TESTING DOCUMENT

Project: DPM Final Design Project
Task: Document the various tests and experiments to ensure that the robot is functioning properly.

Document Version Number: 6.0
Date: November 28th 2019
Authors: Stefan Barbu

Edit History
Stefan Barbu - 2019/10/27 - Formatting and Layout
Stefan Barbu - 2019/10/27 - Filled in Test Reports Format
Stefan Barbu - 2019/10/27 - Filled in Tests Coverage
Aurelia Haas - 2019/10/27 - Updated formatting and layout to match other documents
Stefan Barbu - 2019/11/03 - Added test design for testing the us sensor's field of view
Stefan Barbu - 2019/11/03 - Added test design for testing the motors displacement
Stefan Barbu - 2019/11/03 - Added test design for testing the robot's displacement when launching
Pierre Robert-Michon - 2019/11/07 - Tested launching displacement experiment
Pierre Robert-Michon - 2019/11/08 - Tested motor displacement
Aurelia Haas - 2019/11/08 - Tested motor displacement
Stefan Barbu - 2019/11/08 - US localization test design, Light sensor localization test design
Stefan Barbu - 2019/11/09 - US sensor fov experiment report
Brendan Marks - 2019/11/10 - US localization experiment report
Brendan Marks - 2019/11/10 - US localization, complete Ball Launch and LS localization
Stefan Barbu - 2019/11/11 - Light sensor line detection experiment
Stefan Barbu - 2019/11/12 - Beta Demo Testing
Brendan Marks - 2019/11/25 - Revise Document
Stefan Barbu - 2019/11/25 - Obstacle Avoidance Experiment
Stefan Barbu - 2019/11/25 - Complete Demonstration Experiment
Stefan Barbu - 2019/11/26 - Added more obstacle avoidance test results
Stefan Barbu - 2019/11/27 - Added more complete demonstration test results
Aurelia Haas - 2019/11/28 - Revision of the entire document

1.0 TABLE OF CONTENTS

2.0 TESTS COVERAGE

2.1 PROPRIOCEPTIVE CONSTRAINTS

The robot should be able to:

1. **Know** which colored square it starts in.
2. **Know** its current orientation within a certain error.
3. **Know** its current position within a certain error.
4. **Know** the deviation of the wheels over a distance.
5. Knowing which square it is in, **know** whether the bridge overlaps or is on the edge of its starting square.
6. **Detect** the closest thing to itself in the direction it **intends to move towards**.
7. **Know** the field of view of its sensors.
8. **Detect** the closest thing to itself in the next direction it **intends to orient to**.
9. **Recognize** the location of an obstacle.
10. **Know** the error launching a ball causes to its position and orientation.

2.2 PHYSICAL CONSTRAINTS

11. The furthest unmovable component of the robot's cross-section should be at a distance to the center of the robot smaller than the radius of the tunnel specified in the Constraints document section Hardware Constraints.

2.3 BEHAVIORAL CONSTRAINTS

The robot should:

12. **Update** the next waypoint when an obstacle is **recognized** at the next waypoint.
13. **Complete** the track in less than the amount of time specified in the Requirements document section 2.0 Capabilities, subsection 2.2 Scope.
14. **Issue** a sequence of 3 beeps upon completing the localization.
15. **Issue** a sequence of 3 beeps upon arriving at their launch point.
16. **Issue** a sequence of 5 beeps upon arriving at their initial corner at the end of the course.

The obstacle avoidance algorithm should:

17. **Engage** upon detecting an obstacle in its path towards its next waypoint.
18. **Terminate** upon arriving at its desired waypoint.
19. **Terminate** upon having **compensated** its trajectory to avoid the obstacle.

The navigation algorithm should:

20. **Orient** itself **towards** its next waypoint.

21. **Advance towards** its next waypoint.
22. **Stay further to** the edge of the land as to not have its wheels cross into the water.
23. **Stay further to** the maps walls as to not have any of its physical components touch a wall.
24. **Verify** that a waypoint is within a valid region.

The launching algorithm/mechanism should:

25. **Launch** a ping-pong ball to the maximal possible distance the bin can be placed at, with a 95% error tolerance within the bin's width. (See the Requirements document section 2.0 Capabilities subsection 2.2 Scope.) (As of 27/10/19 the bin's width and maximal possible distance the bin can be placed at is still unclear.)
26. **Engage** upon arriving at the launching location.
27. Fire all ping-pong balls **one at a time until depleted**.
28. Hold as many balls as allowed. (See the Requirements document section 2.0 Capabilities.)
    - To maximize chances of getting one ball in the bin.
29. **Terminate** when all balls are depleted.

3.0 TEST DESIGN FORMAT

The test designs should be included in the Testing Document under Test Designs with the following sections:

Authors: Name of the people responsible for the test design
Outline: Basic description of how the test should proceed and what should be tested for.
Sample Table: A sample table specifying important data to record.

The results of tests will be recorded in Test Reports.

4.0 TEST REPORT FORMAT

The test reports should be included in the Testing Document under Test Reports with the following sections:

Date: Date the test was performed
Testers: Name of the people performing the test
Hardware version: Version of the robot's hardware with which the test was performed
Software version: Version of the robot's software with which the test was performed. Possibly the commit id of the software ran
Purpose of test: Description of the scope of the test performed. The test should cover one of the items in the Tests Coverage section of this document, otherwise add it in the tests coverage as a suggestion to be reviewed.
Test reliability : How would you know the test has passed or failed? What are the error bounds? How do you know the results are reliable? Is a failure a terminal event or can the system gradually degrade in performance?
Test Procedure: Step by step instructions for carrying out the test. Detailed enough for someone else to reproduce the test. Should contain information on under what time constraints data was collected or at what location data was collected and what type of data was collected.
Expected Results: Hypothesis of the expected behavior prior to running the test.
Test Report: Summary of the test results such as average over number of runs, standard deviation, general behavior of the robot, etc. Name of the file containing the test results.
Conclusion: Did the robot pass the test? Are the results reliable?
Actions: What are the impacts on the Gantt chart? Should the results be forwarded to a team to correct the behavior ? Does it impact other tests? Importance of the results.
Distribution: Which team are the results important to?

Note: Only the most recent reports will be kept in this document, refer to previous versions to view previous reports and Section 7 on test report history.

5.0 TEST DESIGNS

- **Beta Demo Test**

Author: Stefan Petru Barbu
Outline: This test is meant to determine if the robot is able to achieve the required functionalities for the Beta Demo by completing full runs of the software and recording its status at each major milestone.
Sample Table:

Receive parameters	Localize, move to (0,0), less than 30 sec & beep1	Navigate to tunnel (3,3)	Traverse tunnel to island (3,6)	Navigate to bin coords (5,6), orient to (3,6), beep3	Launch at least 4 tiles away, beep1

Tab 5.1. Sample Table of Beta Demo Test

- **Ultrasonic Localization Test**

Author: Brendan Marks
Outline: This test is meant to determine if the robot is able to accurately localize itself with Ultrasonic Localization, within the range that it could be placed along the 45 degree axis of a corner tile. This is achieved by running the US localization algorithm on a range of possible starting angles.
Sample Table:

Initial Angle (degrees)	Final Angle (degrees)	Error (225 degrees desired)	Success (Y/N)

Tab 5.2. Sample Table of Ultrasonic Localization Test

- **Light Sensor Localization Test**

Author: Brendan Marks
Outline: This test is meant to determine if the robot is able to accurately localize itself with light sensor localization within the possible range of values that it may see after ultrasonic localization is completed. This is achieved by running the LS localization algorithm on a range of realistic starting angles.
Sample Table:

Initial Angle (Degrees)	Final Angle Error (Degrees)	Final X Pos (cm)	Final Y Pos (cm)	Euclidean Error in Position (cm)

Tab 5.3. Sample Table of Light Sensor Localization Test

- **Light Sensor Line Detection Test**

Author: Stefan Petru Barbu

Outline: This test is meant to determine the accuracy with which the light sensors are able to detect lines, as well as establishing the threshold for detecting dark lines and determining which light sensors can be seen as being the most reliable. This will be achieved by testing each sensor's readings while passing over dark lines.

Sample Table:

This test will utilize the raw data to generate graphs, there will be no table used.

- **US Sensor Field of View Test**

Author: Stefan Petru Barbu

Outline: This test is meant to characterize the field of view of the Ultrasonic Sensors. This will be achieved by measuring the error in the sensors recorded values according to angle and distance.

Sample Table:

This test will utilize the raw data to generate graphs, there will be no table used.

- **Motors Displacement Test**

Authors: Pierre Robert-Michon, Aurelia Haas

Outline: This test is meant to determine the differences in the amount each motor displaces in comparison to what is expected. This will be achieved by telling the motors to turn different distances and then measuring the actual distance that they have covered once they think they have reached the destination.

Sample Table:

Run Number	Theoretical Y Displacement	Theoretical X Displacement	Actual Y Displacement	Actual X Displacement

Tab 5.4. Sample Table of Motor Displacement Test

- **Launching Displacement Test**

Author: Pierre Robert-Michon

Outline: This test is meant to determine the amount that the launching a ball causes the robot to be displaced. This could be useful to correct the robot's position after the launching stage, and will be achieved by launching a number of balls and measuring the robot's change in position.

Sample Table:

Preliminary test results will determine the need for a table of data.

- **Ball Launching Accuracy Test**

Author: Brendan Marks

Outline: This test is meant to determine if the robot is able to launch balls towards a target point with a reasonable amount of accuracy. This will be tested by launching a given amount of balls and measuring their final position in comparison to the average (expected) launch distance.

Sample Table:

Launch #	X (cm)	Y (cm)	Error Y (cm)	Euclidean Error (cm)

Tab 5.5. Sample Table of Ball Launching Accuracy Test

6.0 TEST REPORTS

- **Beta Demo Experiment**

Date: November 12th, 2019
Testers: Stefan Barbu, Aurelia Haas
Hardware version: Final Hardware
Software version: Commit id cea45f0519ad38d8db60f072e9893f6608fdd89b on the betaDemo branch
Purpose of test: Integration of all the requirement needed to pass the beta demo. This test covers the requirements 1 - 7, 11, 13 - 16, 20 - 23, 26.
Test reliability : The test has passed if all tasks within the test have passed. Any deviation from the 45 degree axis when initially placed will result in a biased odometry and all navigation objectives will be off.
Test Procedure: Define the field areas on the client/server app. Place the robot at a random angle on the square (0,0) with its wheelbase on the 45 degrees making sure to place far enough from the walls so it doesn't touch them as it rotates during US localization. Start the program. Record the time it takes to localize (time until first beep) and whether the localization is successful (the wheelbase is on the corner (1,1) facing east). Record if it navigates to one square before the entrance of the tunnel. Record if it traverses the tunnel successfully to the island. Record if it reaches the launching spot and orients to the upper right corner of the tunnel. Record if the robot launches at least 4 tiles away. Throughout, record any relevant details related to the robot's behavior.
Expected Results: Localization and launch is expected to work. Navigation can sometimes fail because the motors sometimes act out of sync which throws off odometry.
Test Report:

Receive parameters	Localize, move to (0,0), less than 30 sec & beep1	Navigate to tunnel (3,3)	Traverse tunnel to island (3,6)	Navigate to bin coords (5,6), orient to (3,6), beep3	Launch at least 4 tiles away, beep1
Pass	Localized in 28 sec	Pass	Fail, not parallel enough to pass	Fail, navigation, orients north	Pass
Pass	Localize in 28 sec	Pass	Fail b/c not parallel to tunnel to pas through	Fail, navigation orients north	Pass

Pass	Fail, False positive detects wall twice and fails	N/A	N/A	N/A	N/A
Pass	Fail, Initial motor impulse throws robot off the 45degree axis. Test stopped	N/A	N/A	N/A	N/A
Pass	Pass, but initial motor impulse throws the robot off 45 axis and odometry off a bit	Pass but not in the center of square	Fail b/c not parallel to tunnel to pass through	Fail, odometry too off from start	Pass

Tab 6.1. Beta Demo Experiment

Conclusion: The robot never passes all sequential tests successfully. The beta demo can be expected to fail mainly because the robot drifts when it is expected to go straight when passing through the tunnel.

Actions: Results should be forwarded to team lead and software team. Team lead should update the gantt chart to reflect that odometry and navigation do not work reliably. Software should know that odometry and navigation do not work reliably so they can correct them.

Distribution: Team lead, Software team

- **Ultrasonic Localization Experiment**

Date: November 15th , 2019

Testers: Brendan Marks

Purpose of test: Test to determine the reliability of the ultrasonic localization algorithm to determine the robot's initial orientation. Covers the requirement that the robot should know its current orientation within a certain error.

Test reliability : The test has been passed if the difference between the angle to which the robot orientates itself is within 5 degrees of its actual angle to the desired point. Values should be less reliable depending on the noise and reliability of the US sensors.

Test Procedure: Place the robot on a corner square on the 0 degree axis with respect to the point (1,1). Record the robots orientation with respect to the desired point. Start the US localization. When the robot stops moving, measure and record the robots orientation and compare to the 0 degree (See *Figure 1*). Since our algorithm turns the back of the robot towards the point, we are measuring the angle error from the angle of the corner (180 degrees).

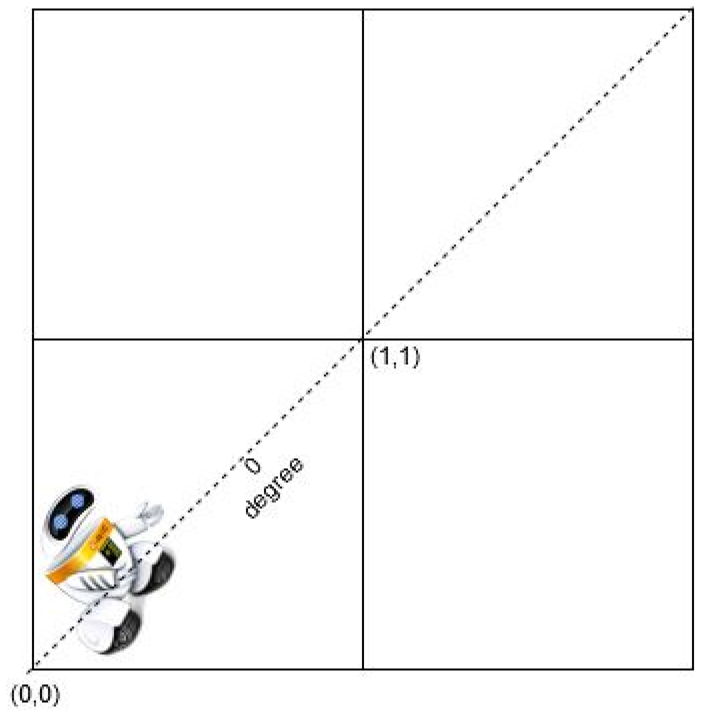

Figure 6.1. Ultrasonic Localization diagram

Expected Results: The robot should be able to localize itself accurately, finishing the localization pointing towards the 180 degree angle (the point (0,0)), within 5 degrees.

Test Report:

Ultrasonic Localization Accuracy Depending on Starting Angle:

Initial Angle (degrees)	Final Angle (degrees)	Error (180 degrees desired)	Success (Y/N)
-45	167	13	N
-25	169	11	N
0	172	8	N
25	174	6	N
45	176	4	Y
Average		8.4	N

Tab 6.2. Ultrasonic Localization Experiment

Mean: 8.4 degrees error

Conclusion: Although the algorithm works fairly well, the average error is over the desired 5 degree error tolerance. However, it can be seen that the robot consistently under-turns and therefore it may be the case that adding a slight adjustment (having it turn a few degrees more than calculated) may solve this issue.
Actions: Inform the software team of the failed tests, suggest that they use the gathered data to make suitable adjustments.
Distribution: Software Team Lead

- **Light Sensor Localization Experiment**

Date: November 15th , 2019
Testers: Brendan Marks
Hardware: Most recent hardware version.
Software: GitHub commit #cea45f0519ad38d8db60f072e9893f6608fdd89b.
Purpose of test: Test to determine the reliability of the robot's light sensor algorithm to determine the robot's initial position. Covers the requirement that the robot should know its current position within a certain error.
Test reliability : The test has been passed if the euclidean distance error between the return position and the robot's position is within 0.5 cm of the desired point and 5 degrees of the 0 degree axis (See *Figure 1*).
Test Procedure: Place the robot at angles in the range -15 to +15 degrees from the 0 degree axis (ie: assume that US localization worked within accurate range), run the light sensor localization and record the final position and angle of the robot.
Expected Results: The robot should be positioned on the point (1,1) and be aligned with the Y-axis.
Test Report:
Light Localization Accuracy Depending on Initial Angle:
*Final angle and final angle error are merged because they both describe the distance from the 0 degree axis.

Initial Angle (Degrees)	Final Angle Error (Degrees)	Final X Pos (cm)	Final Y Pos (cm)	Euclidean Error in Position (cm)
-15	15	1	3	3.162278
-10	12	1	4	4.123106
-5	10	0.5	2	2.061553
0	0	0	0	0

5	5	0.5	1	1.118034
10	3	1	1	1.414214
15	2	2	0	2
Avg:	6.714286			1.982741

Tab 6.3. Light Sensor Localization Experiment

Mean error in angle: 6.714 degrees

Mean error in position: 1.983 cm

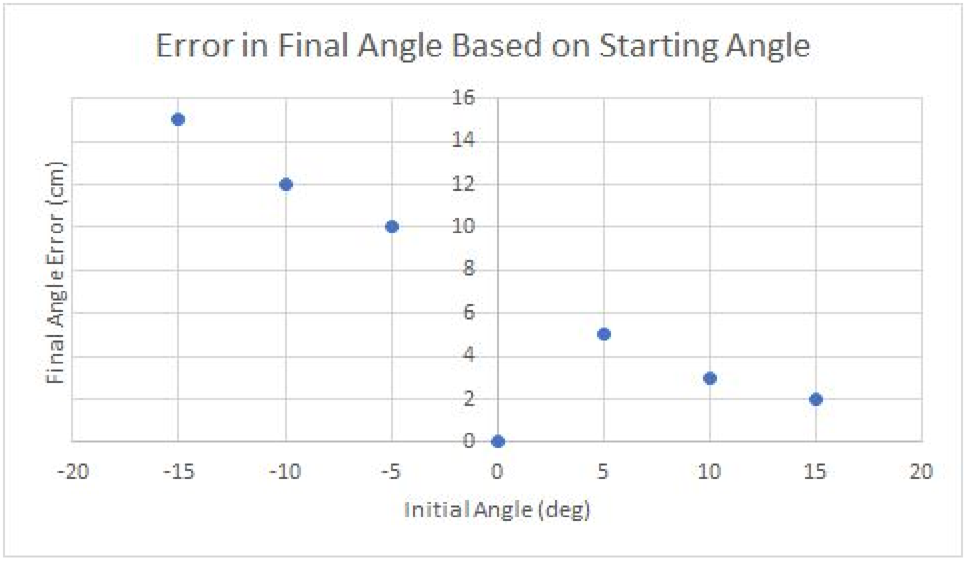

Figure 6.2. Final Angle Error

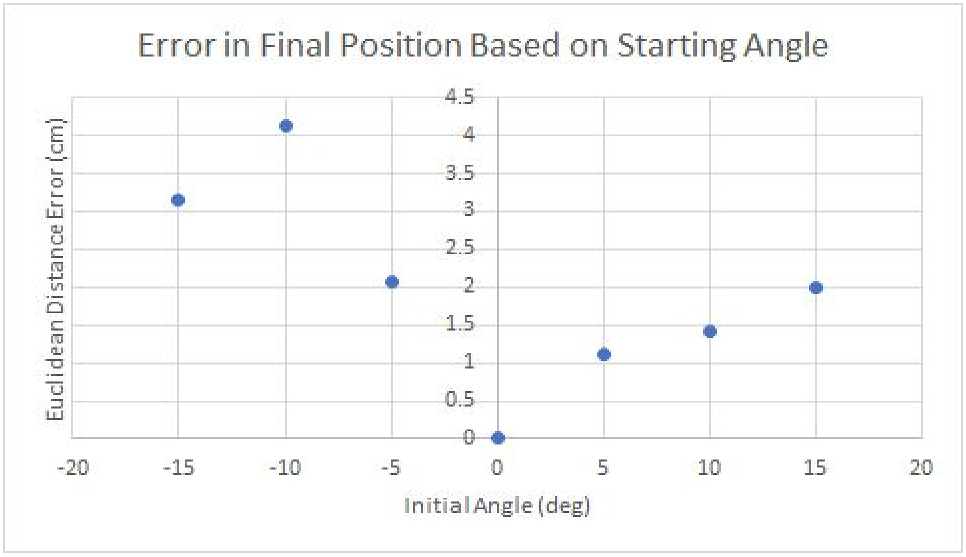

Figure 6.3. Euclidean Distance Error

Conclusion: From the data, it is possible to conclude that the light localization's accuracy is highly dependent on the initial angle it is given. The tested angles were within what could be expected for the robot's initial angle after finishing ultrasonic localization, so overall the light localization is seen to have failed the experiment. In the majority of cases, aside from when it is placed at a perfect initial angle, its final position and angle fall outside of the desired range.
Actions: The software team will be notified of this failed experiment, but there are several possible courses of action to resolve this issue. Either the light localization algorithm is corrected to ensure that the initial angle has less of an effect on the final position and angle, or they could also improve the ultrasonic sensor localization's accuracy, such that the robot will only need to localize accurately within a smaller range of initial angles.
Distribution: Software Team Lead

- **Light Sensor Line Detection Experiment**

Date: November 11^{th}, 2019
Testers: Stefan Barbu
Hardware version: Final Hardware
Software version: Commit id d2d2fee3b77a3154f187729a85686dae86013f99 on the github Testing branch.
Purpose of test: Experiment to determine the light threshold for detecting a dark line. Comparison between the light sensors can tell which is more reliable. Covers the requirement that the robot should know its current orientation within a certain error and that it should know its current position within a certain error.
Test reliability : Experiment will determine the reliability of the sensor and help determine the threshold to filter unreliable data out.
Test Procedure: Place the robot with its wheelbase on a corner. Run the program. It turns in circles on itself, polling the light sensor at constant intervals and saves the data to "LightSensor Intensity.csv".
Expected Results: Results should not fluctuate significantly in the center of the center where there is no line and should fluctuate a lot when overlapping differently with a line.
Test Report: Raw test data located in "Test Data Raw/Light sensor intensity/".

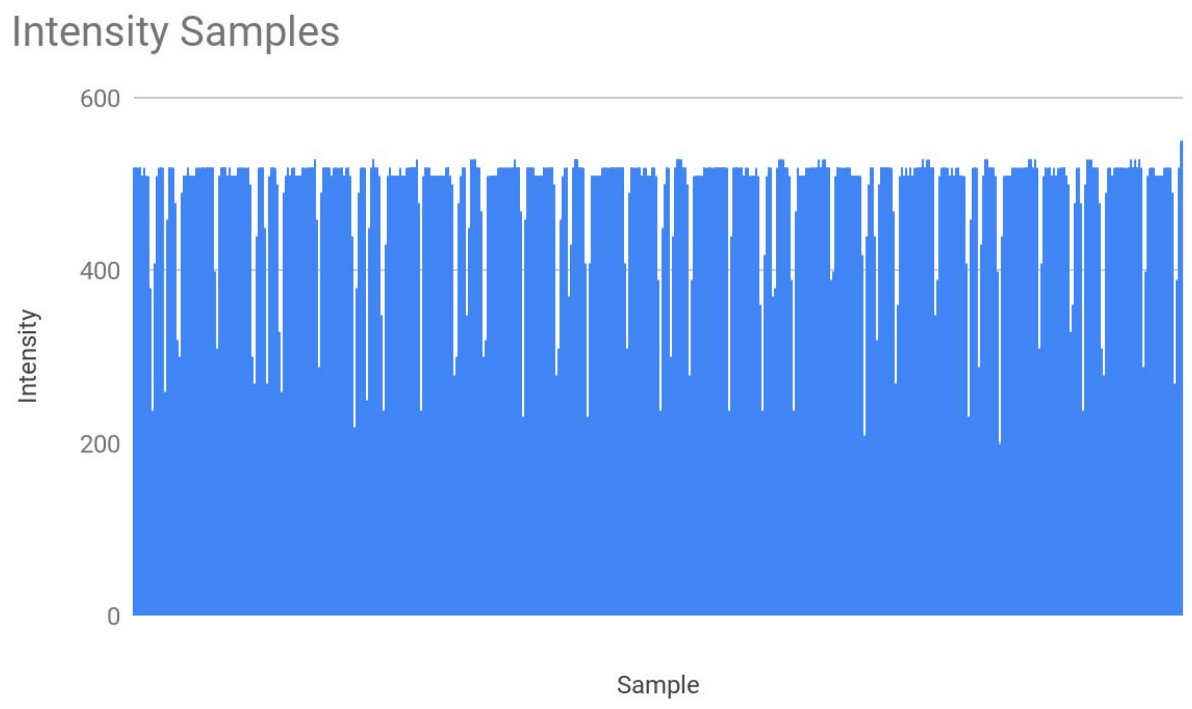

Figure 6.4. Light Sensor Returned Intensity Value

Conclusion: Data shows that light intensity is reduced between 83% and 66% of its original value when the sensor detects a line compared to when it is not. False positive have much more of an impact because they make odometry correct itself at the wrong location which throws it completely off. False negative are not so impactful because the line could be detected at the next poll with multiple polls per line. False negatives have more chances of being detected.

Actions: Results should be forwarded to the team lead, the software team and the testing team. The testing team should test every sensor in all kits to select the more reliable ones. The software team should implement a differentiator filter rather than a filter given that a fixed threshold can't be determined. Or maybe they could only poll the light sensor about when a line is expected within a certain range rather than anywhere on the board, relying on the assumption that odometry is somewhat reliable across one square. If the software team chooses to implement a threshold filter, they should run their tests to choose which value between 83% and 66% is the most reliable. The team lead should expect more budget expenditure for the testing.

Distribution: Team lead, Software team

- **US Sensor Field of View Experiment**

Date: November 9th, 2019 and November 10th, 2019

Testers: Stefan Barbu

Hardware version: Final Hardware

Software version: Commit id 1beb087bd8645bcb77bdbb13f4e4d02556b8b088 on the github Testing branch

Purpose of test: Experiment to characterise the field of view of the Ultrasonic Sensor. Covers the requirement that the robot should know the field of view of its sensors.

Test reliability : Angle values are off by less than 5 degrees because the odometry constants were determined after this test was run. Experiment will indicate accuracy of the sensors.

Test Procedure: Place the robot at 5 cm from the wall. Run the code that turns the robot, polls the sensor and print the values to a file. When the robot stops moving, place the robot at the next distance and press any button to continue. Distances to test are 5 cm to 30 cm by 5 cm increments. Press the back button to terminate gracefully and save the data to the file. The test data is located in the "US_sensor_data.csv" file.

Expected Results: The noise should be greater for bigger distance from the wall.

Test Report: Raw test data is located in "Test Data Raw/ US sensor fov/".

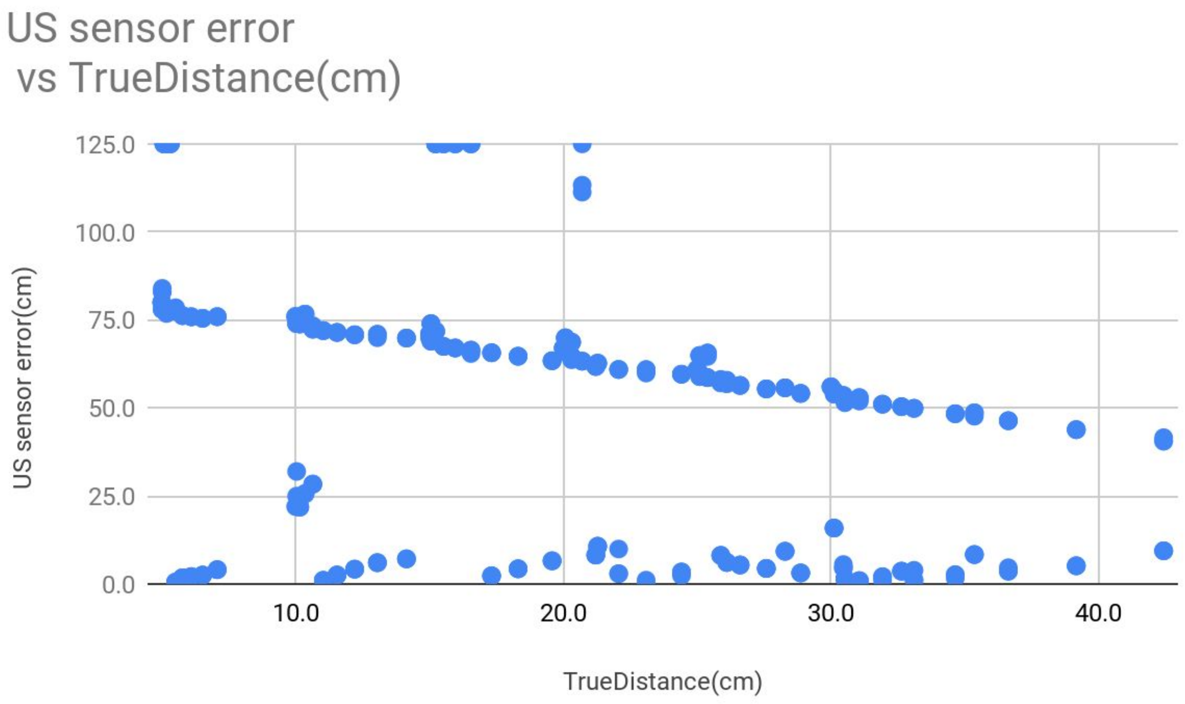

Figure 6.5. US returned value error vs Robot's distance from the wall

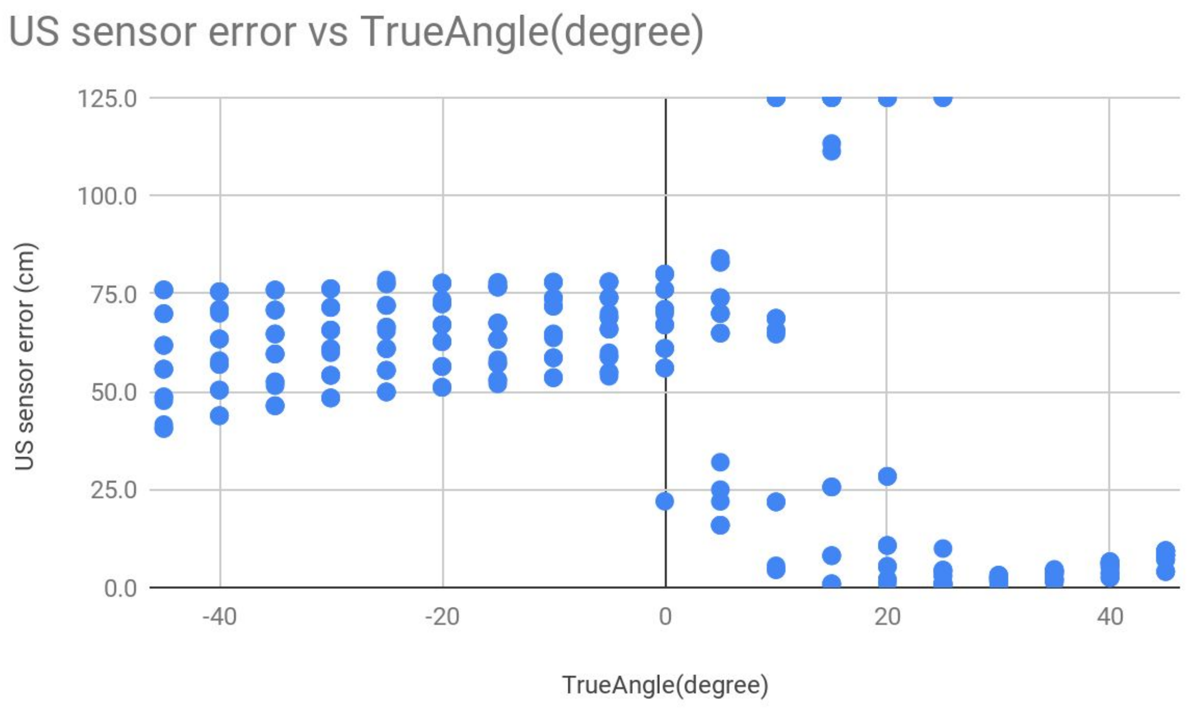

Figure 6.6. US returned value error vs Robot's angle relative to the perpendicular of the wall

Conclusion: Data shows that the US sensors are most reliable to detect obstacles when they are located to its right. Data also shows that the US sensor is more reliable to detect obstacles further away following the trend presented in graph 1.
Actions: The results should be forwarded to the software and hardware team. The software team can better design their algorithms and the hardware team so they can place the sensors at a convenient orientation. The robot should have only one sensor which rotates on itself to detect an obstacle right in front of it.
Distribution: Software team, hardware team

- **Motors Displacement Experiment**

Date: November 8^{th}, 2019
Testers: Pierre Robert-Michon, Aurelia Haas
Hardware version: Final Hardware
Software version: GitHub commit id #cea45f0519ad38d8db60f072e9893f6608fdd89b.
Purpose of test: Experiment to test the displacement between the two wheels motors. Covers the requirement that the robot should know the deviation of the wheels over a distance.
Test reliability : The test has passed if the actual displacement of X and Y is equal to the theoretical displacement. However, errors of $\pm 0.05\ cm$ should be accepted.

Furthermore, the results also depend on the battery of the EV3 as the accuracy of the motors are linked to it.

Test Procedure: This test is separated into 3 parts:

1. The robot is manually and as precise as possible placed at the origin (0, 0) of the field. The robot is then launched to go straight for a distance of 1 tile (30.48 cm). Once stopped, using a ruler, the positions of X and Y are measured. The final error are written into the Tab 1 below. This test was realized 5 times.
2. Realize the same test as the one above, but here the distance should be equal to 3 tiles (change it in the software).
3. Realize the same test as the one above, but here the distance should be equal to 5 tiles (change it in the software).

Expected Results: Hypothesis: A displacement should exist between the theoretical and actual positions of the robot because of dust, weight or slippage in both wheels.

Test Report: Summary of the test results such as average over number of runs, standard deviation, general behavior of the robot, etc. Name of the file containing the test results.

This test was conducted with 3 sets of 5 runs. The distance to reach for the 5 first tests is 1 tile (30.48 cm); the distance to reach for the 5 next tests is 3 tiles (91.44 cm) and the distance to reach for the 5 last tests is 5 tiles (152.4 cm). There is a displacement between the theoretical and actual positions, even more important after a few tests.

Standard deviations:

$\boldsymbol{\sigma} = \sqrt{\frac{\sum_i^n (x_i - \bar{x})^2}{n-1}}$ and (STDEVA()) on Excel.

Standard deviation for tests 1 to 5:

- σ(X) = 0.11 cm
- σ(Y) = 0.23 cm

Standard deviation for tests 6 to 10:

- σ(X) = 0.55 cm
- σ(Y) = 0.32 cm

Standard deviation for tests 15 to 15:

- σ(X) = 1.23 cm
- σ(Y) = 0.16 cm

Run Number	Theoretical Y Displacement	Theoretical X Displacement	Actual Y Displacement	Actual X Displacement
1	30.48 cm	0 cm	29.58 cm	-0.1 cm
2	30.48 cm	0 cm	29.98 cm	-0.2 cm

3	30.48 cm	0 cm	30.08 cm	-0.2 cm
4	30.48 cm	0 cm	29.58 cm	-0.3 cm
5	30.48 cm	0 cm	29.78 cm	-0.4 cm
6	91.44 cm	0 cm	89.14 cm	-0.9 cm
7	91.44 cm	0 cm	88.94 cm	-1.1 cm
8	91.44 cm	0 cm	89.24 cm	-1.5 cm
9	91.44 cm	0 cm	89.34 cm	-1.7 cm
10	91.44 cm	0 cm	89.24 cm	-2.3 cm
11	152.4 cm	0 cm	149.1 cm	-3.7 cm
12	152.4 cm	0 cm	148.7 cm	-5.7 cm
13	152.4 cm	0 cm	149 cm	-2.6 cm
14	152.4 cm	0 cm	149 cm	-5.2 cm
15	152.4 cm	0 cm	148.8 cm	-4.4 cm

Tab 6.4. Motor Displacement Experiment

Conclusion: The robot did not pass the test. The displacements are still too important and consequently on a larger distance, the robot could not be able to reach the desired point.
After realizing the tests, the conclusion was to change the wheels and use the most two similar wheels in the different kits given.
Actions: The test results should be forwarded to the hardware and software team, to find the more similar wheels and put them on the final hardware, as well as balancing correctly the motor correction. Finally, the results should be forwarded to the testing team to realize again the same tests until reaching a positive conclusion.
Distribution: Hardware, Software, Testing.

- **Launching Displacement Experiment**

Date: November 7th , 2019
Testers: Pierre Robert-Michon, Stefan Barbu
Hardware version: Final Hardware
Software version: Launch Class 1.0
Purpose of test: Experiment to determine the displacement launching a ball causes to the robot's positioning and orientation. Covers the requirement that the robot should know the error launching a ball causes to its position and orientation.

Test reliability : It should be noted that this test was ran with two rubber bands for the launcher (1 green and 1 blue). The green rubber band has a small slit which may slightly impact the distribution of the force and the overall performance of the launcher.
Test Procedure: Place the robot's wheelbase on a line, as perpendicular as possible. Press the button to active the launching mechanism. Record the displacement of the robot's wheelbase and its angle. Without moving the robot or correcting its position, press the button to active the launching mechanism. Repeat to have the displacement for 4 launches. Replace the robot to its initial position. Repeat the procedure to have 10 sets of experiments.
Expected Results: Initial Hypothesis: The error should increase with the amount of consecutive launches. One launch should not change its position or orientation significantly. Revised Hypothesis: This test was designed prior to the final hardware build. While a launching arm mechanism would cause sway and displacements (therefore error) in consecutive launches, the rubber band launching mechanism under test is not expected to cause much if any displacement of the robot during consecutive launches.
Test Report: This test was conducted with 5 sets of 5 consecutive launches. It was deemed sufficient to conduct only 5 sets of launches instead of 10, due to the obtained results. There is no displacement at all when the robot launches a ball: no X, no Y displacements and no change in the Theta orientation. The average reach of the launcher is around 6.5 blocks. It is important to not that the first 4 shots are steady, reaching 6.5 to 7 blocks on average. However, the 5th and last shot is usually higher and shorter, because there is no ball remaining in the reloading mechanism behind it to push properly in place. The launching piece consequently does not hit it at the intended angle, resulting in a shorter 4-5 block shot.
Conclusion: The robot easily passes this test. The slit in the rubber band does not have an effect on the reliability of the test. It is safe to say that there is no displacement caused by launching a ball. This test raises one hardware concern: the disparity in results between the first 4 balls and the 5th one. It also raises the need for a new test to determine the exact launcher reach, which will be needed for determining the launch position with software.
Actions: The test results should be forwarded to the hardware team, to fix the 5th ball launch defect. The results should also be forwarded to the testing team to design a new test conjointly with the software team to determine the precise reach of the launcher, and the
Distribution: Hardware, Software, Testing

- **Ball Launching Accuracy Experiment**

Date: November 15^{th} , 2019
Testers: Brendan Marks
Hardware: Final hardware.
Software: GitHub commit id #cea45f0519ad38d8db60f072e9893f6608fdd89b.
Purpose of test: Experiment to determine the accuracy of the robot's ball launching. Covers the requirement that the robot should be able to launch a ball.

Test reliability : The ball should be able to launch the balls within the accuracy of one square (30.48cm).
Test Procedure: Place the robot's wheelbase on a line, as perpendicular as possible. Press the button to active the launching mechanism. Record the landing position of the ball after being launched. Repeat the procedure to have 10 sets of experiments.
Expected Results: The ball launching mechanism should be able to launch the balls consistently to a desired distance, within 1 square accuracy.
Test Report:
Ball Launch Accuracy Experimental Data:
* Note: Error X merged with X column since they are both distance for 0 on the X-Axis. The Y-Error is based on the distance of the landing from the average (assumed) distance.

Launch #	X (cm)	Y (cm)	Error Y (cm)	Euclidean Error (cm)
1	10	198.12	-64.008	64.78444
2	12	243.84	-18.288	21.87352
3	7	289.56	27.432	28.31103
4	1	274.32	12.192	12.23294
5	4	243.84	-18.288	18.72034
6	-16	281.94	19.812	25.46596
7	4	281.94	19.812	20.21176
8	3	274.32	12.192	12.55567
9	7	259.08	-3.048	7.634809
10	6	274.32	12.192	13.58841
Average:	3.8	262.128		22.53789

Tab 6.5. Ball Launching Accuracy Experiment
Mean X Position: 3.8 cm
Mean Y Position: 262.13 cm
Mean Euclidean Error in Position for (0, Average Y Launch Distance) = 22.54 cm
Conclusion: Given that the average euclidean distance from the expected point is 22.54 cm and this is less than the tile length/width, this is deemed acceptable performance for the launcher.
Actions: There are no required actions at this point for the launcher.

Distribution: N/A

- **Obstacle Avoidance Experiment**

Date: November 25th , 2019, November 26th , 2019
Testers: Stefan Barbu
Hardware version: Final Hardware
Software version: Git commit 99d98db80c16100fd70ddc5f5b710ae2e34acfd9 on the Testing branch.
Purpose of test: Integration testing of the obstacle avoidance. Covers the requirements that the obstacle avoidance should engage upon detecting an obstacle in its path towards its next waypoint, terminate upon arriving at its desired waypoint and terminate upon having compensated its trajectory to avoid the obstacle.
Test reliability : The test has passed if the robot reaches its final waypoint without colliding with any obstacles and staying within the island's borders.
Test Procedure: Place the robot on a square corner. Program a waypoint for the robot to reach. Place an obstacle on the shortest path from the robot to the programmed waypoint. Press start. The robot's obstacle avoidance should engage when the US sensor retrieves a distance shorter than a set threshold. The avoidance algorithm follows a defined path, stops at the end and resumes going to the waypoint after. Record the robot's behavior and how the obstacle was placed on the robot's path.
Expected Results: The robot should be successful when the obstacle is not on the waypoint and that the obstacle is not at an angle that deflects the US sensor's waves.
Test Report:

Trial	1(Nov 25)	2(Nov 25)	3(Nov 26)	4(Nov 26)	5(Nov 26)	6(Nov 26)
Description of obstacle on the path of the robot	Path crosses the obstacle in the center, perpendicular	Path crosses the obstacle in the center, perpendicular	Path crosses the obstacle in the center, perpendicular	Path crosses the obstacle in the center, perpendicular	Path crosses the obstacle in the center, perpendicular	Path crosses the obstacle in the center, perpendicular
Behavior of the robot	Obstacle avoidance engages with obstacle at ~20 cm distance and	Obstacle avoidance engages with obstacle at ~10 cm distance. Fails to	Starts obstacle avoidance, comes back to where it engaged obstacle	Engages obstacle avoidance ~10cm away from obstacle. Moves on x-axis one	Detects obstacle after it hit it and starts reversing.	Engages obstacle avoidance ~10cm away from obstacle. Reverses to corner it

	collides with it at the end	reach destination . Disengages too soon.	avoidance. Crashes into obstacle.	square, moves on y-axis one square, stops.		came from, moves on x-axis one square, moves on y-axis one square, move in opposite direction on x-axis. Turns towards obstacle, detects it, reverses to corner and stops.

Tab 6.6. Obstacle Avoidance Experiment

Conclusion: Obstacle avoidance should work, but odometry is off when it ends.
Actions: Recalibrating after obstacle avoidance ends is necessary when it should come back to its initial corner. Results should be forwarded to software so they correct the error in odometry.
Distribution: Software team

- **Complete Demonstration Experiment**

Date: November 25th , 2019, November 27th , 2019
Testers: Stefan Barbu, Isaac Di Francesco
Hardware version: Final Hardware
Software version: GitHub commit id 974901ea4c83bd643187234e5c76fc7c9f7c3a74 on master branch.
Purpose of test: Experiment to test the robot under the same conditions expected during the final demonstration. This test covers all the requirements listed in the test coverage section.
Test reliability : The test has passed if requirements listed in the test coverage are met. Any failed requirements are to be adjusted before the final demonstration.
Test Procedure: Define the field areas on the client/server app. Place the robot at a random angle on the square (0,0) with its wheelbase on the 45 degrees making sure to place far enough from the walls so it doesn't touch them as it rotates during US localization. Start the program. Record the time it takes to localize (time until first beep) and whether the localization is successful (the wheelbase is on the corner (1,1) facing east). Record if it navigates to one square before the entrance of the tunnel. Record if it traverses the tunnel successfully to the island.

Record if it reaches the launching spot and orients to the upper right corner of the tunnel. Record if the robot launches the ball in the bin. Record if the robot navigates back to starting corner. Throughout, record the robot's ability to avoid any obstacles placed in its path and return to the desired waypoint after doing so. Record the time it takes for the robot to complete the circuit.

Test Report:

Trial	1(Nov 25)	2(Nov 25)	3(Nov 25)	4(Nov 27)	5(Nov 27)	6 (Nov 27)
Localize, move to (0,0), less than 30 sec, 3 beeps	Localized in 24 sec.	Motors unsynchronized but localized successfully in 20 sec.	Thread crash, headed into the wall.	Localized in ~20 sec.	Localized in ~20 sec.	Tread crash, head into wall.
Navigate to tunnel	Pass, navigates one tile off the tunnel	Pass	N/A	Pass	Pass	N/A
Traverse tunnel to island	Pass	Pass	N/A	Pass	Pass	N/A
Navigate to launch location avoiding obstacles, orient towards bin, 3 beeps	Engages at ~20 cm from the obstacle, collides with it at the end.	We know obstacle avoidance unreliable for now. Pass without obstacles	N/A	Pass. No obstacles in path	Pass. No obstacles in path	N/A
Launch all ping pong balls into bin	First 4 balls launched in bin 7 away. Fifth ball launched at ~4 distance.	First 4 balls launched in bin 7 away. Fifth ball launched at ~4 distance.	N/A	Pass, launch distance of 4. 1 ball reaches bin, 3 balls hit rim of the bin, last misses.	Pass, launch distance 4. 4 balls reach the bin, last misses	N/A

Navigate back to tunnel	N/A	N/A	N/A	Fail. Instead of going to the entrance of the tunnel on the launch island, tries to go to the entrance of the tunnel on the starting island.	Pass	N/A
Traverse tunnel to start island	N/A	N/A	N/A	N/A	Pass	N/A
Navigate to start corner, 5 beeps	N/A	N/A	N/A	N/A	Pass	N/A
Terminates in less than 5 minutes	N/A	N/A	N/A	N/A	Pass. Terminates with 30 sec left.	N/A

Tab 6.7. Complete Demonstration Experiment (Part 1)

Trial	7(Nov 27)	8(Nov 27)	9(Nov 27)	10(Nov 27)	11(Nov 27)	12(Nov 27)
Localize, move to (0,0), less than 30 sec, 3 beeps	Localized in ~20 sec.	Localized in ~20 sec.	Localized in ~20 sec.	Localized in ~20 sec.	Localized in ~20 sec.	Localized in ~20 sec.
Navigate to tunnel	Pass.	Pass.	Pass.	Pass.	Pass.	Pass.
Traverse	Pass.	Pass.	Pass.	Pass.	Pass.	Pass.

tunnel to island						
Navigate to launch location avoiding obstacles, orient towards bin, 3 beeps	Pass. No obstacles.	Pass. No obstacles.	Fails to detect obstacle.	Detects obstacle. Moves one square following x-axis. Comes back to where started avoidance. Resumes behavior and crashes into obstacle.	Detects obstacle. Reverses backwards to closest corner. Moves along x-axis to avoid obstacle. Detects wall as obstacle, crashes into it.	Pass. Detects obstacle. Move along x-axis to avoid obstacle and resumes path on y-axis.
Launch all ping pong balls into bin	Pass, launch distance 4. 4 balls reach the bin, last misses	Pass, launch distance 4. 4 balls reach the bin, last misses	N/A	N/A	N/A	Fail. First ball touches rim.
Navigate back to tunnel	Pass	Pass	N/A	N/A	N/A	Fail, navigates 2 squares off.
Traverse tunnel to start island	Fail. Detects tunnel as obstacle.	Pass	N/A	N/A	N/A	N/A
Navigate to start corner, 5 beeps	N/A	Pass	N/A	N/A	N/A	N/A
Terminates	N/A	Pass. ~ 30	N/A	N/A	N/A	N/A

in less than 5 minutes		sec left.				

Tab 6.8. Complete Demonstration Experiment (Part 2)

Conclusion: Obstacle avoidance is failing point of robot’s behavior. The robot can finish the circuit without obstacles.

Actions: Forward results to software team so they know to keep on working on it.

Distribution: Software team

7.0 HISTORY OF EXPERIMENTS RUN

November 7th, 2019:
Launching displacement experiment. Concluded that there is no displacement caused to the robot by launching balls. No further changes needed.

November 8th , 2019:
Motors displacement experiment. Concluded that the robot did not pass the test, and that the displacement is too large over longer distances. Actions to correct this issue required.

November 9th & 10th, 2019:
US sensor field of view experiment. Showed that the sensors were more accurate for obstacles placed right in front of the robot. Data to be passed to software team to be taken into account for algorithm design.

November 11th, 2019:
Light sensor line detection experiment. Determined the light threshold, which sensors most accurate. Passed data to hardware and software teams for implementation.

November 12th, 2019:
Beta demo experiment. Did several test runs of the beta demo and recorded the results. Found that the robot never passes all tests successfully. Robot still needs a lot of work.

November 15th, 2019:
Ultrasonic localization experiment. Determined that the robot was not yet reliable enough with its US localization. Algorithm needs to be modified to correct this.

November 15th, 2019:
Light sensor localization experiment. It was determined that the light sensor localization was not accurate enough, but that it was accurate when the initial angle was correct. This information to be passed to software team to fix as they see fit.

November 15th, 2019:
Ball launching accuracy experiment. Found that the launching ability was adequately accurate.

November 25th, 2019:
Obstacle avoidance experiment. Found that the obstacle avoidance was not working as desired. Forwarded to software team to attempt to resolve before deadline.

November 25th, 2019:
Complete demonstration experiment. It was determined that the robot was very unlikely to complete all tasks since the obstacle avoidance was not working correctly.

November 26th, 2019:
Quick testing rounds to provide feedback for software development on obstacle avoidance. Multiple software changing from one test to the other leads to clear improvements.

November 27th, 2019:
The robot was expected to succeed the presentation demos but failed. Quick testing rounds in between demos to provide feedback for final software updates. Multiple software changes from one test to the other leads to clear improvements.

8.0 GLOSSARY OF TERMS

All terms used should be easily understood by anyone with a basis in electrical, computer or software engineering, so no definitions will be needed.