ECSE211 DESIGN PROJECT

TEST DOCUMENT

Version 1.11
2019/4/9
Team 14

VERSION HISTORY

Title	Test Document			
Description	Test the robot to make sure it meets all the requirements			
Created By	Hanwen Wang (Testing Manager)			
Date Created	2019/2/26			
Version Number	**Modified By**	**Modifications Made**	**Dote Modified**	**Status**
1.00	Hanwen Wang	Created the Testing Document Template	26th February	First version of the document, planned for possible future tests.
1.01	Hanwen Wang	Completed 1.1, 1.2, 4.4	11th March	All other tests remaining.
1.02	Hanwen Wang	Completed 3.1, 3.2, 3.3	12th March	All other tests remaining.
1.03	Hanwen Wang	Completed one version of 3.4	13th March	All other tests remaining.
1.04	Hanwen Wang	Completed 1.3	14th March	All other tests remaining.
1.05	Hanwen Wang	Completed 4.7	18th March	All other tests remaining.
1.06	Hanwen Wang	Completed 3.4, 4.1	21st March	All other tests remaining.
1.07	Hanwen Wang	Completed parts of 4.2	27th March	All other tests remaining.
1.08	Hanwen Wang	Completed 4.2	29st March	All other tests remaining.
1.09	Hanwen Wang	Completed 4.3, 4.5	2nd April	All other tests remaining.

1.10	Hanwen Wang	Completed part of 4.6, part of 5	4th April	Two tests remaining.
1.11	Hanwen Wang	Completed 4.6, 5	9th April	Complete version.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose of Test Document

Test document records all necessary steps of testing. It keeps track of what needs to be done and what has been done and what should be improved. Hardware team and software team can revise their parts according to the test results. Methods of doing each test is also recorded for convenience of understanding how tests approach. Thus it is easier for clients to comprehend this huge project by going through each small sections step by step.

1.2 Useful Tools for Testing

Following software tools will be used for testing:

Process	Tools
Test Creation	Microsoft Word
Data Collection	Draft Paper
Data Analysis	Microsoft Excel
Test Management	Microsoft Word

Table 1.2.1 Tools list

1.3 Testing Process

1.3.1 Hardware Component Testing

Hardware components that need to be used in final project design will be tested separately to decide each one's accuracy and its condition to work properly. For example, there are three ultrasonic sensors, three light sensors and three gyro sensors. Those are the sensors we are going to use in the final project. So it is necessary to test all of them and choose the relatively best one or two.

1.3.2 Software Component Testing

Testing on software part corresponds to hardware part. For each section, we will see what are the flaws for both hardware and software part. If the logic of code is right, then we will need to negotiate with hardware team to adjust the structure. If it is difficult to change the structure and there is leeway to change the code a bit to fit the hardware, then we will need

to negotiate with software team to tell them what we have seen during testing and give them advise on which part should be revised.

1.3.3 Overall Testing

If we accumulate all parts and let the robot run the code completely, then small errors will accumulate and form a large error. This is the problem that we might not foresee before. This error may finally lead to corruption of the whole project and fail. So it is important to overall testing to avoid such things taking place and allows to ensure all clients' requirements are met.

2. TEST REPORT TEMPLATE

Date: Year/Month/Day

Tester: Name

Author: Name

Hardware version: Hardware version number

Software version: Software version number

Goal: Purpose of this test

Procedure: Detailed information about initial conditions and steps to carry out the test

Expected Result: Ideal result of the test

Test Report: Detailed information about the data collected during the test. May need to use Excel to include all test data and show the name of that file. Also, observation of each test outcome should be included.

Conclusion: Check which part is incorrect comparing the actual outcome with the expected outcome

Action: Based on the result, decide which should be done to improve

Distribution: Indicate which team member should pay attention to the results and make modifications

3. HARDWARE TESTING

3.1 Light Sensor

Light Sensor Test

Date: 2019/3/12

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: N/A

Software Version: N/A

Goal: The objective of this test is to know the accuracy of each of the three light sensors provided. Depending on the comparison of errors, decide which one should be used for final design.

Procedure:

1. Place a red sheet on the board.
2. Place a ruler vertical to the board, and using a wall to help stabilize it.
3. Use a wire to connect the light sensor to Port 1 in EV3 brick.
4. Use the tool on EV3 brick to help testing. Select Tools-> Test Sensors-> Go-> Port 1-> EV3 Color-> Color ID.
5. Move up the light sensor slowly and record the value of height moved until the ID becomes 2.0.
6. Repeat step 5 until the ID becomes 7.0.
7. Repeat step 5 until the ID becomes -1.0.
8. Do the same procedure for the other 2 light sensors.

Expected Result: As the light sensor moves up, the ID value shown on the screen will decrease. The first and second distance measured correspond to the smallest and largest distance that a light sensor can detect color correctly. Furthermore, the third distance corresponds to the longest distance that a light sensor can detect an obstacle, but not able to distinguish the color of it.

Test Report:

Sensor	First distance(cm)	Second distance(cm)	Third distance(cm)
1	0.3	0.7	5.3

2	0.3	1.4	4.4
3	0.3	1.0	3.9

Table 3.1.1 Measured values

Conclusion: From the result, we can see that only when the distance is greater than 0.3 cm, the ultrasonic can distinguish the right color. But it can't be too far as well since otherwise it will detect it as an incorrect color. The limit is around 1.0 cm. Furthermore, if the distance is even farther and greater than around 4.5 cm, then it will not detect anything. Since all values don't differ too much and all three light sensors work well, there's no need to rank them, we just need to make sure the distance should between 0.3 cm to 0.7 cm to be safe.

Action: Make sure the distance between the sensor and the can is within 0.3 cm and 0.7 cm when the robot is rotating its light sensor to scan the can.

Distribution: Hardware team

3.2 Ultrasonic Sensor

Ultrasonic Sensor Test

Date: 2019/3/12

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: N/A

Software Version: N/A

Goal: The objective of this test is to know the accuracy of distance for each of the three ultrasonic sensors provided. Depending on the comparison of errors, decide which one should be used for final design.

Procedure:

1. Place the robot on the board.
2. Use a wire to connect the ultrasonic sensor to Port 1 in EV3 brick.
3. Use the tool on EV3 brick to help testing. Select Tools-> Test Sensors-> Go-> Port 1-> EV3 Ultrasonic -> Distance.
4. Move the ultrasonic sensor just behind the first black grid line and face towards the wall. Record the value shown on the EV3 brick.

5. Repeat step 4 by only changing moving the ultrasonic sensor to the second black grid line.
6. Repeat step 4 again by changing moving the ultrasonic sensor to the third black grid line.
7. Do the same procedure for the other two ultrasonic sensors.

Expected result:

The most ideal result is that the value shown on the EV3 brick will be the same as 30.48 (side length of a square) times number of grid lines. But there might be some unavoidable errors exist, so we can ignore the error if that is in tolerated range of 1.5 cm.

Test Report:

	Ultrasonic Sensor 1		Ultrasonic Sensor 2		Ultrasonic Sensor 3	
Real distance (cm)	Measured distance (cm)	Error (cm)	Measured distance (cm)	Error (cm)	Measured distance (cm)	Error (cm)
30.48	32.00	1.52	32.00	1.52	32.00	1.52
60.96	61.80	0.84	61.80	0.84	61.80	0.84
91.44	92.10	0.66	91.70	0.26	91.70	0.26
Average Error (cm)		1.01		0.87		0.87

Table 3.2.1 Measured values

Conclusion: All of three sensors have an average value of error less than 1.5 cm. So all three sensors are accessible, but compared to sensor 1, sensor 2 and 3 are a little better, which can slightly increase the accuracy of localization.

Action: Since one ultrasonic sensor is sufficient for this project, so either sensor 2 or sensor 3 is fine.

Distribution: Hardware team

3.3 Gyro Sensor

Gyro Sensor Test

Date: 2019/3/13

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: N/A

Software Version: N/A

Goal: The objective of this test is to know the accuracy of angle for each of the three gyro sensors provided. Depending on the comparison of errors, decide which one should be used for final design.

Procedure:

1. Place the robot on the board.
2. Use a wire to connect the gyro sensor to Port 2 in EV3 brick.
3. Use the tool on EV3 brick to help testing. Select Tools-> Test Sensors-> Go-> Port 2-> EV3 Gyro-> Angle.
4. Place the gyro sensor in one corner of a grid intersection.
5. Rotate it 360° and stops to record its value shown on EV3 brick every 90°.
6. Repeat step 5 for other two gyro sensors.

Expected result: The ideal result is that all measured values are the same as real values. But according to unavoidable errors, errors with in 3° is acceptable.

Test Report:

	Gyro Sensor 1		Gyro Sensor 2		Gyro Sensor 3	
Real angle (°)	Measured angle (°)	Error (°)	Measured angle (°)	Error (°)	Measured angle (°)	Error (°)
90.0	89.0	-1.0	90.0	0.0	89.0	-1.0
180.0	180.0	0.0	180.0	0.0	178.0	-2.0
270.0	270.0	0.0	270.0	0.0	267.0	-3.0
360.0	358.0	-2.0	361.0	1.0	355.0	-5.0

3.3.1 Measured values

Conclusion: Both sensor 1 and 2 works within the acceptable error range while sensor 3 is not working that well. Thus sensor 3 should be excluded for the project. And compared to sensor 1, sensor 2 works a little better, which may improve the accuracy a bit more on localization and travel.

Action: Use gyro sensor 2, if a second gyro sensor is needed, then add gyro sensor 1.

Distribution: Hardware team

3.4 Track

Track Value Test 1

Date: 2019/3/13

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 1.0

Software Version: N/A

Goal: The objective of this test is to determine what is the exact value of track that ensures the robot to turn exactly how much we want it to turn.

Procedure:

1. Measure the value track.
2. Place the robot on the board.
3. Place the robot in the corner of a grid line intersection.
4. Change the track input to what we just measured into Lab2's (let the robot move in a large square) code and run it.
5. Measure the angle error when it stops based on vertical grid line.
6. Repeat this procedure 2 times for the same track value.
7. Repeat all steps using several different track values until the error is acceptable.

Expected result: Track value should be close to the value measured. Under this condition, the angle turned at each corner should almost equal to 90° and the angle at ending position should around 360°, ±3° error in angle is allowed. This error might due to personal error, measurement and quantity of electric charge.

Test Report:

Test	Track value (cm)	Error (°)
1.	10.40	+15
2	10.40	+13
3.	10.30	+10

4.	10.30	+12
5.	10.00	+2
6.	10.00	+3
7.	9.95	-1
8.	9.95	+1
9.	9.96	+2
10.	9.96	+5
11.	9.94	-2
12.	9.94	-3

3.4.1 Measured values

Conclusion: Due to the fact that increase track value will let the robot turn more while decrease track value will let the robot turn less, we adjust the value of track gradually depending on the result of the first measured track. After comparison, we find that value should be around 9.95. Since the track value depends on quantity of electric charge and human error so much (this time is 7.7), so it is recommended to test again to check if it changes a bit or not before other tests.

Action: Report the relatively most accurate track value to the software team.

Distribution: Software team

Track Value Test 2

Date: 2019/3/17

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.0

Software Version: N/A

Goal: The objective of this test is to determine what is the exact value of track that ensures the robot to turn exactly how much we want it to turn. Since we change the structure a lot, so we have to measure the track value again, which has no relation to the previous measured value.

Procedure:

1. Measure the value track.
2. Clean the wheel surface.
3. Place the robot on the board.
4. Place the robot in the corner of a grid line intersection.
5. Change the track input to what we just measured and plug it into test code and run it.
6. Measure the angle error when it stops based on vertical grid line.
7. Repeat this procedure 2 times for the same track value.
8. Repeat all steps using several different track values until the error is acceptable.

Expected result: Track value should be close to the value measured. Under this condition, the angle turned at each corner should almost equal to 90° and the angle at ending position should around 360°, ±3° error in angle is allowed. This error might due to personal error, measurement and quantity of electric charge.

Test Report:

Test	Track value (cm)	Error (°)
1.	16.80	+5
2	16.80	+6
3.	16.00	-10
4.	16.00	-12
5.	16.50	-3
6.	16.50	-4
7.	16.60	-1
8.	16.60	-2
9.	16.65	+1
10.	16.65	+1
11.	16.62	+0.5
12.	16.62	+0.5
13.	16.61	-0.5
14.	16.61	-0.5
15.	16.615	+0
16.	16.615	+0

Table 3.4.2 Measured values

Conclusion: The value of track is tested again under the quantity of electric charge of 8.0 and with the wheels cleaned. The track value is more accurate because we can have the error angle within 1°, with the help of correction, we can make sure that the robot is not far away from the route which we expected.

Action: Report value 16.615 to the software team.

Distribution: Software team

4. SOFTWARE TESTING

4.1 Localization

Ultrasonic Localization Test

Date: 2019/3/21

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.0

Software Version: 1.0

Goal: The objective of this test is to ensure that the robot can localize using ultrasonic localization.

Procedure:

1. Place the robot on the 45° line at the corner of the board facing in any direction.
2. Run the test code and check where it is facing when the robot stops.
3. Measure the angle difference between the final orientation and the vertical angle.
4. Repeat step 2 and 3 using different starting angle or position.

Expected Results:

The most ideal result is that the robot is facing towards the positive axis of y-axis but an error of ±3° is tolerated.

Test Report:

	Falling Edge			Rising Edge		
Trial #	Expected angle (°)	Actual angle (°)	Error in angle (°)	Expected angle (°)	Actual angle (°)	Error in angle (°)
1	0.0	2.5	2.5	0.0	3.1	3.1
2	0.0	2.4	2.4	0.0	2.5	2.5
3	0.0	2.0	2.0	0.0	2.7	2.7
4	0.0	0.5	0.5	0.0	3.3	3.3
5	0.0	1.0	1.0	0.0	3.2	3.2
6	0.0	0.5	0.5	0.0	2.8	2.8
7	0.0	1.7	1.7	0.0	2.1	2.1
8	0.0	1.8	1.8	0.0	2.6	2.6

9	0.0	2.0	2.0	0.0	2.6	2.6
10	0.0	1.9	1.9	0.0	3.1	3.1

Table 4.1.1 Measured values

Conclusion:

Both rising edge and falling edge are in tolerance level, though rising edge has a bit more error than falling edge.

Action: Report the result to software team and tell them ultrasonic localization is good, there's no need to change it.

Distribution: Software team

Light Localization Test

Date: 2019/3/21

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.0

Software Version: 2.0

Goal: The objective of this test is to ensure that the robot can localize using light localization.

Procedure:

1. Place the robot on the 45° line at the corner of the board facing in any direction.
2. Run the test code and continue testing after it finishes ultrasonic localization.
3. Measure the angle difference between the final orientation and the vertical angle.
4. Repeat step 2 and 3 using different starting angle or position.
5. The error was calculated using the following formula, where x_f and y_f are the coordinates of the expected final position and x and y are the actual coordinates shown on the odometer.

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

6. The standard deviation is calculated using the formula below, we will need to use it for the error calculated above.

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

7. The final angle error is calculated using the actual angle minus expected angle which is 0°.

Expected Results:

The most ideal result is that the robot is facing towards the positive axis of y-axis and the center of two wheels is on the intersection of two grid lines. But an error of $\pm 3°$ and 2cm is tolerated.

Test Report:

Trial #	Expected X x_f (cm)	Expected Y y_f (cm)	Actual X x (cm)	Actual Y y (cm)	Euclidean error ε	Final angle error
1	0.0	0.0	0.4	0.2	0.447213595	1
2	0.0	0.0	0.4	0.7	0.806225775	1
3	0.0	0.0	0.6	0.1	0.608276253	11
4	0.0	0.0	0.7	0.3	0.761577311	3
5	0.0	0.0	0.3	0	0.3	0
6	0.0	0.0	0.5	0.2	0.538516481	6
7	0.0	0.0	0.2	0.1	0.223606798	3
8	0.0	0.0	0.2	0.1	0.223606798	5
9	0.0	0.0	0.4	0.5	0.640312424	2
10	0.0	0.0	0.6	0.3	0.670820393	1

Table 4.1.2 Measured values

	Mean	Standard deviation
Final angle error	4.8	2.658320272
Euclidean error ε	0.94191	0.60992273

Table 4.1.3 Mean and standard deviation

Conclusion:

Out of ten trials, the x and y error are acceptable but the angle error is above the tolerance level. However, most of the time the angle error is within the tolerance level, if we exclude value 11 which might be a special case due to some human made error, the mean is going to below 3.0, which is in the tolerance range.

Action: Report the result to software team and explain the special case to them.

Distribution: Software team

4.2 Navigation

Navigation Test (Normal)

Date: 2019/3/27

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.3

Software Version: 2.0

Objective: The objective of this test is to test the accuracy of navigation code.

Procedure:

1. Place the robot at point (0, 0).
2. Let the robot move to a certain point with the help of odometer correction.
3. Measure the real distance error between its actual coordinate and expected coordinate.
4. Repeat the above-mentioned steps 10 times.

Expected Results: The most ideal result is that the distance error is zero. But considering different errors, a tolerance of 3cm is accepted.

Test Report:

#Trail	Starting point	Destination	Actual destination	Horizontal distance error / cm	Vertical distance error / cm
1	(0, 0)	(1, 2)	(0.97, 2)	1.0	0.0
2	(0, 0)	(2, 3)	(1.92, 3)	2.3	0.0
3	(0, 0)	(2, 4)	(1.98, 4)	0.7	0.0

4	(0, 0)	(2, 5)	(2, 5)	0.0	0.0
5	(0, 0)	(2, 6)	(1.89, 6)	3.3	0.0
6	(0, 0)	(3, 3)	(2.96, 3)	1.2	0.0
7	(0, 0)	(3, 4)	(2.99, 4)	0.2	0.0
8	(0, 0)	(3, 5)	(3.00, 5)	-0.5	0.0
9	(0, 0)	(3, 6)	(2.96, 6)	1.2	0.0
10	(0, 0)	(3, 7)	(3.02, 7)	-0.7	0.0
11	(0, 0)	(5, 6)	(4.82, 6)	3.3	0.0

Table 4.2.1 Measured values

Conclusion:

Out of eleven trials, most of them are within the error of 3cm, so this is enough to navigate to some specified point with little error. And under this prerequisite, all other missions like travel through the tunnel, search and go back to the starting point can ideally process without large errors.

Action: There's no need to change the navigation method.

Distribution: Software team.

Navigation Test (Tunnel)

Date: 2019/3/27

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.3

Software Version: 2.0

Objective: The objective of this test is to test the accuracy of navigation code.

Procedure:

1. Place the robot at point (0, 0).
2. Let the robot move to the entrance of the tunnel with the help of odometer correction.
3. Let the robot move through the tunnel and see whether it passes or not and whether it collides or not.
4. Repeat the above-mentioned steps 10 times.

Expected result: The robot passes through the tunnel successfully.

Test Report:

#Trail	Starting point	Tunnel lower left corner	Tunnel upper right corner	Pass or not
1	(0, 0)	(3, 3)	(5, 2)	Pass
2	(0, 0)	(3, 4)	(5, 3)	Pass
3	(0, 0)	(3, 5)	(5, 4)	Pass
4	(0, 0)	(3, 6)	(5, 5)	Pass
5	(0, 0)	(4, 3)	(5, 5)	Pass
6	(0, 0)	(4, 4)	(6, 3)	Pass
7	(0, 0)	(4, 5)	(6, 4)	Pass
8	(0, 0)	(4, 6)	(5, 7)	Pass
9	(0, 0)	(5, 5)	(7, 4)	Pass
10	(0, 0)	(5, 6)	(6, 8)	Pass
11	(0, 0)	(6, 6)	(8, 5)	Pass

Table 4.2.2 Measured values

Conclusion:

Out of eleven trials, all of them successfully pass the tunnel. So going through the tunnel at any position is fine.

Action: There's no need to change the navigation method.

Distribution: Software team.

Navigation Test (Starting point)

Date: 2019/3/29

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.3

Software Version: 2.1

Objective: The objective of this test is to test the accuracy of navigation code.

Procedure:

1. Place the robot at a corner.
2. Let the robot move to the entrance of the tunnel with the help of odometer correction.
3. Let the robot move through the tunnel.
4. Let the robot move to the left corner of the search zone.
5. Repeat the above-mentioned steps 10 times.

Expected result: The robot passes through the tunnel successfully.

Test Report:

#Trail	Starting point	Corner	Search zone lower left corner	Bridge orientation	Arrived or not
1	(15, 0)	1	(12, 6)	Horizontal	Arrived
2	(15, 0)	1	(10, 7)	Vertical	Arrived
3	(0, 0)	0	(10, 5)	Horizontal	Arrived
4	(0, 0)	0	(9, 6)	Vertical	Arrived
5	(0, 9)	3	(9, 5)	Horizontal	Arrived
6	(0, 9)	3	(6, 3)	Vertical	Arrived
7	(15, 9)	2	(9, 6)	Horizontal	arrived
8	(15, 9)	2	(11, 4)	Vertical	Arrived

Table 4.2.3 Measured values

Conclusion:

Out of eight trials at all four corners, all of them successfully arrive the left corner of the search region.

Action: There's no need to change the navigation method.

Distribution: Software team.

4.3 Searching

Search Test

Date: 2019/4/2

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.1

Software Version: 2.2

Goal: The objective of this test is to make sure the claw is working for both grabbing and lifting.

Procedure:

1. Place the robot on the board.
2. Set the position of search area through Wi-Fi.
3. Place some random cans in random places in the search zone.
4. Set one color as target color and run the code.
5. Change the color of the target can and repeat the above-mentioned steps.

Expected Result: The robot can detect the right color can based on color calibration and make the right action afterwards.

Test Report:

Trial#	Target can	Detected or not	#Beeps	Action afterwards
1	Blue	Detected	1	Take the can back to the starting point
2	Blue	Detected	1	Take the can back to the starting point
3	Green	Detected	2	Take the can back to the starting point
4	Green	Detected	2	Take the can back to the starting point
5	Yellow	Detected	3	Take the can back to the starting point
6	Yellow	Detected	3	Take the can back to the starting point
7	Red	Detected	4	Take the can back to the starting point
8	Red	Detected	4	Take the can back to the starting point

Table 4.3.1 Measured values for target cans

Trial#	Target can	Actual can's color	Detected or not	Action afterwards
1	Blue	Green	Detected	Take the can out of the search zone
2	Blue	Red	Detected	Take the can out of the search zone
3	Green	Yellow	Detected	Take the can out of the search zone
4	Green	Red	Detected	Take the can out of the search zone
5	Yellow	Blue	Detected	Take the can out of the search zone
6	Yellow	Red	Detected	Take the can out of the search zone
7	Red	Blue	Detected	Take the can out of the search zone
8	Red	Green	Detected	Take the can out of the search zone

Table 4.3.2 Measured values for non-target cans

Conclusion: Now the robot can get the can it is supposed to get and remove those non-target cans to avoid re-detection, which is a save of time and easier for coding.

Action: Report the result to the software team and re-evaluate this method during integrated testing.

Distribution: Software team.

4.4 Color Calibration

Color Calibration of Four Colors

Date: 2019/3/11

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: N/A

Software Version: 1.0

Goal: The objective of this test is to know the range of values of R, G and B for all 4 different colors under RGB mode of light sensor, so that the robot can distinguish each can when detect it.

Procedure:

1. Place the robot in a position on the board.
2. Place a can right beneath the center of rotation of light sensor.
3. Ensure the distance from the light sensor to the can is 0.6cm.
4. Rotate the can several times until a cycle is completed.
5. Record values of R, G and B shown on EV3 for each turn.
6. Repeat the above procedure using another can until we have 30 groups of values for a can.
7. Calculate the Gaussian distribution of each normalized R, G and B value.

Expected Result: The robot can distinguish the color under all conditions.

Test Report:

Blue can:

	R	G	B	$\sqrt{R^2+G^2+B^2}$	$\hat{R}=\frac{R}{\sqrt{R^2+G^2+B^2}}$	$\hat{G}=\frac{G}{\sqrt{R^2+G^2+B^2}}$	$\hat{B}=\frac{B}{\sqrt{R^2+G^2+B^2}}$
1.	52.9	58.8	58.8	98.55602468	0.536750545	0.596614973	0.596614973
2.	64.7	66.7	51.9	106.4358492	0.607877895	0.626668557	0.487617662
3.	5.9	16.7	42.2	45.76614469	0.128916255	0.364898554	0.92207898
4.	15.7	55.9	50.9	77.21470067	0.203329157	0.723955406	0.659200898
5.	33.4	94.2	45.1	109.650399	0.304604455	0.859094001	0.411307213
6.	49.0	79.4	53.9	107.752355	0.454746442	0.736874846	0.500221086
7.	25.5	73.6	63.8	100.6858977	0.253262876	0.730986182	0.633653783
8.	41.2	77.5	77.5	117.089453	0.35186773	0.661887113	0.661887113
9.	20.6	79.6	43.1	92.8338839	0.221901736	0.857445543	0.464270137

10.	11.7	38.2	50.0	64.00101562	0.182809599	0.596865528	0.781237603
11.	10.8	26.5	48.0	55.88282384	0.193261529	0.47420653	0.858940131
12.	9.8	31.4	39.2	51.17264894	0.191508554	0.61360904	0.766034216
13.	62.7	74.5	60.8	114.7962543	0.546185068	0.648975879	0.529634006
14.	13.7	45.1	50.0	68.71462726	0.199375308	0.656337694	0.727647111
15.	23.5	37.3	59.8	74.29387593	0.316311401	0.502060224	0.804911566
16.	15.6	26.5	47.1	56.24962222	0.277335196	0.471114275	0.837338957
17.	38.2	31.4	48.0	68.91443971	0.554310536	0.455637456	0.696515857
18.	51.7	72.6	52.0	103.1874508	0.501029918	0.703573927	0.503937248
19.	26.5	55.9	53.9	82.05041133	0.322972177	0.68128848	0.656913221
20.	13.7	33.3	51.9	63.16795073	0.216882135	0.527166065	0.821619182
21.	28.4	56.9	56.8	85.26669924	0.333072586	0.667317962	0.666145172
22.	29.4	44.1	49.0	72.18150733	0.40730654	0.61095981	0.678844233
23.	11.7	25.5	50.0	57.33358527	0.204068871	0.444765487	0.872089191
24.	25.5	41.1	55.9	73.92070075	0.344964262	0.556001223	0.756215775
25.	59.8	53.9	59.8	100.2860409	0.596294354	0.537462637	0.596294354
26.	59.9	60.8	62.8	105.9645696	0.565283285	0.57377669	0.592650923
27.	13.7	45.1	50.0	68.71462726	0.199375308	0.656337694	0.727647111
28.	9.8	19.6	44.1	49.24439054	0.199007438	0.398014876	0.895533471
29.	16.7	27.4	48.0	57.73776927	0.289238746	0.47455938	0.8313449
30.	28.4	55.9	51.0	80.82307839	0.351384785	0.691634136	0.631007888
Mean μ	29.0	50.2	52.5		0.33517449	0.603336339	0.685645132
			Gaussian Distribution σ		0.142979122	0.120943592	0.135881179
			2σ		0.285958244	0.241887184	0.271762358

Table 4.4.1 All values of blue can

Green can:

	R	G	B	$\sqrt{R^2+G^2+B^2}$	$\hat{R}=\frac{R}{\sqrt{R^2+G^2+B^2}}$	$\hat{G}=\frac{G}{\sqrt{R^2+G^2+B^2}}$	$\hat{B}=\frac{B}{\sqrt{R^2+G^2+B^2}}$
1.	47.0	89.2	36.3	107.1603005	0.438595261	0.832397815	0.338744851

2.	51.9	111.7	35.3	128.1272414	0.405066084	0.871789627	0.275507375
3.	45.1	120.6	22.5	130.7081482	0.345043524	0.922666274	0.17213923
4.	64.8	79.4	41.2	110.4574126	0.586651439	0.718829077	0.372994433
5.	77.5	85.2	50.0	125.559906	0.61723525	0.678560559	0.39821629
6.	75.5	91.2	42.2	125.6922034	0.600673693	0.725581997	0.335740793
7.	16.7	60.8	15.7	64.97707288	0.257013732	0.935714665	0.241623688
8.	17.7	70.6	15.7	74.458982	0.237714773	0.948173049	0.210854347
9.	63.7	85.3	32.3	111.2522809	0.572572531	0.766725853	0.290331126
10.	69.6	85.3	33.3	115.0179986	0.605122684	0.741623059	0.289519905
11.	69.6	87.3	37.3	117.714655	0.591260281	0.741623887	0.316867938
12.	16.7	68.7	15.7	72.42285551	0.23059019	0.948595571	0.216782394
13.	23.5	73.5	31.4	83.30942324	0.282080935	0.882253137	0.376908143
14.	11.7	39.2	10.8	42.31040061	0.276527753	0.926486146	0.255256387
15.	12.7	52.9	11.8	55.66812373	0.228137741	0.950274528	0.2119705
16.	31.3	75.5	25.5	85.61652878	0.365583614	0.881839069	0.297839685
17.	57.8	91.2	34.3	113.2906439	0.510192175	0.805009106	0.3027611
18.	18.6	77.5	14.7	81.0450492	0.229501989	0.956258288	0.181380604
19.	56.8	79.4	44.1	107.1233401	0.530229919	0.741201683	0.41167499
20.	48.0	95.1	41.2	114.21668	0.420253854	0.832627949	0.360717892
21.	13.7	43.1	12.8	47.00148934	0.291480125	0.916992219	0.272331796
22.	57.8	95.1	48.0	121.197566	0.476907267	0.784669224	0.396047558
23.	42.1	83.3	34.3	99.43736722	0.423382086	0.837713249	0.34494075
24.	62.7	82.4	38.1	110.3297784	0.568296256	0.746851858	0.345328347
25.	25.5	75.4	22.6	82.74158567	0.308188437	0.911270909	0.273139556
26.	44.1	69.6	35.3	89.63849619	0.491976125	0.776452115	0.393804018
27.	15.7	69.6	15.7	73.05573215	0.214904423	0.952697317	0.214904423
28.	13.7	41.2	13.8	45.55842403	0.300712772	0.904333301	0.302907756
29.	61.7	76.5	53.9	112.0908114	0.550446546	0.682482347	0.480860111
30.	18.6	50.9	19.6	57.62751079	0.32276251	0.883258695	0.340115333

Mean μ	41.1	76.9	29.5		0.409303466	0.840165086	0.307407044
			Gaussian Distribution σ		0.136964609	0.088912326	0.073685176
			2σ		0.273929217	0.177824652	0.147370352

Table 4.4.2 All values of green can

Yellow can:

	R	G	B	$\sqrt{R^2+G^2+B^2}$	$\hat{R}=\frac{R}{\sqrt{R^2+G^2+B^2}}$	$\hat{G}=\frac{G}{\sqrt{\square^2+G^2+B^2}}$	$\hat{B}=\frac{B}{\sqrt{R^2+G^2+B^2}}$
1.	144.1	77.5	17.7	164.573236	0.875598023	0.470914967	0.107550902
2.	143.1	71.6	15.7	160.7814044	0.8900283	0.445325131	0.097648108
3.	139.2	56.9	17.7	151.4184269	0.919306869	0.375779891	0.116894623
4.	122.6	56.9	13.7	135.8530824	0.902445479	0.41883481	0.100844234
5.	125.5	66.7	15.7	142.9882163	0.877694703	0.466472005	0.109799258
6.	120.6	60.8	13.7	135.7523112	0.888382665	0.447874511	0.100919092
7.	116.7	45.1	12.8	125.7646214	0.927923916	0.358606415	0.10177743
8.	127.5	65.7	15.7	144.2887036	0.883645059	0.455337101	0.108809627
9.	131.3	73.6	15.7	151.337834	0.867595343	0.486329149	0.103741408
10.	141.2	63.7	16.7	155.8012195	0.906283022	0.408854309	0.107187864
11.	141.2	70.6	14.7	158.5493299	0.890574562	0.445287281	0.092715624
12.	175.5	101.9	17.7	203.7084927	0.8615252	0.500224604	0.086888866
13.	131.4	75.5	16.7	152.4634382	0.861845971	0.495200691	0.109534458
14.	147.1	82.3	18.6	169.5808362	0.867432921	0.485314272	0.109682205
15.	140.2	119.6	16.7	185.0380772	0.75768189	0.646353452	0.090251694
16.	150.9	77.5	17.7	170.5589341	0.884738175	0.454388393	0.103776446
17.	151.0	77.4	17.7	170.6020223	0.88510088	0.453687471	0.103750236
18.	143.1	109.8	23.5	181.8952996	0.786716316	0.603643966	0.129195202
19.	139.2	59.8	18.6	152.6389203	0.911956136	0.39177426	0.121856208
20.	133.4	90.2	23.6	162.7530645	0.819646625	0.554213835	0.14500495
21.	137.3	72.6	17.7	156.3180732	0.878337336	0.464437659	0.113230669

22.	141.2	60.8	14.7	154.4350025	0.9143005	0.393693133	0.095185675
23.	250.0	85.2	32.3	266.0870722	0.939542075	0.320195939	0.121388836
24.	167.6	85.2	20.6	189.1379391	0.88612576	0.450464885	0.108915219
25.	169.7	83.3	22.6	190.3883925	0.891335852	0.437526673	0.118704716
26.	155.9	73.5	18.6	173.3580687	0.899294744	0.423977958	0.107292381
27.	164.7	75.4	23.5	182.656782	0.901691129	0.41279606	0.128656597
28.	165.7	85.3	23.5	187.8425671	0.882121676	0.454103675	0.125104764
29.	168.6	69.6	24.5	184.0390448	0.916109949	0.37818062	0.133123925
30.	133.3	70.6	15.7	151.6566517	0.878959139	0.465525245	0.103523319
Mean μ	144.1	77.5	17.7	164.573236	0.875598023	0.470914967	0.107550902
			Gaussian Distribution σ		0.037265405	0.065474555	0.013064422
			2σ		0.07453081	0.130949111	0.026128844

Table 4.4.3 All values of yellow can

Red can:

	R	G	B	$\sqrt{R^2+G^2+B^2}$	$\hat{R}=\frac{R}{\sqrt{R^2+G^2+B^2}}$	$\hat{G}=\frac{G}{\sqrt{R^2+G^2+B^2}}$	$\hat{B}=\frac{B}{\sqrt{R^2+G^2+B^2}}$
1.	145.2	15.7	11.8	146.5222509	0.990975767	0.107150961	0.080533843
2.	165.7	15.7	15.7	167.1809499	0.991141635	0.093910221	0.093910221
3.	187.2	12.8	29.4	189.9264068	0.98564493	0.067394525	0.1547968
4.	150.9	9.8	13.8	151.8462709	0.993768231	0.064538957	0.090881389
5.	150.9	14.3	11.8	152.0346671	0.992536787	0.094057495	0.077613877
6.	155.9	14.7	10.7	156.9566501	0.993267886	0.093656433	0.068171689
7.	154.9	17.6	12.7	156.4131069	0.990326214	0.11252254	0.081195242
8.	149.0	17.6	12.7	150.5724078	0.989557132	0.116887285	0.084344802
9.	161.7	20.6	14.7	163.6683842	0.987973339	0.12586426	0.089815758
10.	155.9	14.7	14.7	157.2799733	0.991226008	0.093463902	0.093463902
11.	141.2	8.8	12.8	142.0518215	0.994003445	0.061949223	0.090107961
12.	140.2	8.8	12.7	141.0488213	0.993982074	0.062389745	0.090039746

13.	200.0	8.8	28.4	202.1979228	0.989129845	0.043521713	0.140456438
14.	150.9	13.7	11.7	151.971675	0.992948193	0.090148378	0.076988031
15.	167.7	31.4	14.7	171.2464306	0.979290484	0.183361486	0.085841205
16.	160.7	16.7	10.7	161.9193318	0.992469511	0.103137777	0.066082289
17.	172.5	15.7	15.6	173.9140592	0.991869207	0.090274473	0.089699476
18.	181.4	17.7	17.7	183.1189231	0.990613078	0.096658498	0.096658498
19.	166.7	21.6	14.7	168.7351179	0.987938979	0.128011289	0.087118794
20.	169.5	26.4	16.7	172.3545764	0.983437769	0.153172608	0.096893279
21.	166.7	21.6	13.7	168.6509413	0.988432076	0.128075182	0.08123287
22.	129.4	13.7	9.8	130.4917239	0.991633769	0.104987501	0.075100548
23.	115.7	8.8	8.8	116.3673923	0.994264782	0.075622559	0.075622559
24.	143.1	13.7	11.7	144.2296433	0.992167745	0.615030373	0.525244917
25.	121.6	10.8	9.8	122.4713844	0.992884996	0.088183865	0.080018692
26.	118.7	13.8	9.8	119.9006672	0.989986151	0.115095273	0.081734324
27.	118.8	12.8	9.8	119.8887818	0.990918401	0.106765619	0.081742427
28.	135.3	14.7	10.8	136.5240638	0.99103408	0.107673326	0.079106933
29.	123.6	13.0	8.8	124.5929372	0.99203055	0.104339783	0.070630007
30.	115.7	16.7	8.8	117.2297744	0.986950633	0.142455277	0.075066254
Mean μ	146.2	15.4	13.5		0.990413457	0.119010018	0.102003759
			Gaussian Distribution σ		0.003257299	0.09636383	0.094796834
			2σ		0.006514597	0.192727659	0.189593667

Table 4.4.4 All values of red can

	Blue can	Green can	Yellow can	Red can
Mean μ	$\hat{R}$: 0.33517449 $\hat{G}$: 0.60333634 $\hat{B}$: 0.68564513	$\hat{R}$: 0.40930347 $\hat{G}$: 0.84016509 B: 0.30740704	$\hat{R}$: 0.87895914 $\hat{G}$: 0.46552525 $\hat{B}$: 0.10352332	$\hat{R}$: 0.99041346 $\hat{G}$: 0.10167525 $\hat{B}$: 0.08719962

Standard deviation σ	$\hat{R}$: 0.14297912 $\hat{G}$: 0.12094359 $\hat{B}$: 0.13588118	$\hat{R}$: 0.13696446 $\hat{G}$: 0.08891233 $\hat{B}$: 0.07368518	$\hat{R}$: 0.0372654 $\hat{G}$: 0.06547456 $\hat{B}$: 0.01306442	$\hat{R}$: 0.00325799 $\hat{G}$: 0.09636383 $\hat{B}$: 0.09479683

Table 4.4.5 Summary of useful values listed above

Sample:	Blue $\hat{R}$	Blue $\hat{G}$	Blue $\hat{B}$	Green $\hat{R}$	Green $\hat{G}$	Green $\hat{B}$	Yellow $\hat{R}$	Yellow $\hat{G}$	Yellow $\hat{B}$	Red $\hat{R}$	Red $\hat{G}$	Red $\hat{B}$
Total Number	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000	30.000
Max Value	0.608	0.859	0.922	0.617	0.956	0.481	0.940	0.646	0.145	0.994	0.183	0.155
Min Value	0.129	0.365	0.411	0.215	0.679	0.172	0.758	0.320	0.087	0.979	0.044	0.066
Range	0.479	0.494	0.511	0.139	0.090	0.075	0.182	0.326	0.058	0.015	0.140	0.089
Upper Limit	0.814	1.098	1.196	0.688	1.021	0.532	1.064	0.778	0.168	1.005	0.242	0.176
Lower Limit	0.129	0.365	0.411	0.131	0.659	0.158	0.758	0.320	0.087	0.979	-0.096	-0.002
Span	0.685	0.733	0.785	0.557	0.362	0.375	0.306	0.458	0.081	0.026	0.338	0.177
Number of Cylinder	6.000	6.000	6.000	6.000	6.000	6.000	6.000	6.000	6.000	6.000	6.000	6.000
Span of Cylinder	0.125	0.134	0.143	0.023	0.066	0.068	0.056	0.084	0.015	0.005	0.062	0.032

Table 4.4.6 All extra values needed to use to build a Gaussian distribution

The below tables and diagrams are for the sample listed above from left to right.

Group Statistics	Number of Frequency	Rate of Frequency
0.129	1.000	0.986

0.272	11.000	2.530
0.415	10.000	2.389
0.558	5.000	0.830
0.701	3.000	0.106
0.844	0.000	0.005

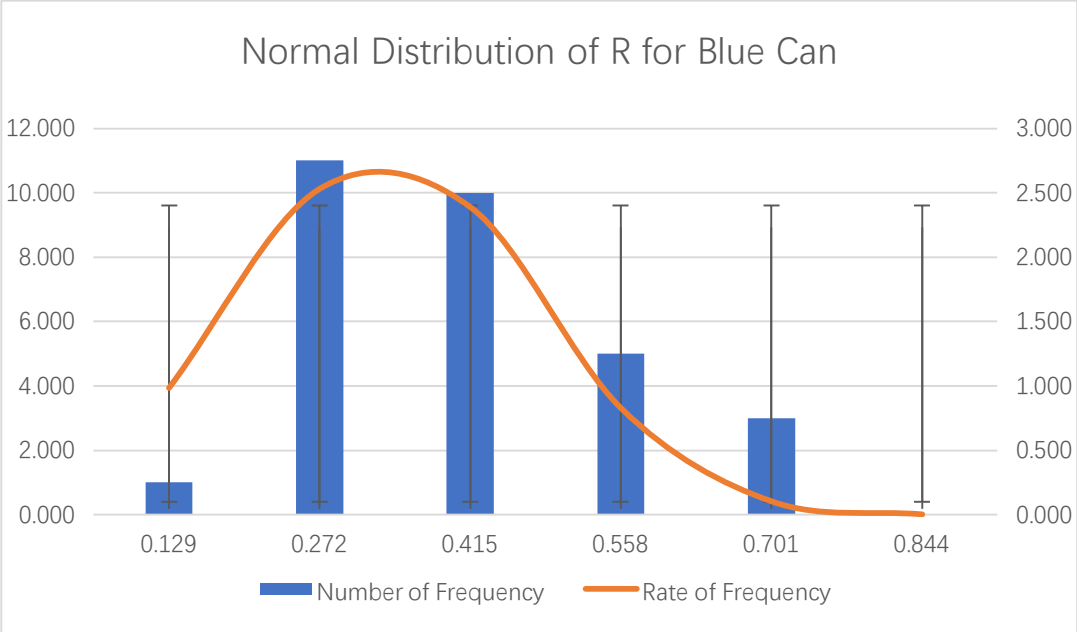


Figure 4.4.1: Normal Distribution of R for Blue Can

Group Statistics	Number of Frequency	Rate of Frequency
0.365	1.000	0.472
0.486	6.000	2.058
0.607	7.000	3.297
0.728	12.000	1.944
0.849	2.000	0.421
0.970	2.000	0.034

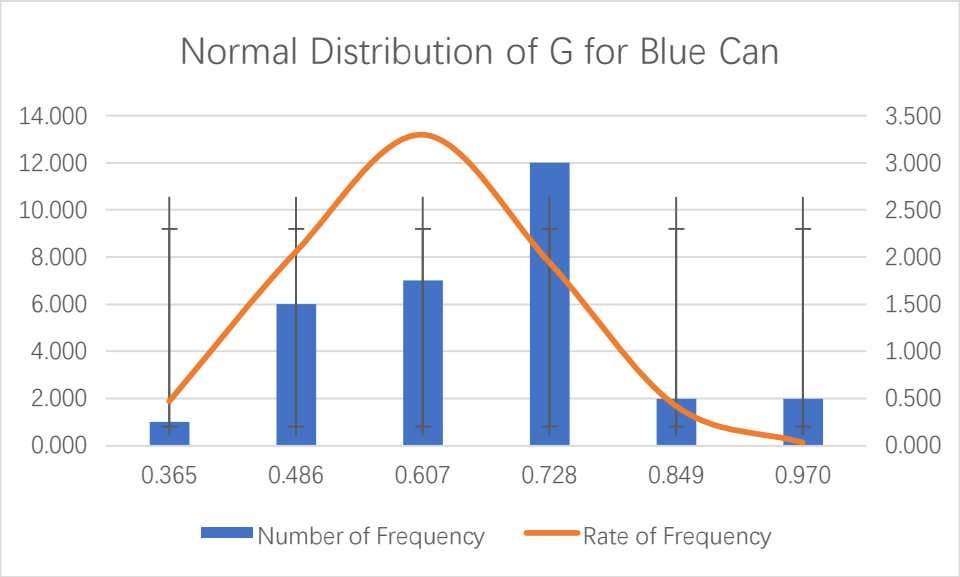


Figure 4.4.2: Normal Distribution of G for Blue Can

Group Statistics	Number of Frequency	Rate of Frequency
0.411	1.000	0.382
0.547	5.000	1.747
0.683	10.000	2.935
0.819	7.000	1.815
0.955	7.000	0.413
1.091	0.000	0.035

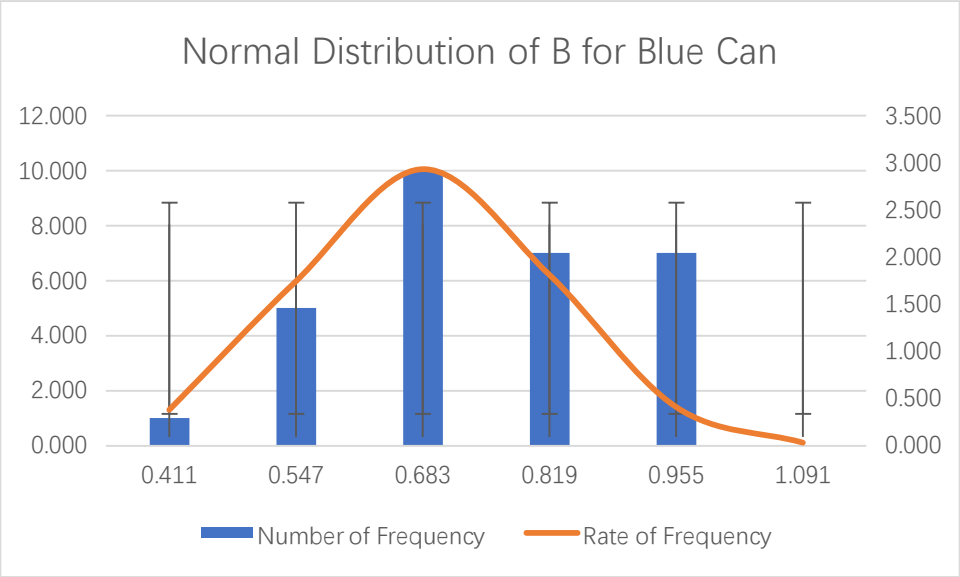


Figure 4.4.3: Normal Distribution of B for Blue Can

Group Statistics	Number of Frequency	Rate of Frequency
0.131	0.000	0.368
0.268	6.000	1.706
0.405	8.000	2.911
0.542	8.000	1.827
0.679	8.000	0.422
0.816	0.000	0.036

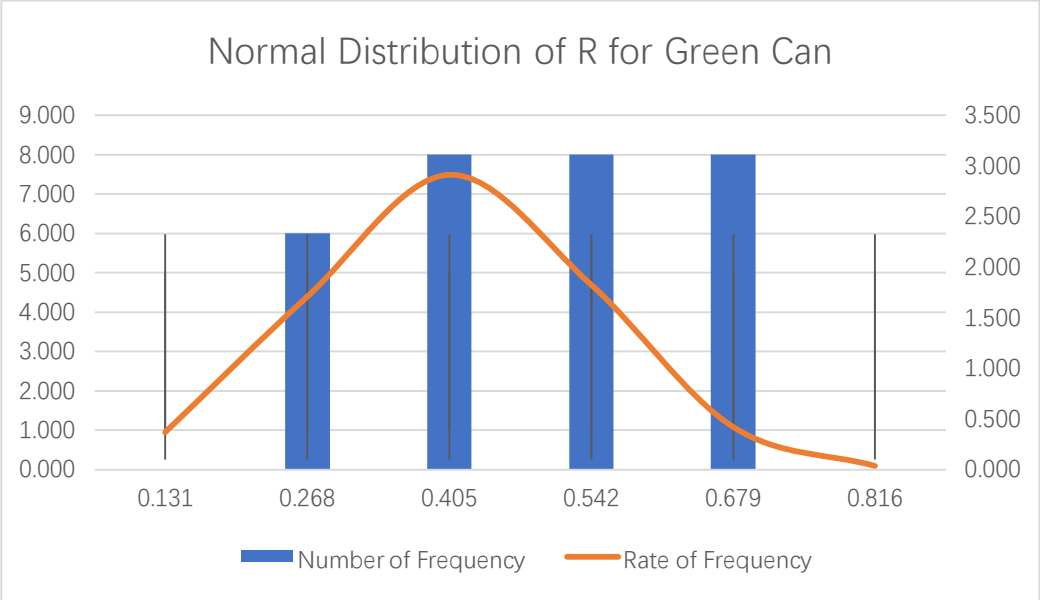


Figure 4.4.4: Normal Distribution of R for Green Can

Group Statistics	Number of Frequency	Rate of Frequency
0.659	0.000	0.567
0.748	8.000	2.628
0.837	6.000	4.484
0.926	9.000	2.814
1.015	7.000	0.650
1.104	0.000	0.055

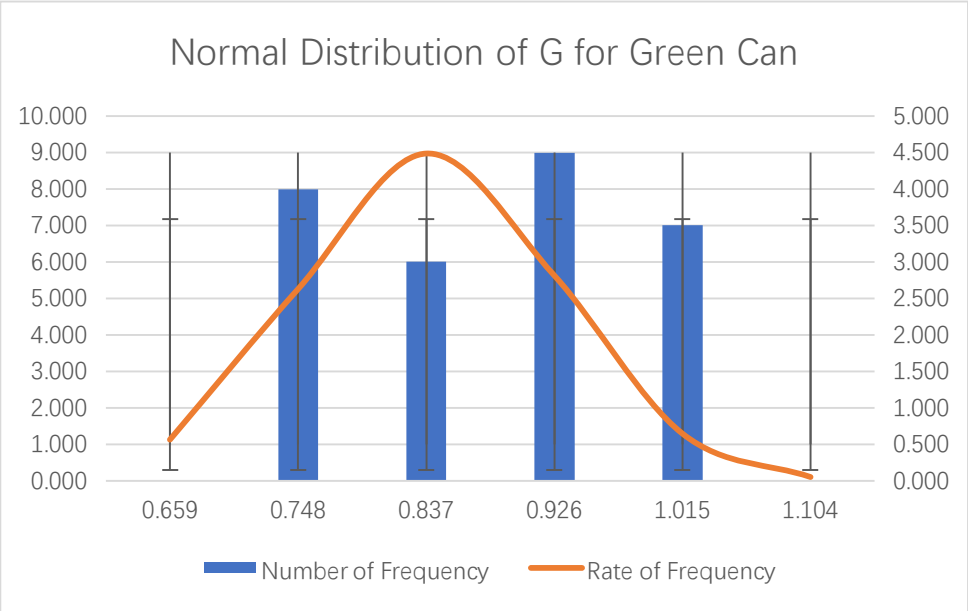


Figure 4.4.5: Normal Distribution of G for Green Can

Group Statistics	Number of Frequency	Rate of Frequency
0.158	0.000	0.684
0.231	6.000	3.172
0.305	10.000	5.411
0.379	9.000	3.396
0.452	4.000	0.784
0.526	1.000	0.067

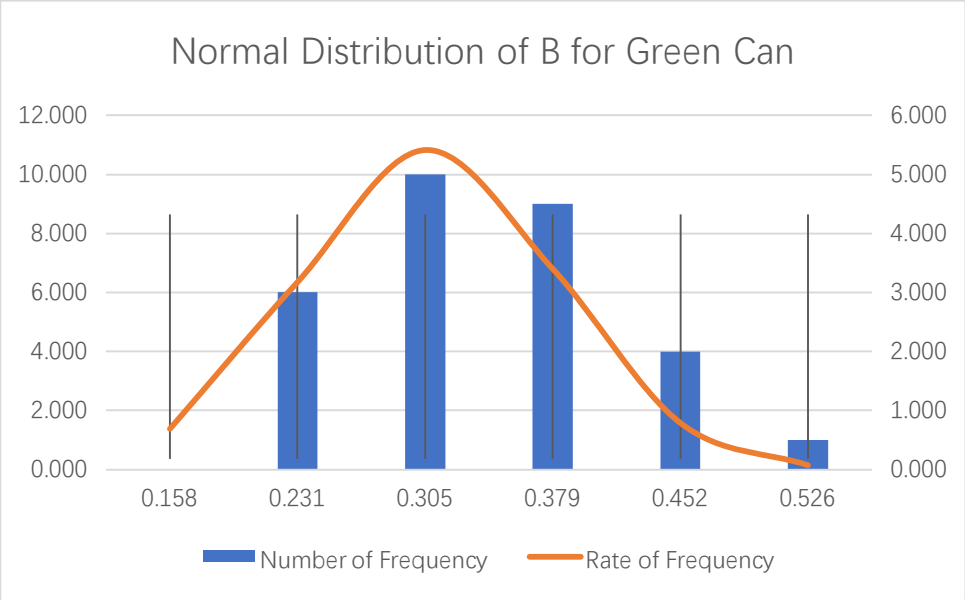


Figure 4.4.6: Normal Distribution of B for Green Can

Group Statistics	Number of Frequency	Rate of Frequency
0.758	1.000	0.042
0.795	1.000	0.708
0.832	1.000	4.417
0.869	4.000	10.136
0.907	17.000	8.557
0.944	6.000	2.657

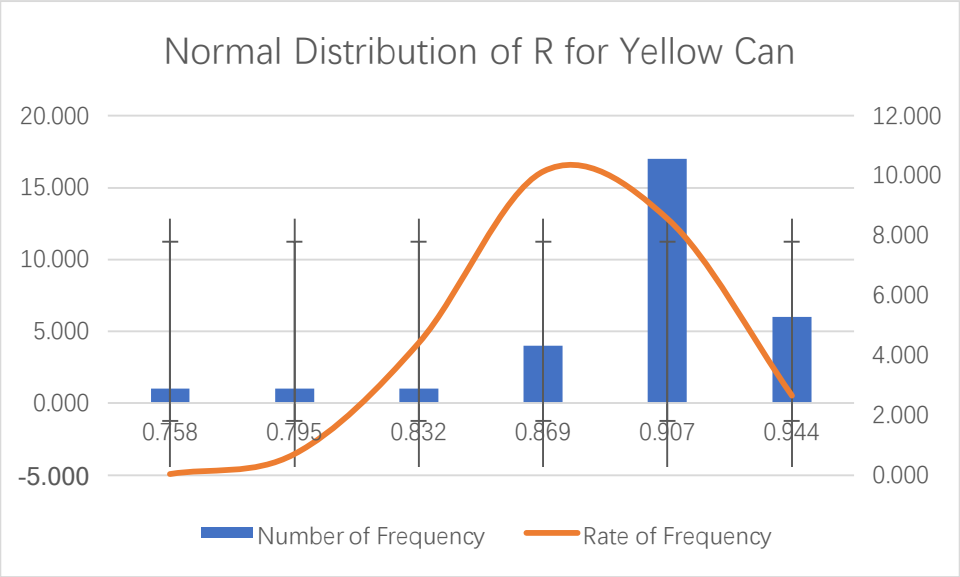


Figure 4.4.7: Normal Distribution of R for Yellow Can

Group Statistics	Number of Frequency	Rate of Frequency
0.320	1.000	0.799
0.386	3.000	3.637
0.451	11.000	6.092
0.517	12.000	3.754
0.582	1.000	0.851
0.648	2.000	0.071

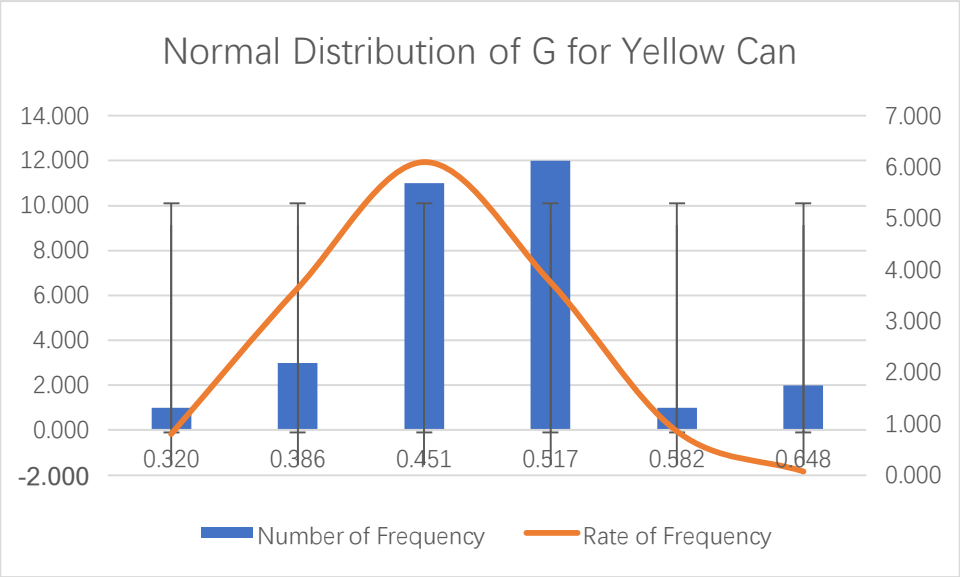


Figure 4.4.8: Normal Distribution of G for Yellow Can

Group Statistics	Number of Frequency	Rate of Frequency
0.087	1.000	6.302
0.100	4.000	22.588
0.113	15.000	29.784
0.126	6.000	14.447
0.139	3.000	2.578
0.152	1.000	0.169

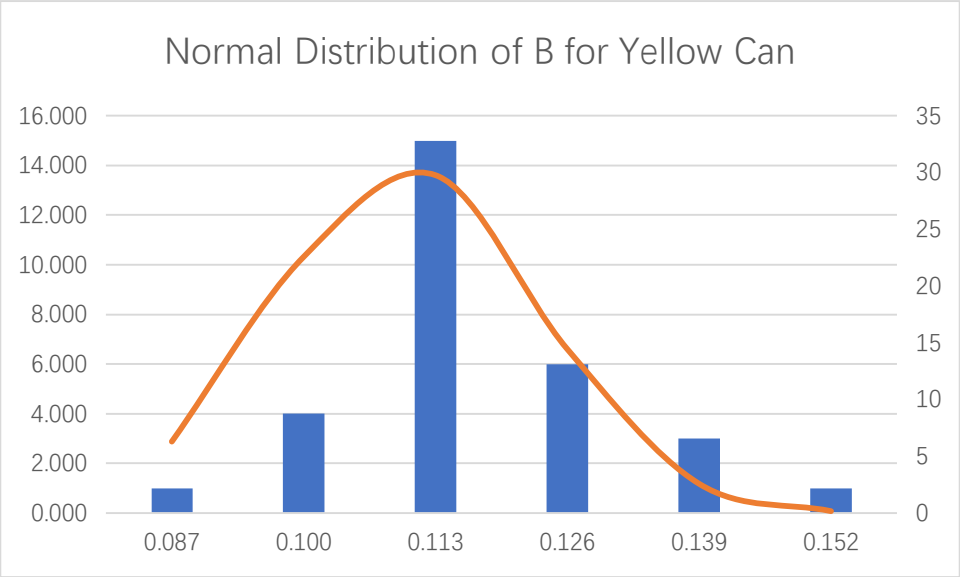


Figure 4.4.9: Normal Distribution of B for Yellow Can

Group Statistics	Number of Frequency	Rate of Frequency
0.979	1.000	3.60E-01
0.983	0.000	6.635
0.986	2.000	45.020
0.989	4.000	112.381
0.992	14.000	103.201
0.996	9.000	34.864

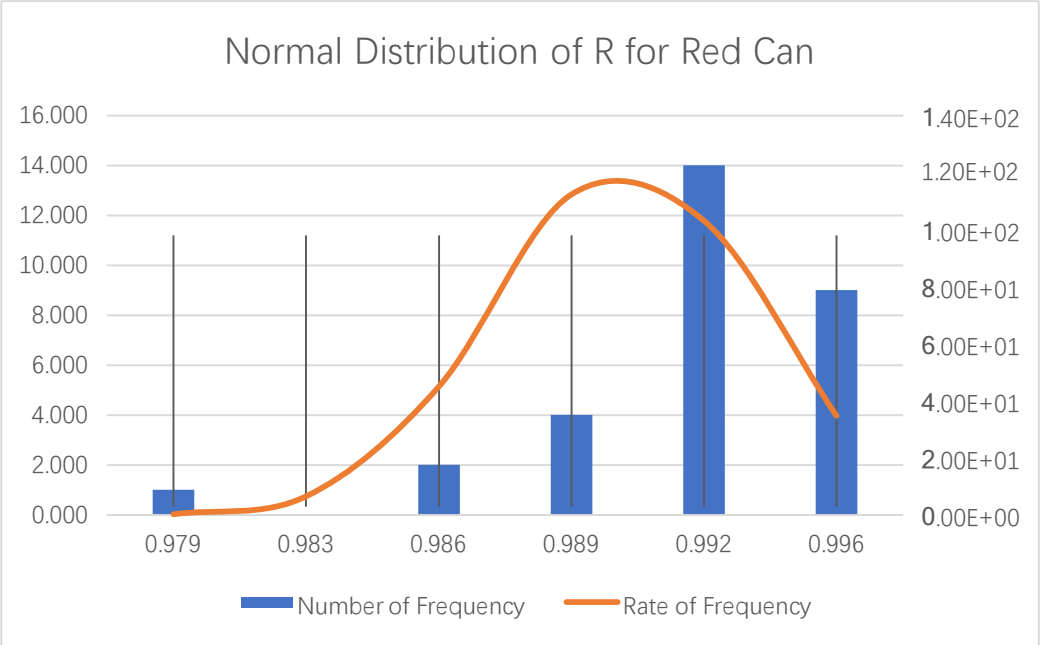


Figure 4.4.10: Normal Distribution of R for Red Can

Group Statistics	Number of Frequency	Rate of Frequency
-0.096	0.000	0.502
0.000	0.000	2.374
0.096	14.000	4.134
0.193	16.000	2.648
0.289	0.000	0.624
0.386	0.000	0.054

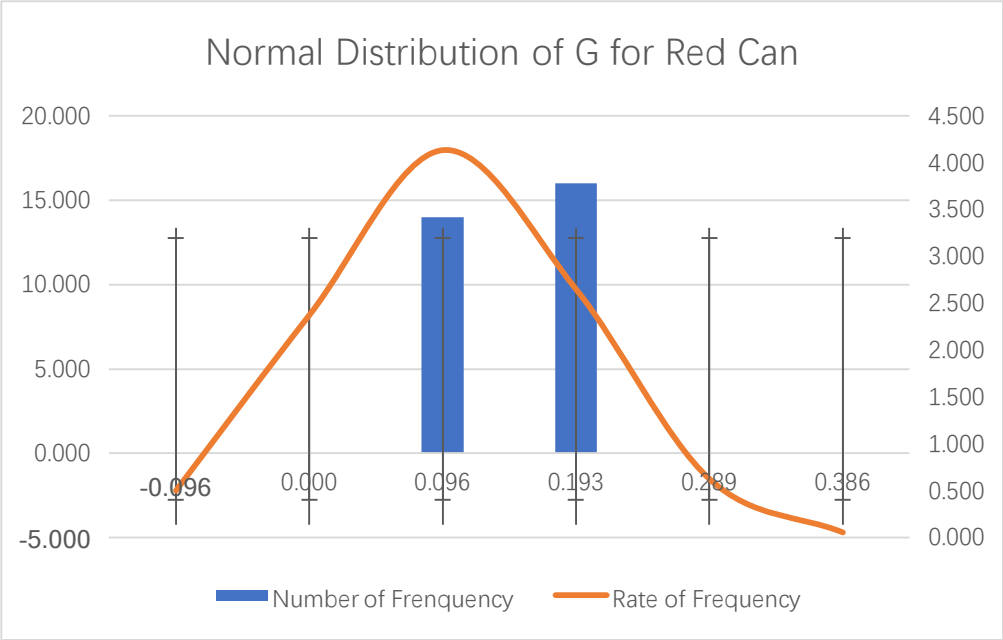


Figure 4.4.11: Normal Distribution of G for Red Can

Group Statistics	Number of Frequency	Rate of Frequency
-0.002	0.000	2.716
0.093	24.000	4.200
0.188	6.000	2.389
0.283	0.000	0.500
0.378	0.000	0.038
0.472	0.000	0.001

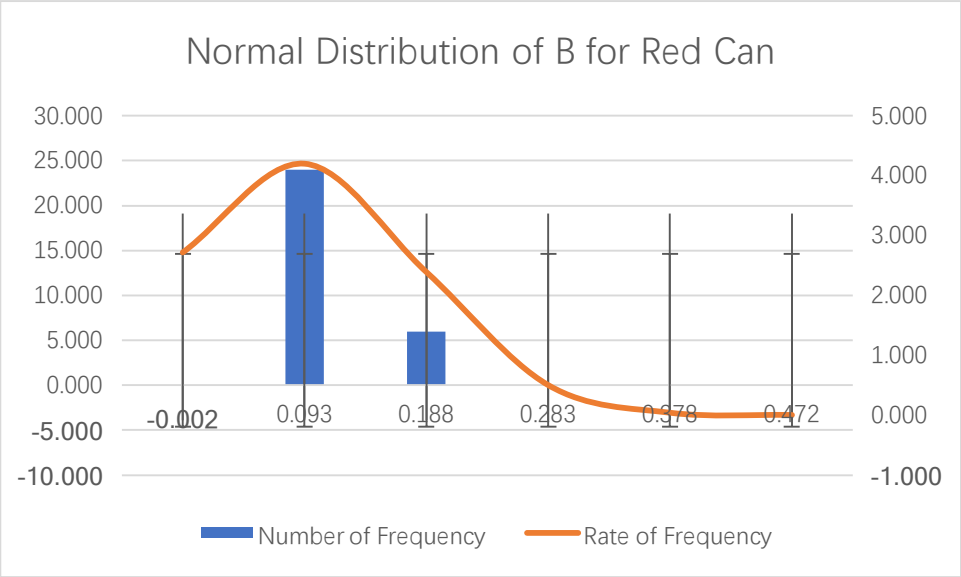


Figure 4.4.12: Normal Distribution of B for Red Can

Conclusion: Although there still might exist some errors, this way of calculation can minimize the error caused by environment.

Action: Report useful values to software team and notify them values need to be updated.

Distribution: Software team

4.5 Grabbing and Lifting

Grab and Lift Test

Date: 2019/4/2

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.1

Software Version: 2.2

Goal: The objective of this test is to make sure the claw is working for both grabbing and lifting.

Procedure:

1. Place the robot on the board.
2. Place a can right beneath the center of rotation of light sensor.
3. Run the code for the motor of claw.

4. Check if the can is grabbed and lifted.
5. Repeated the steps above using different weight of cans.

Expected result: The robot can lift any can weight of can.

Test Report: The robot can grab all weight of cans, but it can only lift those light cans.

Conclusion: Currently the claw structure is not suitable for lifting.

Action: Report the result to the hardware team.

Distribution: Hardware team.

4.6 Weight Measurement

Weight Measurement Test 1

Date: 2019/4/4

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.3

Software Version: N/A

Goal: The objective of this test is to make sure the robot can successfully distinguish light can from heavy can.

Procedure:

1. Place the robot on the board.
2. Run the testing code using debug mode.
3. Then record the median value shown in Eclipse.
4. Repeat this process 10 times for both light can and heavy can.

Expected Result: The logic of testing code is to let the claw close and lift the can up and then open three times. Record 12300 values of current flow in the motor in an array and use quicksort to make a sorted array. Then we find the 6150^{th} number which is the median number. The expected result is that the median number for heavy can is greater than that for light can.

Test Report:

Trial#	Median value for light can	Median value for heavy can
1	0.31908831	0.325600326

2	0.312576324	0.481888503
3	0.306064308	0.397232383
4	0.488400489	0.31908831
5	0.306064308	0.332112342
6	0.31908831	0.35164836
7	0.306064308	0.35164836
8	0.306064308	0.325600326
9	0.299552292	0.306064308
10	0.358160347	0.312576324

Table 4.6.1 Measured values

Conclusion: Out of ten values for light can, eight of them have value below 0.32 and out of ten values for heavy can, seven of them have values above 0.32. So we can assume that if the value is below 0.32, it is a light can and if the value is above 0.32, it is a heavy can.

Action: Report the value to the software team.

Distribution: Software team.

Weight Measurement Test 2

Date: 2019/4/4

Tester: Elias Mouannes

Author: Hanwen Wang

Hardware Version: 2.3

Software Version: 2.4

Goal: The objective of this test is to make sure the robot can successfully distinguish light can from heavy can and make improvements compared to the former tested one.

Procedure:

1. Place the robot on the board.
2. Run the testing code using debug mode.
3. Then record the median value shown in Eclipse.
4. Repeat this process 10 times for both light can and heavy can.

Expected Result: This time the main procedure is the same as the former one, the only

thing that is changed is we change the rotation times for the motor when the claw opens. We expect this time there is a larger difference in values between light and heavy cans.

Test Report:

Trial#	Median value for light can	Median value for heavy can
1	0.338624328	0.501424491
2	0.325600326	0.520960509
3	0.357560090	0.560032546
4	0.348474908	0.449328452
5	0.455291507	0.462352484
6	0.319985276	0.478776891
7	0.355908702	0.409600785
8	0.418530789	0.491458765
9	0.349609827	0.519678354
10	0.334378213	0.536012586

Table 4.6.2 Measured values

Conclusion: This time the difference is very clear, out of ten trials, eight of the them has values below 0.4 for the light cans and eight of them has values greater than 0.46 for the heavy cans. This probability of success is greater than the former test. So we set the boundary as 0.46. If the values are greater than 0.46, it is considered as a heavy can, otherwise it is considered as a light can.

Action: Report the value to the software team.

Distribution: Software team.

4.7 Wi-Fi Connection

Wi-Fi Connection Test

Date: 2019/3/18

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.0

Software Version: 2.0

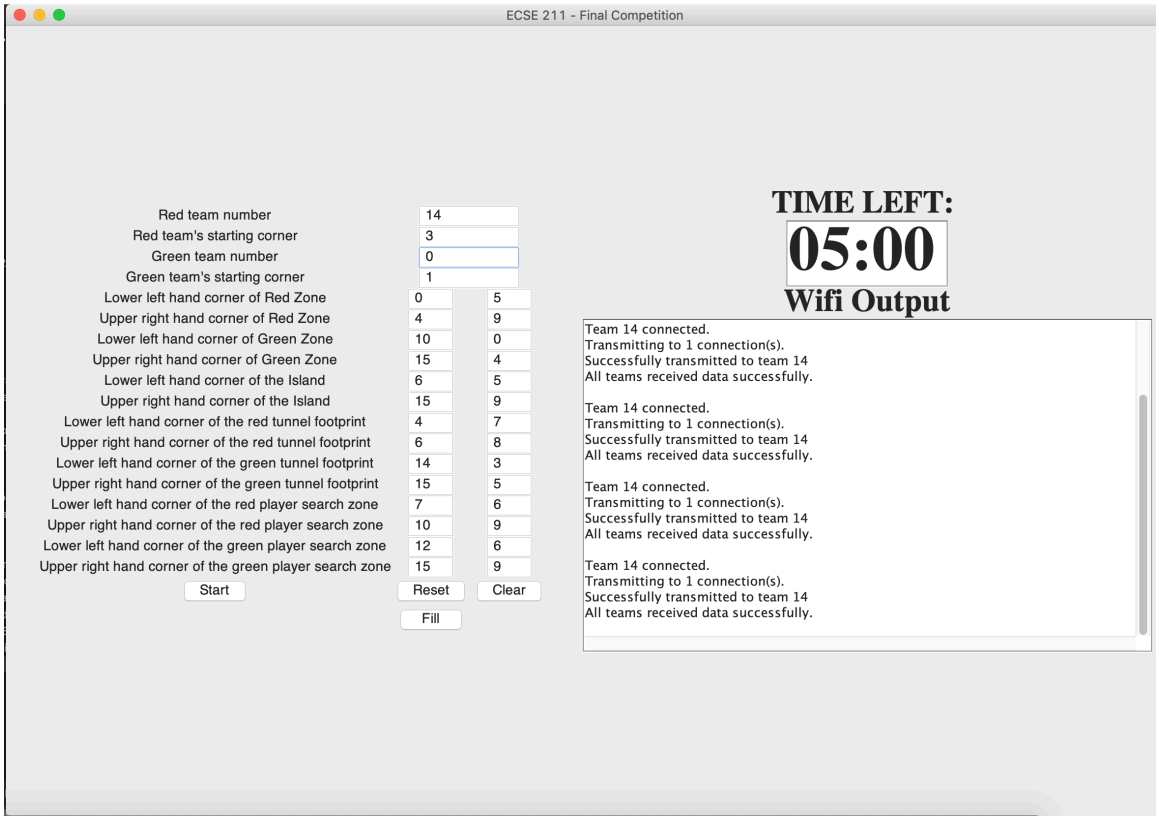
Goal: The objective of this test is to ensure the robot can connect to professor's laptop on Beta demo and receive correct data for our own use.

Procedure:

1. Follow the instructions on DPM_Wifi_Guide.pdf, which professor posted on mycourses.
2. Change the IP address in example code.
3. Run the server program by double clicking DPM Server Winter 2019.jar in the folder professor gave us.
4. Run the code.
5. Fill in the data using fill button and select a file called example_data_fill.xml.
6. Change red team number to 14 and green team number to 0.
7. Press start button.
8. Repeat step 4 to 7 for different values.

Expected result: The robot receives data filled and moves to the position it is ordered to.

Test Report: The robot correctly navigates to the place where it is asked to for all data.



Conclusion: We can now receive the data so we do not need to worry about not connecting to professors' laptops during Beta demo or final competition.

Action: Communicate with software team and ask them to keep current Wi-Fi class remain unchanged.

Distribution: Software Team

5. OVERALL TESTING

Integration Test 1

Date: 2019/4/3

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.3

Software Version: 2.4

Goal: The objective of this test is to make sure the robot can work under integrated code. Small mistakes should be pointed out. And we should make sure every detail meets the requirement of clients.

Procedure:

1. Place the robot on the board.
2. Upload all the values listed in the power point professor posted on mycourses.
3. Set our team as red team and run the code.
4. Waiting for team connected in DPM Server Winter 2019.jar.
5. Press start button when connected.
6. Supervise the whole process.
7. Test again using different target can.
8. Repeat the whole process above using different starting corner.

Test Report:

Trial#	Starting corner	Localization	Beep three times	Through the tunnel	Reach search zone	Beep three times	Get the can	Return
1	3	Success	Yes	Success	Success	Yes	Yes	No
2	3	Success	Yes	Success	Success	Yes	Yes	No
3	3	Success	Yes	Success	Success	Yes	Yes	No
4	3	Success	Yes	Success	Success	Yes	Yes	No
5	3	Success	Yes	Success	Success	Yes	Yes	No
6	1	Success	Yes	Success	Success	Yes	Yes	No
7	1	Success	Yes	Success	Success	Yes	Yes	No

8	1	Success	Yes	Success	Success	Yes	Yes	No
9	1	Success	Yes	Success	Success	Yes	Yes	No
10	1	Success	Yes	Success	Success	Yes	Yes	No

Table 5.1 Tested Result

Conclusion: The method of returning to the starting point is not working. And since the time limit is five minutes. There is a potential problem that the time will exceed the time limit. And this version of code has not included the weight measurement, so the robot cannot make the appropriate types of beep required though the number is correct. In addition, we have to press start button twice before the robot actually run because we receive two messages of Team 14 connected. This is an issue need to be fixed because this is not allowed in the final competition.

Action: Report the result to the software team. And supervise them to add the weight measurement method into the integrated code.

Distribution: Software team.

Integration Test 2

Date: 2019/4/9

Tester: Hanwen Wang

Author: Hanwen Wang

Hardware Version: 2.3

Software Version: 2.5

Goal: The objective of this test is to make sure the robot can complete weighing the can and go back to the original point now. And we should make sure every detail meets the requirement of clients.

Procedure:

1. Place the robot on the board.
2. Upload all the values listed in the power point professor posted on mycourses.
3. Set our team as red team and run the code.
4. Waiting for team connected in DPM Server Winter 2019.jar.
5. Press start button when connected.

6. Supervise the whole process.
7. Test again using different target can.
8. Repeat the whole process above using different starting corner.

Test Report:

Trial#	Starting corner	Localization	Beep three times	Through the tunnel	Reach search zone	Beep three times	Get the can	Make the right beep	Retu
1	3	Success	Yes	Success	Success	Yes	Yes	Yes	Yes
2	3	Success	Yes	Success	Success	Yes	Yes	Yes	No
3	3	Success	Yes	Success	Success	Yes	Yes	Yes	Yes
4	3	Success	Yes	Success	Success	Yes	Yes	No	Yes
5	3	Success	Yes	Success	Success	Yes	Yes	Yes	No
6	1	Success	Yes	Success	Success	Yes	Yes	Yes	Yes
7	1	Success	Yes	Success	Success	Yes	Yes	Yes	No
8	1	Success	Yes	Success	Success	Yes	Yes	No	Yes
9	1	Success	Yes	Success	Success	Yes	Yes	Yes	Yes
10	1	Success	Yes	Success	Success	Yes	Yes	Yes	Yes

Table 5.1 Tested Result

Conclusion: Based on what we have for the former integrated test, this time we include weight measurement and returning to the starting point and both have a high probability of success. But the time spent is always exceeding 5 minutes. It is not able to finish all the requirements within the limited time.

Action: Report the result to the software team. And accelerate the speed of motors if we can.

Distribution: Software team.