Test Plan:

Automatic and Semi-Automatic Tests:

1. **Circular Buffer FIFO Testing**: The CBFIFO is rigorously tested with the **automated test suite** to make sure it is solid and does not let the UART based command processor down by dropping useful information or not picking up useful information.

2. **LED Testing:** The LED is tested by making it light up in a particular sequence and making sure it matches the pre-defined sequence. The sequence is White -> Red -> Green -> Blue. I am using the systick timer for achieving this. Once all the LEDs are lit-up, the PWM testing starts. Slow transitions from one color to another is tested now. The transition sequence is as follows: White -> Red -> Green -> Blue -> White -> Red. All the LED testing is **semi-automatic** as it requires the user to check if the sequence is correct. In this test, there is no way for the software to return an error unless the LED/PWM is incorrectly configured. Thus, the user must manually check if the LEDs are lighting up as per the requirements.

3. **Interrupt Testing:** I am using the touch sensor interrupt and GPIO interrupt for my project. Both interrupts are tested and made sure it is working. This is a **semi-automatic** test as it requires the user to press the switch and touch-sensor when prompted. The touch sensor is tested for different ranges of touch (left, middle, and right). In case if the switch/touch sensor is not working, I have given a time of 10 seconds after which it will return an error and not be stuck in an infinite loop forever if the switch/touch sensor is not pressed. If both the interrupts are working as expected it will pass the interrupt testing phase.

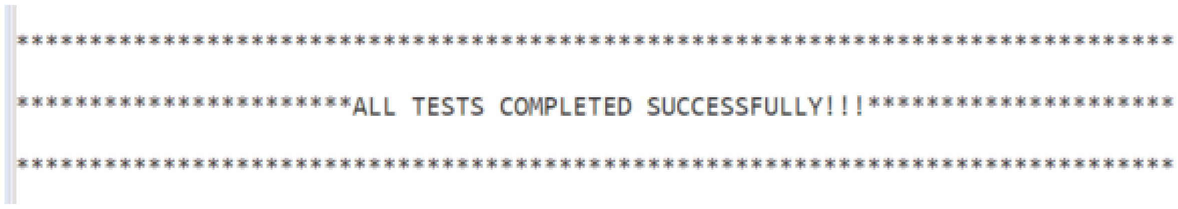
4. **Accelerometer and I2C testing:** This is a **semi-automatic** test that requires the user to tilt the DAG to the angle shown on the screen. Once the required angle is attained by the DAG, the next angle is printed on the screen and the user must move the DAG towards that angle. Once all the angles are completed, the test is over. The angles sequence is 0-> 30 -> 60 -> 90 -> 120 -> 150 -> 180 -> 0. This test is done to make sure that the accelerometer is correctly configured and working. Like the Interrupt test, I have given a 10 second time frame to achieve each angle, after which it returns an error and not be stuck in an infinite loop forever if the desired angle is not achieved. This is also done to make sure I2C is up and running as this test would fail if I2C was not working.

Manual Testing:

1. **Command Processor:** A variety of test-cases are done by giving various inputs into the command processor. Both valid and invalid inputs are given to make sure that the command processor is robust and can handle erroneous inputs. The different erroneous inputs are shown in the test results file.

“Happy Cases” Testing:

When all the testable items as mentioned above are working as expected we would not face any errors and we would get the following output on the serial terminal.



```
******************************************************************************

**********************ALL TESTS COMPLETED SUCCESSFULLY!!!*********************

******************************************************************************
```

Error cases:

In the CBFIFO test cases, we are using assert to check if all the cases have passed. Thus, we would get to know if there is a problem when an assert fails.

In the LED testing, it is not possible to check using software if the LEDs are working. Thus, we must manually verify if the sequence is correct, and the LEDs are working as expected.

In the interrupt testing phase, the user is asked to press the switch. Until the switch is pressed, we are inside an infinite loop. Thus, I have a 10 second timeout after which an error is returned. So, if the user presses the switch and if the switch is not working, we will return an error after 10 seconds. Similarly, the same is done for the touch sensor interrupt.

In the accelerometer and I2C testing phase, the user is asked to move the DAG to the desired angle. If the accelerometer sensor is not working, we would be stuck in an infinite loop. Thus, I have given a timeout, like the interrupt testing above, where if the sensor is not working and the user is unable to reach the shown angle within 10 seconds and error is returned.

In the command handler testing various error cases are tested and the results are shown in the test results file.

Corner Cases for testing:

In the interrupt testing phase, if the switch/touch sensor was pressed before entering the test function the interrupt triggers and sets the global variable in the switch.c file. Thus, as soon as you enter the test_interrupts() it will pass the switch press case as the global variable was already set by pressing a switch. Thus, it must be cleared before checking if the switch was pressed. This was done by resetting the global variable before checking for a switch press.

The corner cases present in the functionality are discussed in the Functionality and Corner cases file.

The test results are present in the test results file.