Diagnostic Testing

Testing Table

Test/calculation	Purpose of the test/calculation and procedural steps.
Buzzer Functionality Video of result: Buzzer v1.MOV	Purpose of test: This test was conducted to see if the buzzer's sound output is working properly Equipment required: The Arduino Uno with its components and breadboard, and the Circuito.io test code. Procedural steps for the test: 1. Connect all components to the breadboard. 2. Connect Arduino Uno to the computer. 3. Run test code. 4. Type "1" in the console Results: The buzzer was silent when the code was run. Review: From this point, as the there are no code errors, I will firstly replace the buzzer. If that still does not work, I will examine the connections of the circuit.

Buzzer Functionality v2 Video of result: Buzzer v2.MOV	Purpose of test: This test was conducted to see if the buzzer's sound output is working properly Equipment required: The Arduino Uno with its components and breadboard, and the Circuito.io test code. Procedural steps for the test: 1. Connect all components to the breadboard. 2. Connect Arduino Uno to the computer. 3. Change the Arduino port from 3 to 2. 4. Run test code. 5. Type “1” in the console Results: The buzzer sounded clicks at 0.5 second intervals. Review: From this point, as the there are no component errors regarding the buzzer, I will continue to test the other components.
HC-SR04 Functionality (Ultrasonic Sensor) Video of result: HC-SR04 v1.MOV	Purpose of test: This test was conducted to see if the distance reading output from the HC-SR04 is correct Equipment required: The Arduino Uno with its components and breadboard, and the Circuito.io test code. Procedural steps for the test: 1. Connect all components to the breadboard. 2. Connect Arduino Uno to the computer. 3. Place a ruler in front of the sensor as a measurement indicator. 4. Run test code. 5. Type “3” in the console Results: The HC-SR04 outputted the correct values Review:

	From this point, as the there are no component errors regarding the HC-SR04, I will continue to test the other components.
RGB LED Functionality Video of result: RGB LED v1.MOV	Purpose of test: This test was conducted to see if the LED can turn on as well as change its colour output. Equipment required: The Arduino Uno with its components and breadboard, and the Circuito.io test code. Procedural steps for the test: 1. Connect all components to the breadboard. 2. Connect Arduino Uno to the computer. 3. Run test code. 4. Type "4" in the console Results: The RGB LED did not output a light Review: From this point, as the there are no code errors, I will firstly replace the LED. If that still does not work, I will examine the connections of the circuit.
RGB LED Functionality v2 Video of result: RGB LED v2.MOV	Purpose of test: This test was conducted to see if the LED can turn on as well as change its colour output. Equipment required: The Arduino Uno with its components and breadboard, and the Circuito.io test code. Procedural steps for the test: 1. Connect all components to the breadboard. 2. Connect Arduino Uno to the computer. 3. Flip the RGB LED pins and connect the port 11 wire to the parallel breadboard row. 4. Run test code. 5. Type "4" in the console Results:

	The RGB LED turned on (without console command) and cycles through green and blue Review: From this point, as the there are no component errors regarding the RGB LED, I will continue to test the other components.
Button Functionality Video of result: Buttons v1.MOV	Purpose of test: This test was conducted to see if the button will output a signal when it is pressed. Equipment required: The Arduino Uno with its components and breadboard, and the Circuito.io test code. Procedural steps for the test: 6. Connect all components to the breadboard. 7. Connect Arduino Uno to the computer. 8. Run test code. 9. Type “5” in the console (Button 1) 10. Type “6” in the console (Button 2) 11. Type “7” in the console (Button 3) Results: All buttons returned a ‘1’ in the serial monitor. Note: a ‘0’ indicates the button is not being pressed Review: From this point, as the there are no component errors regarding the buttons, I will continue to test the other components.
L298N Functionality Video of result: L298N v1.MOV	Purpose of test: This test was conducted to see if the L298N will spin the DC Motors at a given speed and direction. The use of an L298N is required for the direction to be controlled. By using PWM ports on the Arduino Uno, the speed of the DC Motors are able to be controlled in one direction however, the L298N chip has a H gate which allows for a 2 direction spinning. Equipment required: The Arduino Uno with its components and breadboard, L298N with DC motors and the Circuito.io test code. Procedural steps for the test:

Commented [IW1]: explain the need for using the L298 Board

	1. Connect all components to the breadboard. 2. Connect L298N to Arduino Uno PWM ports. 3. Connect Arduino Uno to the computer. 4. Run test code. 5. Type “2” in the console Results: Right DC motor (connected to Motor B ports) started turning first at which Motor B followed shortly after. Also, it was noted that the rotation speed of the DC motors was inconsistent. Review: From this point, as the motors are spinning at a low performance, I will try to incorporate another power supply as it seems that the Arduino Uno cannot provide enough power to both the 5V and VMS ports.
L298N Functionality v2 Video of result: L298N v2.MOV	Purpose of test: This test was conducted to see if the L298N will spin the DC Motors at a given speed. Equipment required: The Arduino Uno with its components and breadboard, L298N with DC motors, Power Supply Module (PSM) and the Circuito.io test code. Procedural steps for the test: 6. Connect all components to the breadboard. 7. Connect L298N to Arduino Uno PWM ports. 8. Connect PSM to L298N VMS port and plug 9V Battery into component. 9. Connect Arduino Uno to the computer. 10. Run test code. 11. Type “2” in the console Results: Both DC motors started spinning at the same time with the same speed. Review: From this point, as the there are no component errors regarding the L298N and its DC motors, I will continue to test the other components.

Pinboard Functionality Video of result: Pinboard Testing.MOV	Purpose of test: This test was conducted to see if the pinboard has successfully replaced the breadboard Equipment required: The Arduino Uno with its components and pinboard, L298N with DC motors, Power Supply Module (PSM) and the Circuito.io test code. Procedural steps for the test: 1. Connect L298N to Arduino Uno PWM ports. 2. Connect PSM to L298N VMS port and plug 9V Battery into PSM. 3. Plug another 9V Battery into Arduino 4. Connect Arduino Uno to the computer. 5. Run test code. 6. Type “1” in the console (Buzzer) 7. Type “2” in the console (L298N Motor Driver) 8. Type “3” in the console (Ultrasonic Sensor – HC-SR04) 9. Type “5” in the console (Mini Pushbutton Switch #1) 10. Type “6” in the console (Mini Pushbutton Switch #2) 11. Type “7” in the console (Mini Pushbutton Switch #3) Results: All components were functioning as expecting however, the DC motors did not spin. The L298N lit up (indicating the ports were being used to receive instructions from the Arduino) however, the DC motors did not spin, instead a beep-like sound was emitted for the duration that the DC motors were expected to spin. Review: From this point, I will continue to test the L298N and its DC motors and figure out what the error is.
L298N Functionality Video of result: L298N v3.MOV	Purpose of test: This test was conducted to figure out why the DC motors are not spinning Equipment required: The Arduino Uno, pinboard, L298N with DC motors, Power Supply Module (PSM) and the Circuito.io test code.

	<u>Procedural steps for the test:</u> 1. Connect L298N to Arduino Uno PWM ports. 2. Connect PSM to L298N VMS port and plug 9V Battery into PSM. 3. Plug another 9V Battery into Arduino 4. Connect Arduino Uno to the computer. 5. Run test code. 6. Type “2” in the console (L298N Motor Driver) <u>Results:</u> When active, both the Motor A and Motor B ports on the L298N released a voltage value of 0.25 ±0.05V. <u>Review:</u> The L298N was working fine on the breadboard. From this point, I will visually inspect the connections on the pinboard to locate a potential ‘short-circuit’ then, I will continue to test the L298N and its DC motors and figure out what the error is.
L298N Functionality <u>Video of result:</u> L298N v4.MOV	<u>Purpose of test:</u> This test was conducted to observe the L298N performance with a 9V power supply <u>Equipment required:</u> The Arduino Uno, pinboard, L298N with DC motors, 9V Battery and the Circuito.io test code. <u>Procedural steps for the test:</u> 1. Connect 9V Battery to L298N VMS port. 2. Connect Arduino Uno to the computer. 3. Run test code. 4. Type “2” in the console (L298N Motor Driver) <u>Results:</u> When active, both the Motor A and Motor B ports spin. <u>Review:</u> From this point, as the there are no component errors regarding the L298N and its DC motors, I will continue to test the other components.

TCS230 Functionality Video of result: TCS230 v1.MOV	Purpose of test: This test was conducted to observe the colour reading nature of the TCS230 Equipment required: The entire construction (including TCS230 and Arduino Uno) and the TCS230 colour frequency code. Procedural steps for the test: 1. Connect Arduino Uno to the computer. 2. Run test code. 3. Open serial monitor 4. Observe value outputs. Results: The red, green, and blue frequency values fluctuate, however there is a clear pattern, the lowest value is the color that is being picked up. This mean she TCS230 and the code are working properly as colors can be distinguished by using the lowest value. Review: From this point, as the there are no component errors regarding the TCS230, I will continue to test the other components.
TCS230 Colour Detection Functionality Video of result: TCS230 v2.MOV	Purpose of test: This test was conducted to ensure that the TCS230 would be able to accurately identify the colour that is directly under the sensor (referring to pre coded colours such as red, green and blue) Equipment required: The entire construction (including TCS230 and Arduino Uno), white sheet with both red and blue tape and, the TCS230 colour sensing code. Procedural steps for the test: 1. Connect Arduino Uno to the computer. 2. Run TCS230 colour sensing code. 3. Open serial monitor 4. Place down white sheet 5. Place TCS230 over red tape

	6. Observe value outputs. 7. Place TCS230 over blue tape 8. Observe value outputs. <u>Results:</u> When the TCS230 is placed above the red tape, the serial monitor returns “Red Detected”, showing that the correct color is detected. Once the TCS230 is no longer presented with red tape, both its value and output changes. This is also the same for the blue tape. However, there is a problem with the code knowing when it is not looking at the line anymore (specifically the blue line). Since the RGB value of white meets the criteria for blue, both colors (blue and white) are seen and read as blue. This creates problems for line differentiation as if set to blue, the TCS230 will still believe it is always following the line. <u>Review:</u> A physical solution to this problem would be to cover the sides of the blue tape with red tape, however this is not cost efficient. From this point, as the TCS230 is unable to identify the colour/shade white, I will incorporate an if statement in the code that should be able to pick up on white values.
TCS230 Colour Detection Functionality <u>Video of result:</u> <u>TCS230 v3.MOV</u>	<u>Purpose of test:</u> This test was conducted to ensure that the TCS230 would be able to accurately identify the colour that is directly under the sensor (referring to pre coded colours such as red, green, blue and white) <u>Equipment required:</u> The entire construction (including TCS230 and Arduino Uno), white sheet with both red and blue tape and, the <u>TCS230 colour sensing code</u>. <u>Procedural steps for the test:</u> 1. Connect Arduino Uno to the computer. 2. Run TCS230 colour sensing code. 3. Open serial monitor 4. Place down white sheet 5. Place TCS230 over red tape 6. Observe value outputs. 7. Place TCS230 over white space 8. Observe value outputs.

	Results: When the TCS230 is placed above the red tape, the serial monitor returns "Red Detected", showing that the correct color is detected. Once the TCS230 is no longer presented with red tape, the serial monitor now returns "White Detected". Review: From this point, as the there are no component errors regarding the TCS230, I will continue to develop my code to account for line following.
HC-SR04 Functionality Video of result: HC-SR04 v2.MOV	Purpose of test: This test was conducted to see if the distance recorded by the HC-SR04 is able to accurately alter the speed of the DC motors in real time. Equipment required: The entire construction (including HC-SR04 and Arduino Uno). Procedural steps for the test: 1. Connect Arduino Uno to the computer. 2. Run main code. 3. Open serial monitor 4. Move object towards HC-SR04 5. Observe value outputs. Results: By default, the value of speed is 200 as the distance is greater than 20. When the object (hand) moves closer to the sensor, despite there being code to change the speed based off increments of five, the speed is only changed to account for a distance level of 4. There are 5 levels of speed, 1 being close and speed = 0 and 5 where distance is far and speed = 200. Review: From this point, as the code is only reactive at 15 cm or higher, I will alter the code to incorporate all levels of distance.
HC-SR04 Functionality	Purpose of test: This test was conducted to see if the distance recorded by the HC-SR04 can accurately alter the speed of the DC motors in real time.

Video of result: HC-SR04 v3.MOV	Equipment required: The entire construction (including HC-SR04 and Arduino Uno). Procedural steps for the test: 1. Connect Arduino Uno to the computer. 2. Run main code. 3. Open serial monitor 4. Move object towards HC-SR04 5. Observe value outputs. Results: By default, the value of speed is 200 as the distance is greater than 20. When the object (hand) moves closer to the sensor, the DC motor speed changes in increments of 50, from 200 to 0. At less than or equal to 20cm, the speed is set to 150. At less than or equal to 15cm, the speed is set to 100. At less than or equal to 10cm, the speed is set to 50. Finally, at less than or equal to 5cm, the speed is set to 0. Review: From this point, as the code is changing the speed variables based off the distance readings, I will move onto testing a different subsystem. Also noted, when the L298N was powered on, the speed of the DC motors did change in real time. This was not shown in the video as it was not needed; only the value of the speed variable is needed.
Buzzer Functionality with Colour Detection Video of result: Buzzer v3.MOV	Purpose of test: This test was conducted to see if the buzzer will sound when the colour sensor is sensing white and if the buzzer will stop buzzing once the TCS230 is sensing a line. Equipment required: The entire construction (including Buzzer, TCS230 and Arduino Uno) and white sheet with both red and blue tape. Procedural steps for the test: 1. Connect Arduino Uno to the computer. 2. Run main code. 3. Place TCS230 over red tape

	4. Observe noise outputs. 5. Place TCS230 over white paper 6. Observe noise outputs. 7. Place TCS230 over blue tape 8. Observe noise outputs. Results: The buzzer was silent when the TCS230 was reading both the red and blue colored tape however, the buzzer let out a constant noise when it was presented with the RGB values of white. Review: From this point, as the buzzer is sounding only when is sees white RGB values, I will move onto testing a different subsystem.
L298N Functionality with HC-SR04 Video of result: L298N v5.MOV	Purpose of test: This test was conducted to see if the distance recorded by the HC-SR04 can accurately alter the speed of the DC motors in real time Equipment required: The entire construction (including HC-SR04 and Arduino Uno). Procedural steps for the test: 1. Connect Arduino Uno to the computer. 2. Run main code. 3. Open serial monitor 4. Move object towards HC-SR04 5. Observe value outputs. Results: When the red button is pressed the L298N spins the wheels at a speed of 200. However, the previously ‘distance alters speed’ system no longer works for the speed. The serial monitors output the speed variable and shows that it is actively changing based off the distance however, this is not mirrored in the actual spinning speed. It was also sound the repressing of the button did not turn off the motors as coded, it simply reruns the initial push button if statement. On top of that, the speed at which the motor spins are decided by what the current reading is as the button is pressed. This can be seen at the end where the hand is

	smothering the sensor, causing a speed of 0 which results in the DC motors to stop spinning as the button is pressed. Review: From this point, as the DC motor speed does not active change when it is triggered by the pressing of a button, I will continue to experiment with the arrangement of code to find why the executed operations are not updating the variables.
L298N Functionality with TCS230 Video of result: Evaluation Video 1.MOV	Purpose of test: This test was conducted to see if the colour detected by the TCS230 can accurately alter the rotation of the DC motors in real time to guide the robot along the coloured path Equipment required: The entire construction (including TCS230 and Arduino Uno). Procedural steps for the test: 1. Connect Arduino Uno to the computer. 2. Run main code. 3. Place TCS230 sensor over red line 4. Press button 2 (red) 5. Observe physical outputs (wheel direction) 6. Reset the Arduino Uno 7. Place TCS230 sensor over blue colour 8. Press button 2 (red) 9. Observe physical outputs (wheel direction) Results: When the TCS230 is detecting the red line and button 2 is pressed (red), The L298N spins both DC motors forward at a speed of 200. However, once the robot ventures off the line, it still moves forward and does not execute the secondary while loop which allows it to spin left to find the line once again. Furthermore, the opposite was tested. When the TCS230 is detecting a color other than red and button 2 is pressed (red), The L298N spins both DC motors in opposite directions at a speed of 200 to spin left. However, once the robot eventually detects the red line again, it still spins left in a never-ending cycle and does not execute the primary while loop which allows it to move forward when the line is being detected.

	Review: From this point, as the DC motor rotation does not actively change according to the colour detected by the TCS230, I will continue to experiment with the arrangement of code to find why the executed operations are not updating the variables that influence the direction of the DC motors