TEAM 11

MCGILL ENGINEERING DESIGN

ECSE 211

ECSE 211 DESIGN PROJECT

TESTING DOCUMENT

Version *1.08*

04/05/2018

ECSE 211 TEAM 11

VERSION HISTORY

Title	Testing Document			
Description	Test the built system to make sure it meets all the requirements			
Created By	Tianyi Zou, Testing leader			
Date Created	1st March 2018			
Version Number	**Modified By**	**Modifications Made**	**Date Modified**	**Status**
1.00	Tianyi Zou	Created the Testing Document Template	1st March	Preliminary version of the document; added testing template in the appendix
1.01	Luka Jurisic	Peer reviewed the document. Fixed some small errors and formatted the document. Added the introduction, 2 appendixes, and the test plan document. Created section 1.1-1.3.2 and section 2	2nd March	Preliminary template complete
1.02	Tianyi Zou, Enan Zaman	Completed section 4.2 and 4.4; Light Sensor and Wheels preference tests	13th March	All other tests remain
1.03	Tianyi Zou	Completed section 4.1	20th March	All other tests remain
1.04	Volen	Completed 5.2 and 5.3	22nd March	All other tests remain
1.05	Luka Jurisic	Following criticisms in the 3rd weekly meeting, changes regarding test procedures were updated and corrected. Also Added testing process. Table of contents presentation was improved as well as the Testing Template	26th March	Outstanding tests remain
1.06	Luka Jurisic, Tianyi Zou	Continued to improve the document through procedural correction. Tianyi added section 4.3	29th March	Software testing has been corrected fully. However, many tests are still required. In hardware, the motor test section has not

				been completed, but this might be removed following a team discussion.
1.07	Enan Ashaduzzaman	Enan-Added subparts to section 5	4th April	Localization still needs to be tested fully. Software lag is limiting this test. Following that, capture testing needs to tested. Otherwise, testing is on schedule for the final demo.
1.08	Tianyi Zou	Added the localization part. Edited the procedures of obstacle traversal. Edited the data collected and tolerance of data for the ultrasonic sensor and light sensor testing. Added obstacle traversal Began Edit History Section Recompiled the wheel test data and improved the presentation of that data.	5th April	Localization data needs to be collected. Capture testing remained.

TABLE OF CONTENTS

1 Introduction

1.1 Purpose of The Test Plan Document

The Test Plan document documents and tracks the necessary information required to effectively define the approach to be used in the testing of the project's product. The Test Plan document is created during the Planning Phase of the project. Its intended audience is the project manager, project team, and testing team. Some portions of this document may on occasion be shared with the client/user.

1.2 TESTING TOOLS

The following tools will be used for testing:

PROCESS	TOOLS
Test Case Creation	Microsoft Word
Test Case Tracking	Microsoft Excel
Test Case Execution	Manual
Test Case Management	Microsoft Excel
Defect Management	Microsoft Excel

1.3 Quality objective

1.3.1 Primary Objective

The primary objective of this testing phase is to assure that the system meets the full requirements, including quality requirements, and maintain the metrics for each quality re-equipment of the final design. At the end of the project development, the user should find that the project has met or exceeded all their specifications detailed in the requirements.

1.3.2 secondary Objective

The secondary objective of this testing phase is to identify issues and propose solutions to all hardware and/or software issues, and to communicate all this to the project team. This requires careful and methodical testing of the design to ensure all areas of the system are scrutinized appropriately.

1.4 TESTING PROCESS

1.4.1 Hardware Components

The hardware components utilized in the final design were separately tested to clearly identify weaknesses and allow a swift evaluation of performance. For example, at our disposal were 6 ultrasonic sensors, and tests were run solely on these sensors to determine which one was most accurate. This procedure was also performed for the light sensors and the motors. Motor testing was especially critical as there were two types of motors available, and a decision had to be made as to which type would be implemented into the final design.

Similarly, precise values were obtained of both the wheel radius and wheelbase length, which are key for the software department to have and utilize.

1.4.2 Software Components

The testing of software components follows on from its respective hardware testing, as an initial or even progressive design is required before any meaningful software testing phase can commence. Once again, key classes that governed the robot's main behaviors were tested in isolation to fully assess their performance and flaws.

Regarding software, the primary objective is to realize any flaws within the architecture. This requires that the robot be tested in a varying number of unfamiliar and unorthodox situations to ensure full functionality of the code. For example, localization has to be tested many times because the starting orientation of the robot is unknown, and thus as many cases as possible must be accounted for during this testing phase.

1.4.3 Integration testing

Software and hardware testing ensures that each respective individual component performs accordingly. However, when all the components are working in tandem, this gives rise to problems previously unforeseen. The accumulation of small errors within each subsystem could result in a poor working final design. Thus, integration testing is the key final phase of testing that allows us to ensure that all the client's behavioral specifications are met.

2 *TEST DELIVARABLES*

The testing phase will allow a general progression of the project in terms of both hardware and software. The testing phase will provide key deliverables that fall into 3 basic categories: Documents, Test Cases and Reports. The figure below illustrates the dependencies of these 3 categories.

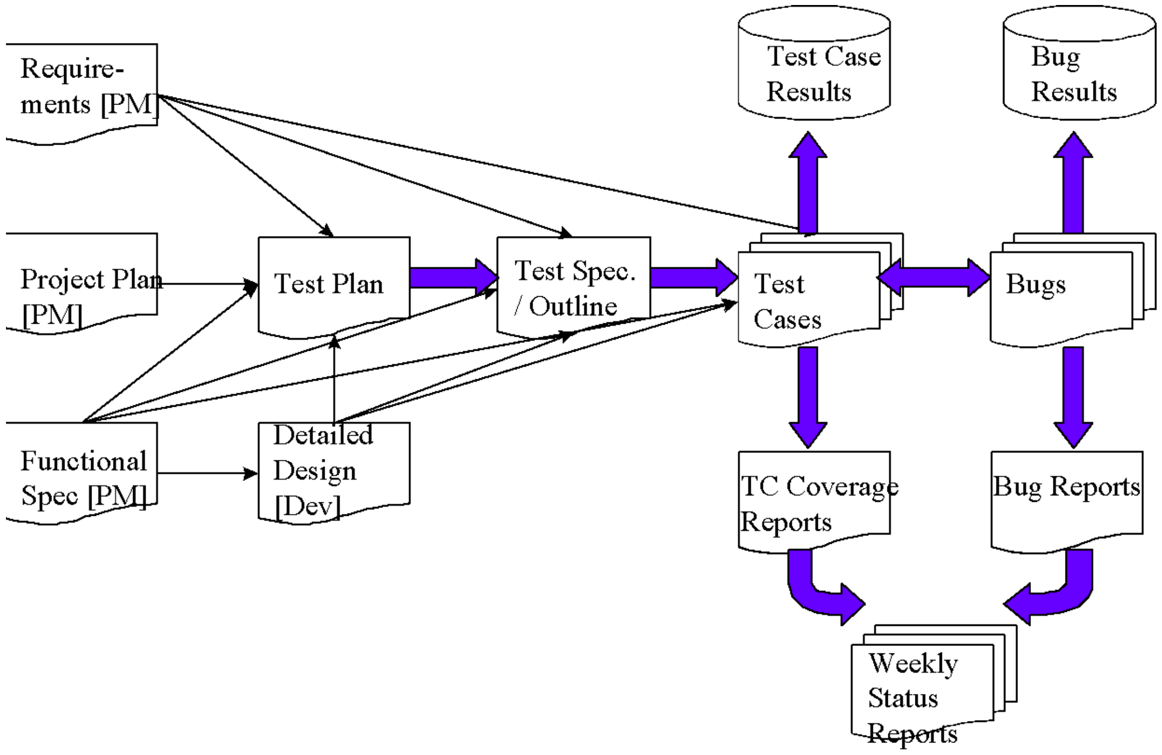


Diagram Source: https://strongqa.com/qa-portal/testing-docs-templates/test-report

3 TEST REPORT TEMPLATE

Test's Title: *Name of Test*

Tester: *Concerned Team Members* **Test Date:** *Month/Day/Year*
Software Version: *Software version of Robot*
Hardware Version: *Hardware version of Robot*

Objective: *What is the purpose of the test?*

Background knowledge (if needed): *Relevant information disclosed here*

Procedure: *Detailed description of the steps performed to carry out the test*

Expected Results: *What is the predicted outcome of the test?*

Test Report: *Provide the relevant data acquired during the test, along with an analysis of the results. The number of times the test was performed must be included. Tables and/or graphs are key in providing visual information. See appendix for table templates.*

Conclusion: *Specify the outcome of the test. Were the requirements met and did they meet the expected outcome?*
Action: *Based on the results, provide a list of what needs to done going forward.*
Distribution: *Indicate which sections of the team need to be made aware of the results of the test to allow for necessary modifications.*

4 HARDWARE TESTING

4.1 Ultrasonic Sensor

Ultrasonic Sensor Consistency Test

Tester: *Tianyi Zou* **Test Date:** *03/15/18*
Software Version: *N/A**
Hardware Version: *N/A**
**No hardware and software version due to the use of the built-in sensor testing application located in the tools option in the EV3 selection menu*

Objective:

Determine the accuracy of ultrasonic sensors available. This test will serve to identify which of the 3 ultrasonic sensors provided to our team have the least incongruities in their distance measurements. Thus, the most accurate and/or consistent ultrasonic sensors will be determined and implemented onto our final design.

Procedure:

1. Turn on the EV3 brick. Connect the Port 1 of the brick to the light sensor via a cable.
2. Assemble the ultrasonic sensor vertically in front of the robot. Make sure that the sensor should direct to the front of robot.
3. Use the tools application on the EV3 brick. Select **Tools>Test Sensors >Go> Port 1>EV3 Ultrasonic >Distance**.
4. Record the real distance from each tile and the measured value on the screen of brick. Please refer to the Figure 3.1.1.

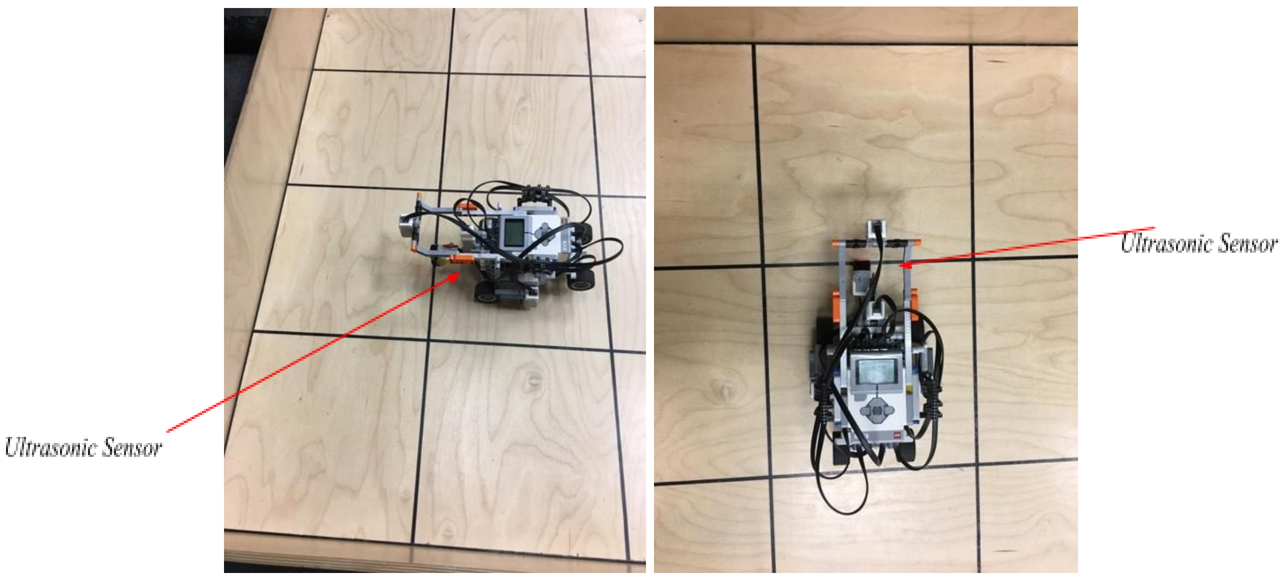


Figure 3.1.1 Placement of ultrasonic sensor during the test: 2 different views

5. Calculate the average value and standard deviation of error that between real distance and measured distance.

6. Test other two ultrasonic sensors by using the same procedure. Compare the result from all different ultrasonic sensors.

Expected result:

We expect all the sensors would provide small inconsistencies when measuring distances and different ultrasonic sensor would perform differently. Some sensors would perform better than others, which give us a sense of which sensor to choose for object avoidance and localization. Note that 1.5 cm of error can be tolerated and the standard deviation of error should be as low as possible.

Test Report:

	US Sensor 1		US Sensor 2		US Sensor 3	
Real distance (cm)	**Measured distance (cm)**	**Error (cm)**	**Measured distance (cm)**	**Error (cm)**	**Measured distance (cm)**	**Error (cm)**
29.8	30.4	0.6	30.4	0.6	30.2	0.4
60.1	62.2	2.1	67.0	6.9	64.3	4.2
90.4	91.6	1.2	91.7	1.3	91.7	1.3

120.7	121.8	1.1	122.3	1.6	121.7	1.0
151.3	152.9	1.6	152.9	1.6	152.1	0.8
181.6	182.9	1.3	182.9	1.3	182.4	0.8
211.9	213.6	1.7	213.0	1.1	213.2	1.3

	Sensor 1	**Sensor 2**	**Sensor 3**
Average Error (cm)	1.37	2.06	1.40
Standard Deviation (cm)	0.48	2.16	1.27

Conclusion:

As mentioned before, only an error of 1.5 cm can be tolerated. Sensor 2 has an average error of 2.06 cm so that have to be excluded. Between sensor 1 and sensor 3, we pick sensor 1 because it has a significantly lower standard deviation than sensor 3 has. This means sensor 1 provides more accurate values in distances.

Action:

We use ultrasonic sensor 1 as the priority choice. If more than one ultrasonic sensor needs to be used, sensor 3 would be the second-best choice.

Distribution: Hardware team

4.2 Light Sensor

Light Sensor Consistency Test

Tester's names: *Enan Ashaduzzaman, Tianyi Zou* **Test Date:** *03/12/18*
Software Version: N/A
Hardware Version: N/A

Objective:

Determine the best performing light sensors available. This test will serve to identify which of the 3 light sensors provided has the clearest detection of an object from different heights.

Procedure:

1. Put a blue paper on the table. Use a ruler to measure the distance by putting a block adjacent to the ruler so that the ruler can be stabilized and placed perpendicular to the table surface.
2. Connect the Port 1 on the brick to the sensor via a cable.
3. Use the tools application on the EV3 brick. Select Tools>Test Sensors >Go> Port 1>EV3 Color >Color ID.
4. Place the light sensor next to the ruler and on the table surface. Make sure the direction of light should be to the blue paper on the table.
5. Move up the light sensor along the ruler slowly.
6. Record the distance *d1* between the light screen of light sensor and the table surface when the value of color ID shown on the screen of EV3 brick becomes 2.0.
7. Repeat step 5 and record the distance *d2* when the value of color ID becomes 7.0.
8. Repeat step 5 and record the distance *d3* when the value of color ID becomes -1.0.
9. Do the same procedure to test other two light sensors.

Expected result:

The value of *d1* and *d2* are respectively the closest and farthest distance that the light sensor is able to precisely detect an object, which means light sensor can both detect the object in front of it and identify the color of the object. Value of *d3* is the farthest distance that the light sensor can detect an object, but not able to identify the color.

Note that only the light sensor with the largest difference between d1 and d2 would be picked because it can detect objects from a larger range.

Test reports:

Sensor #	*d1(cm)*	*d2(cm)*	*d3(cm)*

Light Sensor 1	0.3	1.8	4.5
Light Sensor 2	0.6	1.0	3.5
Light Sensor 3	0.5	1.5	3.7

Conclusion:
Comparing the average value differences of all 3 sensors for any given height, it can easily be seen that the differences between the sensors are minimal and cannot be detected with tests of this quality. However, there is no need for more precise testing, as this test shows that all 3 color sensors would get the job done equally well as they can each detect the black lines with at least one data point for each black line.

Action:
We must place the sensor at distance between 0.6 cm and 1.0 cm from the block in order to detect the block and identify its color.

Distribution:
Hardware team

4.3 TRACK

Exact Wheelbase Length (Track Value) Test #1

Tester: *Tianyi Zou* **Test Date:** *03/20/18*
Software Version: *01.00.08 to 1.01.00*
Hardware Version: *1.0*

Objective:

The objective of this test is to achieve a very precise value of the robot's wheelbase length. This track value is essential in ensure proper navigation and thus a very accurate value is pivotal.

Background Info: By initially physically measuring the distance between the two front wheels, the starting test value is found to be 12.70cm.

Procedure:

1. Charge up the EV3 brick and make sure the robot works at a battery level of 8.0V.
2. Use the test track code that enables the robot to rotate about itself by 360 °. Set the rotating speed at 150 to avoid slippage of wheels.
3. Put the robot at a cross line on the testing board. Make sure that the axis of wheels and the middle line of robot are respectively right above the two lines of the cross.
4. Run the code. It is expected that the robot will not rotate exactly by 360 °. There is an angle between the black line and the middle line of robot. Measure the angle by using a protractor.
5. If the robot turns more than 360 °, lower the value of track in the code and repeat step 2 and step3. If the robot turns less than 360 °, higher the value of track in the code and repeat step 2 and step 3.
6. Continue the process until a minimum amount of error is found.

Expected Result: The Research and Development phase of this project clearly indicated that the precise fine-tuning of an accurate track value is difficult, and it known to be off its true value. We expect that the robot will not turn exactly by 360 °. There would be an error of angle. If the track value we set in the code is greater than the real track, the robot will turn more. Conversely, if the track value is less than the real track, the robot will turn less. Past testing within this phase indicates that the angle errors of the track value is expected to be within the range of 0°± 3°. This error is primarily due to human error and the inexactitude of measurements.

Test Report:

	Value/cm	n Angle/°

Conclusion: Through trial and error, it has been determined that the most precise measurement of our track is 12.62cm. This value of track produces the least error when measuring the angle.

Action: Change the track to the experimentally determined valued of 12.62cm.

Distribution: Software Team

Exact Wheelbase Length (Track Value) Test #2

Tester: *Bryan Jay, Volen Mihaylov, Tianyi Zou* **Test Date:** *04/07/18*
Software Version: *06.01*
Hardware Version: *2.2*

Objective:
The objective of this test is to achieve a very precise value of the robot's wheelbase length. This track value is essential in ensure proper navigation and thus a very accurate value is pivotal. This test was redone due to the change in motors and brick.

Background Info: A new track test must be performed. This is because the motors on the current hardware design have stopped working, and thus the robot had to deconstructed and rebuilt. This has affected the previously verified track value of 12.62cm. A new track needs to be determined. That is the purpose of the 2nd iteration of the test.

Procedure:

Please refer to the procedure outlined in *Track Value Test* #1 (page __). They are identical.

Expected Result: The Research and Development phase of this project clearly indicated that the precise fine-tuning of an accurate track value is difficult, and it known to be off its true value. We expect that the robot will not turn exactly by 360 °. There would be an error of angle. If the track value we set in the code is greater than the real track, the robot will turn more. Conversely, if the track value is less than the real track, the robot will turn less. Past testing within this phase indicates that the angle errors of the track value is expected to be within the range of 0°± 3°. This error is primarily due to human error and the inexactitude of measurements.

Test Report:

	Value/cm	n Angle/°

gh tests, it was examined that the robot was consistently off by about 11 degrees. Taking this into account, the following formula was created: TRACK = $\frac{180-11.1}{180} \times 12.62$. "11.1" is the average of the "Error in Angle" for tests 1.1-1.5.		

Conclusion: Through trial and error, it has been determined that the most precise measurement of our track is 13.45cm. This value of track produces the least error when measuring the angle.

Action: Change the track to the experimentally determined valued of 13.45cm.

Distribution: Software Team

4.4 Wheels

Wheel type: Treads vs Regular Wheels

Tester: *Enan Ashaduzzaman* **Test Date:** *03/12/18*

Software Version: *N/A*

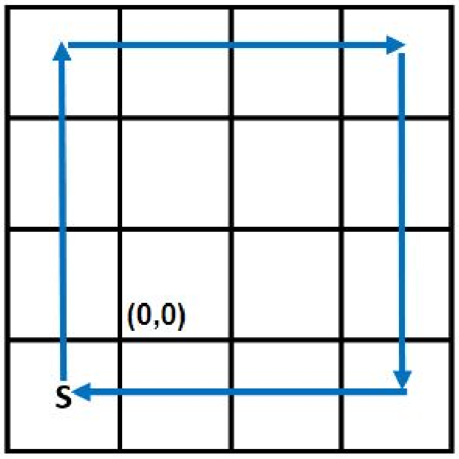
Hardware Version: *N/A*

Objective:

The objective of this test is to determine whether to use the treads or the regular wheels. Treads can be very useful at overcoming the bumps and avoiding a variable track. Fears include treads not being accurate during navigation. This test serves to provide the necessary *quantitative* data so that the best form of transportation can be implemented on the final robot.

Background knowledge:

The figure below illustrates the meaning of the square navigation that will be referenced in the procedure.



3-BY-3 Tile Trajectory using *SqaureDriver* class given in R&D phase

Procedure:

1. Build a very simple robot that implements treads as its type of wheel.
2. Make the robot complete the square navigation to see the accuracy using the square driver code that was given/developed during the Research and Development phase.
3. To test the odometer's accuracy, measure the robot's end position and note the position reported by the odometer 10 independent times. Note that in this test, measurements were taken from the center of the robot's brick, and not the center of the light sensor. This was done because our sensor was positioned very slightly to the left.
4. The error of the result has to calculated in order to allow easy comparison between the two methods. The error was calculated by using the formula below, where x_f and y_f are the co-ordinates of the final position of the robot and x and y are the actual co-ordinates displayed by the odometer.

$$\varepsilon = \sqrt{(x_f - x)^2 + (y_f - y)^2}$$

5. The standard deviation must be calculated to serve as an effective comparison as to the consistency in the readings between the two methods. The expression for standard deviation used is as follows:

$$\sigma = \sqrt{\frac{\sum(x - \bar{x})^2}{N}}$$

Where σ is the standard deviation, x an element of the data set, $\bar{x}$ is the average value of the data set and N the number of elements in the data set.

6. Record these calculation for each of the 10 runs and present them in a tabulated format.
7. Build a simple robot that implements regular wheels as its wheel type.
8. Completes steps 2-6 using the regular wheels.

Expected Results:

It is expected that the treads will be less accurate than the regular wheels during navigation. These little discrepancies can accumulate at the end of the day.

Test Report:

Table 1: Recorded Values & Errors Using Treads

Treads							
Run #	Odometer X X_0 ± 0.05 cm	Odometer Y Y_0 ± 0.05 cm	Actual X X_A ± 0.05 cm	Actual Y Y_A ± 0.05 cm	Error X E_X ± 0.05 cm	Error Y E_Y ± 0.05 cm	Euclidean Error
1	0.05	0.30	1.90	-0.90	1.85	1.20	2.2051
2	0.04	-0.15	0.95	0.80	0.91	0.95	1.3134
3	-0.35	-0.19	0.50	1.20	0.85	1.39	1.6293
4	0.01	-0.39	-1.60	-0.60	1.61	0.21	1.6266
5	0.13	-0.28	1.30	1.60	1.17	1.88	2.2133
6	0.32	-0.88	2.90	-0.50	2.58	0.38	2.6078
7	1.06	-0.71	-1.60	-2.30	2.66	1.59	3.0990
8	0.59	0.45	-1.40	-1.60	1.99	2.05	2.8570
9	0.31	0.52	-0.20	-2.30	0.51	2.82	2.8657
10	1.02	-1.11	-0.70	-0.70	1.72	0.41	1.7682
			Mean (cm)		1.5848	1.288	2.218553
			Standard Deviation (cm)		0.724824239	0.833943776	0.621206

Table 2: Recorded Values & Errors using Regular Wheels

Regular Wheels							
Run #	Odometer X X_0 ± 0.05 cm	Odometer Y Y_0 ± 0.05 cm	Actual X X_A ± 0.05 cm	Actual Y Y_A ± 0.05 cm	Error X E_X ± 0.05 cm	Error Y E_Y ± 0.05 cm	Euclidean Error
1	-15.09	-13.02	-15.30	-12.20	0.21	-0.82	0.8465
2	-13.42	-14.10	-13.20	-11.33	-0.22	-2.77	2.7787
3	-13.02	-15.31	-13.00	-13.98	-0.02	-1.33	1.3302
4	-13.05	-16.06	-13.55	-14.35	0.5	-1.71	1.7816
5	-11.39	-15.05	-11.00	-16.30	-0.39	1.25	1.3094
6	-12.46	-12.03	-13.33	-10.50	0.87	-1.53	1.7601
7	-16.11	-15.04	-15.50	-16.50	-0.61	1.46	1.5823
8	-11.86	-13.01	-11.40	-14.22	-0.46	1.21	1.2945
9	-10.91	-12.54	-12.20	-12.10	1.29	-0.44	1.3630
10	-14.59	-13.02	-14.50	-12.30	-0.09	-0.72	0.7256
			Mean (cm)		0.108	-0.54	1.47718
			Standard Deviation (cm)		0.61383675	1.427250659	0.571218

Conclusion:

Clearly, the mean of the error in the x, y and ε substantially decreases when regular wheels are implemented. Looking at table 1, mean errors for x, y and ε across 10 independent trials were 1.58 cm, 1.28 cm and 2.21 cm respectively when using the treads. This is compared to the regular wheels, where the mean errors were 0.108cm, -0.54 cm, and 1.478 cm for x, y and ε respectively. These results clearly indicate that regular wheel implementation led to much more accurate results.

However, it must be noted that when looking at the standard deviation between the two tables, our Y values, although more accurate overall, proved to be much less consistent with regular wheels. The inconsistency in the Y-values could certainly be improved upon in within the navigation code in order to for us to be successful in the final demo.

Action:
Implement regular wheels in the final design of our robot. Notify software team that navigation needs to be improved.

Distribution:
Hardware/ Software Team

5 SOFTWARE TESTING

5.1 localization

5.1.1 Ultrasonic localization test

<u>Test's Title: Ultrasonic Localization Test</u>

Tester's names: *Tianyi Zou and Volen* **Test Date:** *03/26/18*

Software Version: *4.01(tentative)*

Hardware Version: *2.0*

Objective: The objective of this test is to ensure that the robot can successfully localize using ultrasonic localization. The client's requirement is initial localization must be performed under 30 seconds.

Background knowledge: None

Procedure:

1. Place the robot at the corner tile (surrounded by two walls) along the 45 ° line (bottom left-top right diagonal of the tile).
2. Run a test code that enables the robot rotate about itself by 360 °. Make sure that the wall does not touch any piece of robot when the robot is rotating.

Note: The robot is relatively long, especially with two wheels at the back. It is highly necessary to check if the robot would touch the wall.

3. After checking the rotation of robot, set the orientation of robot along the vertical gridline.
4. Run the ultrasonic localization program.
5. Record the final orientation of robot when the ultrasonic localization is done.
6. The orientation of robot is expected to direct to the y-axis. (along with the vertical line) Measure the angle between the final orientation and the y-axis direction by using a protractor.
7. Repeat step 1-step 6 with different initial orientations (20° difference between each run)

Note: These steps should be repeated if there is any change on the localization code. If done so, each version of localization code should be recorded and considered as separate tests.

8. The raw data must be analyzed accordingly. The error in angle is calculated simply as

$$Angle\ Error = Actual - Expected$$

Expected Results:

The robot would orient to the y-axis direction after doing the ultrasonic orientation with minimal error. The tolerance on the error is ±3 degrees.

Results obtained:

Table 1: Ultrasonic Localization Results

	Falling Edge Ultrasonic Localization			Rising Edge Ultrasonic Localization		
Run #	**Expected angle (°) ± 0.05°**	**Actual Angle (°) ± 0.05 °**	**Error in Angle (°) ± 0.05 °**	**Expected angle (°) ± 0.05 °**	**Actual Angle(°) ± 0.05 °**	**Error in Angle (°) ± 0.05°**
1	0.0	9.34	9.34	0.0	12.04	12.04
2	0.0	10.14	10.14	0.0	12.56	12.56
3	0.0	3.55	3.55	0.0	3.04	3.04
4	0.0	4.54	4.54	0.0	5.08	5.08
5	0.0	10.53	10.53	0.0	2.52	2.52
6	0.0	3.12	3.12	0.0	7.59	7.59
7	0.0	5.03	5.03	0.0	3.03	3.03
8	0.0	2.57	2.57	0.0	2.05	2.05
9	0.0	7.78	7.78	0.0	4.57	4.57
10	0.0	8.43	8.43	0.0	4.35	4.35

Conclusion:

Out of 10 test runs, localization was below the tolerance of ±3 degrees on only 1 occasion, and on it was relatively accurate (±5 degrees) on 4 occasions. The other 5 occasions were deemed too inaccurate.

Action:

Considering that the Beta demo is 2 days away, it will be difficult to improve the localization method to make it more reliable. It is relatively accurate 5/10 times, thus this version of the software will be used for the beta demo, and following that, the software team will strive to improve the localization in preparation for the final demo.

Distribution:

Software Team

5.1.2 Light localization test

Test's Title: Light Localization Test

Tester's names: *Tianyi Zou & Volen* **Test Date:** *03/26/18*

Software Version: *5.01*

Hardware Version: *2.00*

Objective: The objective of this test is to ensure that the robot can successfully localize using light localization. The client's requirement is initial localization must be performed under 30 seconds.

Procedure:

1. Place the robot on one tile. Set the orientation of the robot along the y-axis.
2. Make sure that the first line that each light sensor detects is a horizontal line.
3. Run the localizeY() method.
4. Record the final orientation of the robot. Measure the angle between the final orientation and the y-axis direction by using a protractor.
5. Repeat step 1-step 4 with different initial orientations (5° difference between each run).
6. Place the robot on one tile. Set the orientation of the robot along the x-axis. Set the value of odometer to be (0,0,90).
7. Make sure that the first line that each light sensor detects is a vertical line.
8. Run the localizeX() method.
9. Record the final orientation of the robot. Measure the angle between the final orientation and the x-axis direction by using a protractor.
10. Repeat step 6-step 9 with different initial orientations (5° difference between each run).
11. The raw data must be analyzed. The error was calculated by using the formula below, where x_f and y_f are the co-ordinates of the final position of the robot and x and y are the actual co-ordinates displayed by the odometer.

$$\varepsilon = \sqrt{(x_f - x)^2 + (y_f - y)^2}$$

12. Similarly, calculating the standard deviation of the error in x, the error in y and the error e for the trials serves as an effective marker as to the consistency in the readings. The expression for standard deviation used is as follows:

$$\sigma = \sqrt{\frac{\Sigma(x_i - \mu)^2}{n}}$$

Where σ is the standard deviation, x_i an element of the data set, u is the average value of the data set and n the number of elements in the data set.

Expected Results:

After running the localizeY() method, the orientation of robot should be along the y-axis (along the vertical line).

After running the localizeX() method, the orientation of robot should be along the x-axis (along the horizontal line).

The tolerance on the error is ±3 degrees.

Results obtained:

Light Localization							
Run #	Theoretical X X_A ± 0.05 cm	Theoretical Y Y_A ± 0.05 cm	Actual X X_A ± 0.05 cm	Actual Y Y_A ± 0.05 cm	Error X E_X ± 0.05 cm	Error Y E_Y ± 0.05 cm	Euclidean Error ± 0.05 cm
1	0.0	0.0	10.00	9.57	10.00	9.57	1.151043
2	0.0	0.0	5.67	7.23	5.67	7.23	1.994442
3	0.0	0.0	6.76	6.45	6.76	6.45	1.518223
4	0.0	0.0	2.43	3.34	2.43	3.34	1.782723
5	0.0	0.0	5.55	4.78	5.55	4.78	0.954411
6	0.0	0.0	1.24	1.80	1.24	1.80	1.475669
7	0.0	0.0	7.54	8.12	7.54	8.12	1.243382
8	0.0	0.0	3.21	3.70	3.21	3.70	1.712921
9	0.0	0.0	6.47	7.32	6.47	7.32	1.975677
10	0.0	0.0	6.34	7.10	6.34	7.10	1.733667
			Mean (cm)		1.112	0.941	1.554216
			Standard Deviation (cm)		0.516049	0.427693	0.350782

Conclusion: Out of 10 test runs, localization was below the tolerance of ±3 degrees on only 1 occasion for Y-orientation, and twice for X-orientation. It was relatively accurate (±5 degrees) on 3 occasions. The other 6 occasions were deemed too inaccurate. This is almost identical to our ultrasonic localization.

Action: Considering that the Beta demo is 2 days away, it will be difficult to improve the localization method to make it more reliable. It is relatively accurate 4/10 times; thus this version of the software will be used for the beta demo, and following that, the software team will strive to improve the localization in preparation for the final demo.

Distribution: Software team

5.2 Landing Gear Test

Test's Title: Back Wheel Functionality

Tester's names: *Volen Mihaylov* **Test Date:** *03/22/18*

Software Version: *N/A*

Hardware Version: *2.00*

Objective:

The objective of this test is to determine the optimal angle at which to turn the variable back wheels of the robot. This test will serve to highlight the angles at which the back wheels rotate from a stored to lowered position and vice versa that cause no overturning of the motor. Please refer to the procedure for a definition of overturning.

Background knowledge: None

Procedure:

1. Place the robot parallel to any one black line on the grid.
2. Ensure the back wheels are lowered.
3.Run the relevant version of the software code.
4. Observe the robot's performance, keeping note of how the back wheels overturn. Overturning in this case refers to the back wheels coming too much into contact with other hardware

components, producing a “clicking” sound.

Expected Results: The robot is not expected to overturn/under turn at all during this test run. The back wheels should lock smoothly back into place.

Results obtained:
Note: A positive angle refers to the back wheels moving from an up position(stored) to a lowered position. A negative angle refers to the back wheels moving from a lowered position to a stored position.

Tests	Variables				Passed?	Comments:
	Angle down	Angle Up				
1	270	-270			Yes	Overturning, clicking sound can be heard
2	270	-270			Yes	Overturning, clicking sound can be heard
3	270	-270			Yes	Overturning, clicking sound can be heard
4	250	-250			Yes	No overturning sound, wheels well placed
5	250	-250			Yes	No overturning sound, wheels well placed
6	250	-250			Yes	No overturning sound, wheels well placed

Conclusion: Utilizing a 250-degree angle within the software proves to be superior to an angle of 270 degrees. This is highlighted in the above test results since at 270 degrees the robot’s back wheels overturned during each test run. However, the issue could also lie within the hardware design of the robot. A

Action: As of now, implement 250-degree angle within the software. Following this test, a discussion needs to take place between the software and hardware teams to discuss the possibility of a hardware improvement.

Distribution: Hardware/Software Team

Test's Title: Back Wheel Functionality -with stopper

Tester's names: *Volen Mihaylov* **Test Date:** *03/22/18*

Software Version: *N/A*

Hardware Version: *2.1*

Objective: The objective of this test is to determine the optimal angle at which to turn the variable back wheels of the robot. This test will serve to highlight the angles at which the back wheels rotate from a stored to lowered position and vice versa that cause no slight overturning/under turning of the motor.

Background knowledge: This test is identical to the "Back Wheel Functionality test" except that in this case a **"stopper"** has been implemented within the hardware. The stopper serves to fully ensure that the motors don't overturn. Also, the stopper acts as a support so that the robot can stand up on its back wheels.

Procedure:

1. Place the robot parallel to any one black line on the grid.
2. Ensure the back wheels are lowered.
3.Run the relevant version of the software code.
4. Observe the robot's performance, keeping note of how the back wheels overturn. Overturning in this case refers to the back wheels coming too much into contact with other hardware components, producing a "clicking" sound.

Expected Results: The robot is not expected to overturn/under turn at all during this test run. The back wheels should lock smoothly back into place.

Results obtained:

Note: A positive angle refers to the back wheels moving from an up position(stored) to a lowered position. A negative angle refers to the back wheels moving from a lowered position to a stored position.

Tests	Variables				Passed?	Comments:
	Angle down	Angle Up				
1	200	-200			No	Too much Rotation
2	195	-195			No	Too little rotation, does not go all the way
3	192	-192			No	Too little rotation, almost goes all the way but is missing a couple of degrees
4	190	-190			No	Not enough rotation
5	185	-185			Yes	Looks good
6	180	-180			No	Too little rotation, almost goes all the way but is missing a couple of degrees
7	180	-180			No	Too little rotation, almost goes all the way but is missing a couple of degrees
8	180	-180			Yes	Perfect

Conclusion: When implementing a stopper, a wheel rotation of 195 degrees proves to be the most smooth and well-done action. The rotation is qualitatively “perfect”. The test also shows that the addition of a stopper to deal with overturning/underturning and to provide support is proving effective.

Action: Implement 195 degree of the wheel rotation within the software.

Distribution: Software Team

5.3 Navigation

Test's Title: Navigation

Tester: *Volen Mihaylov* **Test Date:** *03/22/18*

Software Version: *2.01*

Hardware Version: *2.1*

Objective: The objective of this test is to test the accuracy of navigation code by checking whether the robot can move to a certain coordinate within an error tolerance of 2 cm using the Euclidean distance measure.

.

Background knowledge: None

Procedure:

1) Place the robot at its original position; that is, (0,0) as defined by your co-ordinate orientation.
2) Make the robot turn at certain wanted degrees. These are: (±45, ±90, ±270). Verify the offset, if there is any.
3) Make the robot move by a set wanted distance and verify the offset (1 tile, 2 tiles, 6 tiles).
4) Finally, make the robot move to a certain co-ordinate and verify the distance covered. In this case, our starting co-ordinate is (0,0) and our end co-ordinate is (5,3).

Expected Results:

After conducting many tests runs to correct for small error offsets in angle and travelling, it expected that the Navigation be fully functioning as well as precise.

Results obtained:

Note: *The test data has been sub-split into 6 test sets. Please refer to the notes in the table breaks.*

Tests	**Variables**				**Passed?**	**Comments:**
	Degrees tested at	Distance tested at	Coordinates tested at	Code Constant changed:		
1	45			0	No	Left turn is off by 2 degrees to the left
1.1	-45			0	No	Right turn is perfect
1.2	90			0	No	Left turn is off by 2 degrees to the left
1.3	-90			0	No	Right turn is perfect
1.4	+270			0	No	Right turn is perfect
1.5	-270			0	No	Left turn is off by 2 degrees to the left

Following these initial 6 tests from test set #1, it is observed that the left turn is off by a couple of degrees. Test set #2 below adds a 4-degree correction to the left turn.

2.0	45			Left: +4	No	Left turn is off by 2 degrees to the right.
2.1	-45			Left: +4	No	Right is perfect
2.2	90			Left: +4	No	Left turn is off by 2 degrees to the right.
2.3	-90			Left: +4	No	Right is perfect
2.4	+270			Left: +4	No	Right is perfect.
2.5	-270			Left: +4	No	Left turn is off by 2 degrees to the right

Following test set #2 it is observed that 4 degrees is an overcorrection. Test set #3 below adds a 2-degree correction to the left turn instead.

3.0	45			Left: +2	No	Left Turn is perfect.
3.1	-45			Left: +2	No	Right Turn is perfect
3.2	90			Left: +2	No	Left Turn is perfect.
3.3	-90			Left: +2	No	Right Turn is perfect
3.4	+270			Left: +2	No	Right Turn is off by 2 degrees not enough.
3.5	-270			Left: +2	No	Left is perfect

Following test set #3, it is observed that both right and left turns are perfect. Test set #4 below now records the accuracy of the odometer.

4.0		1 tile			No	0.5cm too short
4.1		1 tile			No	0.5cm too short
4.2		2 tiles			No	0.9cm too short
4.3		2 tiles			No	0.9cm too short
4.4		6 tiles			No	2cm too short.
4.5		6 tiles			No	2cm too short

Following test set #4, it is observed that the robot does not travel the full desired tile length, falling slight short. Test set #5 below adjusts this error by applying: *Rotation multiplied by: (TILESIZE+0.4)/TILESIZE.*

5.0		1 tile		*(TILESIZE+0.4)/TILESIZE*	Yes	Perfect
5.1		1 tile		*(TILESIZE+0.4)/TILESIZE*	Yes	Perfect
5.2		2 tiles		*(TILESIZE+0.4)/TILESIZE*	Yes	Perfect
5.3		2 tiles		*(TILESIZE+0.4)/TILESIZE*	Yes	Perfect

5.4		6 tiles		*(TILESIZE+0.4)/TILESIZE*	Satisfactory	Perfect
5.5		6 tiles		*(TILESIZE+0.4)/TILESIZE*	Satisfactory	Perfect
Following test set #5, it is observed that the correctional formula applied corrects the offset error. Test set #6 now implements both angle turning as well as travelling. This serves to test the integration and overall accuracy of the navigation. Start corner: (0,0) -> End Corner:(5,3)						
6.0			(5,3)			Off by 3cm
6.1			(5,3)			Perfect
6.2			(5,3)			Perfect

Conclusion: This extensive test procedure has concluded by ensuring that Navigation is fully functional and precise.

Action: Left turning is corrected by applying a +2-degree constant and travelling across a tile is corrected by rotateByDistance constant of (*TILESIZE*+0.4)/*TILESIZE* within the Navigation class.

Distribution: Software Team

5.4 Obstacle Traversal

5.4.1 Applying Different Rear Wheels

Obstacle Traversal: Rear Wheel Types

Tester: *Enan Ashaduzzaman* **Test Date:** *03/21/18*

Software Version: *N/A*

Hardware Version: *1.0*

Objective:

The objective of this test is to determine whether the robot is able to drive through the bridge and tunnel successfully. Not being able to traverse either the bridge or tunnel will not allow the robot to move in-between the red and green zones of the playing field.

Background knowledge:

Procedure:

1. Make sure the Hardware Version 1.0 is not equipped with the landing gear. Instead run this test using a single marble attached to the rear of the robot.
2. Implement a test code that allows the robot to drive straight.
3. Place the robot in front of the bridge.
4. Execute the drive-forward code.
5. Analyze if the robot successfully traverses the bridge.
6. Make sure the Hardware Version 1.0 is not equipped with the landing gear. Instead run this test using two marbles attached to the rear of the robot, one on each corner.
7. Place the robot in front of the tunnel.
8. Execute the drive-forward code.
9. Analyze if the robot successfully traverses the bridge.
10. Use the exact Hardware Version 1.0. Make sure the landing gear is pulled and is lowered when the robot is completing this test.
11. Place the robot in front of the tunnel.
12. Execute the drive-forward code.
13. Analyze if the robot successfully traverses the bridge.

Expected Results:

Test #	Obstacle	Rear Wheel type	Passed?	Comments
1.1	Bridge	Single marble	No	The marble from the rear end changes the direction of the robot, causing it to crash into the wall.
1.2	Bridge	Single marble	No	
2.1	Tunnel	Single marble	Yes	The robot has difficulty in the beginning when the marble encounters the surface of the bridge. It causes the robot to buffer for a couple of seconds and it changes the direction of the robot as it enters the tunnel. The robot can traverse the bridge.
2.2	Tunnel	Single marble	Yes	
Following the failure of test set #1, two marbles were attached in the rear end instead of one. They were placed on the corners.				
3.1	Bridge	Two marbles	No	The two marbles performed better than the single marble. the robot still had difficulties traversing the bridge. The marbles in the rear end continued to change the direction of the robot when traversing the bridge. this made it crash into the wall.
3.2	Bridge	Two marbles	No	
4.1	Tunnel	Two marbles	Yes	The robot had a slight second of hesitation when the marbles encountered the surface of the bridge. The two marbles helped the robot to align itself before entering the tunnel.
4.2	Tunnel	Two marbles	Yes	
Following test set #2, a landing gear was attached to the robot. The two marbles were taken off the robot. a single marble is attached instead right before the landing gear (marble used during navigation and localization).				
5.1	Bridge	Landing gear	Yes	When the landing gear was used, the robot traversed the bridge without any issues.
5.2	Bridge	Landing gear	Yes	
6.1	Tunnel	Landing gear	Yes	Using the landing gears, the robot did not have a second of hesitation when traversing the tunnel. Moreover, the
6.2	Tunnel	Landing gear	Yes	

				landing gears did not change the direction of the robot while entering the tunnel.

Conclusion:

The robot performed best when landing gears were utilized while it traversed both obstacles. The landing gears did not change the direction of the robot while it was traversing either obstacles.

Action:
The next step is to implement a landing gear on the robot which will only be used when the robot is traversing obstacles.

Distribution:
Hardware and Software Teams

5.4.2 Encountering Obstacles at an Angle

Obstacle Traversal- Angle of Obstacle Approach

Tester: *Enan Ashaduzzaman, Bryan Jay*, *Tianyi Zou* **Test Date:** *04/04/18*

Software Version: *N/A*

Hardware Version: *2.0 - 2.1*

Objective:

The objective of this test is to determine whether the robot is able to drive through the bridge and tunnel successfully when encountering the obstacles at an angle. Note that this test is determined successful or not through qualitative observations.

Background knowledge: *N/A*

Procedure:

1. Implement a test code that allows the robot to drive straight on Hardware Version 2.0.
2. Place the robot in front of the bridge
3. Execute the drive-forward code.
4. Analyze if the robot successfully traverses the bridge.
5. Redo steps 2-4 placing the robot at an angle (to check if the robot is able to traverse the bridge when it doesn't start straight)

6. Place the robot in front of the tunnel.
7. Execute the drive-forward code.
8. Analyze if the robot successfully traverses the bridge.
9. Redo steps 6-8 placing the robot at an angle (to check if the robot is able to traverse the tunnel when it doesn't start straight).
10. Implement guard rails on the robot near the corners (Hardware Version 2.1)
11. Redo steps 2-9.

Expected Results:

It is expected that the robot will be able to successfully traverse both obstacles using Hardware Version 2.0. The implementation of guard rails will not be necessary.

Results Obtained:

Test #	Obstacle	Angle of approach (°)	Passed?	Comments
1.1	Bridge	-20	No	The robot encountered the walls of the bridge causing it to get stuck.
1.2	Bridge	-10	Yes	The robot encountered the walls of the bridge but, makes it out successfully.
1.3	Bridge	0	Yes	The robot successfully makes it out of the bridge without encountering the walls.
1.4	Bridge	10	Yes	The robot encountered the rails of the bridge but, makes it out successfully.
1.5	Bridge	20	No	The robot encountered the rails of the bridge causing it to get stuck.
2.1	Tunnel	-20	No	The robot encountered the walls of the tunnel and immediately gets stuck.
2.2	Tunnel	-10	No	The robot encountered the walls of the tunnel and immediately gets stuck.
2.3	Tunnel	0	Yes	The robot successfully makes it out of the tunnel without encountering the walls.
2.4	Tunnel	10	No	The robot encountered the walls of the tunnel and immediately gets stuck.
2.5	Tunnel	20	No	The robot encountered the walls of the tunnel and immediately gets stuck.
Following these mixed results, guard rails were implemented on the robot (hardware version 2.1) with the intention of having the robot to adjust itself using the rails when it encounters the				

walls of either obstacles.				
3.1	Bridge	-20	Yes	The guard rails of the robot encounter the walls of the bridge and adjusts itself to successfully traverse the obstacle.
3.2	Bridge	-10	Yes	The guard rails of the robot encounter the walls of the bridge and adjusts itself to successfully traverse the obstacle.
3.3	Bridge	0	Yes	The robot successfully makes it out of the bridge without encountering the walls.
3.4	Bridge	10	Yes	The guard rails of the robot encounter the walls of the bridge and adjusts itself to successfully traverse the obstacle.
3.5	Bridge	20	Yes	The guard rails of the robot encounter the walls of the bridge and adjusts itself to successfully traverse the obstacle.
4.1	Tunnel	-20	Yes	The guard rails of the robot encounter the walls of the tunnel and adjusts itself to successfully traverse the obstacle.
4.2	Tunnel	-10	Yes	The guard rails of the robot encounter the walls of the tunnel and adjusts itself to successfully traverse the obstacle.
4.3	Tunnel	0	Yes	The robot successfully makes it out of the tunnel without encountering the walls.
4.4	Tunnel	10	Yes	The guard rails of the robot encounter the walls of the tunnel and adjusts itself to successfully traverse the obstacle.
4.5	Tunnel	20	Yes	The guard rails of the robot encounters the walls of the tunnel and adjusts itself to successfully traverse the obstacle.

Conclusion:

When the robot did not have any guard rails (Hardware Version 2.0), the robot got stuck whenever the wheels encountered the walls of either obstacle. The robot is able to successfully traverse both obstacles without any issues only when there are guard rails implemented on the robot (Hardware Version 2.1). The guard rails were able to adjust the direction of the robot so that it slid against the walls.

Action:
Implement guard rails onto the robot.

Distribution:
Hardware Team

5.6 Capture

Object Capture

Tester: *Bryan Jay* **Test Date:** *04/08/18*

Software Version: *Ask Bryan/Volen*

Hardware Version: *2.2*

Objective:

The objective of this test is to ensure that the software implementation of the capture class is working correctly. Capture behavior is an integral part of the client requirements, and thus it needs to be thoroughly tested to ensure it is reliable and accurate.

Background knowledge:

Procedure:

1) Within the capture class, set the desired block color to be captured.
2) Place the color detection sensor on the robot next to a block of a different desired colour. For example, if the desired color is red, place a blue block. The distance between the sensor and the block is illustrated in the figure below.

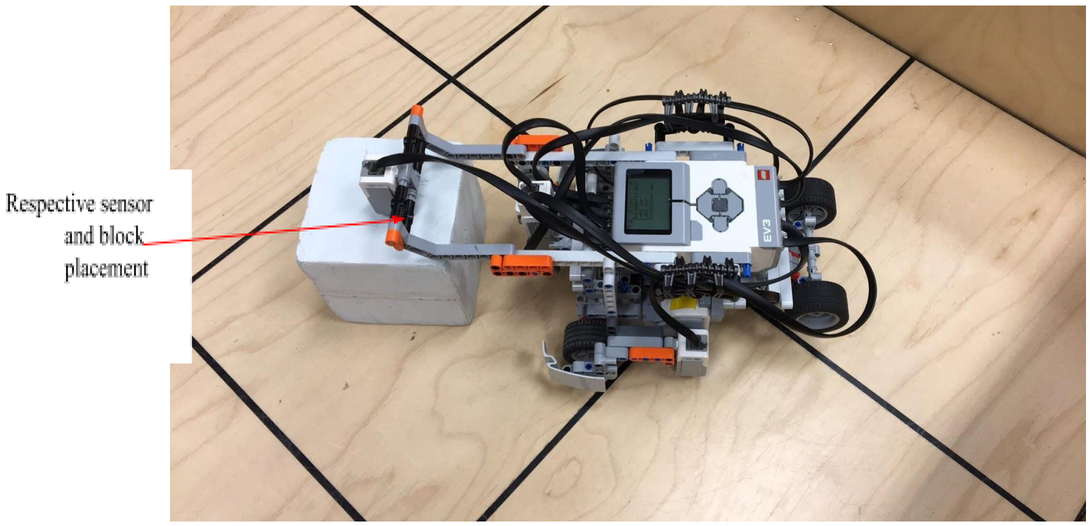


Figure 5.6: Color block detection

3) Run the implemented code.
4) Note the output text displayed on the LCD screen. As this is a test for correct false identification, FALSE should appear on the screen. There should be no beep as well.
5) Repeat steps 2-4 for all the other colors that are not set to the desired one. They should all return FALSE readings.
6) Repeat. Lastly, repeat steps 2-4, but with the desired block color. A reading of TRUE should be displayed on the screen. If TRUE, there should be 3 beeps heard in quick succession indicating object capture.

Expected Results:

It is expected that the color sensor correctly distinguishes the color of each block. When the desired block color is placed in front of the sensor, the LCD display should read TRUE and 3 beeps should be heard in quick succession. When a block color that is not desired is placed in front of the sensor, the LCD display should read FALSE and 6 beeps should be heard in succession.

Results Obtained:

Desired Block Color	Block Color being Compared			
	Red	**Blue**	**Yellow**	**White**
Red	**1. True** **2. 3 beeps heard**	1. False 2. 6 beeps heard	1. False 2. 6 beeps heard	1. False 2. 6 beeps heard
Blue	1. False 2. 6 beeps heard	**1. True** **2. 3 beeps heard**	1. False 2. 6 beeps heard	1. False 2. 6 beeps heard
Yellow	1. False 2. 6 beeps heard	1. False 2. 6 beeps heard	**1. True** **2. 3 beeps heard**	1. False 2. 6 beeps heard

White	1. False 2. 6 beeps heard	1. False 2. 6 beeps heard	1. False 2. 6 beeps heard	**1. True** **2. 3 beeps heard**

Conclusion:

The capture class implementation correctly identifies blocks of the correct color.

Action:

Inform the software team that the software is working, and no change is required.

Distribution:

Software Team

5.8 WI-FI integration

Wi-Fi Integration

Tester: *Bryan Jay* **Test Date:** *03/31/18*

Software Version: *N/A*

Hardware Version: *N/A*

Objective:

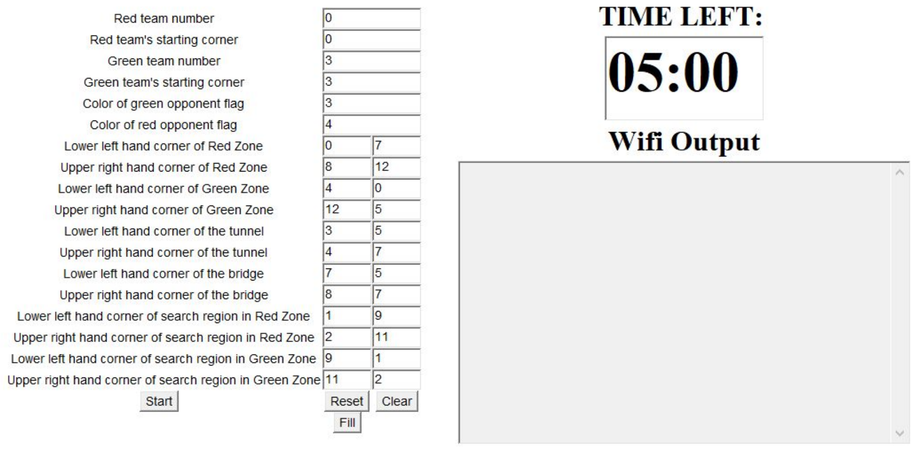
The objective of this test is to determine whether or not the Wi-Fi class functions when it needs to receive data from the Wi-Fi server. Without the data from the Wi-Fi server, the robot would not function since it would not be able to implement data points inside the controller.

Background knowledge:

None

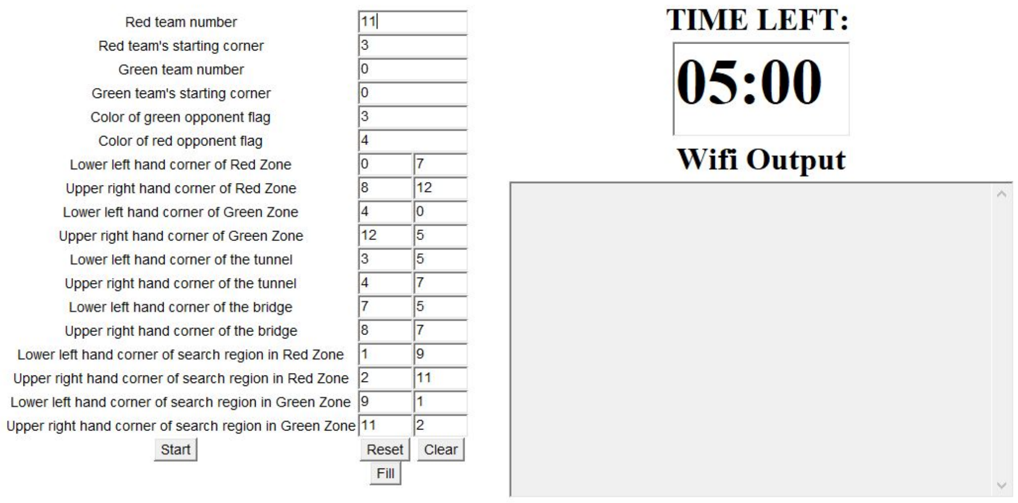
Procedure:

1. Setup Wi-Fi in the brick by connecting it to the DPM server using the wireless USB adapter.
2. Connect your computer to the DPM server and run the DPM server jar file.
3. Using the jar file, fill in the data boxes with the xml data.
4. Adjust the data according to the picture at the bottom. (Testing for Green Team)



Red team number	0	
Red team's starting corner	0	
Green team number	3	
Green team's starting corner	3	
Color of green opponent flag	3	
Color of red opponent flag	4	
Lower left hand corner of Red Zone	0	7
Upper right hand corner of Red Zone	8	12
Lower left hand corner of Green Zone	4	0
Upper right hand corner of Green Zone	12	5
Lower left hand corner of the tunnel	3	5
Upper right hand corner of the tunnel	4	7
Lower left hand corner of the bridge	7	5
Upper right hand corner of the bridge	8	7
Lower left hand corner of search region in Red Zone	1	9
Upper right hand corner of search region in Red Zone	2	11
Lower left hand corner of search region in Green Zone	9	1
Upper right hand corner of search region in Green Zone	11	2

5. Run the robot's code and wait until the robot connects to the DPM server jar
6. Click "Start" in the jar application
14. Watch to see if the robot executed the code
15. Repeat steps 4-7 using the following adjustments in the jar application. (Testing for Red Team)



Red team number	11	
Red team's starting corner	3	
Green team number	0	
Green team's starting corner	0	
Color of green opponent flag	3	
Color of red opponent flag	4	
Lower left hand corner of Red Zone	0	7
Upper right hand corner of Red Zone	8	12
Lower left hand corner of Green Zone	4	0
Upper right hand corner of Green Zone	12	5
Lower left hand corner of the tunnel	3	5
Upper right hand corner of the tunnel	4	7
Lower left hand corner of the bridge	7	5
Upper right hand corner of the bridge	8	7
Lower left hand corner of search region in Red Zone	1	9
Upper right hand corner of search region in Red Zone	2	11
Lower left hand corner of search region in Green Zone	9	1
Upper right hand corner of search region in Green Zone	11	2

Expected Results:

It is expected that the Wi-Fi class functions and is able to receive the data from the Wi-Fi server using the wireless USB adapter. Given that the Wi-Fi class functions, the robot should proceed to localization and navigation.

Results Obtained:

The test was completed 5 times. During each test run, every parameter in the jar application was changed other than the “Team Number.” During every test run, the robot was able to execute the code after receiving data from the Wi-Fi server.

Conclusion:

The Wi-Fi class worked perfectly. Data was received from the Wi-Fi server which was then implemented inside the controller. After receiving the data, the robot executed it’s localization and navigation class.

Action:

The next step is to complete the Search and Capture class, the final step in the software process.

Distribution:

Software Team

Test Plan Approval

The undersigned acknowledge they have reviewed the **Test Plan** document and agree with the approach it presents. Any changes to this document will be coordinated with and approved by the undersigned.

Signature:		Date:	
Print Name:			
Title:			
Role:			

Signature:		Date:	
Print Name:			
Title:			
Role:			

Signature:		Date:	
Print Name:			
Title:			
Role:			

Appendix A: References

1) The following table summarizes the documents referenced in this document.

›cument Name and Version	Description	Location

2) The following is a table template for many of the tests that have been conducted throughout this document.

Tests	Variables				Passed?	Comments:
1	v					
2						
3						
4						
5						
6						

Appendix B: Key Terms

The following table provides definitions for terms relevant to this document.

n	nition

Update History

Friday, March 2nd: This morning, the team conducted its first team meeting with Professor Lowther. The 3 preliminary designs were presented, and their advantages and disadvantages were briefly discussed. The Professor was pleased and indicated that the next step was to test the designs and identify the superior one.

However, during this meeting, discussion led to the final conclusion that preliminary design #3, which is the implementation of a variable track system, is actually completely unfeasible. This is because the hardware design would require the usage of 4 motors, which is simply too costly. Furthermore, it was pointed out that it would in fact be very difficult to ensure that the track is constant, and it would probably double the amount of testing required to figure out a track system. Therefore, this design was discarded during this meeting.

March 5th- 9th: Reading week

Saturday, March 12th : Certain aspects of Preliminary Designs #1 and #2 have been tested in test 4.4: *Wheel type: Treads vs Regular Wheels.* The tests highlight the flaws in navigation that present themselves when implementing treads as the desired wheel type. Furthermore, qualitative observations indicate that the robot actually shakes when using the treads. After observing this, the team conferred with other teams to see if they had experienced a similar problem, and many other teams confirmed our observations. Therefore, preliminary Design #1 has been discarded as a possible idea.

Therefore, following various tests and discussions, preliminary design #2 is going to be used moving forward. The rear lazy wheel that is a key design feature of this design still needs to be tested however.

Also on this day, March 12th, The colour sensor accuracy test has been completed successfully- Tianyi Zou

Regarding the colour sensors, it has been decided that two sensors will implemented for localization instead of one. This is due to the results obverse in the last phase of the research and development stage, the competition of lab 5. During this phase, light localization was implemented with only 1 sensor and the team felt that its dire inaccuracy was due to this. Having spoken to previous teams, it was concluded that using the time delay in a black line being detected by two sensors on each of the front wheels would be an excellent way in marginalizing accuracy error.

It must also be noted that since we are using all 3 light sensors in our design (2 for localization and 1 for colour detection) it seems somewhat redundant to even perform an accuracy test for the sensors. However, we felt that the slightly more accurate sensors should be used for light localization, and the remaining less accurate sensor could be used for colour detection. Therefore, the test was still performed to allow us to implement this.

Sunday, March 13th: It has been decided that no tests will be made for the motors. Following the Research and development phase, it is well known that the difference in motors is almost nonexistent. Withal, no problems have arisen with the current motors that have been used for lab 5. Therefore, no tests are required, and the use of the newer EV3 motors will continue to be utilized for the final design.

Tuesday, March 15th : The US sensors have been tested (Ultrasonic Sensor Consistency Test). – Tianyi Zou

Also, the lazy wheel component of our design was tested today. It was decided that no documented test needs to be presented because the design completely failed. This is because the robot crashed every single time it was run on the bridge. The lazy wheel simply did not provide enough rear end stability to ensure a successful traversal. Because of this, Enan has come up with 3 possible ideas on how to solve this traversal issue. The rear end could be supported by:

1) 2 stabilized rear wheels. However, although this would certainly provided suitable stability, the complexity of the design is questioned. Perhaps the rear could simply be supported by one or two marble wheels. This idea further holds merit because localization is actually more accurate with simply a marble, versus 2 supporting rear wheels. This was concluded after a simple square driver navigation test was done on the same day when the lazy wheel was equipped. The robot had a very difficult time turning due to the friction of the lazy wheel tires. It was immediately clear that although this design might possibly be a solution to obstacle traversal, it would simultaneously serve to hinder the localization and navigation of the robot. Therefore, the other 2 options considered are:
2) 1 marble wheel
3) 2 marble wheels.

An obstacle traversal test must be conducted in order to evaulate which design feature provides the most reliable traversal.

Monday, March 19th : At the meeting with our TA supervisor, it has been brought to our attention that having one test per US sensor per set distance (Ultrasonic Sensor Consistency Test) is not rigorous enough to figure out which one of the sensors is the better one. More data points per distance would be needed per sensor in order to then compare the correlations between the data from each sensor. The sensor with the higher correlation value (most linear) would be the chosen one(s). However, due to the very limited time budget, we do not think that it will be worth it. Testing of the software features still need to be made, and 1 week remains before the beta demo. Furthermore, it is unnecessary to test the US sensor to a higher precision as it will only be used to avoid obstacles. As long as they all work (which the test conducted confirms) its good enough. It won't matter if we start rotating away from an obstacle rom 30 cm away rather than 29 cm, as long as the obstacles is seen. ((Revise this part!! LUKA))

Tuesday, March 20th :

1) The track testing has been successfully completed- Tianyi Zou.
2) The wheel radius Test has been completed successfully- Enan Ashaduzzaman

Wednesday, March 21st: Obstacle Traversal test has been successfully completed.

Following the conclusion reached on *March 15th(Please refer),* the 3 possible rear stability designs were tested. The test clearly indicates that obstacle traversal is most reliable when implementing the stable rear wheels, instead of the marble wheels. However, once again, it must be reiterated that navigation/localization performs very poorly with rear wheels added because of the additional friction.

Hence, the idea of implementing a variable rear wheel system, or "landing gear" if you will, has been proposed by our hardware lead. During localization and navigation to an obstacle, the rear wheels will be in a stored position, not touching the ground, and when the robot stops in front of the tunnel/bridge, the rear wheels will eject like an airplanes' landing gear, and they will be used to aid obstacle traversal. Following a successful traversal, the landing gear will store itself back up, and navigation/localization will continue with just the two front wheels and a supporting gyro ball. This new design, hardware version 1.0 (Please refer to Hardware document, section() , needs to be implemented within the software, and then tested.

Thursday, March 22nd: 1) Following the completion of a verified track value for our robot on *March 20th*, navigation testing was thus possible to be tested. Of course, this test is key in establishing the overall accuracy of our robot's movement towards the first obstacle in the playing field. Minimizing the offset errors during this testing phase is key in producing a very precise final robot for the demo. A testing system was set-up which allowed the navigation to be split into two parts. Firstly, angle theta precision was tested at ±45, ±90 and ±270. Angle offsets were observed and corrected within the software following 3 test sets, and the final result indicates that the constants applied have produced an extremely accurate angle theta. Following this, test sets 4-6 serve to test the TravelTo method by observing the offset errors and adjusting them accordingly. This extensive test procedure concluded by ensuring that Navigation is fully functional and precise. The team is confident that this part of the robot's behavior will perform well for the Beta Demo, which is on Wednesday, March 28th.

2) The landing Gear test- *Back Wheel Functionality* has been completed, but not successfully.

Following the creation of a novel "landing gear" design idea on *March 21st* , the software team tested the practically of this design with respect to the software. It was found that overturning/under turning of the motors occurs frequently, and this affects the overall hardware system integration, as the component is colliding with other pieces such as the chassis. Therefore, the software department has distributed to the hardware team a request to include a "stopper" to deal with this,

3) The 2nd landing Gear test- *Back Wheel Functionality with Stopper* has been completed successfully. Please refer to the relevant test to understand the stopper component. The addition of a stopper has solved the issue of overturning/under turning. Thus, this design is feasible to continue with, and it will form the basis of our final hardware design. This addition is significant and thus the hardware will now be referred to as version 1.1(refer to Hardware document, section)

Sunday, 25th March: The robot has been rebuilt from scratch due to several deficiencies. It is noted here that the new Hardware version is 2.0. Please refer to Hardware Document, section()

Monday, 26th March: Localization has been tested. This is with software version (1.0). Both ultrasonic and light localization were very inconsistent. In both ultrasonic and light localization, there were a few runs which were within the error of tolerance, but overall the accuracy is poor.

Tuesday, March 27th: Further formal testing will have to be postponed until after the Beta demo (happening tomorrow, March 28th at 12.10 PM). That said, the software team has been continuously testing the robot independently while writing the code, hence there is no absolute need for formal testing at this time.

Wednesday, March 28th: The beta demo has passed, and it was completely not successful. During the morning, the software team dedicated itself to try and improve the localization in preparation for the demo at 12:10. Because of this, the Wi-Fi integration was actually not tested to make sure it worked. The class was fully written, and everything looked fine, as it was a simple class, but it was simply not tested.

Later in the day, it was found that due to misnaming of certain variables in this class related to the XML file, during the beta demo, the robot failed to receive the data from the computer. Further edits in this class need to be made and then passed along to the testing team in order to be adjusted for the final demo.

Saturday, March 31st : The Wi-Fi integration has been completed successfully. – Bryan Jay

Wednesday, April 4th: The obstacle traversal- *Angle of obstacle approach* has been tested iteratively.

This test was required because certain runs, localization would make the angle of approach to the bridge/tunnel not necessarily 0 degrees. A deviated approach could cause problems in that the robot would collide with the walls and push off slightly, completely derailing the traversal. This was observed in the test, and that is why guard rails were added on the side of the front wheels. This test conclusion lead to the creation of hardware version 2.1 (See hardware document, section…).

Friday, April 6th : It is noted here that due to a breakdown of the current motors being used, the robot had to be deconstructed and rebuilt. The new Hardware version is 2.2. Refer to Hardware document, section()

Sunday, April 8th: Object Capture has been tested successfully – Bryan Jay

1) The US sensors have been tested (Ultrasonic Sensor Consistency Test). –
2) It has been decided that no tests will be made for the motors. Following the Research and development phase, it is well known that the difference in motors is almost nonexistent. Withal, no problems have arisen with the current motors that have been used for lab 5. Therefore, no tests are required, and the use of the newer EV3 motors will continue to be utilized for the final design.
3) The colour sensor accuracy test has been completed successfully.
4) Regarding the colour sensors, it has been decided that two sensors will implemented for localization instead of one. This is due to the results obverse in the last phase of the research and development stage, the competition of lab 5. During this phase, light localization was implemented with only 1 sensor and the team felt that its dire inaccuracy was due to this. Having spoken to previous teams, it was concluded that using the time delay in a black line being detected by two sensors on each of the front wheels would be an excellent way in marginalizing accuracy error.
5) It must also be noted that since we are using all 3 light sensors in our design (2 for localization and 1 for colour detection) it seems somewhat redundant to even perform an accuracy test for the sensors. However, we felt that the slightly more accurate sensors should be used for light localization, and the remaining less accurate sensor could be used for colour detection. Therefore, the test was still performed to allow us to implement this.
6) At the meeting with our TA supervisor, it has been brought to our attention that having one test per US sensor per set distance (Ultrasonic Sensor Consistency Test) is not rigorous enough to figure out which one of the sensors is the better one. More data points per distance would be needed per sensor in order to then compare the correlations between the data from each sensor. The sensor with the higher correlation value (most linear) would be the chosen one(s). However, due to the very limited time budget, we do not think that it will be worth it. Testing of the software features still need to be made, and two weeks remain before the beta demo. Furthermore, it is unnecessary to test the US sensor to a higher precision as it will only be used to avoid obstacles. As long as they all work (which the test conducted confirms) its good enough. It won't matter if we start

rotating away from an obstacle from 30 cm away rather than 29 cm, as long as the obstacles is seen.

7) The Wheel type test has been completed successfully. A few members of the team had argued in favor of replacing the traditional wheels used during the research and development phase with the treads. This is because of the presence of the speed bumps on the bridge. The use of treads seems like a natural way of traversing an awkward terrain such as this. However, the issue with implementing treads is that it would affect the way our navigation is currently implemented, since it is based on regular wheels. Thus, this test was performed to compare the treads ability to navigate accurately against the regular wheels. The results were not surprising. Having already based our navigation on regular wheels during the Research and Development phase, it is logical to continue using the regular wheels to allow efficient system integration.
8) Obstacle traversal test completed successfully. This test served to highlight a serious issue with our current preliminary design. Through testing, it was realized that the robot is only able to cross the bridge when the back wheels are stabilized. Therefore, real wheels were added to the robot to provide sufficient stability during the traversal.
9) Track testing successfully completed. Through continuous trial and error testing, the optimal track value was found to be 12.62cm.
10) This should go after beta demo. Due to a small change in the hardware configuration of the robot, the track value previously verified is now inaccurate. The motors that are currently being used have stopped working, and thus the robot had be taken apart and rebuilt using new motors. This has affected our track value. Therefore, track testing had to be redone, and the 13.447. 12.62 * (80/170)
11) Navigation Testing successfully completed successfully(v1.0). A new version of the software has affected the reliability of our current navigation. The navigation testing procedure was thus redone using this new software to ensure small changes could be made.
12) Localization- Localization has been tested. This is with software version (1.0). Both ultrasonic and light localization were very inconsistent
13) Beta demo (Friday March 28th) The beta demo has passed, and it was not a success at all. The robot was not even able to get off the start line because the passing of the game parameter through the WIFI class failed. Unfortunately, the Wi-Fi integration was not tested before the demo because we were still having problems with our localization. The team was working hard in fixing the localization, which resulted in the testing of the WiFi class not being tested. After the failure of the demo, it was found that the error lied in the capitalization of the XXY coordinates within the class, where it should have been in small letters.
14) Capture testing has been successfully completed – 8th April 2018
15) **KEY- Navi 2.0 – April 7th bryan jay, Tianyi zou, localization 2.0- April 8th, volen**

Localization: **If it starts to close the line, the expected tile is off. Hence, once it reached the middle point of the next tile, it goes off by half a tile forward.**

March 2nd. Meeting

March4th actually, u tested treads vs wheels navigation if its straight brah. Wheels were superior. Robot shaked when u used treads.

It was not a feasible option because it would take 4 motors, and we cant ensure track is constant , and it would double the tests= unfeasible. Variable rear wheel track. (1st team meeting)

March 14th we had a meeting with prof (Wednesday)

March 15th, u did lazy wheel , u saw it was shit, so u were like actually two wheels at the back. But localization was better wit just a marble, so landing gear was born,. But, complexity of design was questioned?- could we just have single/double marble stability

March 21st