Yes		1.0	0.30	0.80	Yes
3		1.0	0.80	0.70	1.06	28.00		3.0	0.50	0.70	Yes		3.0	0.50	0.70	Yes
4		1.0	0.20	0.30	0.36	29.00		3.0	0.70	0.60	Yes		2.0	0.60	1.10	Yes
5		1.0	0.60	0.70	0.92	29.00		2.0	1.10	1.10	Yes		1.0	0.30	0.50	Yes
6		3.0	0.80	0.60	1.00	28.00		2.0	0.30	0.60	Yes		1.0	0.70	0.20	Yes
7		1.0	1.20	0.50	1.30	27.00		1.0	0.50	1.50	Yes		1.0	0.50	1.70	Yes
8		2.0	1.30	1.10	1.70	29.00		1.0	0.60	0.70	Yes		2.0	0.90	3.40	No
9		1.0	0.40	1.80	1.84	30.00		1.0	1.10	0.90	Yes		2.0	1.30	0.90	Yes
10		1.0	0.10	0.30	0.32	28.00		1.0	0.70	0.30	Yes		2.0	1.10	1.00	Yes
Mean		1.4	0.68	0.69	1.02	28.00		1.8	0.70	0.82			1.8	0.73	1.09	
STD.dev		0.7	0.41	0.44	0.51	1.34		0.9	0.31	0.34			0.7	0.33	0.86	

Table 20: Results from the beta demo test

Test 20: Final Integration Test

Date: 27/11/2018
Tester: Ashish
Author: Ashish
Hardware Version: 4.4.3 from the hardware documentation
Software Version: Final Demo

Goal: Determine if the robot can successfully localize, navigate to the tunnel, navigate to the ring, collect a ring, navigate back to the tunnel and finally navigate back to the starting point in an 8x8/15x9 board.

Procedure:

1. First change the server IP on the BetaDemo.java to your computers IP address.
2. Run DPMServer.jar and insert the coordinates for starting corner, Green_UR, Green_LL, TNG_LL, TNG_UR, Island_LL, Island_UR and TG. Make sure the rest of the fields are left blank. Also insert the team number that is on BetaDemo.java.
3. Place the robot at corner 1 and run BetaDemo.java.
4. Once the robot has connected, (you should see Team 20 connected on DPMServer) click start.
5. The robot should start localizing. Upon finishing localization, the robot should beep 3 times. Measure the angle error, X error and Y error.
6. The robot should then navigate to the tunnel. Measure the X error, Y error and the angle error.
7. The robot should then pass through the tunnel and localize to the nearest grid intersection. Record whether the robot successfully passed the tunnel
8. After completing localization, the robot will navigate to the closest grid intersection before the ring set.
9. The robot should localize again.
10. The robot should then move forward slowly while the color sensor sweeps at 45-degree angles to look for a ring.
11. If there is a ring, the robot will beep according to the color of the ring and the ring sweepers with swipe the ring into the ring collector.
12. The robot will then go back to the grid intersection, go to the next side of the tree and repeat steps 8 to 11 until all sides that can be reached have been visited.
13. The robot then returns to the tunnel and crosses it.
14. After passing the tunnel, the robot will navigate directly to the starting corner, where it will unload the rings using the sweeper.
15. Repeat this procedure with different coordinates of the ring set, tunnel and starting corner.

Expected result: The robot should successfully navigate to the tunnel, navigate through the tunnel, navigate to the tree, get all the reachable rings and return back to the starting corner, through the tunnel in less than 5 minutes.

Test Report: The results of the final demo are shown in the table on the next page. The first 10 tries were done on the 8x8 board because the 15x9 board was no available till 14 hours before the final presentation. Trials 11 to 20 were done on the 15x9 board. The lighting conditions were not the same as in the lab rooms. Rings were not always detected because either the robot was a little too close to the tree or a little too far. Navigation and tunnel traversal had no issues. The robot also failed to the orange ring only when it was on the top part of the tree. When navigating through the tunnel, the robot went up the tunnel due to the small errors in navigation. It passed the tunnel and continued to collect rings but, in the table, below, we marked it as a failure. The color detection of the rings was off for some of the trials because navigation was off thereby not allowing the color sensor to detect the ring at times.

Important statistics based on the last 10 trials:
Localization: 80% success rate
Navigation: 80% success rate (based on tunnel traversal)
Ring Grabbing: 90% success rate
Color Detection: 80% success rate

Conclusion: Localization was much better in the first 10 trials because the board in the 8x8 board was favorable and we had been testing on it since the beginning of the project. On the other 10 trials (15x9), localization did fail sometimes. This could be due to the lighting conditions or it could have been the board itself was not ideal. Ring detection was affected by the battery. At 8 volts the rings were detect at a much higher rate than when it was not at 8 volts. This means for the final demo to work properly we need the battery at 8V. This was because navigation differed when the battery voltage was less than 8V.

Action: No further testing required for beta demo. Update the Gantt Chart.

Distribution: Software Development, Project management.

Trial	Team	Starting corner	Localization	angle error	EU error	Navigation	Angle error	EU error	Tunnel passed	Navigation 2	Angle error	EU error	Rings collected	Navigation 3	Eu error	Angle error	Tunnel passed	Navigation 4	Eu error	Angle error	Reach starting corner
1	Red	0		2.0	0.4		1.0	0.9	Yes		3.0	1.0	3		1.2	3.0	Yes		2.9	1.0	Yes
2	Red	1		1.0	1.1		3.0	1.0	Yes		1.0	1.2	2		1.5	1.0	Yes		1.0	3.0	Yes
3	Red	2		1.0	0.4		1.0	1.2	Yes		2.0	1.5	3		3.0	3.0	Yes		1.0	1.0	Yes
4	Red	3		1.0	0.9		2.0	1.5	Yes		1.0	3.0	4		2.8	1.0	Yes		1.2	2.0	Yes
5	Red	2		3.0	1.0		1.0	3.0	Yes		3.0	2.8	4		3.0	2.0	Yes		1.5	3.0	Yes
6	Green	0		1.0	0.4		3.0	2.8	Yes		1.0	3.0	4		1.9	3.0	Yes		3.0	1.0	Yes
7	Green	1		3.0	1.1		1.0	3.0	Yes		2.0	1.9	4		2.9	1.0	Yes		2.8	1.0	Yes
8	Green	2		1.0	0.4		2.0	1.9	Yes		3.0	2.9	2		1.0	2.0	Yes		3.0	3.0	Yes
9	Green	3		2.0	0.9		3.0	2.9	Yes		1.0	1.0	3		1.0	1.0	Yes		1.9	1.0	Yes
10	Green	0		1.0	1.0		1.0	1.0	Yes		2.0	1.2	2		1.2	2.0	Yes		2.9	2.0	Yes
11	Red	0		1.0	1.3		2.0	1.2	Yes		1.0	1.5	3		1.5	1.0	Yes		1.0	3.0	Yes
12	Red	1		2.0	1.7		1.0	2.1	Yes		2.0	1.6	2		3.0	3.0	Yes		1.2	1.0	Yes
13	Red	2	Failed at localization																		
14	Red	3		1.0	1.7		2.0	1.0	No		1.0	1.2	2		1.0	1.2	Yes		2.8	3.0	Yes
15	Red	2		2	2		3.0	2.2	Yes		1.0	0.4	2		2.0	0.4	Yes		3.0	1.0	Yes
16	Green	0		1.0	1.0		1.0	1.0	No		2.0	1.1	2		3.0	1.1	Yes		1.9	2.0	Yes
17	Green	1		2.0	0.4		2.0	2.5	Yes		1.0	1.7	3		1.0	1.7	Yes		1.9	1.0	Yes
18	Green	2	Failed at localization																		
19	Green	3		1.0	0.4		1.0	1.2	Yes		1.0	0.9	3		1.0	1.0	Yes		3.0	1.0	Yes
20	Green	0		3.0	0.9		2.0	3.2	Yes		3.0	0.8	4		2.0	1.2	Yes		1.9	3.0	Yes
Mean				1.6	0.9		1.8	1.9			1.7	1.6	2.9		1.9	1.6			2.1	1.8	
Std Dev				0.5	0.2		0.5	1.1			0.0	0.1	0.5		0.4	0.9			0.5	1.0	

Table 21: Results from the final integration test