Spring Final Review

Lockheed Martin LLAMAS Team

sateLLite ADCS fault MAnagement System

Dalton Anderson	Andrew Levandoski	Kristyn Sample
Daniel Greer	Andrew Mezich	Corwin Sheahan
Ben Hutchinson	Samuel O’Donnell	Pol Sieira
Kent Lee	Zach Reynolds	Zack Toelkes

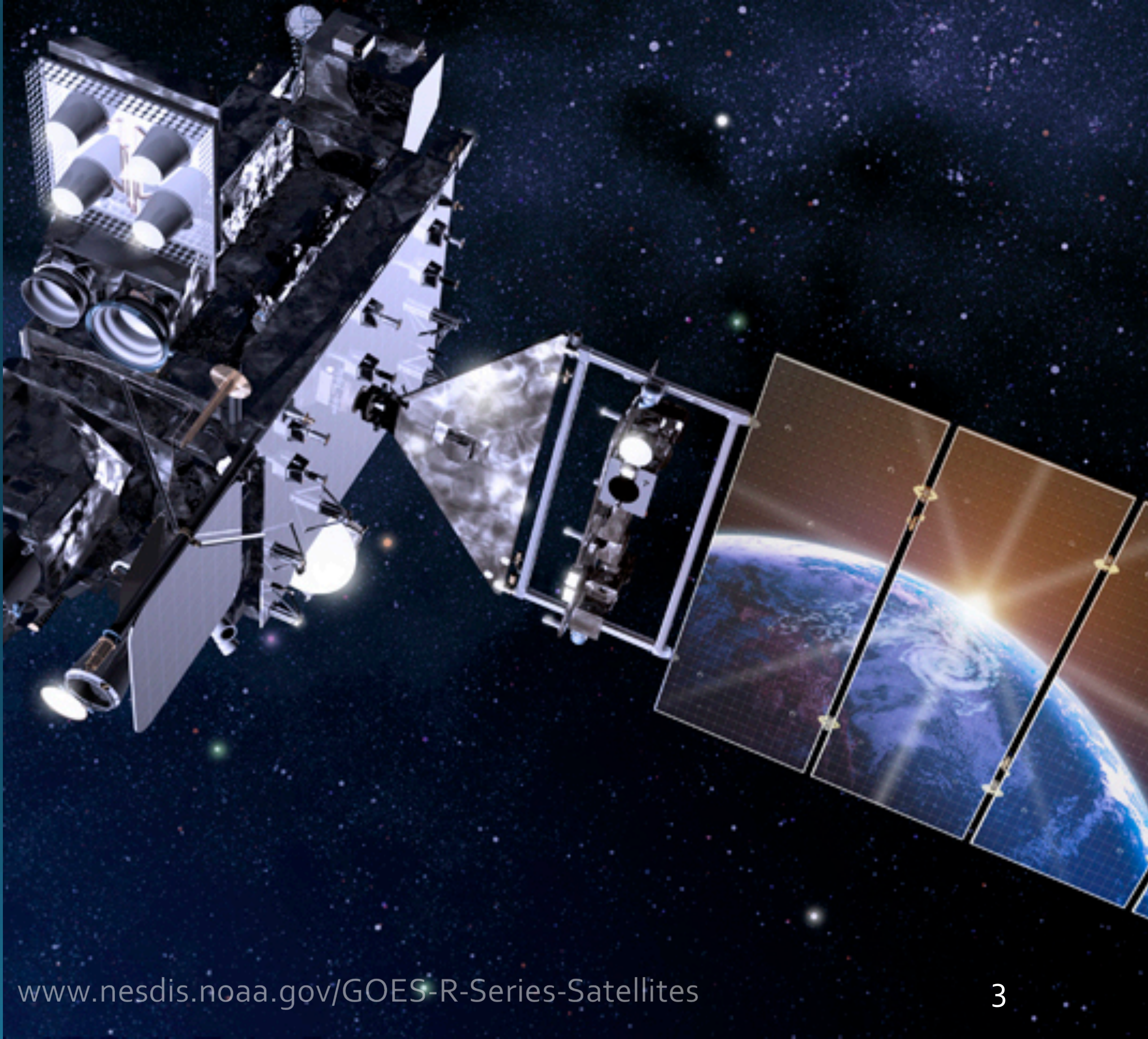
Project Overview

Motivation

Single fault tolerance:

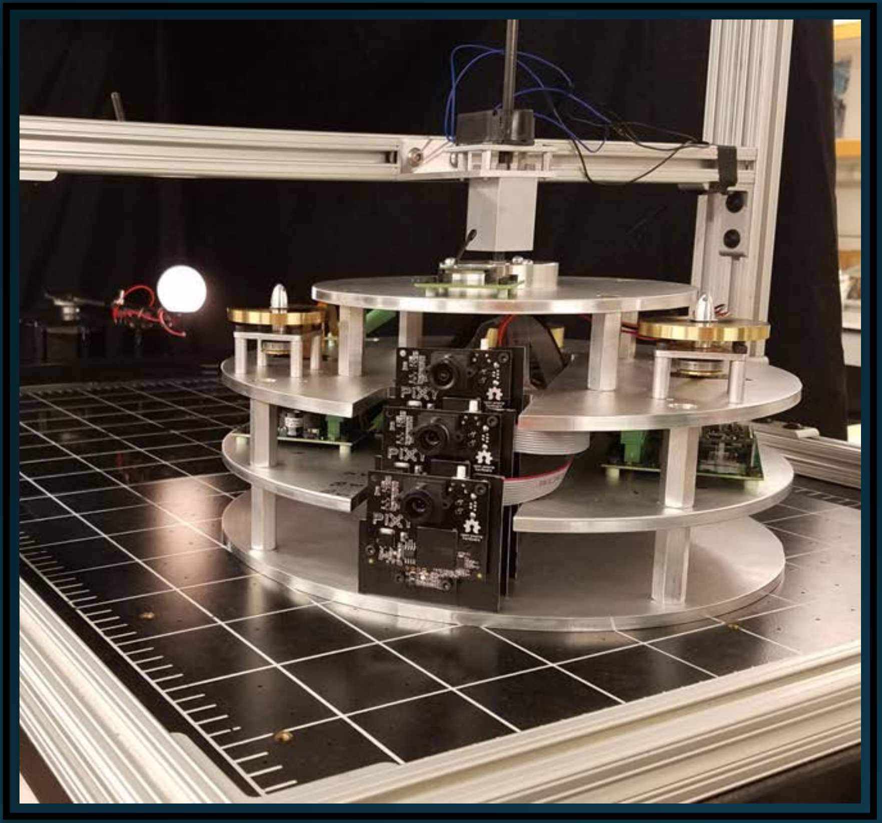
1. System should continue uninterrupted during repair.
2. Fault should be classified optimize the response strategy.
3. Faulty component should be isolated to reduce propagation.
4. No single failure should disable the system.

Separating the ADCS software from fault management software increases <u>modularity of fault testing</u>, allowing for <u>simpler</u> and <u>less costly</u> design.



www.nesdis.noaa.gov/GOES-R-Series-Satellites

Mission Objective



- The team will develop a fault management test bed which allows for testing of fault management software by fault injection into the attitude determination and control system (ADCS) of a mock-satellite (MockSat).
- The MockSat will be representative of the GOES-16 satellite, capable of relaying telemetry and fault data to a ground station unit, allow user selection of faults, and will be tested on a reduced-friction TestTable.

Levels of Success

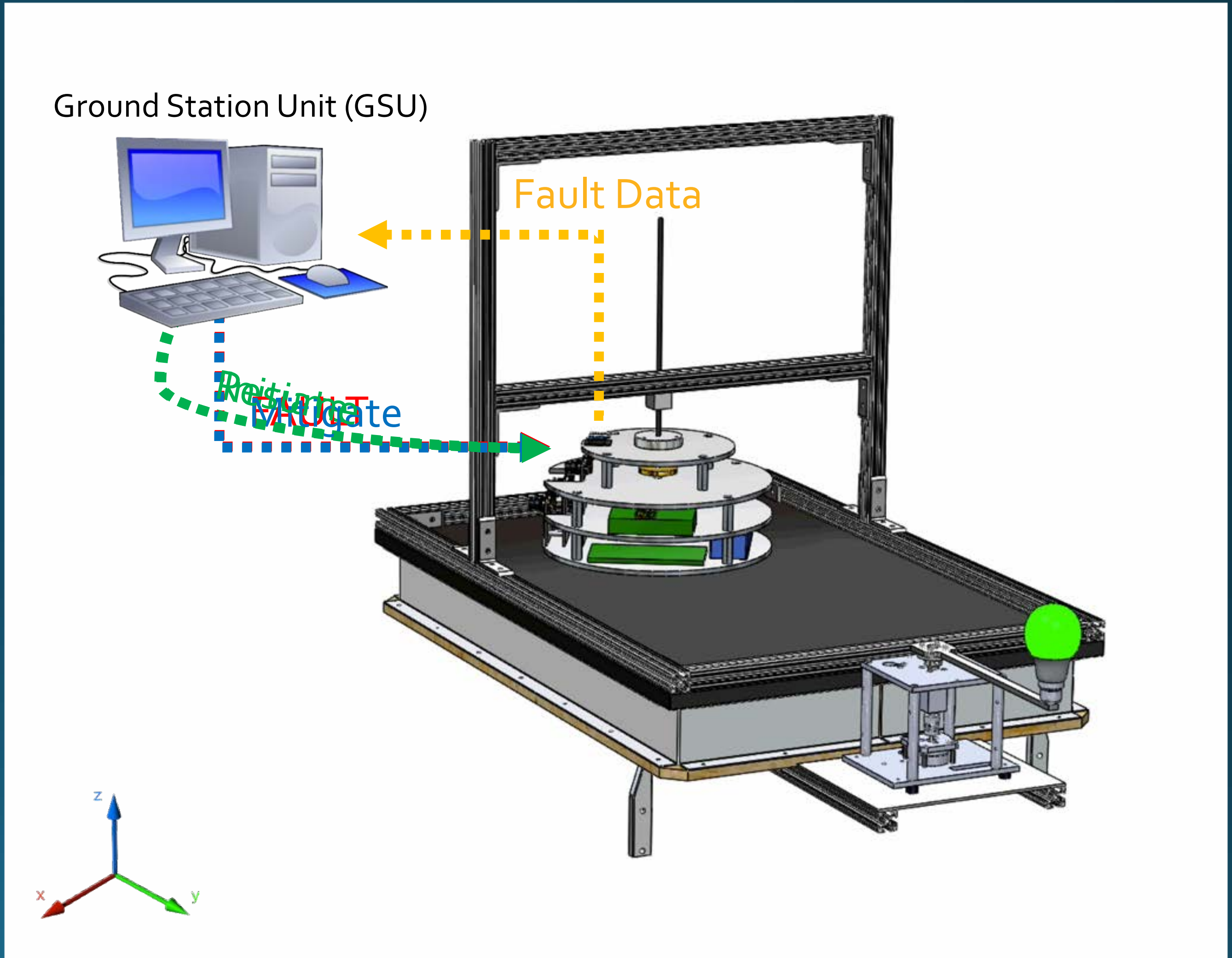
	Level 1	Level 2	**Level 3**
TestTable	Construct a TestTable to allow for 2D translation dynamics with passive control, 1D rotation dynamics, support weight of MockSat, stationary attitude reference		**Moving attitude reference, MockSat attitude encoder**
MockSat Hardware	Power source, coarse orientation sensor, fine orientation sensor, redundant reaction wheels, ADCS/fault injection processor, data storage, 15 minute constant operating time	30 minute constant operating time	**60 minute constant operating time**
Fault Injection	**Inject fatal operating fault into primary reaction wheel after pre-determined time from testing start**	**Inject fatal operating fault into fine sensor**	
Fault Management	Upon fault injection, the MockSat will recognize the presence of the fault and enter a safe mode		**Upon user command, MockSat responds in a way that maintains operational integrity**
MockSat Control	**Active planar rotational control with passive translational control**		
Comm/Data Handling	Flight software and fault uploaded prior to testing, telemetry data stored on-board MockSat, Ground Station data analysis post-test	Wired, real-time telemetry and fault injection	**Wireless, real-time telemetry and fault injection/management**

Functional Requirements

FR 1	The TestTable shall allow for two degrees of freedom in translation and one degree of freedom in rotation in a low friction environment.
FR2	The MockSat shall be equipped with an ADCS that replicates the 0.04 Hz bandwidth response of the GOES-16 satellite to within ±10%
FR 3	The MockSat shall have the ability to maintain a controlled attitude relative to a point of reference within ±2.5°
FR 4	The system shall have the ability to introduce a fatal operating fault in either the MockSat's primary reaction wheel or the fine orientation sensor (but not more than one fault at a time).
FR 5	The MockSat flight control software shall recover from a fatal operating fault in either the MockSat's primary reaction wheel of the fine sensor (but not more than one fault at a time).

Concept Of Operations (CONOPs)

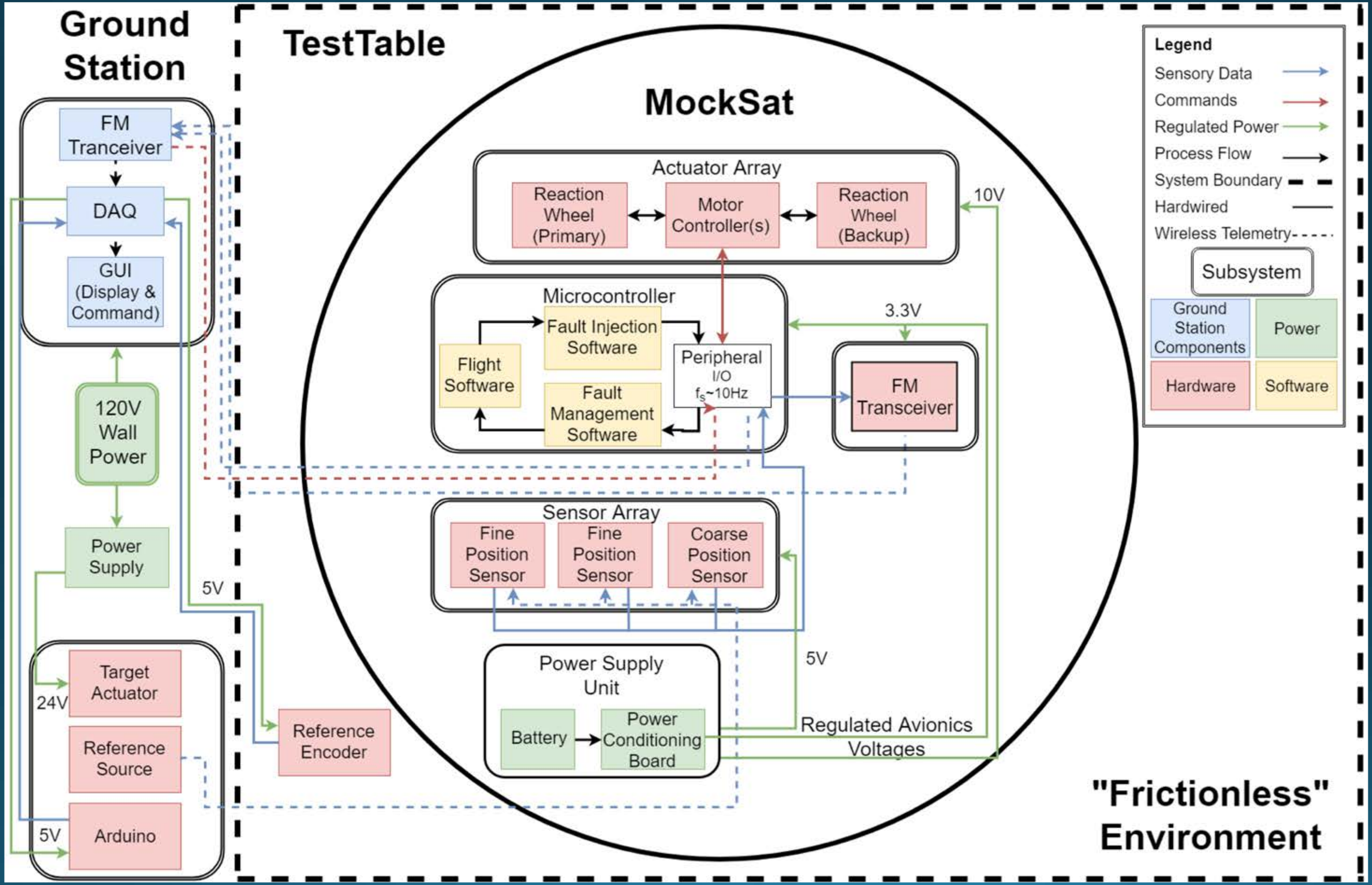
- Test Initiation
  - MockSat initializes and begins searching for target.
- Nominal Operation
  - MockSat has acquired target and tracks motion to within ±2.5°.
- Faulted
  - Fault Injection has introduced a fault that inhibits the MockSat from tracking the target.
- Management of Fault
  - Fault Management has detected and identified the fault and relayed that information to the Ground Station Unit.
  - MockSat is in a faulted state and not maintaining any attitude.
- Initiation of Recovery Sequence
  - MockSat has regained attitude control and is awaiting command to resume searching.
- Recovering
  - MockSat has received command to resume searching for target.
- Return to Nominal Operation
  - Target has re-acquired the target and is tracking to within ±2.5°.



CONOPS (accelerated speed)

Design Description

Functional Block Diagram (FBD)



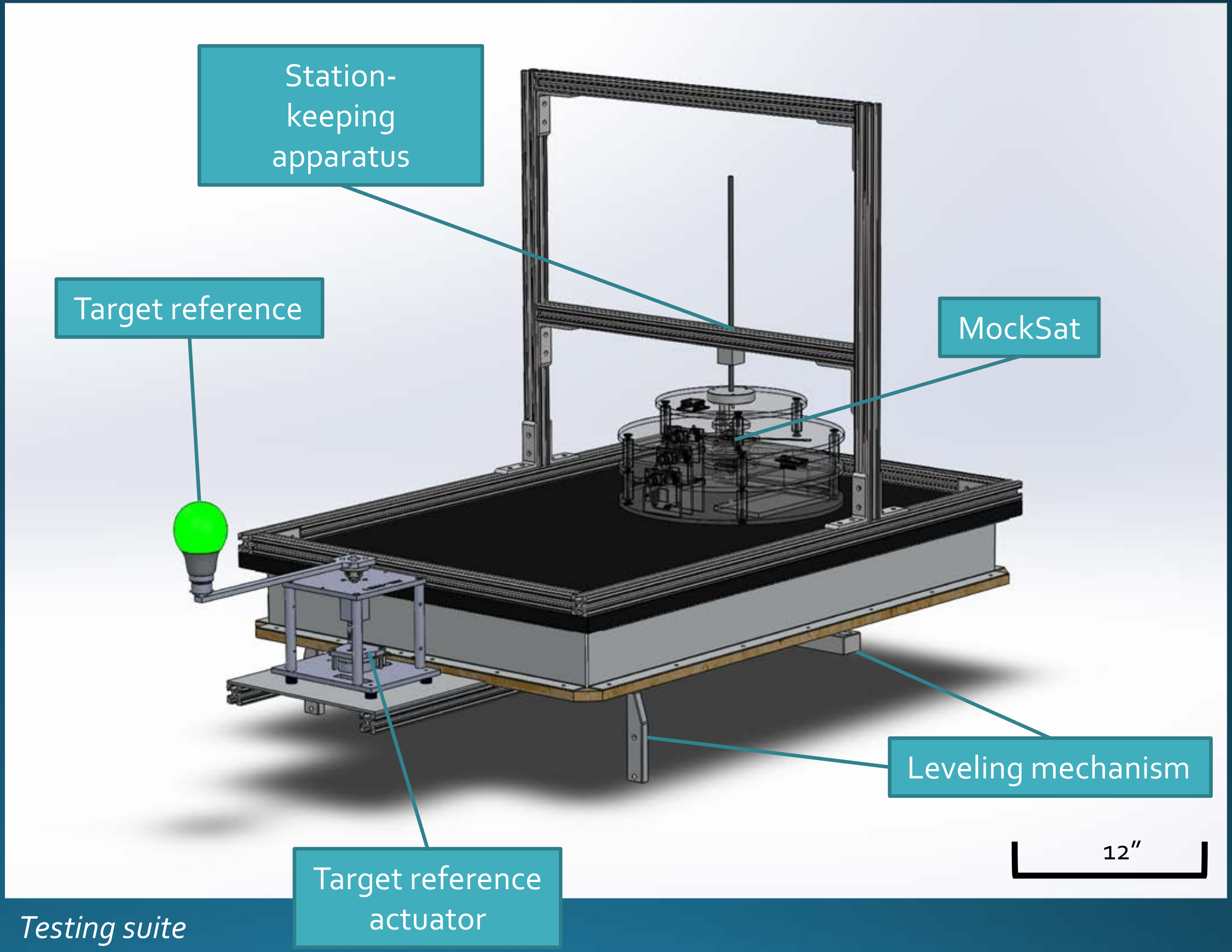
Design Description – TestTable

Operation is similar to an air hockey table

The majority of the TestTable is heritage equipment from the *TracSAT* senior project, with the following modifications:

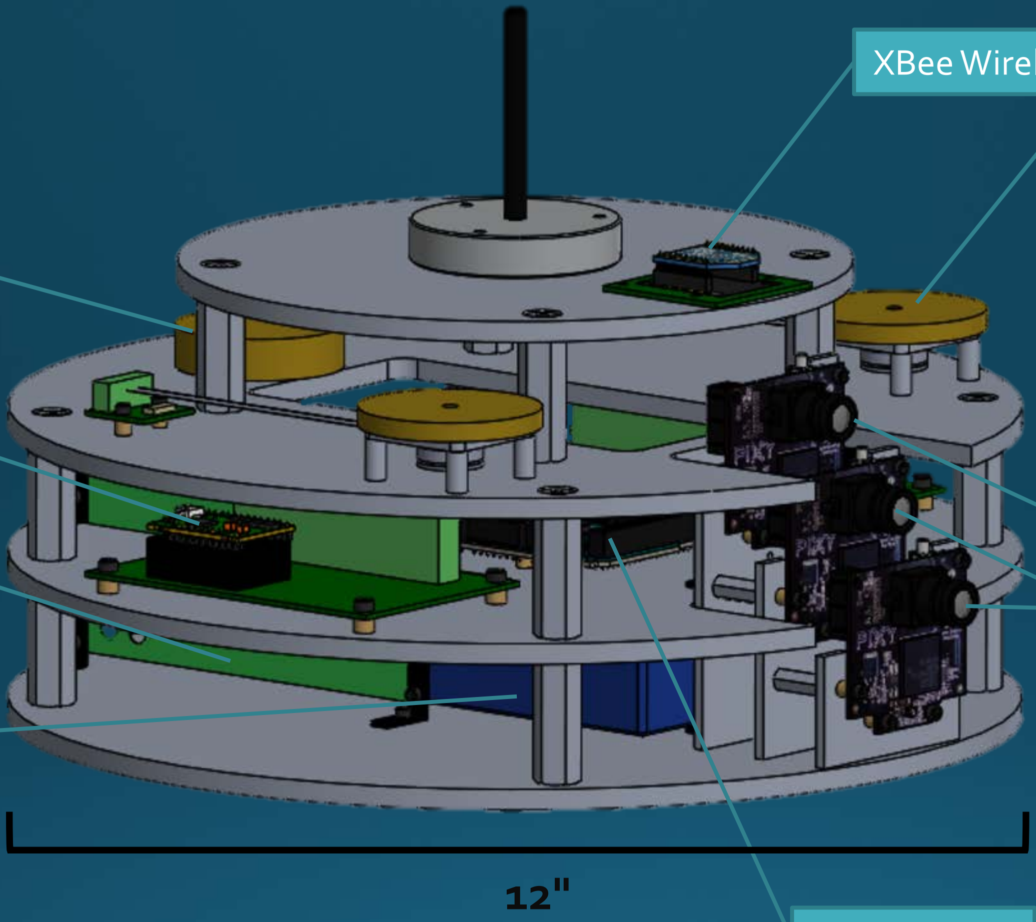
- Half of the TestTable surface has been taped-over.

  Lifting capacity: ~9lb ⟶ ~24lb.
- A table leveling mechanism has been added.
- Station-keeping is accomplished via a removable bearing-block and rigid shaft apparatus.
- The target reference provides an object for the MockSat to track optically.



Testing suite

MockSat Final Design



Mass Balance

- Center of Mass:
  - Fine tuning of center of mass for better rotational dynamics

Reaction Wheel Motor Controller (x2)

Power Conditioning Board

LiPo Battery

XBee Wireless Transmitter/Receiver

Reaction Wheel (x2)

- 1DOF Rotation:
  - One degree of rotational freedom to track target
- Redundancy:
  - Allows for mission continuation after single failure

Coarse FOV Optical Sensor

Fine FOV Optical Sensor (x2)

Fine Field Of View (FOV) = $0.05^\circ/pixel$

Coarse Field Of View (FOV) = $0.16^\circ/pixel$

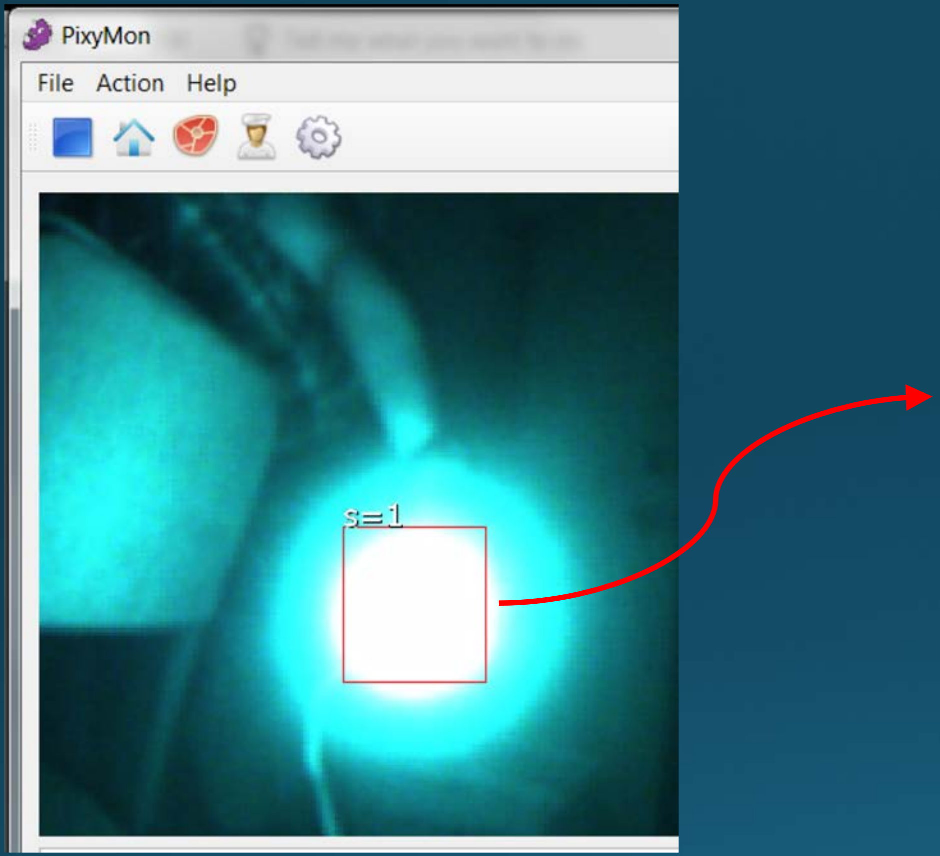
- Tracking:
  - Using hue based cameras with 0.05 degree per pixel
- Redundancy:
  - Allows for mission continuation after single failure

Arduino DUE

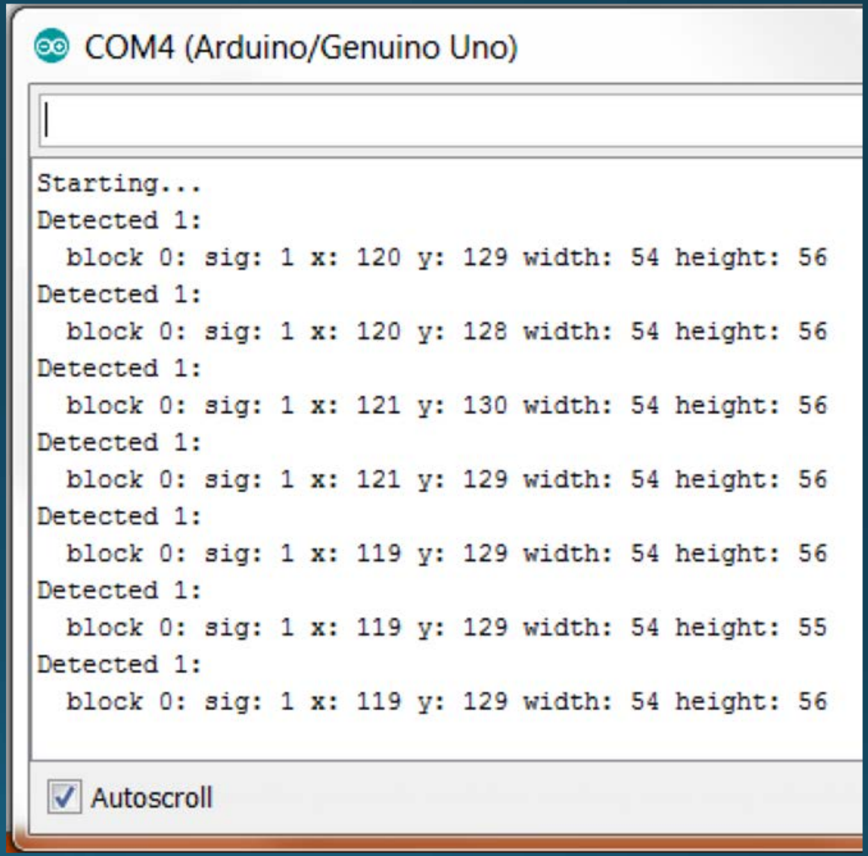
- Communication:
  - Using UART to relay data from ground station to MockSat
- Avionics
  - Controller for pointing and switching of components

Total Weight	11.4 lbm
MOI about Axis of Rotation	188.6 $lb*in^2$
Height	5.9 in
Width	12 in

Design Description – Pixy Operation



Pixy functions based upon a hue detection algorithm



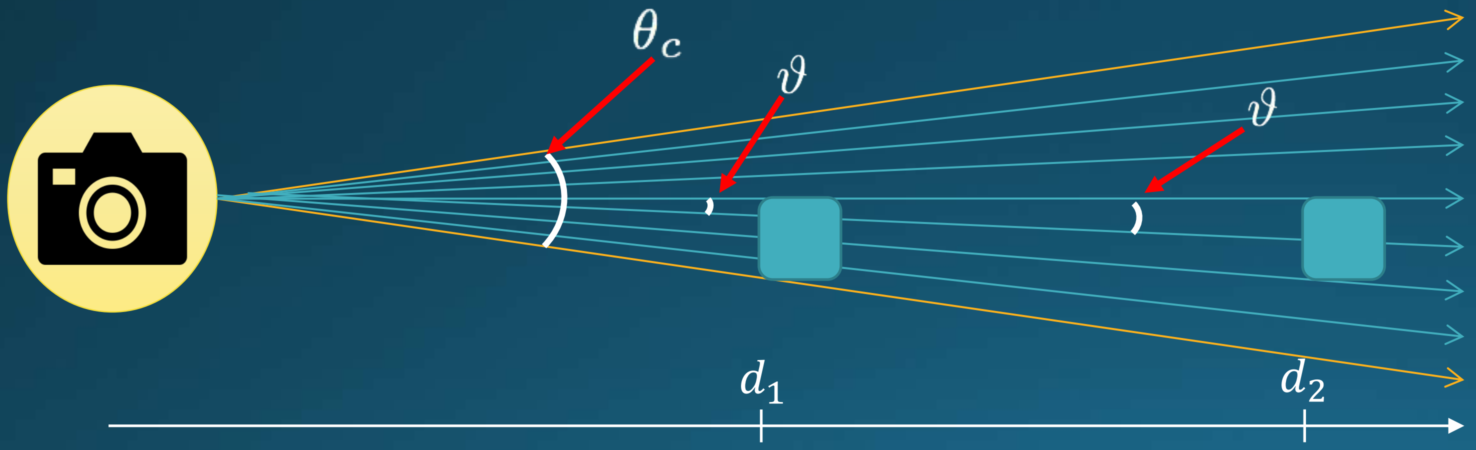
```
Starting...
Detected 1:
  block 0: sig: 1 x: 120 y: 129 width: 54 height: 56
Detected 1:
  block 0: sig: 1 x: 120 y: 128 width: 54 height: 56
Detected 1:
  block 0: sig: 1 x: 121 y: 130 width: 54 height: 56
Detected 1:
  block 0: sig: 1 x: 121 y: 129 width: 54 height: 56
Detected 1:
  block 0: sig: 1 x: 119 y: 129 width: 54 height: 56
Detected 1:
  block 0: sig: 1 x: 119 y: 129 width: 54 height: 55
Detected 1:
  block 0: sig: 1 x: 119 y: 129 width: 54 height: 56
```

Output information details location and size of the detected object relative to the Pixy's internal coordinate frame

Design Description – Pixy Field of View

- Pixy output is related to *pixels*, control law needs *angular distance*
  - Need relationship between pixels and angular distance
- Degrees per pixel (ϑ) can be determined by the following equation⟶

$$\vartheta = \arctan\left[\frac{2\tan\left(\frac{\theta_c}{2}\right)}{\Xi}\right]$$



Pixy is capable of utilizing different lenses.

Coarse Sensor
$\theta_c = 75°$
$\vartheta = 0.16\,°/pixel$

Fine Sensor
$\theta_f = 20°$
$\vartheta = 0.05\,°/pixel$

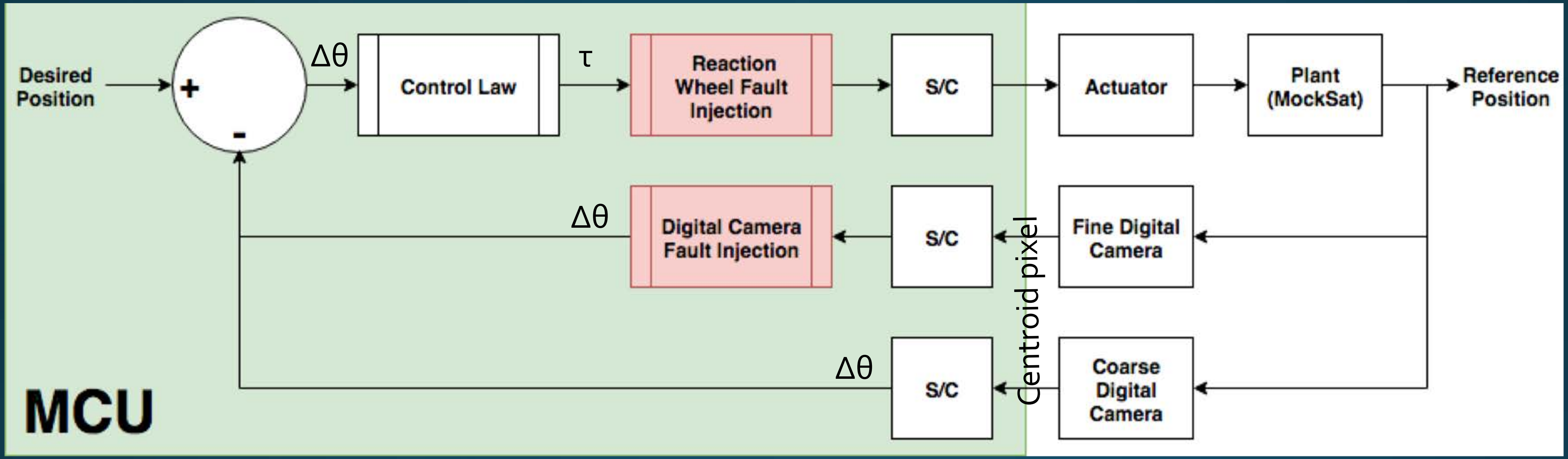
Degree per pixel

Design Description – Fault Injection (FI)

What is a fault?

- Anything that causes the mock-satellite to not meet the desired attitude to ±2.5°
- An induced increase in friction in a reaction wheel, simulating a bad bearing or failing reaction wheel
- A constant bias in pointing angle introduced in the fine attitude determination sensor, causing a systematic error in the pointing of the satellite

Design Description – Fault Injection (FI)



Reaction Wheel

- An additional friction term is subtracted from the commanded torque from the PID control to represent increase friction in the reaction wheel
- Faulted torque is then converted to a PWM signal and sent to motor controller

Digital Camera (Pixy)

- Takes Δθ calculated from the number of pixels between the reference target centroid and the center of the camera FOV*
- 8° is added to Δθ to lie to the control law about the position of the reference target
- This will cause a constant offset in pointing accuracy

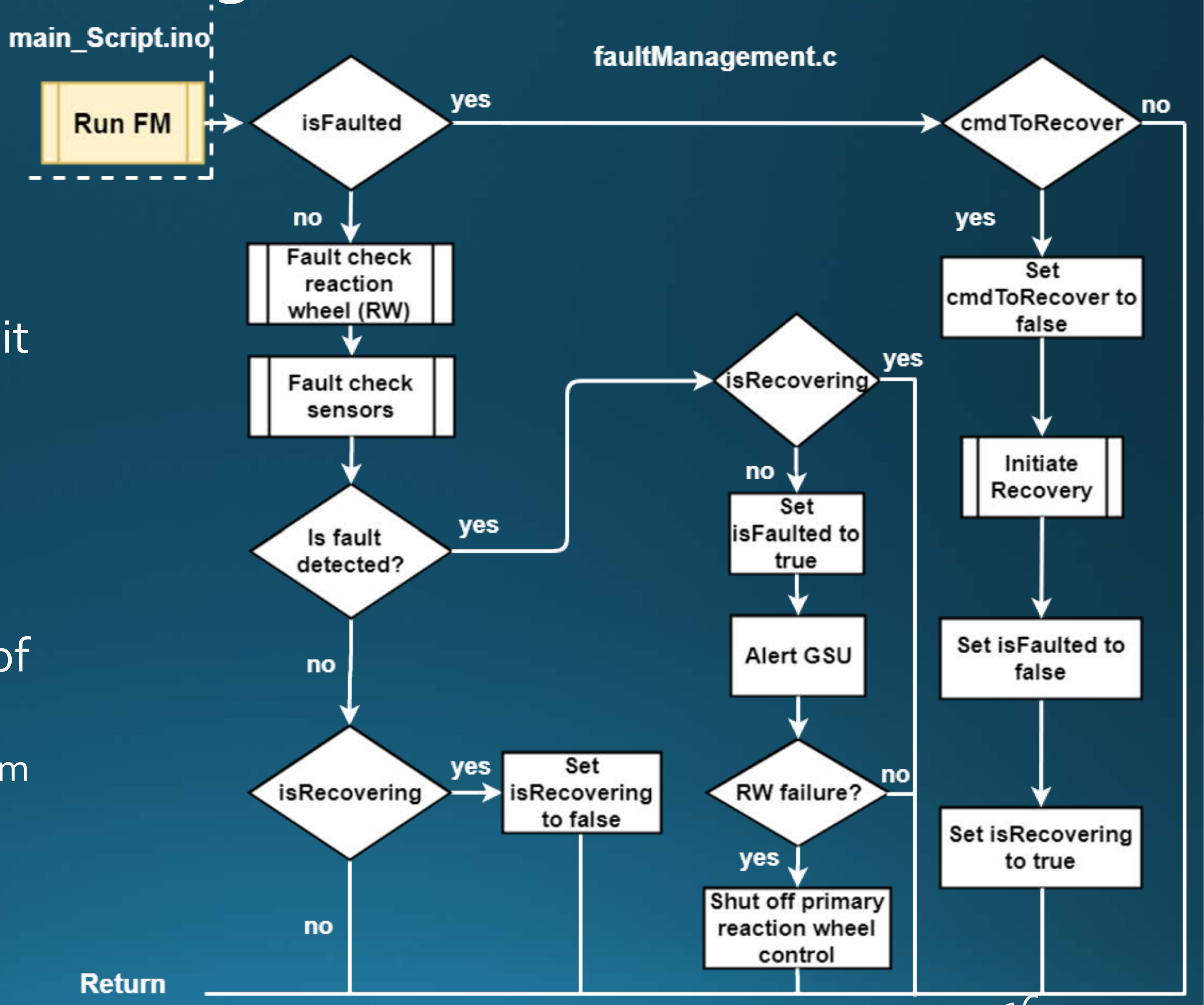
*center of camera FOV is lined up with MockSat pointing centerline

Design Description – Fault Management (FM)

There are 6 states of operation

1. Nominal
2. Faulted
3. Waiting for Ground Station Unit (GSU)
4. Initiate Recovery Sequence
5. Recovering
6. Recovered

- Inputs considered to identify state of system:
  - position of MockSat relative to target (from multiple sensors)
  - reaction wheel speed
  - commanded torque
  - time



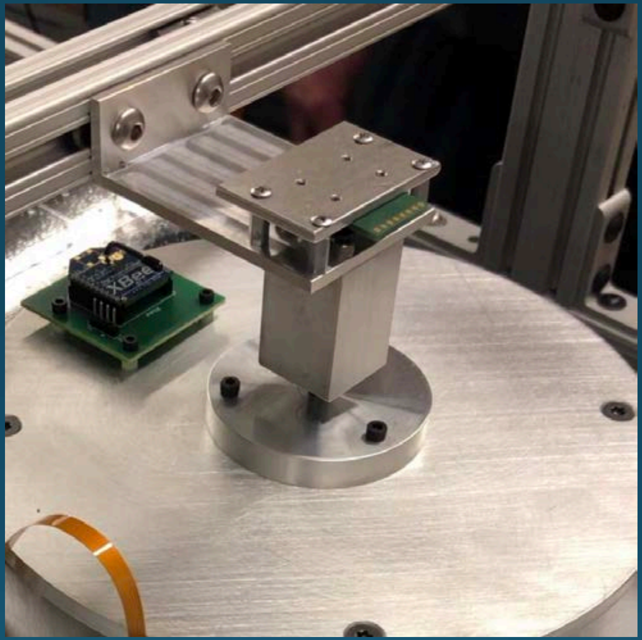
Major Design Changes – MockSat Angular Position Encoder

What changed:

Changed from a 16 bit digital encoder to a dual channel Gray code encoder with 500 pulses per revolution.

Why:

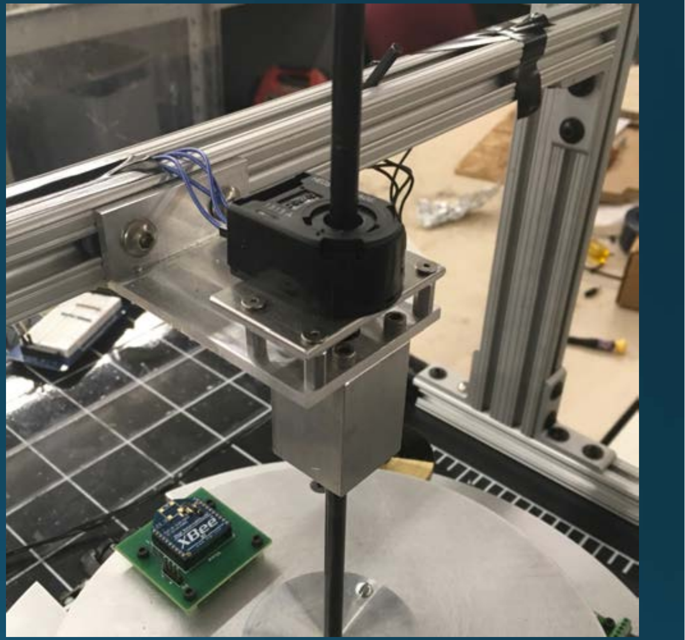
The digital encoder communicated over 16-bit SPI, which proved difficult to integrate with the GSU's LabVIEW software. The HEDS encoder was still capable of the resolution needed to achieve FR 3.



Old Encoder:
MPU 6000

0.05° resolution

→



New Encoder:
HEDS 5505 A06

0.18° resolution

Major Design Changes – Reaction Wheels

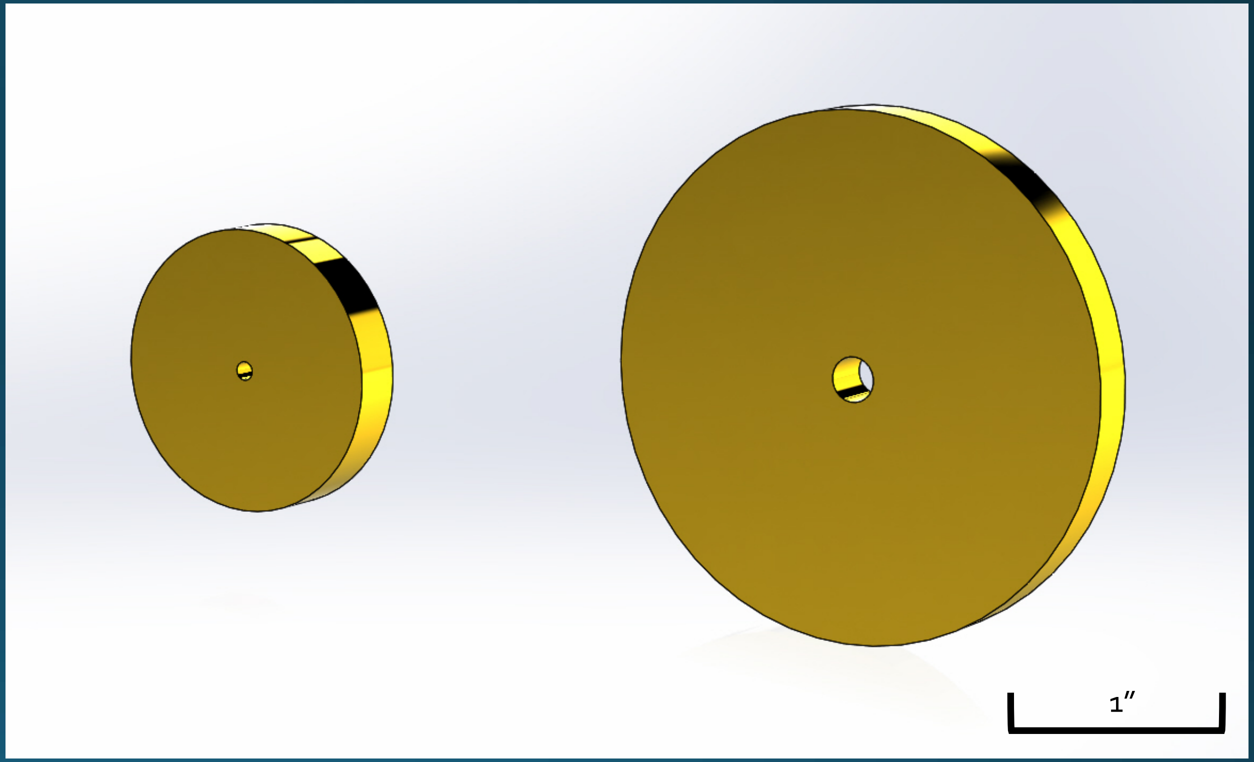
What changed:

Reaction Wheel dimensions were modified to increase moment of inertia (MOI).

Why:

This decision was made to try to reduce how quickly the motors saturate.

This change decreases the angular acceleration for a given torque, thus reducing the time to saturation. This helped achieve the level 3 testing time of an hour.



Old MOI:
0.0035 gm^2

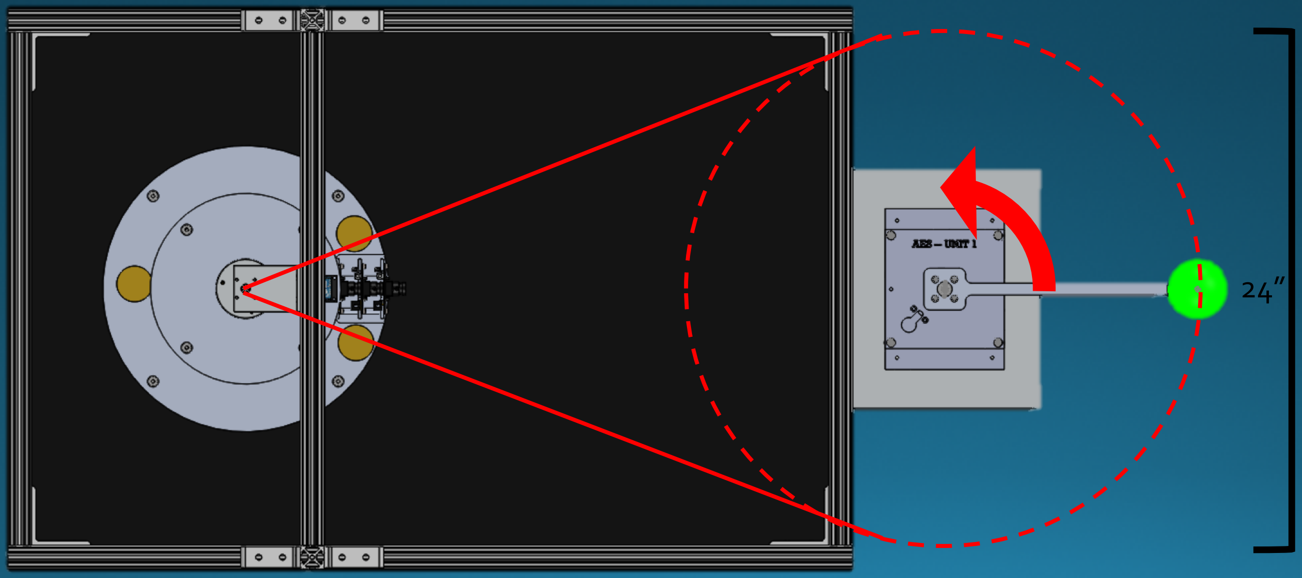
→

New MOI:
0.047 gm^2

Critical Project Elements

Critical Project Elements – Encoders

- Encoders are critical as they are the primary method for V&V for four out of five functional requirements.
- Removed encoder from reference target actuator. Utilizing stepper motor data to determine position.
- Encoder on MockSat changed since TRR due to communication compatibility issues.



Compare encoder data to target location to derive pointing accuracy

Deduce position of the target given the stepper motor's step size and the number of steps taken

Critical Project Elements – Communications and Data Handling

- Communications is a critical element because it is required for analysis of test data and has proved to be more difficult than expected
- Internal communications from sensors to Arduino and Arduino to motor controllers were more difficult than expected
    - The use of SPI communications was foreign and time consuming
    - Structuring communication to the motor controller to prevent unexpected disabling took many trials of testing
- Wireless communications used to isolate dynamics of MockSat and provide fault injection and recovery commands
- LabVIEW used to verify real-time data throughout tests and send commands to MockSat, as well as save data for post-test analysis

Critical Project Elements – Software

- The ultimate goal of the project is the development of the FI/FM software
- Additional software facilitates the testing of the FI/FM software
  - ADCS software to handle sensor inputs and command motors
  - Communication protocols for wireless communications
- Software integration had a higher than expected time cost that delayed project progress
- Software development allows for future flexibility for injecting and detecting new faults

Test Overview and Results

Testing Environment

Preliminary Testing Procedure:

- Fully charge LiPo battery
- Level TestTable*: Adjust set screws on TestTable until MockSat does not translate across table
  - *Detailed leveling of TestTable in backup slides
- Set up black backdrop to ensure Pixy cameras do not pick up erroneous light sources
- Set up Ground Station Unit*
  - *Detailed Ground Station Unit set up in backup slides
- Power on MockSat and TestTable
- Zero MockSat and LabView: Laser attached to reference target arm, once MockSat begins tracking, laser lines up with center line of all 3 Pixy cameras, "zero" button is clicked on LabView

Tests conducted:

1. Pointing Accuracy Test
2. Bandwidth Response Test
3. Fault Injection Test: Fine Pixy Sensor
4. Fault Management Test: Fine Pixy Sensor
5. Fault Injection Test: Reaction Wheel
6. Fault Management Test: Reaction Wheel
7. PID Verification Test

TestTable Friction Quantification

Purpose:
Determine frictional losses due to TestTable and station keeping apparatus for refinement of control law

Requirement Validated:
DR 1.3.1: The frictional damping coefficient (μ) between the MockSat and the TestTable during nominal operation shall be no greater than 1.5 $\text{lbm}\cdot\text{in}^2\cdot\text{sec}^{-1}$

Expectation:
Determining “Frictional Damping Coefficient” value

Method*:
1.) Clear LabView
2.) Turn TestTable on and hold MockSat in place so it does not rotate
3.) Perturb MockSat enough for multiple rotations to ensue
4.) Allowed for MockSat to rotate until rotation in the same direction stopped.
5.) Analysis of this data gave the exponential time decay of the Mocksat's angular velocity.

*Ran the test 9 times in a row for consistency

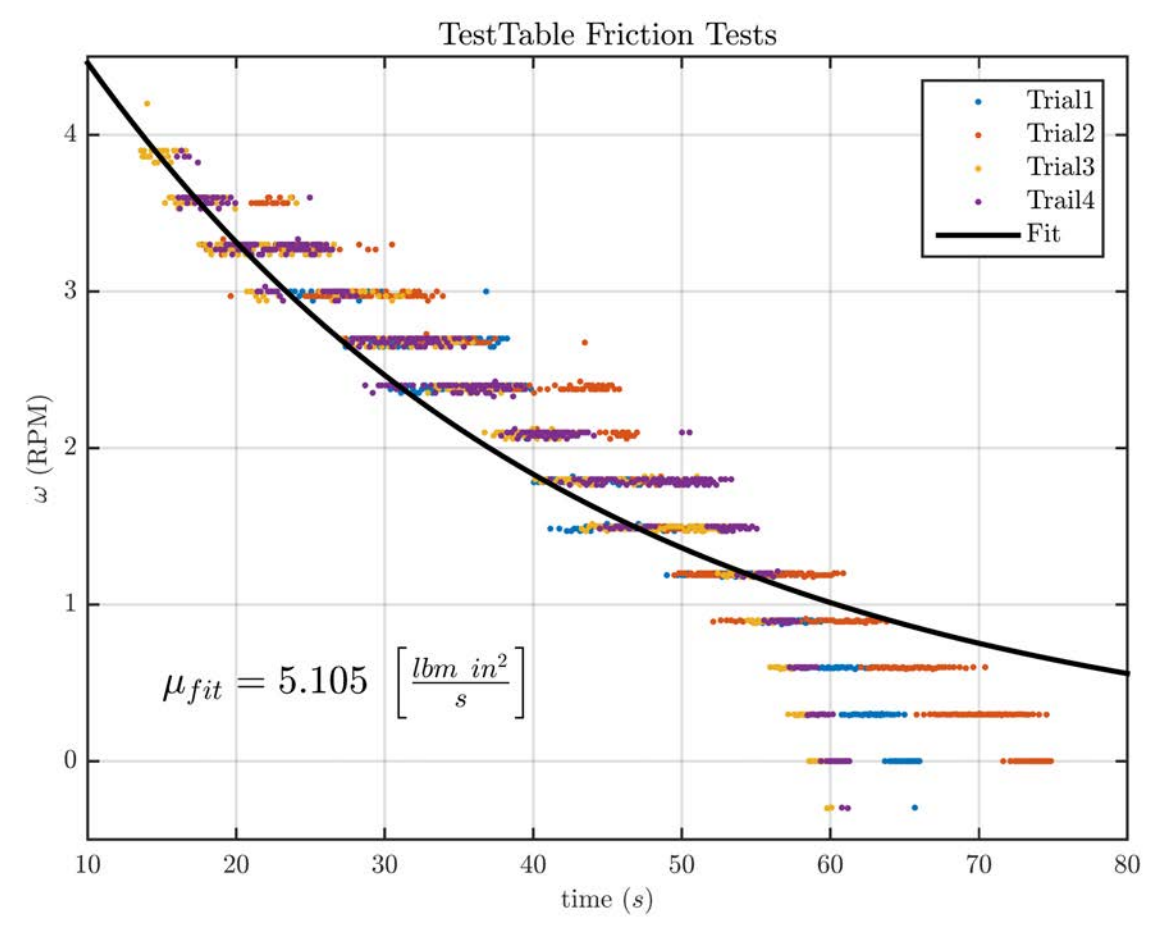
TestTable Friction Quantification – Results

Results:

- $\mu = 5.105$ lbm·in²·sec⁻¹, considerably higher than anticipated
- Original predicted value was estimated using higher angular velocities than the MockSat operating regime
- Lower MockSat angular velocities produce larger and much more unpredictable coefficient values

Importance:

- Does not satisfy DR 1.3.1, $\mu \leq 1.5$ lbm·in²·sec⁻¹
- Requirement was poorly defined using data from outside our operational range
- Proper value still necessary for control system tuning



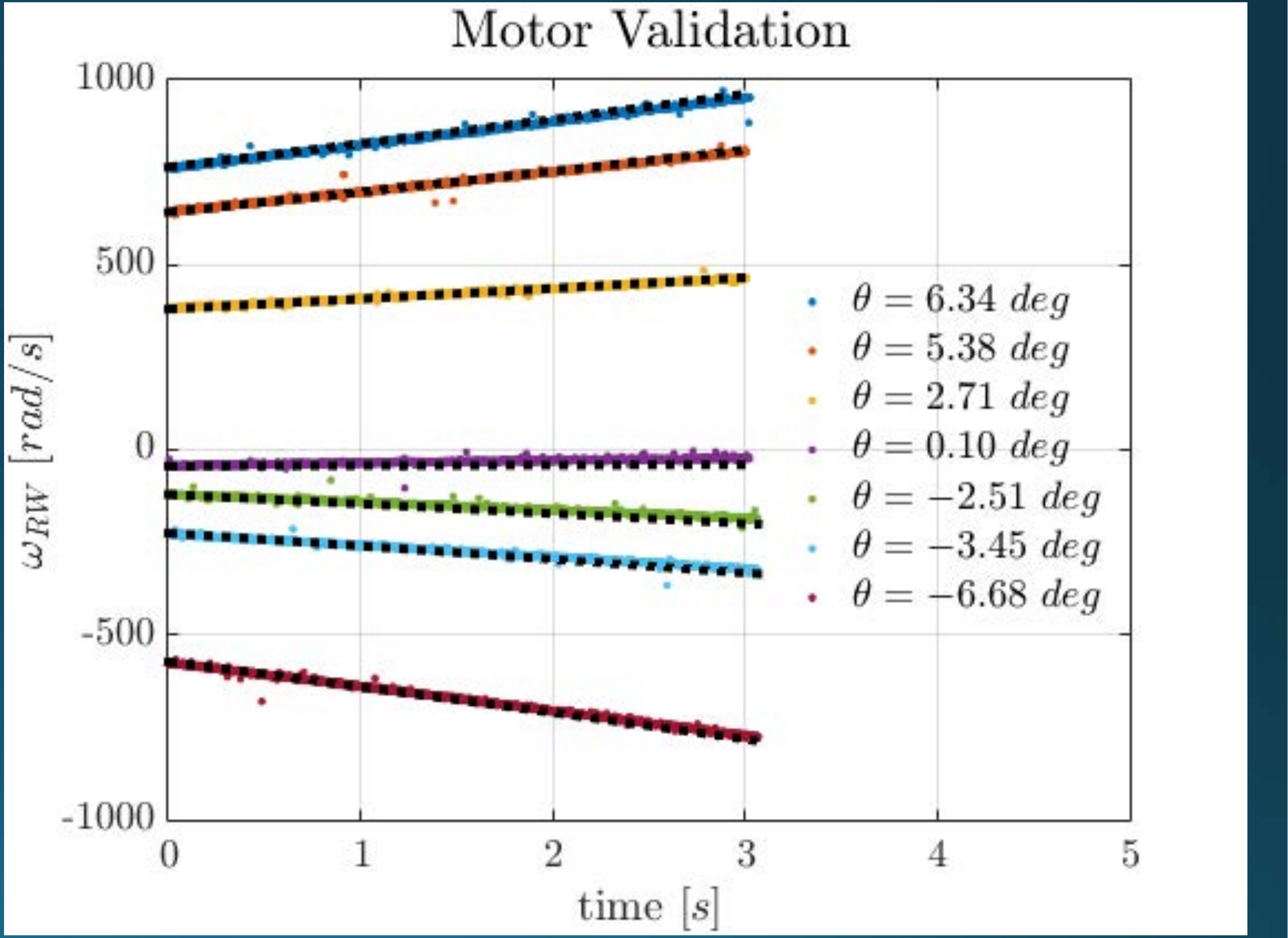
Model Validation – Motors

Purpose:
Determine if commanded torque from PID controller is being applied by the motors

Method:
1.) Turn on the MockSat ADCS without turning on the air for the TestTable
2.) Record the commanded torque and RW speed
3.) Compute the "commanded RW speed" and compare results

Results:
Motors are delivering commanded torques to within 2%.

What we are seeing:
Known offset is inserted into model. MockSat adjusted to offset and torques to correct position are taken and compared to model data.



Pointing Accuracy Test

Purpose:

- Determine pointing accuracy of the MockSat and compare to project requirement
- Large and small perturbations enacted on MockSat were done by physically rotating the MockSat out of the fine sensor FOV so coarse sensor took over

Requirements Validated:

FR 3: Ability to maintain controlled attitude pointing within an accuracy of ± 2.5°

Expectation:

- MockSat will track the reference target to within the ± 2.5° pointing window when reference target is stationary
- The MockSat will be disturbed to outside of the ± 2.5° pointing accuracy and is expected to regain nominal pointing accuracy

Method:

1.) MockSat is actively tracking stationary target
2.) Manually perturb* MockSat a small/large amount
3.) Wait for MockSat to actuate back until the target and Pixy cameras are visually inline
4.) Confirm in LabView data that MockSat encoder data is within ± 2.5° of the target data. This is done by comparing MockSat encoder data and target step command data**

*Large and small perturbations enacted on MockSat were done by physically rotating the MockSat out of the fine sensor FOV so coarse sensor took over

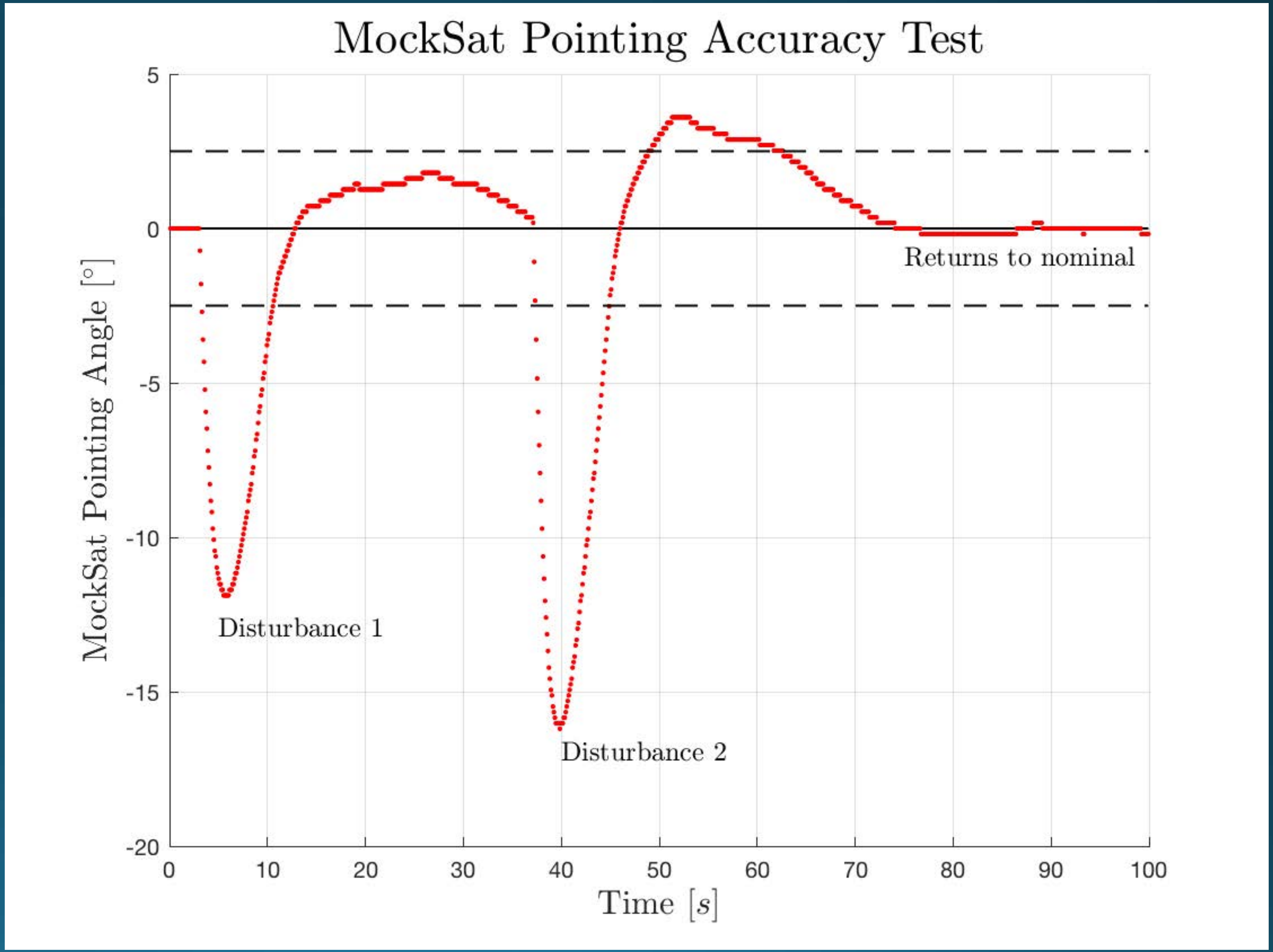
**Target step command data being outputted through LabView is reading values of zero due to the fact that the target is stationary for this test

Pointing Accuracy Test – Results

Results:
The pointing accuracy test shows manual disturbances of the MockSat and the regaining of pointing accuracy within ±2.5°.

Importance:

- This satisfies FR 3 and gives a baseline for future tests of fault injection and management.
- We have to know how well the system works in optimal conditions to understand when the system is faulted.



Bandwidth Response Test

Purpose:
Determine bandwidth response to verify it meets customer functional requirement

Requirements Validated:
FR 2: ADCS replicates 0.04 Hz bandwidth response of the GOES-16 satellite to within ±10%

Expectation:
The MockSat's should traverse 63.2% of any perturbation in 3.98 seconds ±10%.

Method:
1.) MockSat is actively tracking stationary target
2.) Physically rotated MockSat >8° to left *
3.) Let MockSat actuate back until visually the MockSat was inline with the target
4.) Just as the pointing accuracy test, we compared the MockSat encoder data with the target step command data and confirmed the MockSat was within ± 2.5°
5.) Equation to measure bandwidth: $f_{BW} = \frac{1}{2\pi\tau}$. Tau is measured from time of maximum disturbance (>8° off center to 63.2% back to zero line as defined in test plan)

*Multiple tests were done to confirm bandwidth response, physically rotating the MockSat a certain amount was chosen arbitrarily as long as the rotation amount resulted target being out of the FOV of the fine sensors

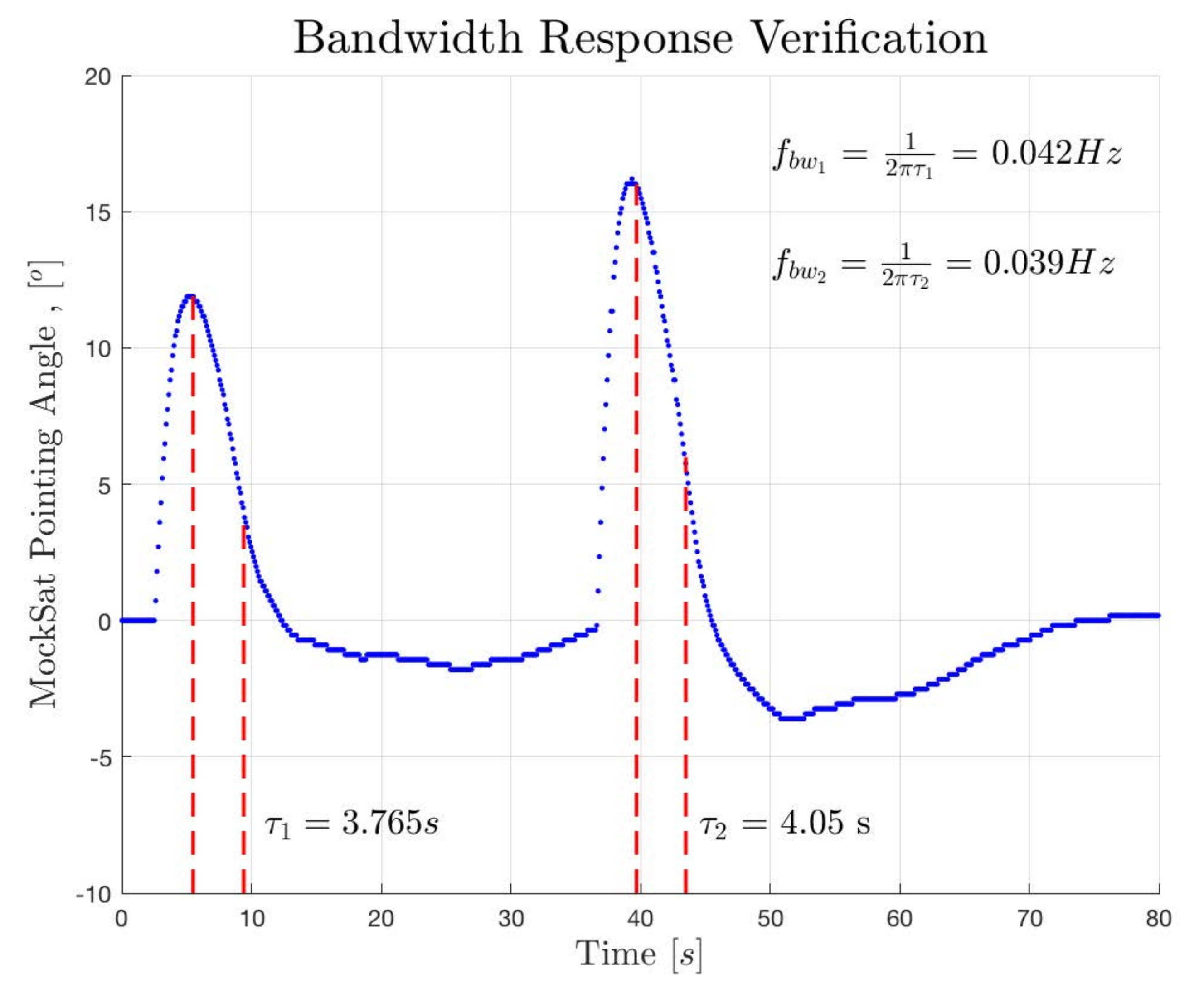
Bandwidth Response Test – Results

Results:

- Data for the same test now shows the bandwidth of this pointing response.
- The pointing angle and the time it takes to reach the desired angle are used to determine the bandwidth, which was found to meet our requirement of $0.04\ Hz$ within $\pm 10\%$

Importance:

Satisfying FR 2, $0.04\ Hz$ within $\pm 10\%$



FI/FM Test: Digital Camera (Pixy) Bias

Purpose:

Demonstrate the ability to inject, detect, and recover from a fault in the fine orientation sensor

Requirements Validated:

- FR 4: System shall have the ability to introduce a single fatal operating fault into the fine orientation sensor
- FR 5: MockSat control flight software shall recover from a fatal operating fault into the fine orientation sensor

Expectation:

- Once the fault is injected into the fine FOV sensor, the MockSat will demonstrate a shift from pointing at the targets center to pointing about 8° away from the target
- After fault has been detected and mitigated by switching to a secondary orientation sensor, the MockSat will reacquire the target and return to nominal pointing of ±2.5°

Method*:

1.) MockSat is actively tracking stationary reference target
2.) Command is given from LabView to inject* bias of 8° into primary fine sensor (Pixy) pointing angle
3.) Observe pointing angle in LabView increase to roughly 8° away from stationary reference target position. The MockSat, will attempt to actuate to the false position fed into the control law
4.) Fault management will alert GSU of detection and type of fault. Then, user gives command to recover.
5.) Fault management initiates recovery sequence and return to nominal pointing angle is observed. Recovery is accomplished by switching to a redundant fine sensor to track the stationary reference target.

*Injection is done by taking the $\Delta\theta$ output by the Pixy and adding a bias of 8° before feeding $\Delta\theta$ to the control law. This is only done if commanded by GSU.

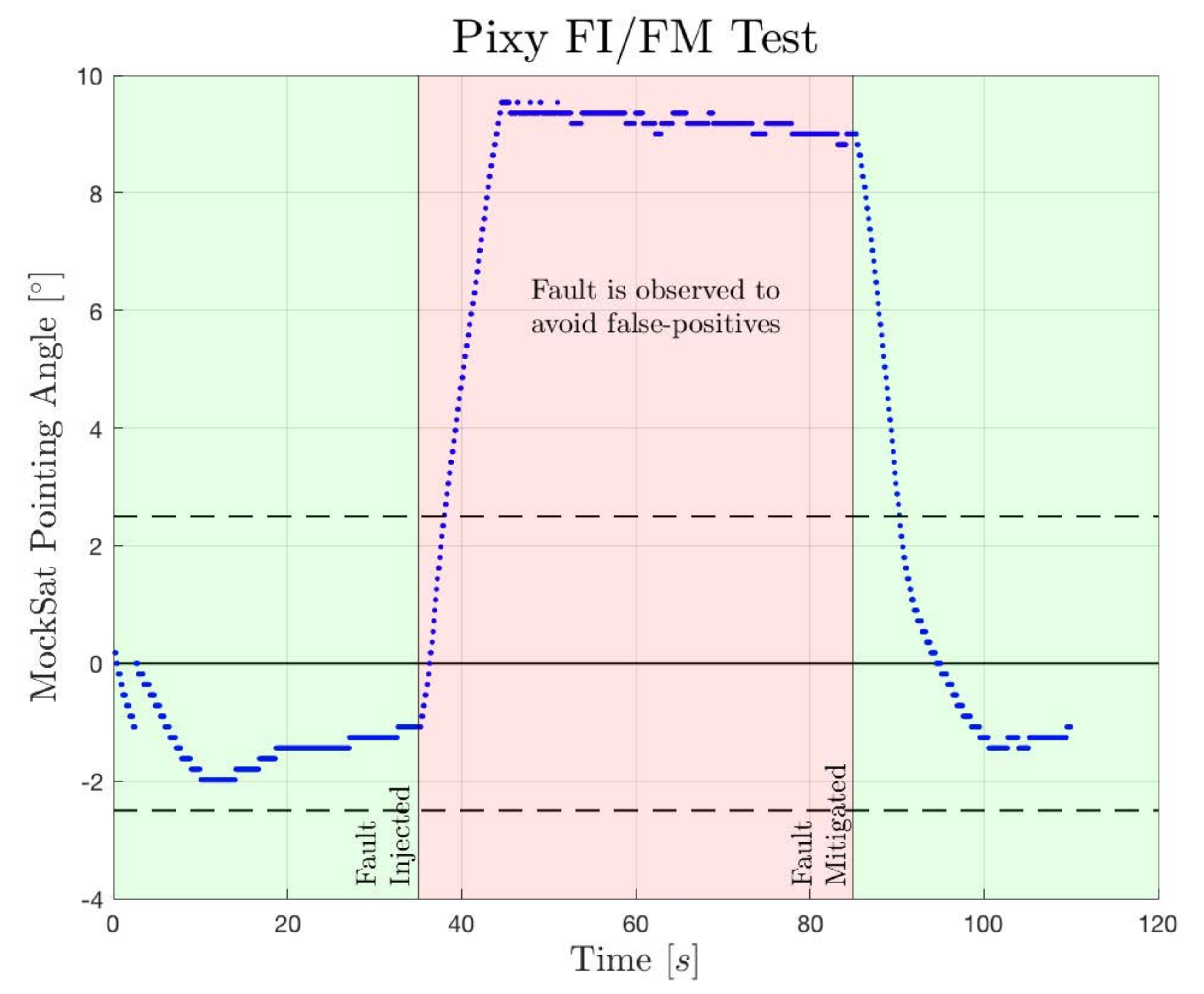
FI/FM Test: Digital Camera (Pixy) Bias – Results

Results:

- After fault injection was initiated via GSU command, MockSat drifted to ~ 7° outside of nominal pointing
- Fault is detected by fault management system and is allowed to fault for 50 seconds
- Fault is then mitigated by switching control to a redundant fine orientation sensor, and returns to nominal pointing requirement of ± 2.5°

Importance:

- Verifies part of FR 4 (causes fatal operating fault in fine orientation sensor)
- Verifies part of FR 5 (regains nominal operation after fatal operating fault in fine orientation sensor)



FI/FM Testing: Reaction Wheel Friction

Purpose:

Demonstrate the ability to inject, detect, and recover from a fault in the primary reaction wheel

Requirements Validated:

- FR 4: System shall have the ability to introduce a single fatal operating fault into the primary reaction wheel
- FR 5: MockSat control flight software shall recover from a fatal operating fault into the primary reaction wheel

Expectation:

- Once the fault is injected into the primary reaction wheel, the MockSat will experience uncontrolled motion, causing off-nominal performance
- After fault has been detected and mitigated, the MockSat will reacquire the target and return to nominal pointing of ±2.5°

Method*:

1.) MockSat is actively tracking stationary reference target
2.) The system is allowed to nominally track within the ±2.5° for 40 seconds
3.) Once hitting the 40 second mark, a timed fault is injected into the primary reaction wheel causing the MockSat to behave unpredictably
4.) After faulting for 10 seconds, a timed recovery switches command to the secondary reaction wheel
5.) The secondary reaction wheel actuates the MockSat back within ±2.5° pointing accuracy
6.) The pointing accuracy of the MockSat is monitored using the encoder mounted to the station keeping shaft and saved using Labview

*Note that detection of the reaction wheel fault was not possible with current hardware. Recovery was instead triggered on a timer (see step 4)

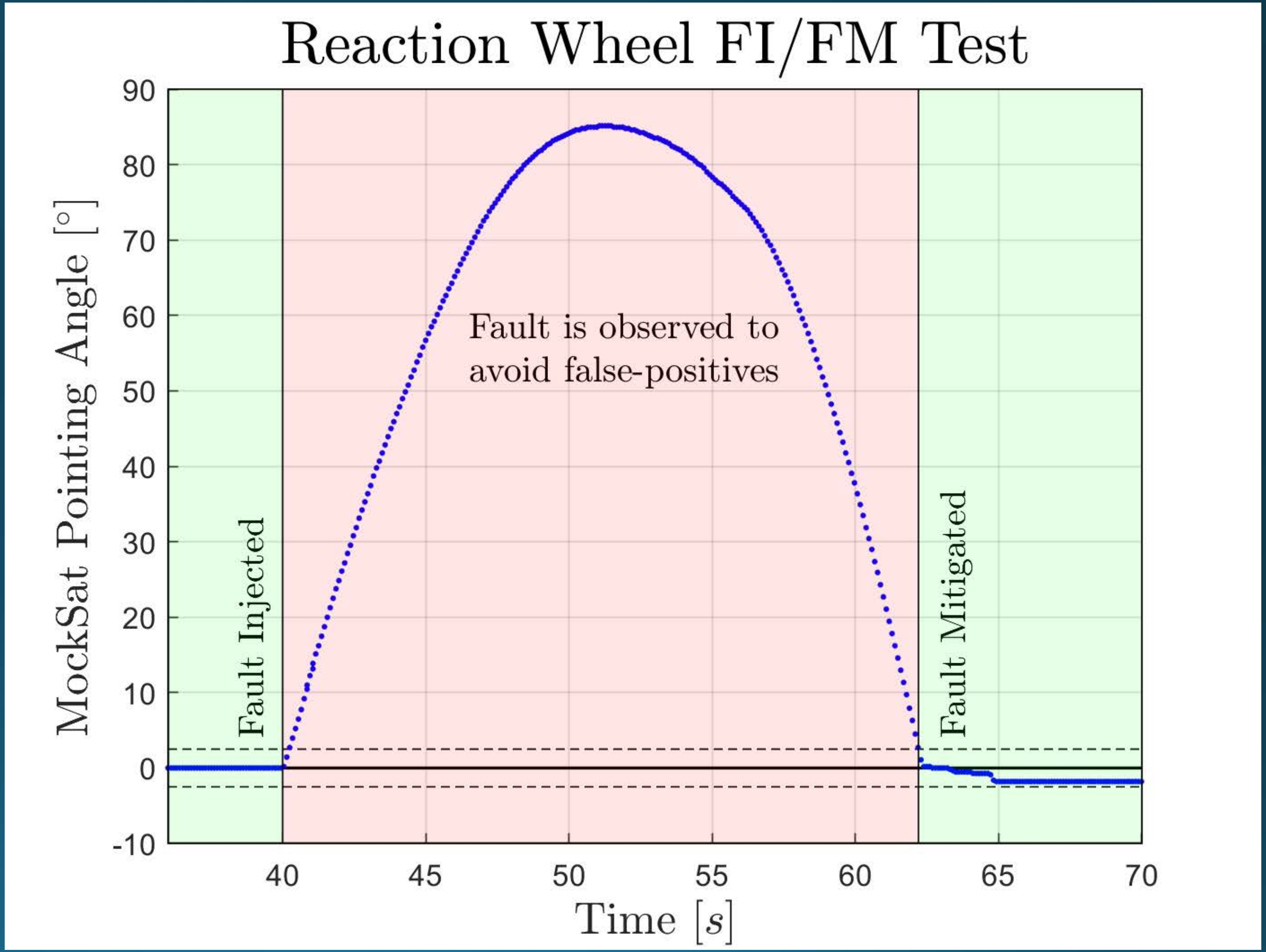
FI/FM Testing: Reaction Wheel Friction – Results

Results:

- Unable to detect reaction wheel fault
- Increasing the friction torque caused the MockSat to point ~85° off of reference
- After timer triggers a command to recover, MockSat control switches to secondary reaction wheel and regains nominal pointing of ±2.5°

Importance:

- Verifies part of FR 4 (causes fatal operating fault in primary reaction wheel)
- Verifies part of FR 5 (regains nominal operation after fatal operating fault in primary reaction wheel)



FM Test: Unable to Accurately Detect Reaction Wheel Friction

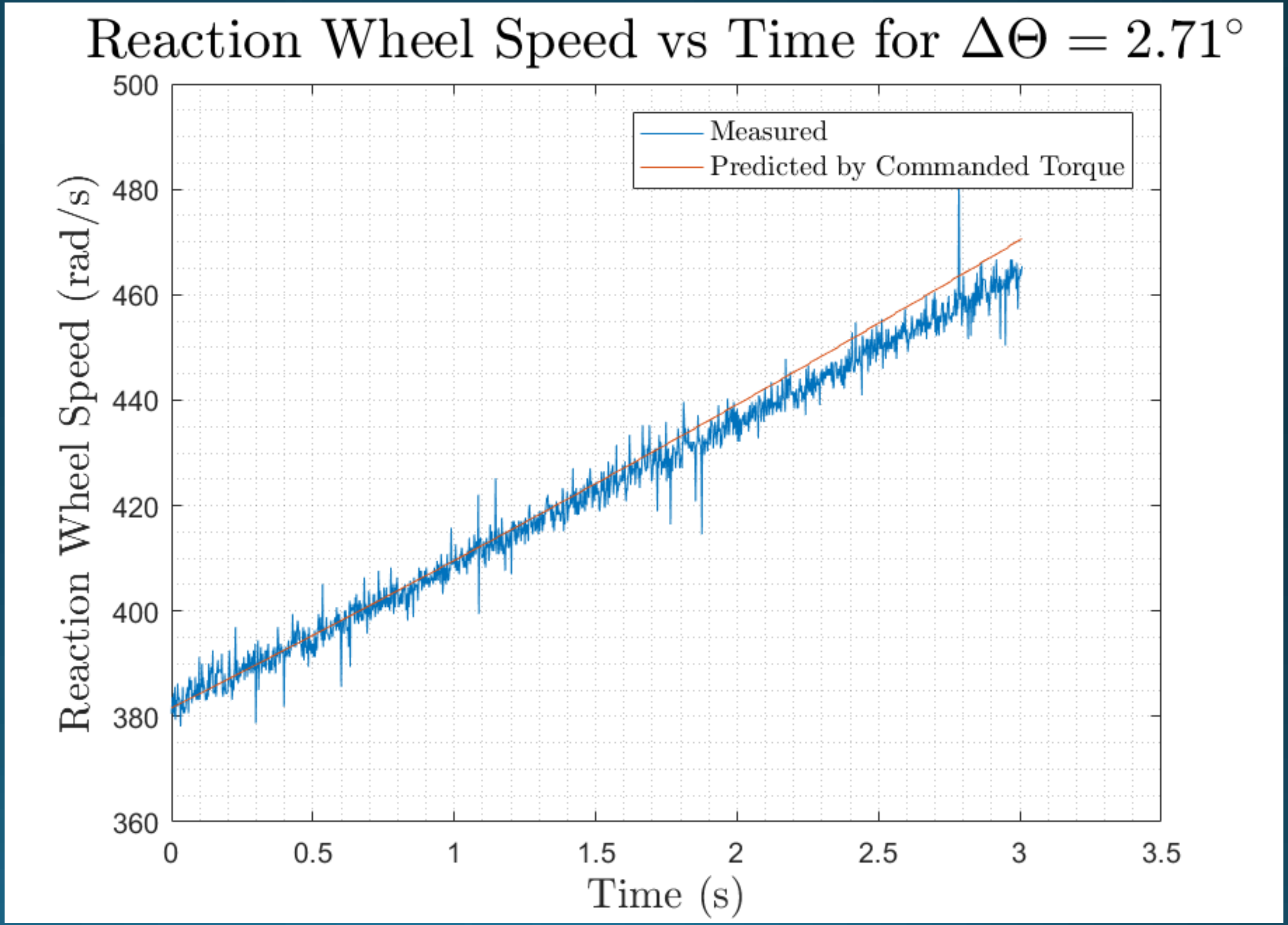
Method:

1.) Fixed MockSat at pointing of ~2.7° off target without running air table (MockSat fixed)

2.) Recorded commanded control torque, reaction wheel speed, and time

Results:

- Noise in reaction wheel speeds from Hall effect sensors of approximately ± 6 rad/s (std. dev. from centerline of 3.0 rad/s)
- Prevents accurate differentiation to find instantaneous angular acceleration and torque
- Would need more accurate measurement device to measure applied torque or reaction wheel speed



FM Test: Unable to Accurately Detect Reaction Wheel Friction

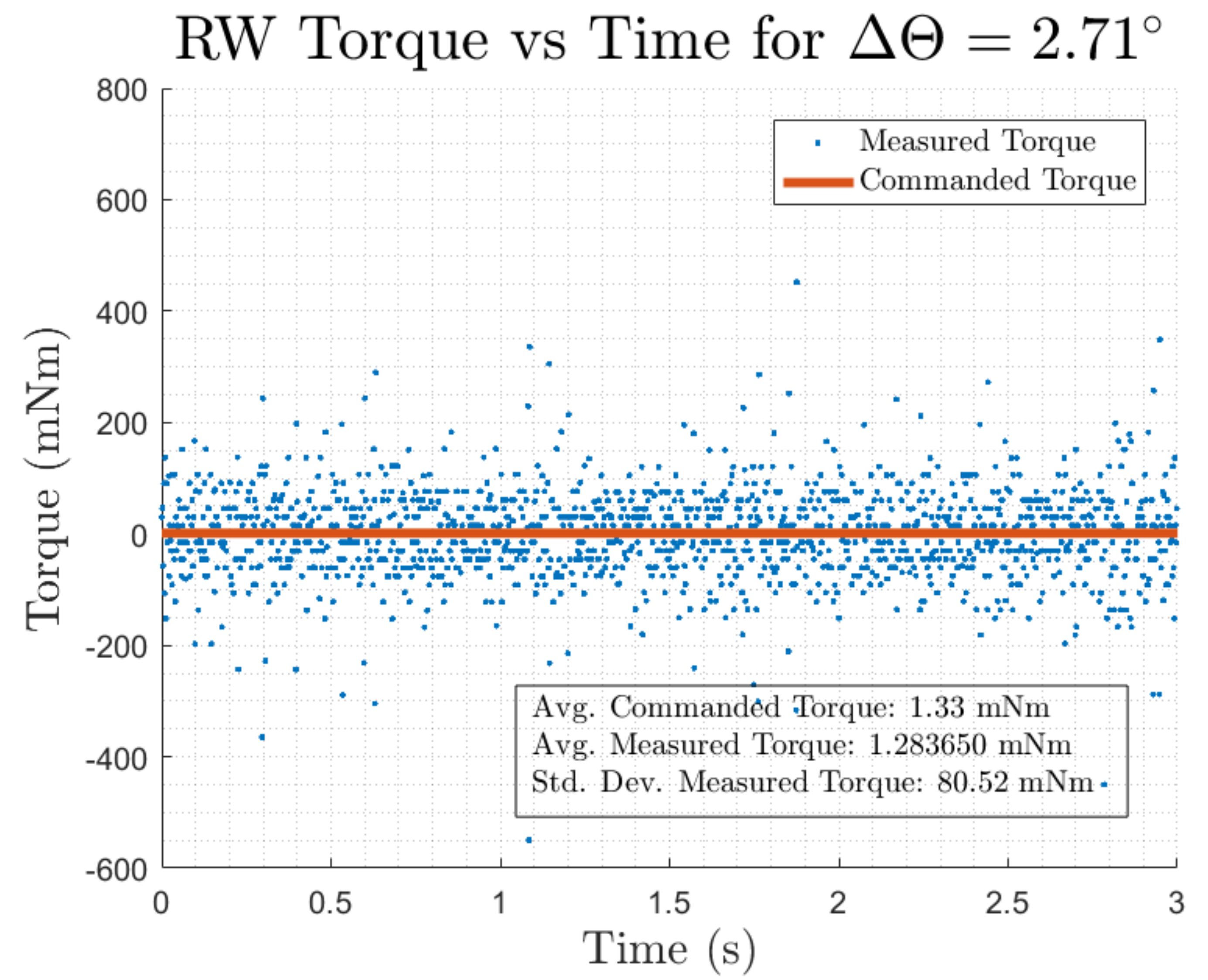
Method:

1.) Used numerical differentiation of reaction wheel speed to determine angular acceleration

2.) Multiplied angular acceleration by MOI to determine measured torque

Results:

- There is no way to accurately predict applied torque using measured reaction speeds from Hall effect sensors, due to noise
- Prevents us from comparing sensed and expected friction torque in order to detect a fault



Control Model Validation

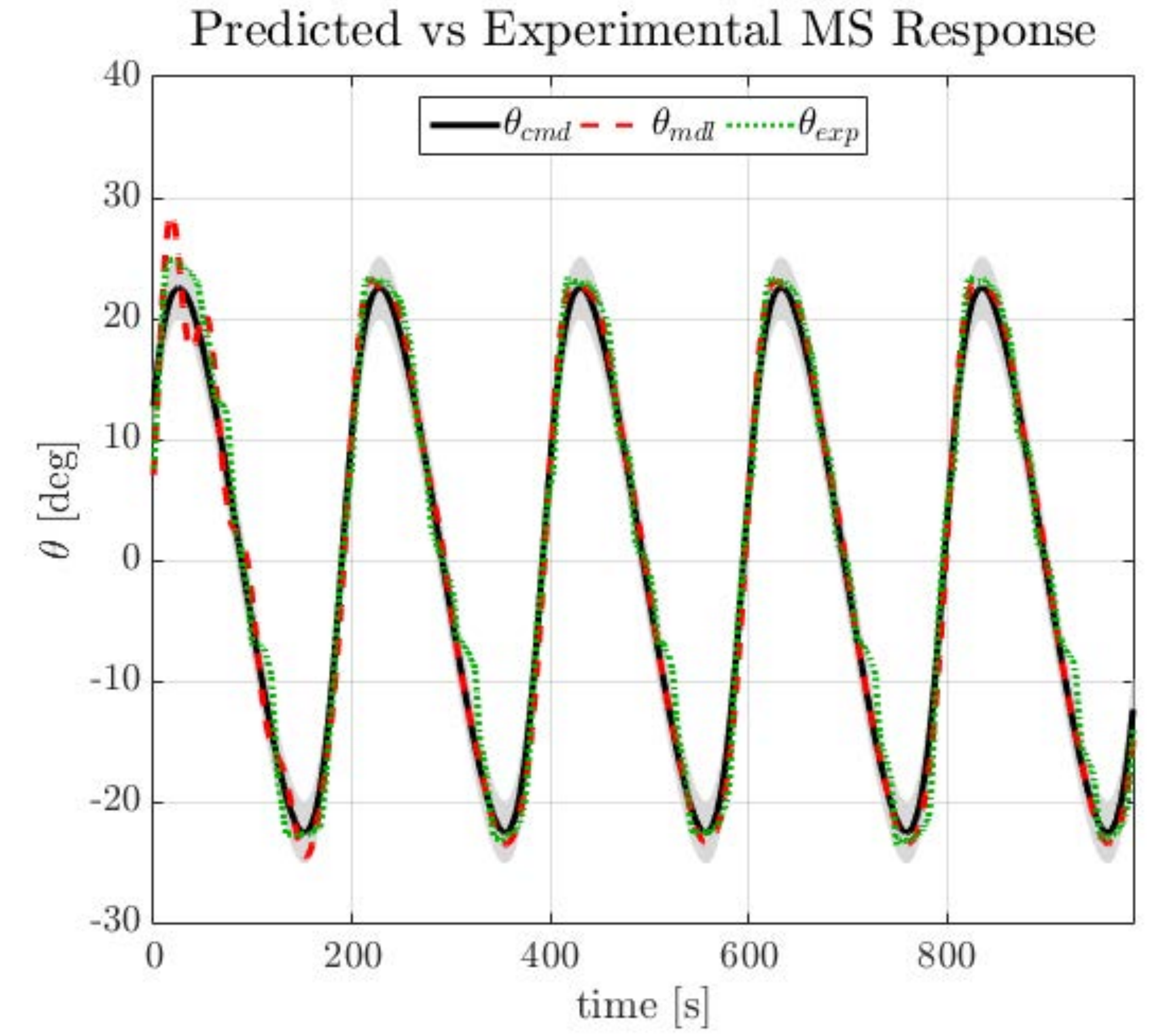
Results:
Modeled and experimental behavior is very similar

Problem:
Major discrepancy between model gains and those tuned experimentally

	Model	Experimental
kp	0.0056	9.3160
ki	5.2724E-6	0.8003
kd	1.3279	46.7969

Potential Explanation:

- **Temporal unit in experimental control law integration**
- **PID discretization error**
- **Improper characterization of friction near zero RPM (operational speeds)**
- **Unmodeled effects: table bias, air disturbance, etc.**



Requirements Met/Unmet

FR 1	The TestTable shall allow for two degrees of freedom in translation and one degree of freedom in rotation in a low friction environment.	Friction quantification unmet, but ability for MockSat to rotate/translate is confirmed
FR 2	The MockSat shall be equipped with an ADCS that replicates the 0.04 Hz bandwidth response of the GOES-16 satellite to within ±10%	Verified through repeated bandwidth response tests
FR 3	The MockSat shall have the ability to maintain a controlled attitude relative to a point of reference within ±2.5°	Verified through pointing accuracy tests
FR 4	The system shall have the ability to introduce a fatal operating fault in either the MockSat's primary reaction wheel or the fine orientation sensor (but not more than one fault at a time).	Verified that faults can be injected into both the primary reaction wheel and fine orientation sensor
FR 5	The MockSat flight control software shall recover from a fatal operating fault in either the MockSat's primary reaction wheel or the fine sensor (but not more than one fault at a time).	Verified that MockSat software can detect and manage the fine sensor fault. For the reaction wheel fault, cannot detect but can still recover.

Systems Engineering

Systems Engineering

Initial Trade Studies:

- Testing Platform
  - In accordance DR 1.1.
  - Crucial in obtaining a frictionless surface for rotation of MockSat.
- Communications and Data Handling
  - Pivotal to ensuring component integration.
- Actuators
  - In accordance DR 2.1.
  - What will drive the rotation of the MockSat to obtain pointing accuracy.
- Sensors
  - In accordance DR 3.1.1, 3.2.1.
  - What MockSat uses to determine target location and relay position change data.
- Station Keeping
  - In accordance DR 1.2.1.
  - Identifying the most efficient way to ensure the movement of MockSat is rotational.
- Controls
  - In accordance with FR 2, 3.
  - Allows MockSat slew to and settle on target within the bounds of .04 Hz
- FI/FM
  - In accordance FR 4.
  - Identifies architecture of the software for injecting and managing the fault. (method)
- Encoder
  - In accordance DR 2.2
  - What ensures we are pointing at the target to validate all data

Systems Engineering

- Objectives laid out by customer was to design a testbed to perform fault management testing of a satellite representative of the GOES-16 satellite, specifically the bandwidth response of 0.04 Hz
- FR 2 was derived from customer objective
- The remaining functional requirements were derived internally to satisfy customer objectives
  - FR 1 – to provide environment for testing and allow for future iterations on current design
  - FR 3 – to demonstrate working ADCS system that can be faulted
  - FR 4 & 5 – to demonstrate fault management testbed works

Systems Engineering

Functional Requirement 1 - Test Table:

- TestTable was necessary to provide a low friction environment in order to isolate system dynamics
- TestTable provided method for station keeping, mimicking the actual operation of GOES-16
- TestTable was portable, complying with OSHA Two-Man Lift Criteria

Functional Requirement 2 - 0.04 Hz Bandwidth Response:

- MockSat ADCS was required to replicate GOES-16 control system performance
- ADCS system utilized redundant reaction wheels to actuate pointing commands

Functional Requirement 3 - ± 2.5° Pointing Accuracy:

- MockSat was equipped with a single coarse sensor, and redundant fine sensors
- Pointing accuracy was used to verify recovery from injected faults

Systems Engineering

Functional Requirement 4 - Fault Injection:

- The system had the ability to inject fatal operating faults into both the fine sensor and the primary reaction wheel
- Fine sensor was faulted by introducing a bias into the data stream
- Reaction wheel was faulted by simulating an induced friction of 5.5 times the natural coulomb friction in the motor

Functional Requirement 5 - Fault Management:

- Fault management software had the ability to detect off nominal system performance
- After detecting fault, management software was able to switch to the redundant component allowing the system to return to nominal performance

Systems Engineering - Risk

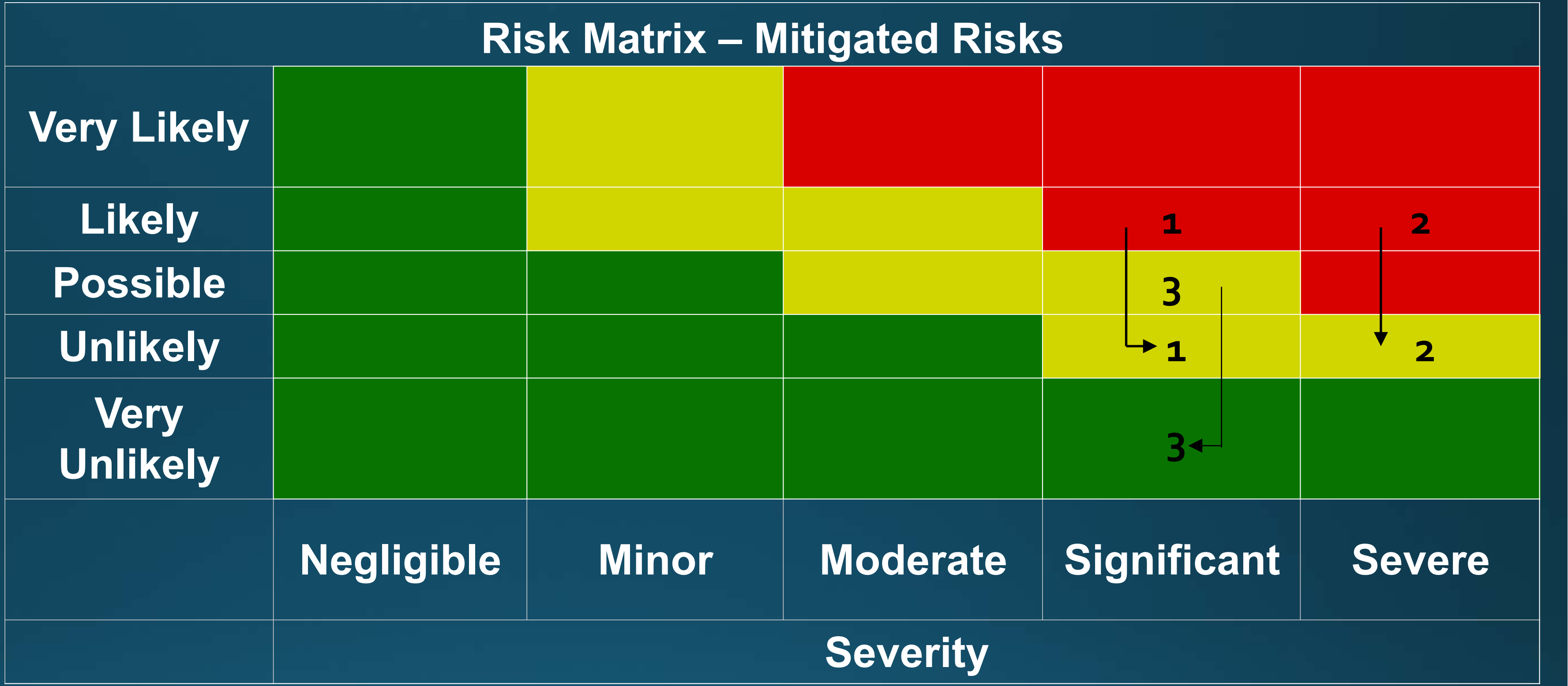
Original 3 major risks:

Overall good assessment of the types of risks predicted in the final stages. All three were a fight to the end.

Other than ADCS, the other risks should have been moved to a higher severity to drive a more extensive mitigation plan.

- Main impact would allow for a padded timeline for integrating components.

For risk mitigation originally predicted see backup slides 63-67.



Risk Matrix – Mitigated Risks					
Very Likely	🟩	🟨	🟥	🟥	🟥
Likely	🟩	🟨	🟨	🟥 1	🟥 2
Possible	🟩	🟩	🟨	🟨 3	🟥
Unlikely	🟩	🟩	🟩	🟨 1	🟨 2
Very Unlikely	🟩	🟩	🟩	🟩 3	🟩
	Negligible	**Minor**	**Moderate**	**Significant**	**Severe**
	Severity				

🟩 **Acceptable**	**1.** Lack of torque resolution
🟨 **Marginal**	**2.** Fault Management Implementation
🟥 **Unacceptable**	**3.** ADCS Integration

Initial risk w/mitigation analysis

Systems Engineering - Risk

Torque Resolution

Plan: Proper motor selection, accurate torque characterization.

Evaluation: Selection and characterization done well, due to risk mitigation.

Lessons Learned: Excess trouble came from needing large torques at small, precise increments. Also adjustments to RRW moment of inertia delayed testing timeline.

Fault Management

Plan: Create a fault management architecture that attempts to solve the modularity aspect.

Evaluation: Biggest issue with faults came from a lack of time more then code modularity.

Lessons Learned: Being able to insert the fault software and spend the time needed to work out the kinks in line with the master code file. Integration time is biggest factor, everything else can be worked around

ADCS

Plan: Careful system integration and understanding of communication protocols.

Evaluation: Component communication was the biggest issue presented in the project.

Lessons learned: Ensuring the similar communication types and sizes from data sheets should be an added forethought when selecting components.

Systems Engineering – Lessons Learned

1. Get on the same page!
   - Every step of the way relate data between subgroups so changes can be related across all components.
2. Put in the engineering analysis.
   - Model the environment the part will be subjected to and the tolerances needed to thrive.
   - Double check with other people to ensure nothing is left out and the environment is valid.
3. Get the groundwork done.
   - Each part needs to be characterized properly, to include data packages for output analysis. Tinkering with component should not only be accepted but highly encouraged.
4. Allow for 3x the tolerances you expect.
   - All the analysis in the world doesn’t prepare for real time testing and strains from other systems.
   - Creating tolerance margin gives flexibility in other changes to system that may effect each component differently
5. Get it into system integration as soon as possible.
   - Where everything gets worked out, just requires time and diligence, be flexible.

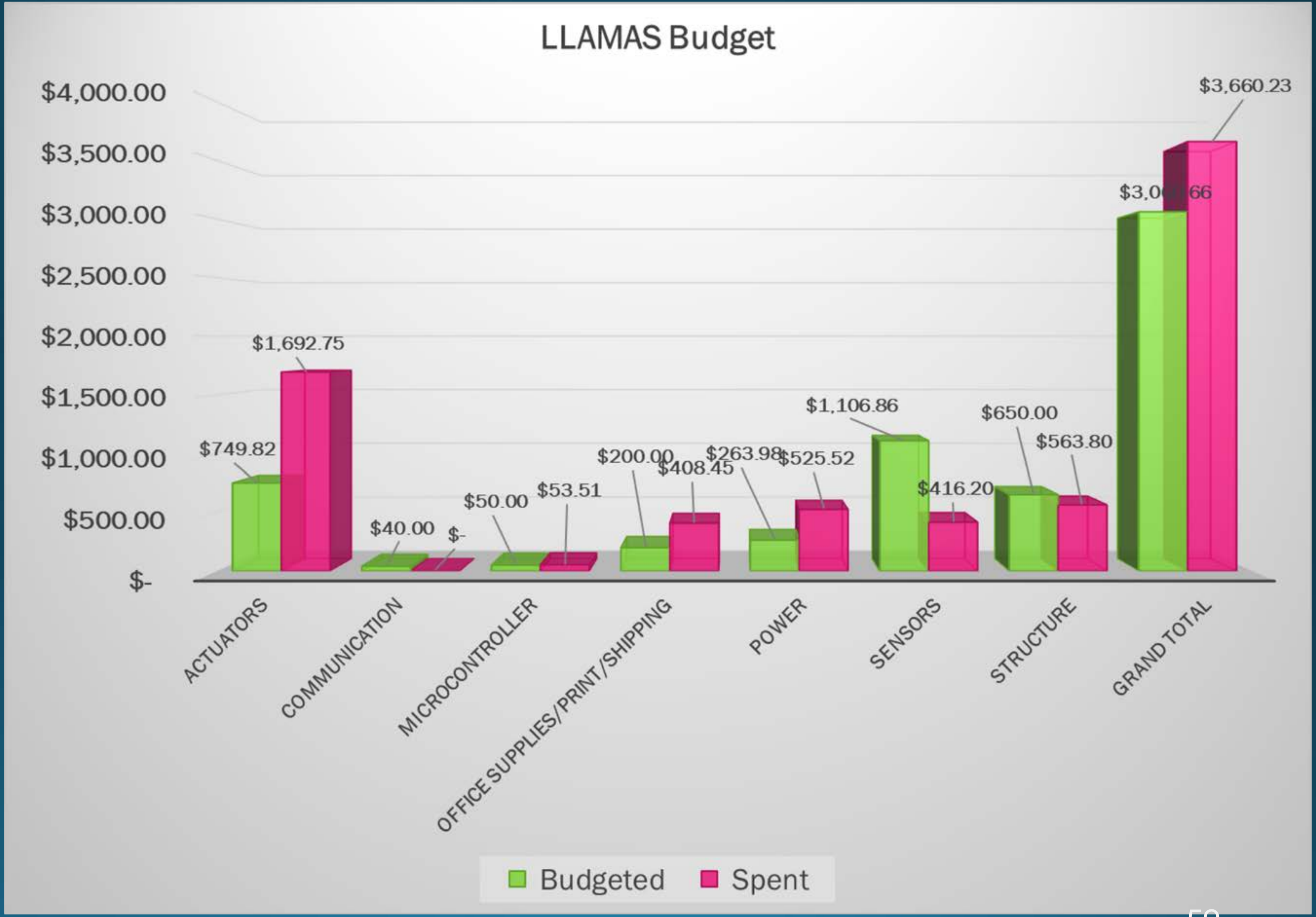
Project Management

Project Management – Lessons Learned

- Everything will take longer than anticipated
  - “$2\pi5$” Rule: should use this to overestimate the time needed based on experience
- Communication is important in every aspect of development
  - Between customer and team to develop project requirements and progress
  - Between subsystems to ensure compatibility
  - Between team members to foster efficient progress

Project Management - Budget

- Discrepancies derive from:
    - Purchase of extra motors to test two different families of motors
    - Borrowing communication components
- Identified at TRR to buy down risk. Money spent on additional:
    - Motor Controllers
    - Motor Adaptors
    - PCBs



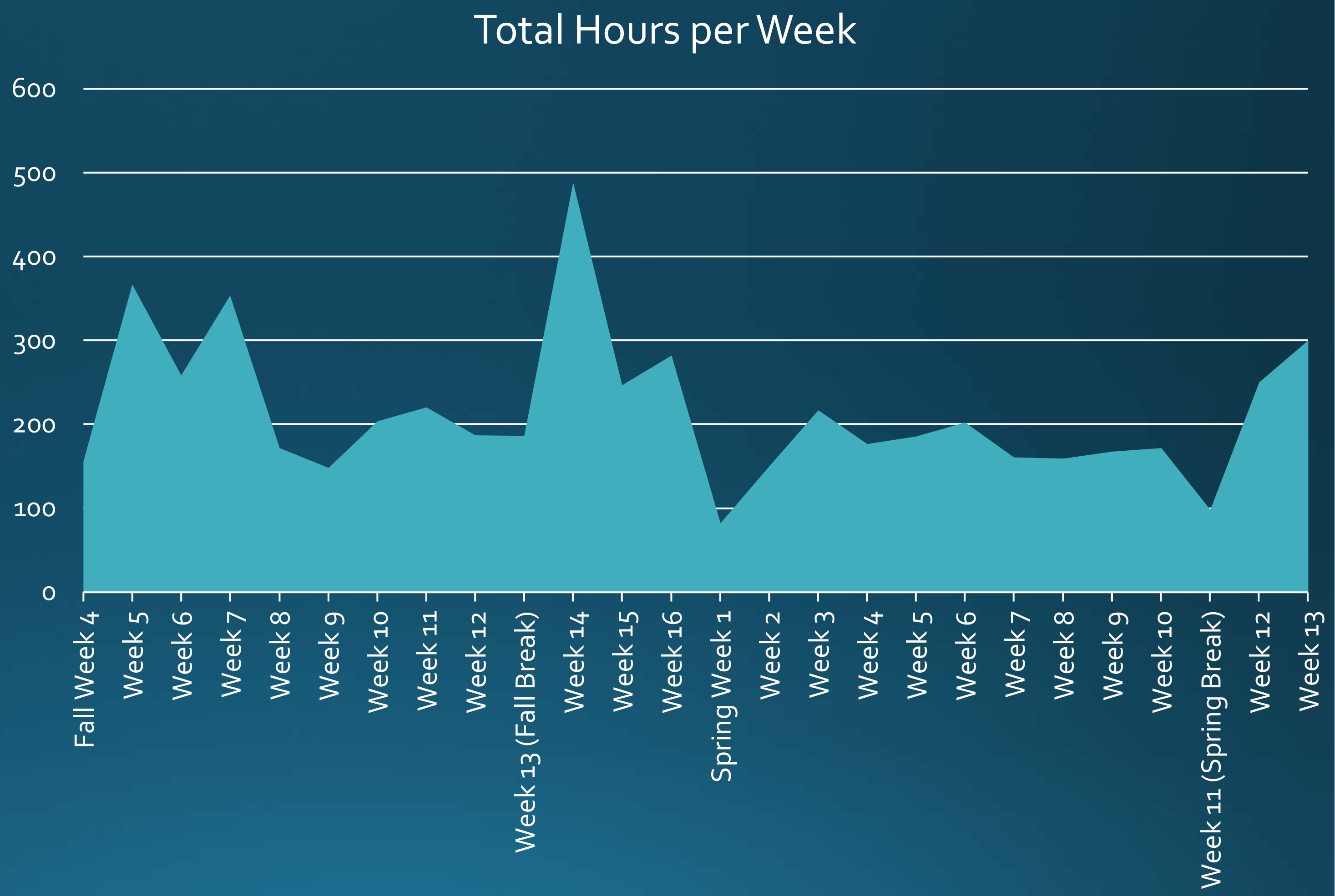
Project Management – Work Hours

Industry cost analysis:

- Total Hours = 4,644.5
- Cost per Hour = $30.77

- Total Hour Cost = $142,941.5
- Materials Cost = $3,660.23

- Project Cost = $146,601.8



LLAMAS Team

Backup Slides

Design Description – Fault Injection

- **What is a fault?**
  - A fault is anything that causes the mock-satellite to not meet the desired attitude. This can result from bad data collection or malfunctions in hardware. In this system, there are two simulated faults: One is an induced increase in friction in a reaction wheel, simulating a bad bearing or failing reaction wheel. Second is a constant bias in pointing angle introduced in an attitude determination sensor, making the satellite point away from the desired direction.
- **Fault Injection**
  - Reaction Wheel Fault
    - Reaction wheels operate by commanding a torque to reach a desired pointing angle. This known torque is then fed to the motor with an additional commanded friction torque via fault injection.
  - Pixy Fault
    - Pixy data relays position of the reference target. Fault injection forces a constant bias of this position and causes the MockSat to consistently lead or follow the reference target.

Test Procedure

Leveling

1.) Move TestTable to desired location where testing will take place and support TestTable wood block so it doesn't move

2.) Use Iphone level feature and adjust table set screws to achieve rough level approximation

3.) Ensure MockSat is disconnected from station keeping apparatus

4.) Turn on air supply and observe direction MockSat translates since exact leveling has not been achieved

5.) Adjust set screws again till MockSat remains in position for no less than 5 seconds (does not rotate/translate on TestTable)

6.) Reintegrate MockSat with station keeping apparatus

7.) Use acrylic spacers to set distance between MockSat and bearing block which ensures the two are parallel which ensures the rod is perpendicular to the MockSat bearing block

8.) Tighten station keeping apparatus down

9.) Connect encoder

Ground station (Hardware)

- 1.) Plug MYRIO into power and computer USB
- 2.) Plug breadboard into channel A of the MYRIO
- 3.) Make following connections:
  - Arduino TX into breadboard RX
  - Arduino RX into breadboard TX
  - Ground Arduino to ground breadboard
  - Channel A on encoder to DIO11 on breadboard
  - Channel B on encoder to DIO12 on breadboard
  - Ground on encoder and breadboard
  - Arduino digital pin 2 to brown wire on ribbon cable to stepper motor
  - Arduino digital pin 3 to red wire of the same ribbon cable
- 4.) Plug 34-pin ribbon cable from XBee to channel B of MYRIO
- 5.) Plug stepper motor into power supply (24V)

Ground station (Software)

- 1.) Press "do nothing" on starting wizard after MYRIO is plugged in
- 2.) Open up "NI LabView 32-bit" application
- 3.) Click on file name "LLAMAS GROUND STATION.lvproj"
- 4.) In explorer window: right click on the MYRIO and press connect
- 5.) Click "+" and then click on "LLAMAS GROUND STATION"
- 6.) Run VI using arrow in top left
- 7.) Input data

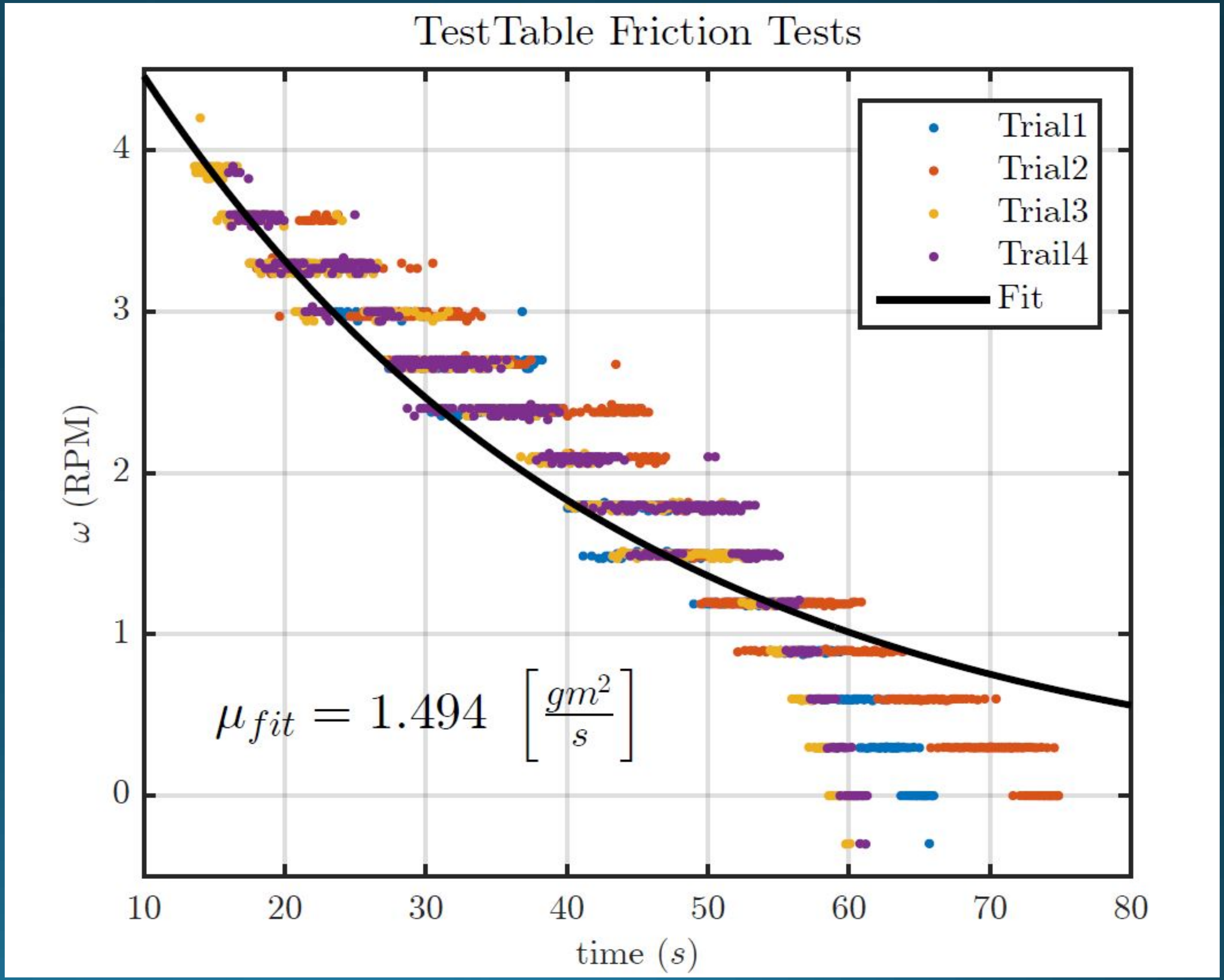
TestTable Friction Quantification: Results

Results:

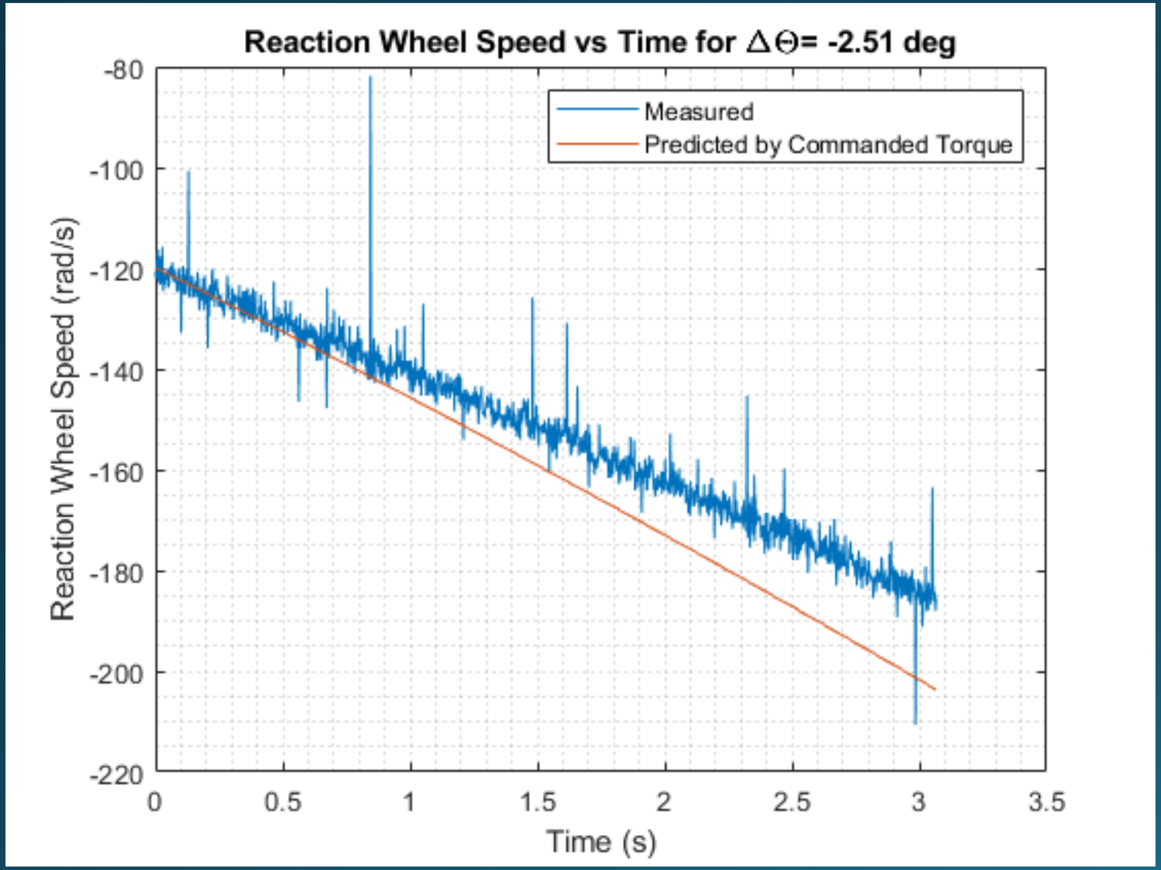
- μ = 5.105 lbm·in2·sec-1, considerably higher than anticipated
- Original predicted value was estimated using higher angular velocities than the MockSat operating regime
- Lower MockSat angular velocities produce larger and much more unpredictable coefficient values

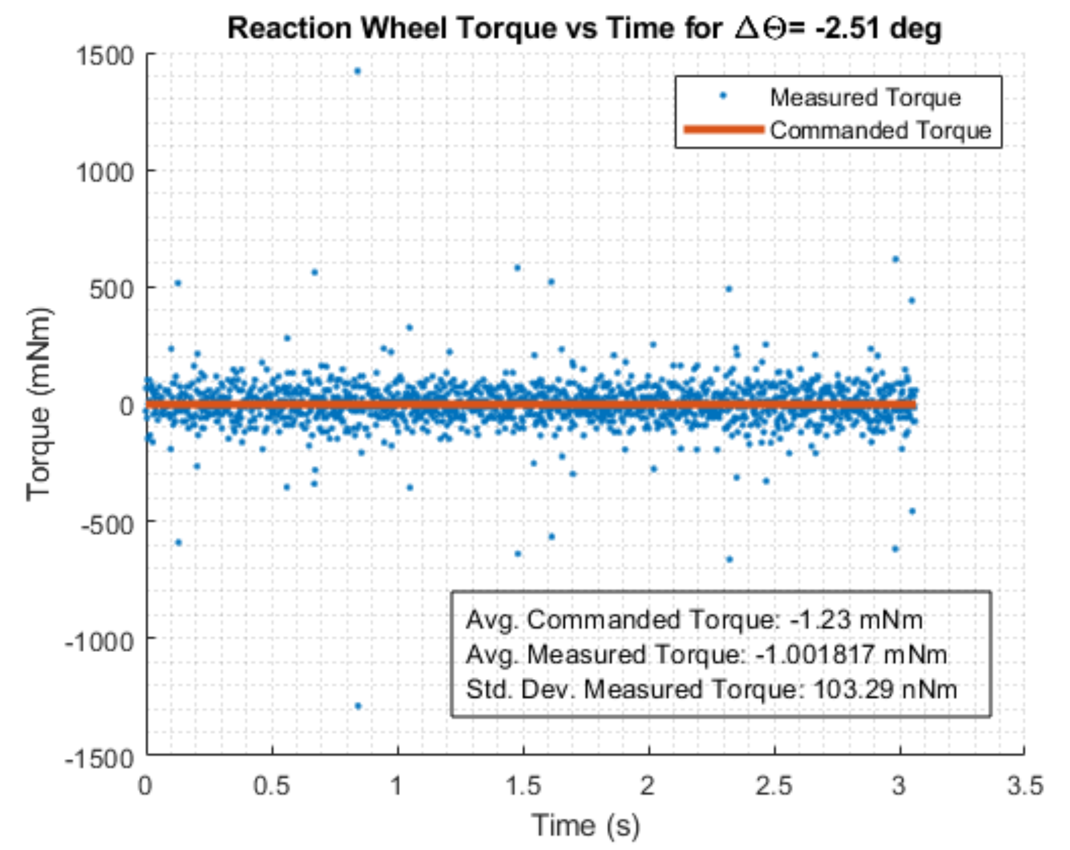
Importance:

- Satisfies FR 1, μ ≤ 1.5 lbm·in2·sec-1
- Value necessary for control system tuning



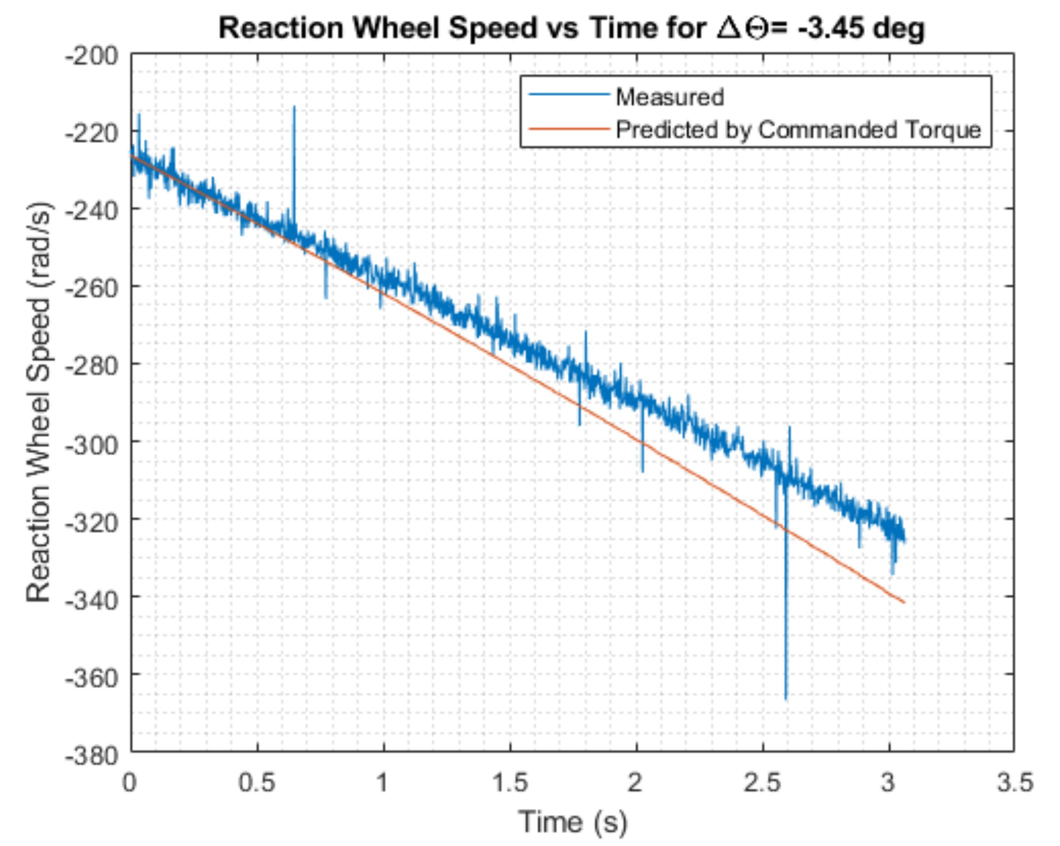
Reaction Wheel Torque Test

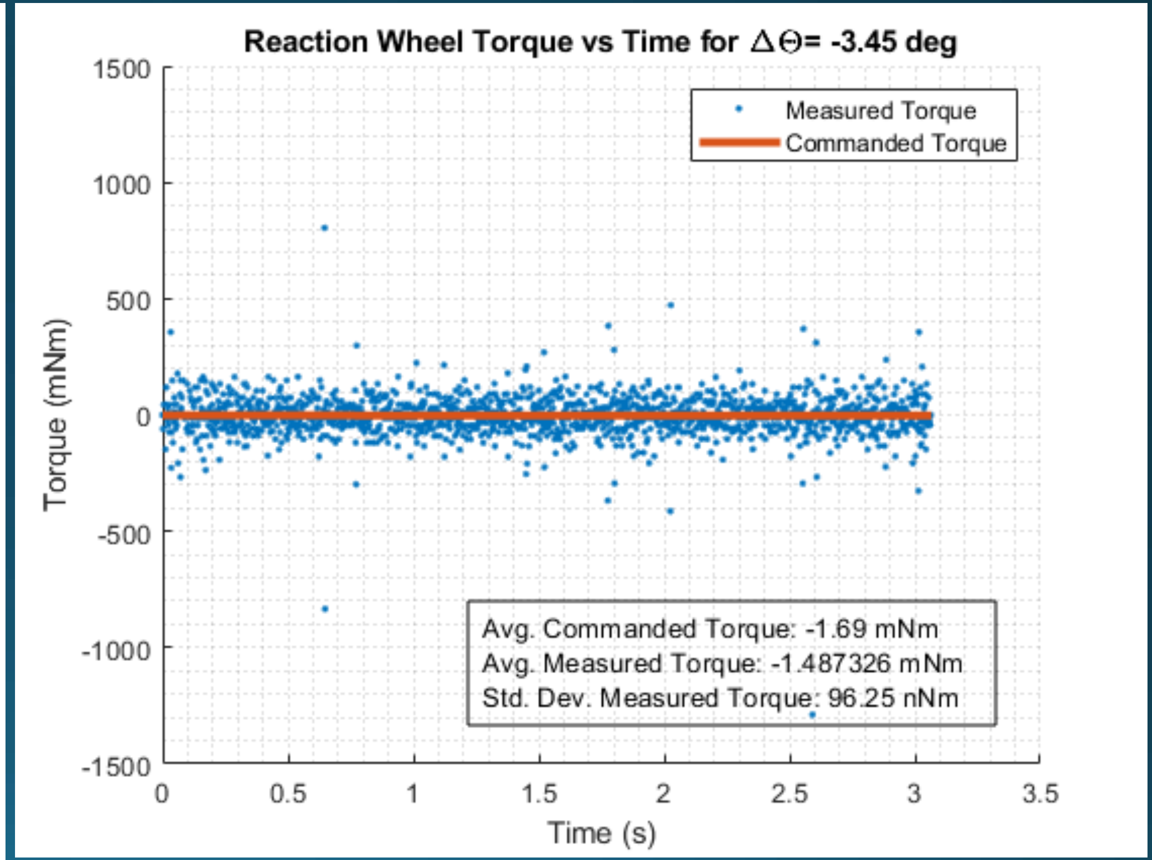




Avg. Commanded Torque: -1.23 mNm
Avg. Measured Torque: -1.001817 mNm
Std. Dev. Measured Torque: 103.29 nNm

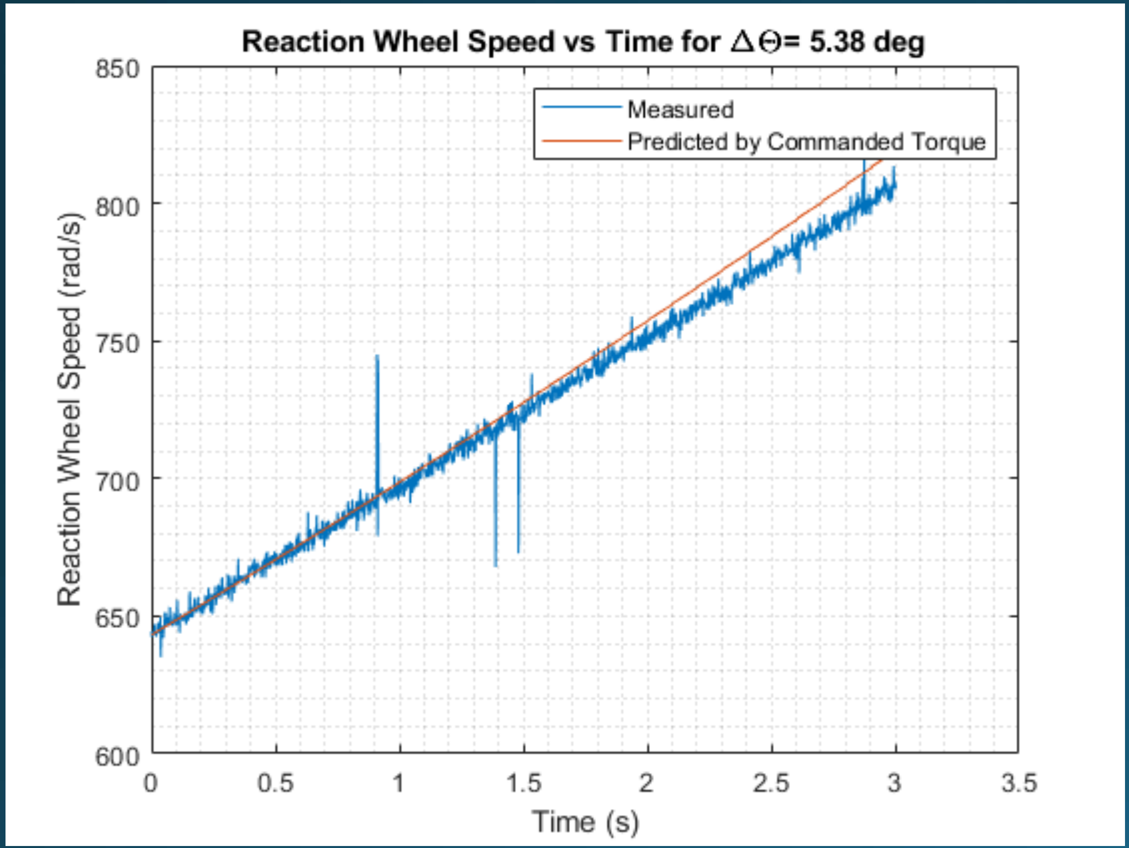
Reaction Wheel Torque Test

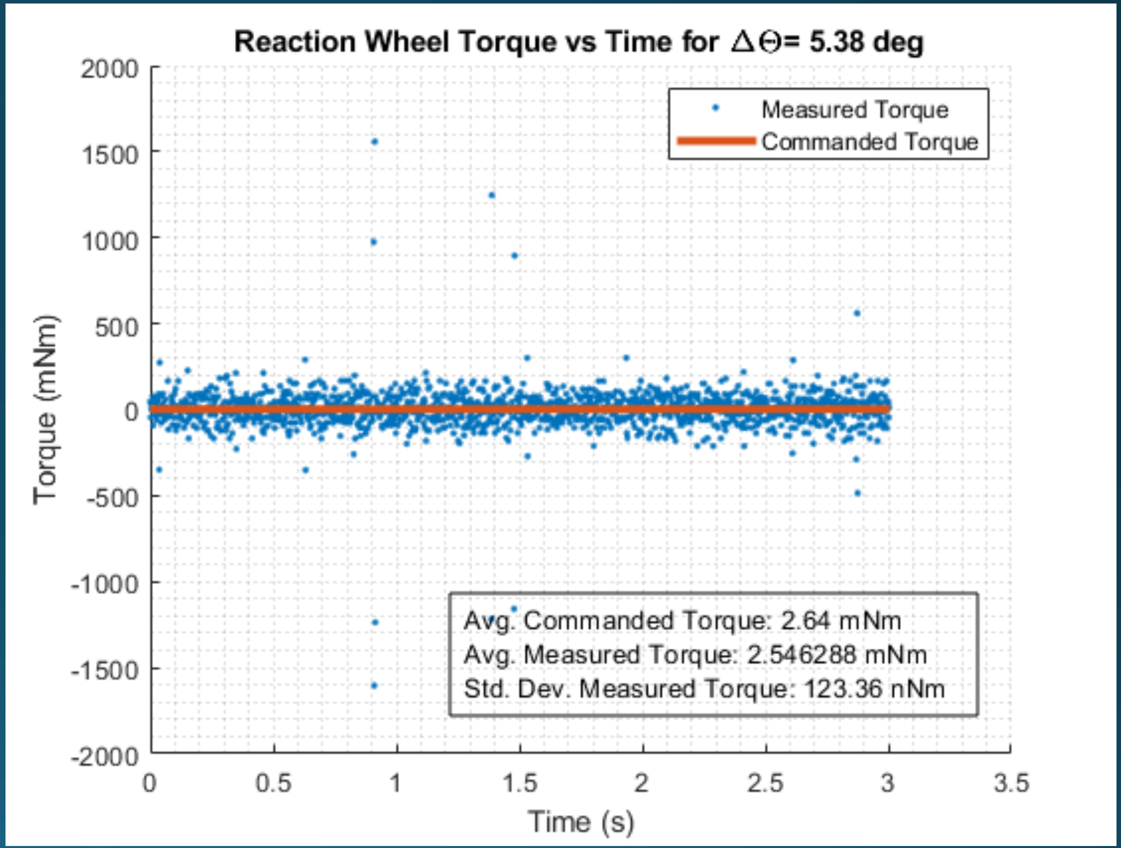




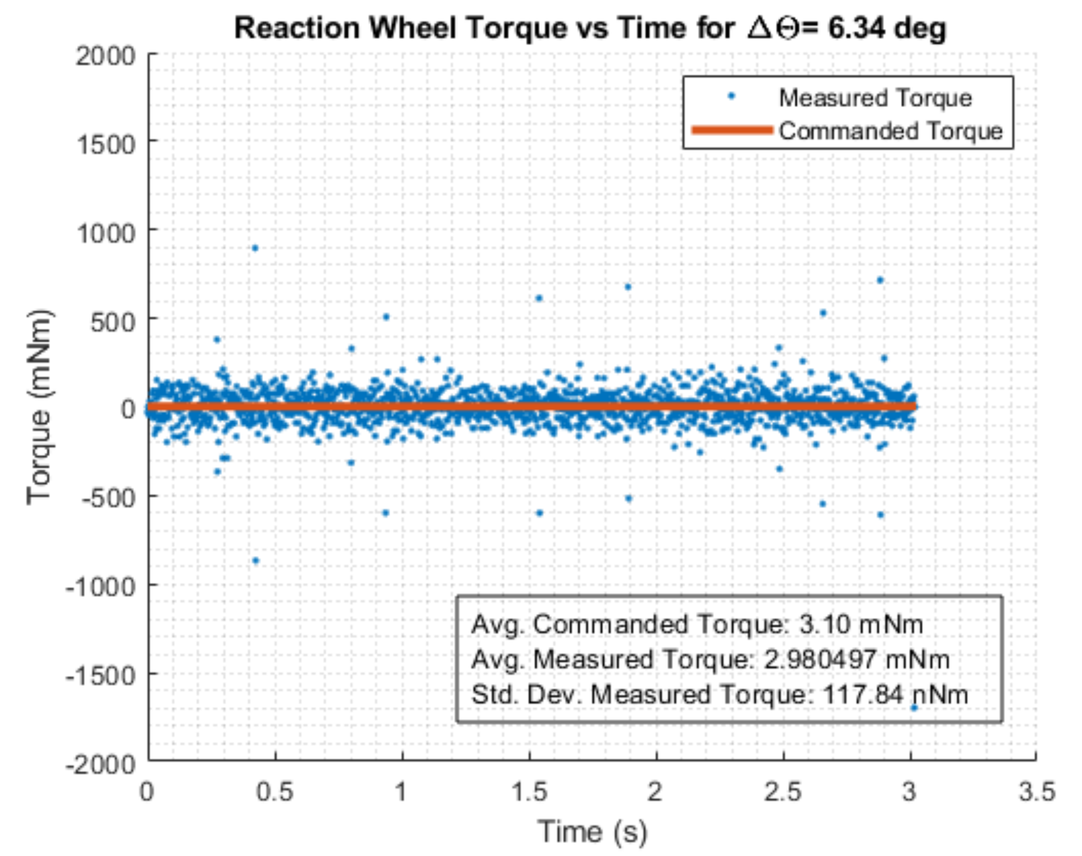
Avg. Commanded Torque: -1.69 mNm
Avg. Measured Torque: -1.487326 mNm
Std. Dev. Measured Torque: 96.25 nNm

Reaction Wheel Torque Test

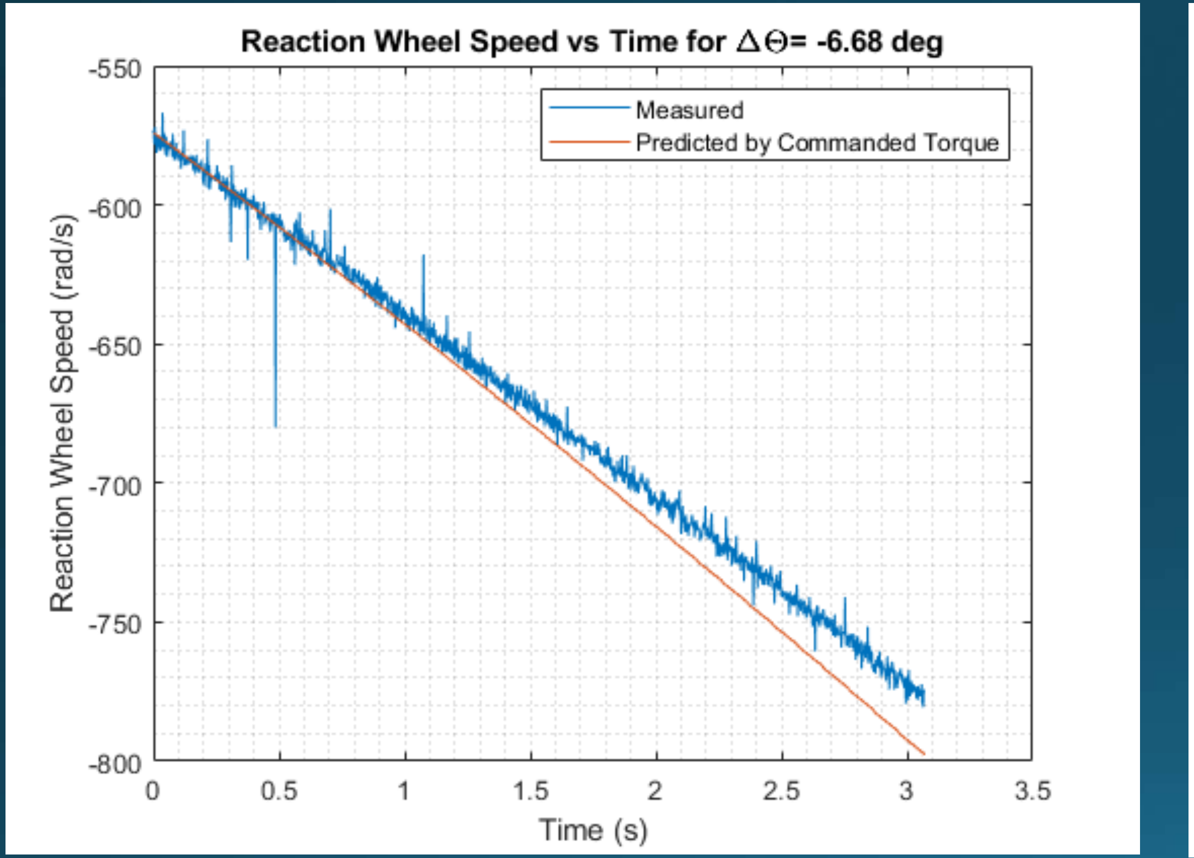

Reaction Wheel Speed vs Time for ΔΘ= 5.38 deg

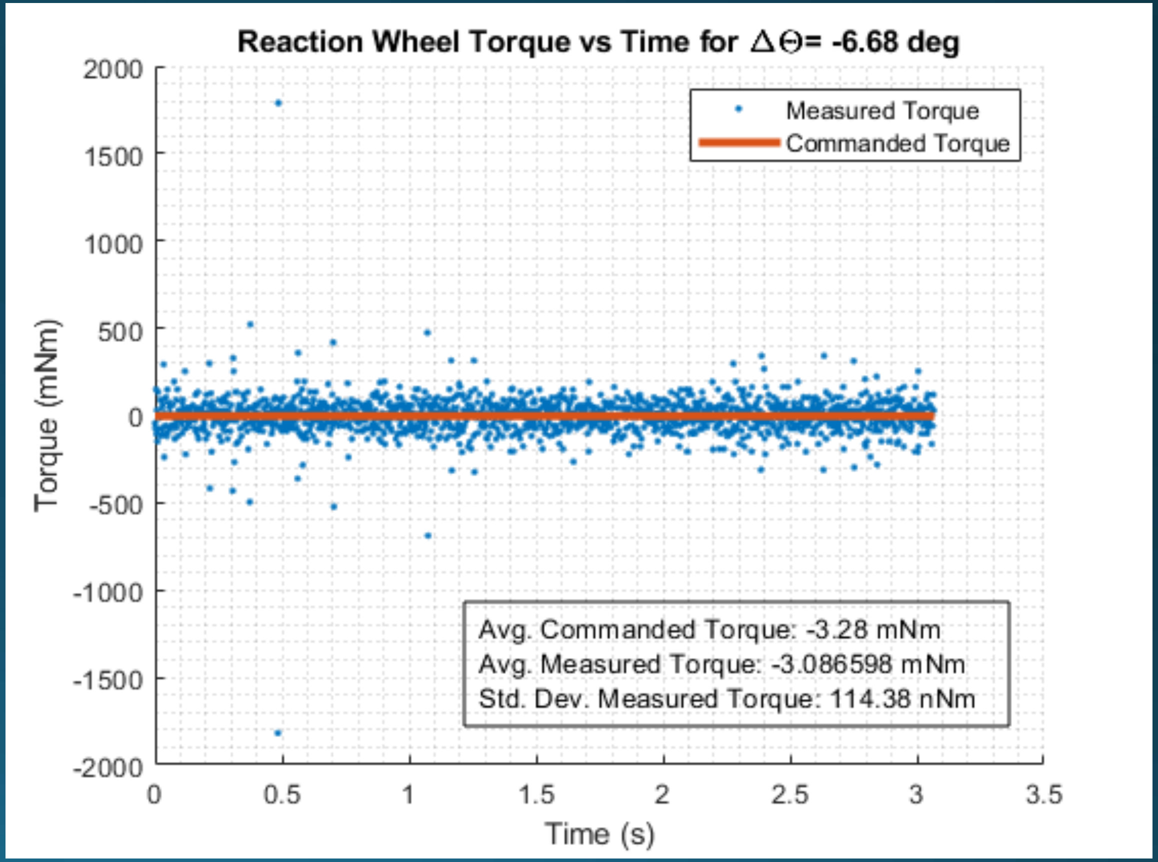

Reaction Wheel Torque vs Time for ΔΘ= 5.38 deg

Reaction Wheel Torque Test

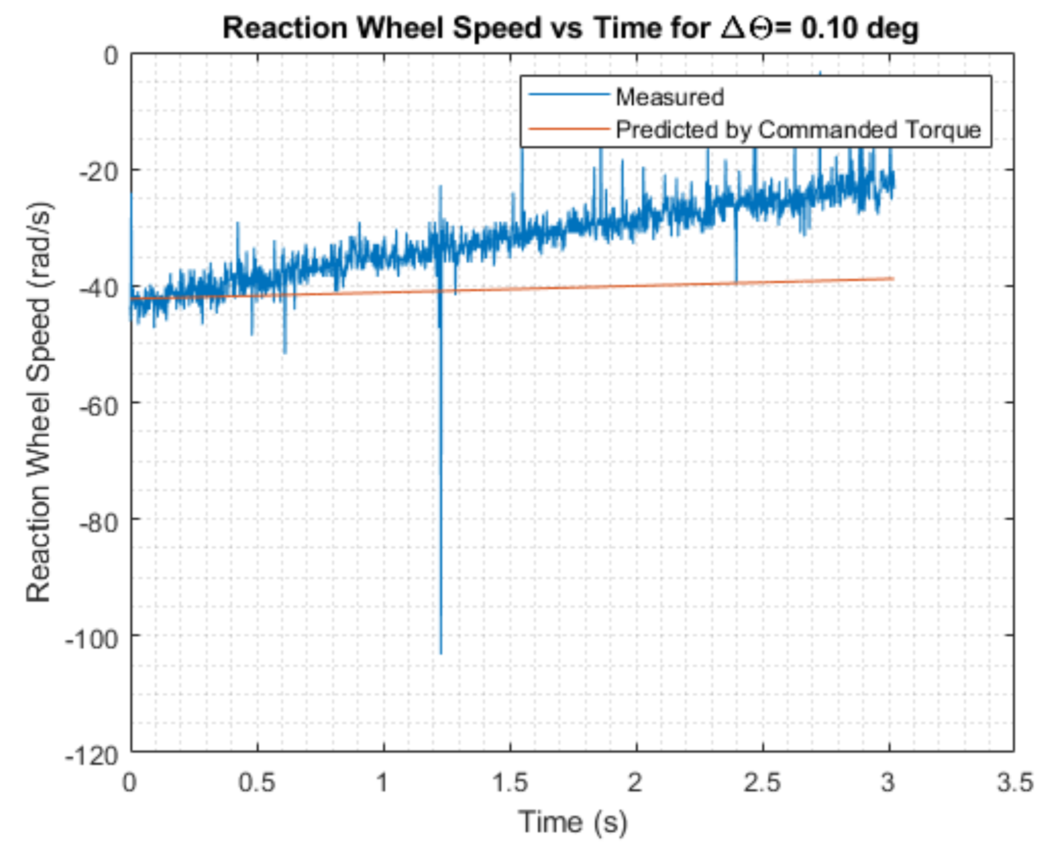


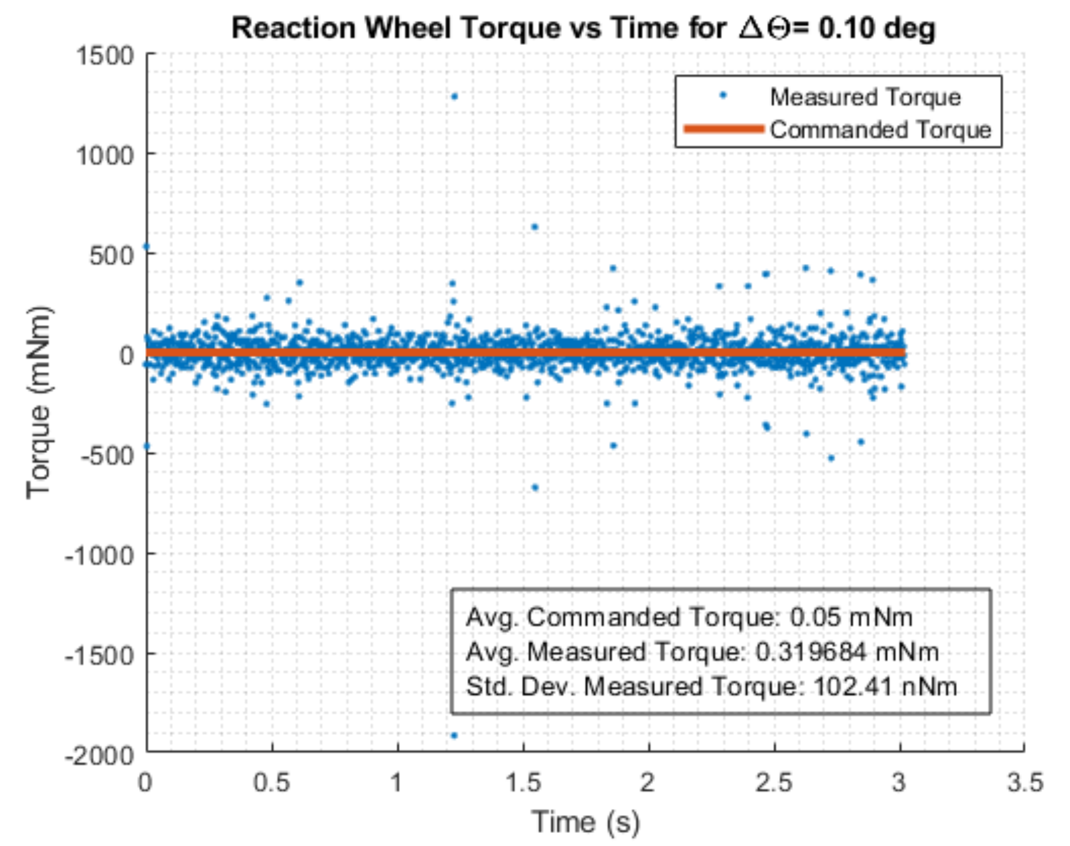
Reaction Wheel Torque Test





Reaction Wheel Torque Test





Risk Assessment

Risk Matrix

Likelihood					
Very Likely	🟩	🟨	🟥	🟥	🟥
Likely	🟩	🟨	🟨	🟥 1	🟥 2
Possible	🟩	🟩	🟨	🟨 3	🟥
Unlikely	🟩	🟩	🟩	🟨	🟨
Very Unlikely	🟩	🟩	🟩	🟩	🟩
🟩 Acceptable 🟨 Marginal 🟥 Unacceptable	Negligible	Minor	Moderate	Significant	Severe
	Severity				

1. Lack of torque resolution
2. Fault Management Implementation
3. ADCS Integration

Risk Assessment – Mitigation

Risk: Lack of torque resolution

Cause: Motors do not have adequate torque resolution

Effect: Unable to provide commanded torque to meet pointing requirements

Risk Mitigation

Action: Proper motor selection, accurate torque characterization

Success Criteria: Pointing requirements satisfied

Old Risk Level: High

New Risk Level: Marginal

Risk Mode: Technological

Risk Assessment – Mitigation

Risk: Fault Management Implementation

Cause: Tailoring a consistent specific response to a generalized suite of hardware

Effect: Fault management system does not work on different sensors and actuators

Risk Mitigation

Action: Create a fault management architecture that attempts to solve the modularity aspect

Success Criteria: End up with a fault management architecture that is applicable to other projects

Old Risk Level: High

New Risk Level: Marginal

Risk Mode: Technological

Risk Assessment – Mitigation

Risk: ADCS Integration

Cause: Breakdown of communication between any of the ADCS components

Effect: ADCS loss of control

Risk Mitigation

Action: Careful system integration and understanding of communication protocols

Success Criteria: ADCS shares and responds to data as anticipated

Old Risk Level: Marginal

New Risk Level: Acceptable

Risk Mode: Technological

Risk Assessment – Post-Mitigation

Risk Matrix

Likelihood	Negligible	Minor	Moderate	Significant	Severe
Very Likely	🟩	🟨	🟥	🟥	🟥
Likely	🟩	🟨	🟨	🟥	🟥
Possible	🟩	🟩	🟨	🟨	🟥
Unlikely	🟩	🟩	🟩	🟨 1	🟨 2
Very Unlikely	🟩	🟩	🟩	🟩 3	🟩

Severity

🟩 Acceptable
🟨 Marginal
🟥 Unacceptable

1. Lack of torque resolution
2. Fault Management Implementation
3. ADCS Integration

Major Trade Studies

Major aspects of the project deemed most important at the beginning of the project:

- TestTable
- Sensors
- Station Keeping
- MCU

The TestTable and Station Keeping were critical as they influenced the conditions that the system would operate in. The sensors would determine how accurately the MockSat would perform and what kind of faults could be injected. The MCU is responsible for ensuring that all the software required for MockSat operation completes in time.

Systems Engineering – Trades: TestTable

- FR1: The TestTable shall allow for two degrees of freedom in translation and in one degree of freedom in rotation in a low friction environment.

- Design Options: Air Table, Ice Table, Air Bearings

Systems Engineering – Trades: Test Table

Design Options	Pros	Cons
Air Table	• Supporting air provided by table (no tanks on-board MockSat) • Heritage, can be reused from previous projects	• Testing area limited to table • Steady air and power supply • Must be leveled
Ice Table	• Melting ice provides thin layer of water to reduce surface friction	• MockSat electronics must be water-resistant • Requires large sub-freezing storage area • Testing must be conducted in cold environment
Air Bearing	• COTS air bearings available	• Air provided by on-board HP air tanks • Requires extremely smooth surface • Minimum 3 air bearings necessary

Systems Engineering – Trades: TestTable

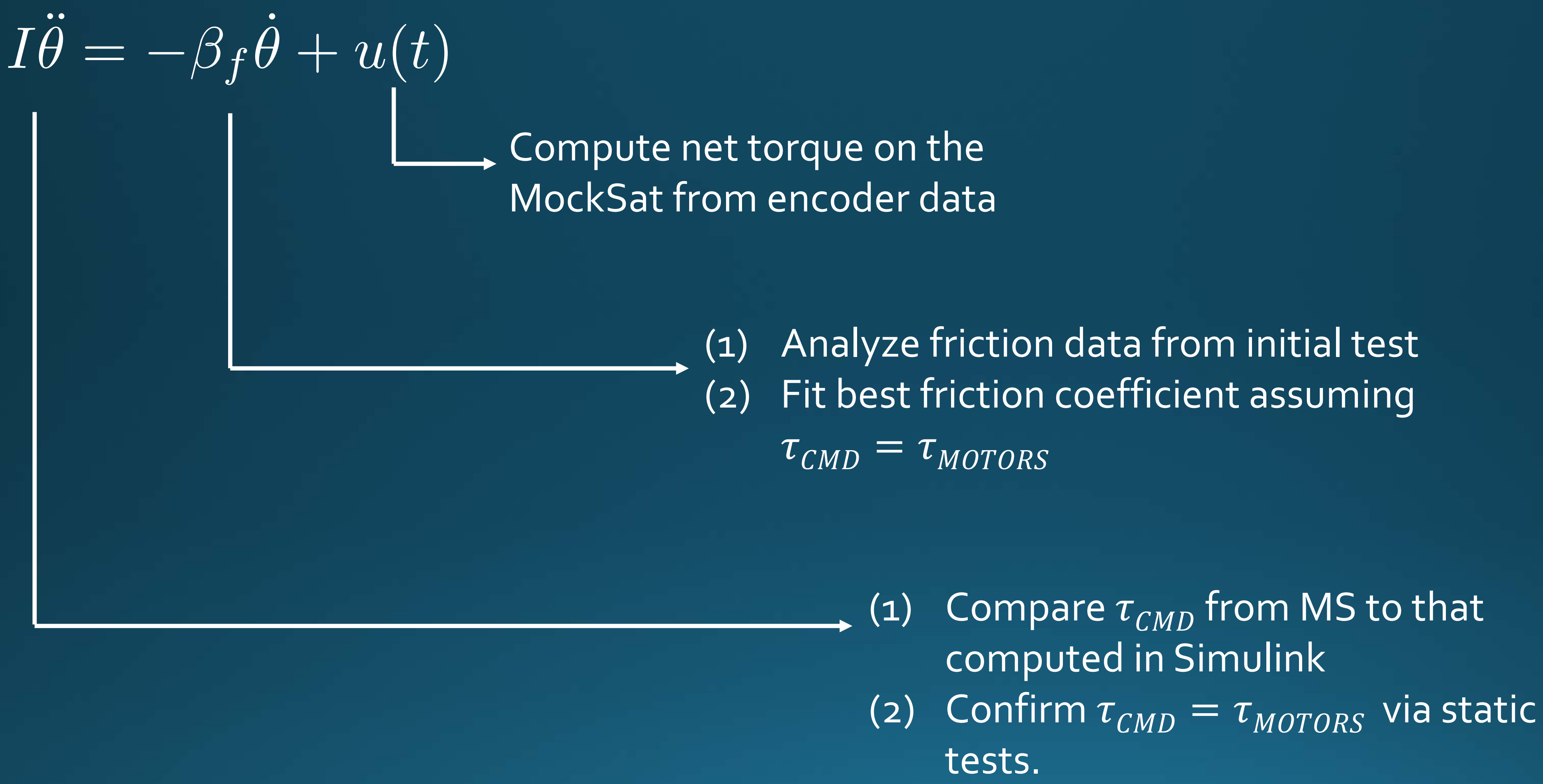
Criterion	Weight	Air Table	Ice Table	Air Bearing
Test Duration	35%	**5**	3	1
Cost	10%	**5**	3	1
Manufacturing Required	15%	**2**	5	4
Heritage	5%	**5**	0	1
Simplicity	15%	**5**	2	1
Logistics	20%	**5**	1	1
Total	100%	**4.55**	2.6	1.45

Score	Criteria
0	Does not fulfill requirement
1	Barely fulfills requirement
2	Marginally fulfills requirement
3	Fulfills requirement
4	Fulfills requirement well
5	Most desirable

Systems Engineering – Trades: Station Keeping

Design Options	Pros	Cons
Air Table	• Supporting air provided by table (no tanks on-board MockSat) • Heritage, can be reused from previous projects	• Testing area limited to table • Steady air and power supply • Must be leveled
Ice Table	• Melting ice provides thin layer of water to reduce surface friction	• MockSat electronics must be water-resistant • Requires large sub-freezing storage area • Testing must be conducted in cold environment
Air Bearing	• COTS air bearings available	• Air provided by on-board HP air tanks • Requires extremely smooth surface • Minimum 3 air bearings necessary

Control Model (where does this go)



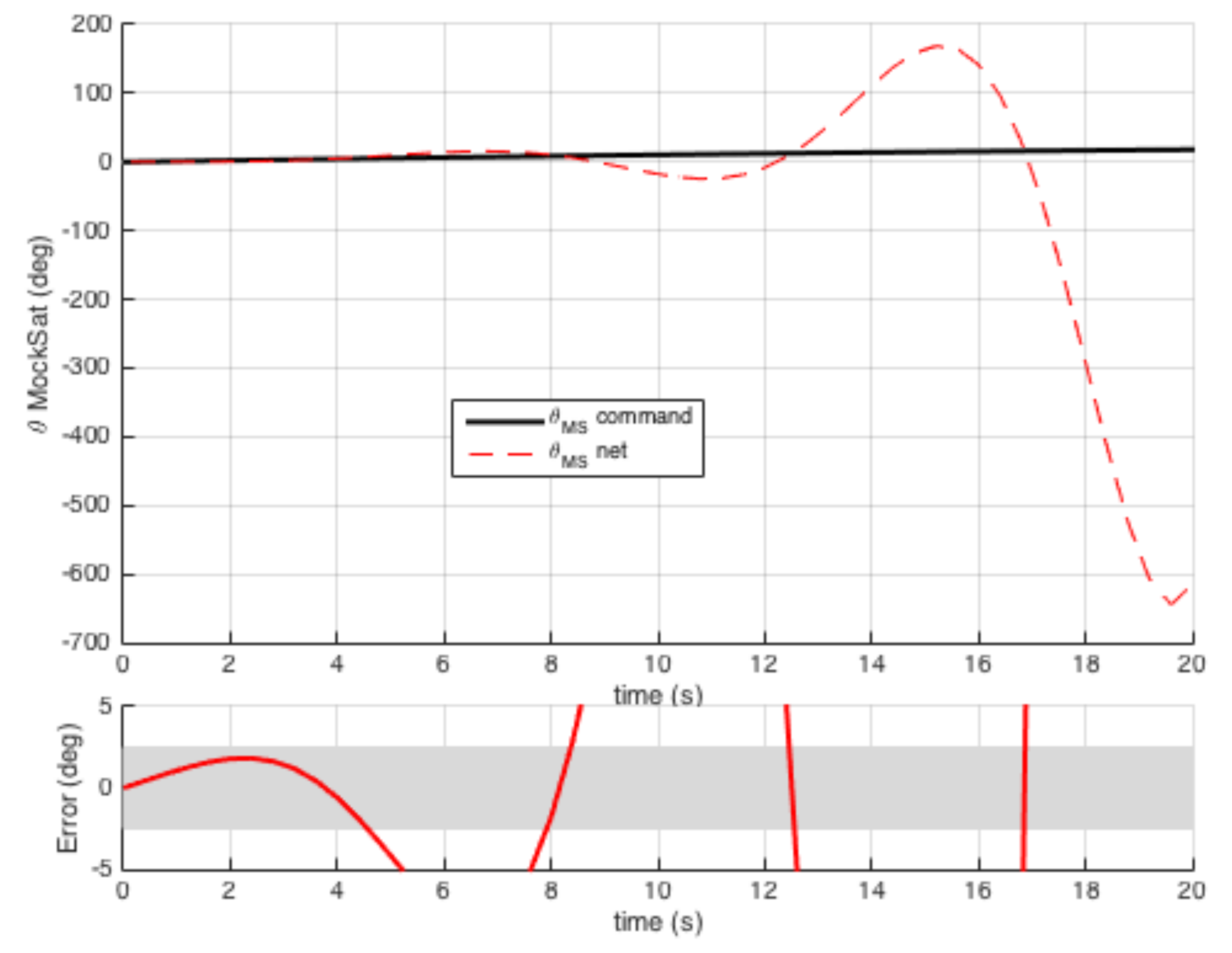
$$I\ddot{\theta} = -\beta_f \dot{\theta} + u(t)$$

Compute net torque on the MockSat from encoder data

(1) Analyze friction data from initial test
(2) Fit best friction coefficient assuming $\tau_{CMD} = \tau_{MOTORS}$

(1) Compare τ_{CMD} from MS to that computed in Simulink
(2) Confirm $\tau_{CMD} = \tau_{MOTORS}$ via static tests.

Simulation with Experimental Gains



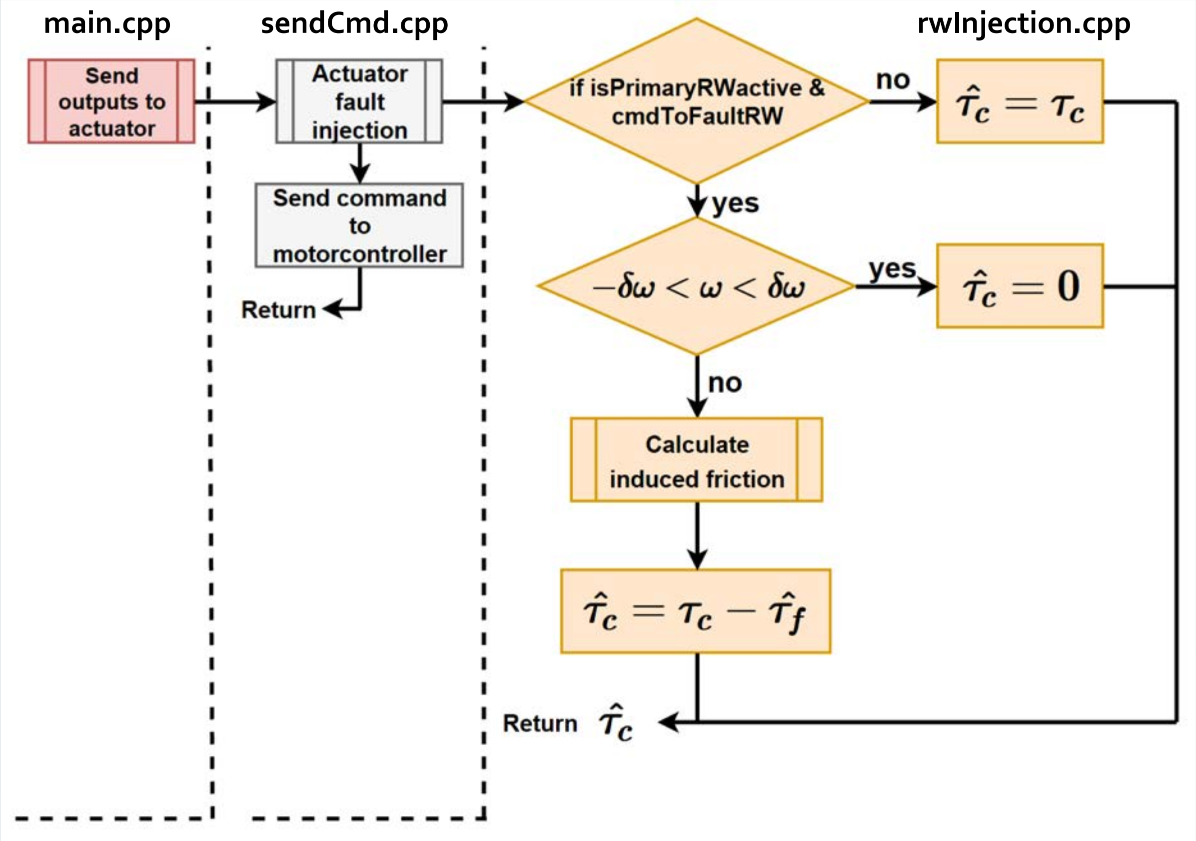
Design Requirements and Satisfaction – Fault Injection and Management

- **FR 4** – The system shall have the ability to introduce a fatal operating fault in either the MockSat's primary reaction wheel or the fine orientation sensor (but not more than one fault at a time).

- **FR 5** – The MockSat flight control software shall recover from a fatal operating fault in either the MockSat's primary reaction wheel or the fine orientation sensor (but not more than one fault at a time) by regaining normal operation.

DR&S – Fault Injection

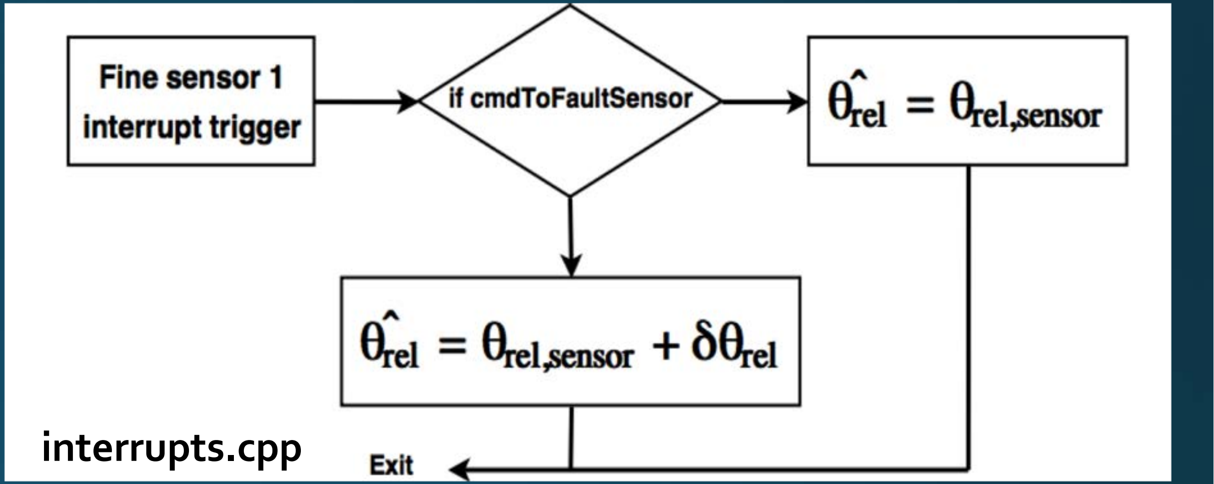
Reaction Wheel Fault Injection



Flowchart for reaction wheel (RW) fault injection

- Reaction wheel fault replicates increased reaction wheel friction by modifying commanded torque
- Increased friction prevents nominal operation, introducing fatal operating fault

Fine Sensor Fault Injection



Flowchart for fine attitude sensor fault injection

- Introduce offset bias in fine sensor data
  - Bias is constant due to interrupt limitations
- This bias causes the satellite to have pointing bias, preventing nominal operation, introducing fatal operating fault

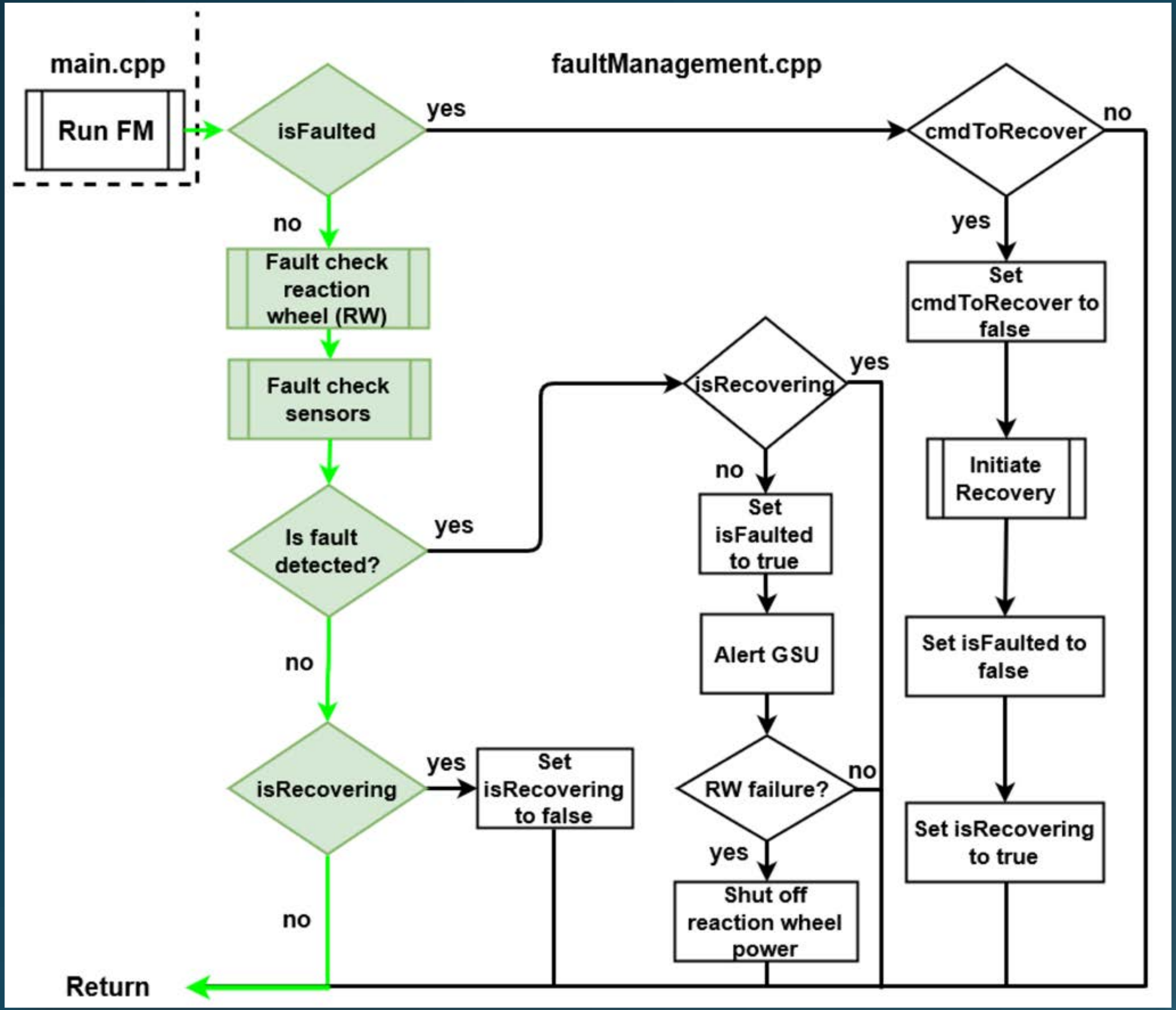
Design Requirements and Satisfaction – Fault Injection and Management

Satisfied

- **FR 4** – The system shall have the ability to introduce a fatal operating fault in either the MockSat's primary reaction wheel or the fine orientation sensor (but not more than one fault at a time).

- **FR 5** – The MockSat flight control software shall recover from a fatal operating fault in either the MockSat's primary reaction wheel or the fine orientation sensor (but not more than one fault at a time) by regaining normal operation.

Design Requirements and Satisfaction – Fault Management

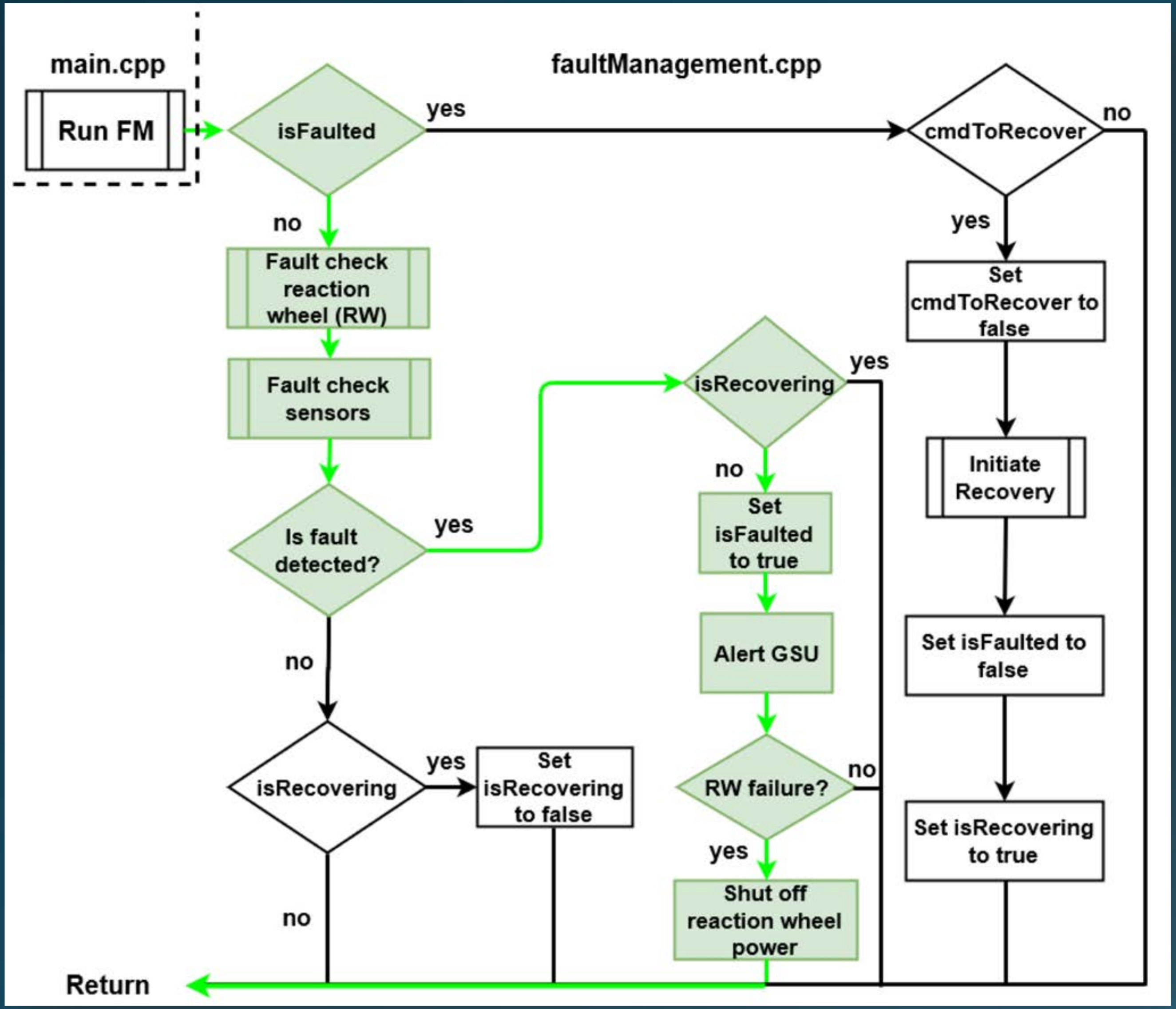


Fault Management Flow Chart: Nominal Operation

Possible MockSat operational states are:

1. Nominal operation
2. Faulted
3. Waiting for Ground Station Unit (GSU)
4. Initiate Recovery Sequence
5. Recovering
6. Recovered

Design Requirements and Satisfaction – Fault Management

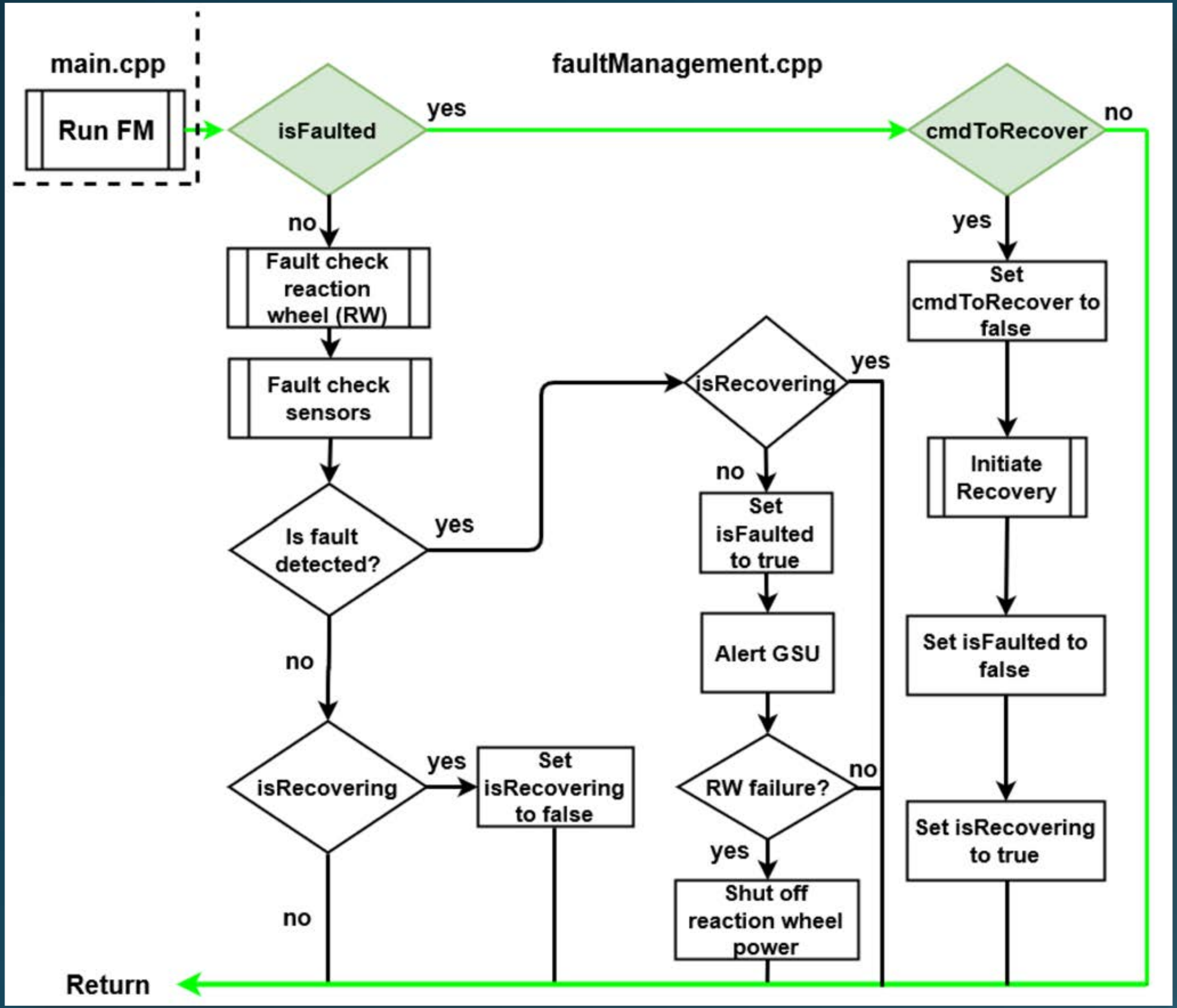


Fault Management Flow Chart: Faulted

Possible MockSat operational states are:

1. Nominal operation
2. Faulted
3. Waiting for Ground Station Unit (GSU)
4. Initiate Recovery Sequence
5. Recovering
6. Recovered

Design Requirements and Satisfaction – Fault Management

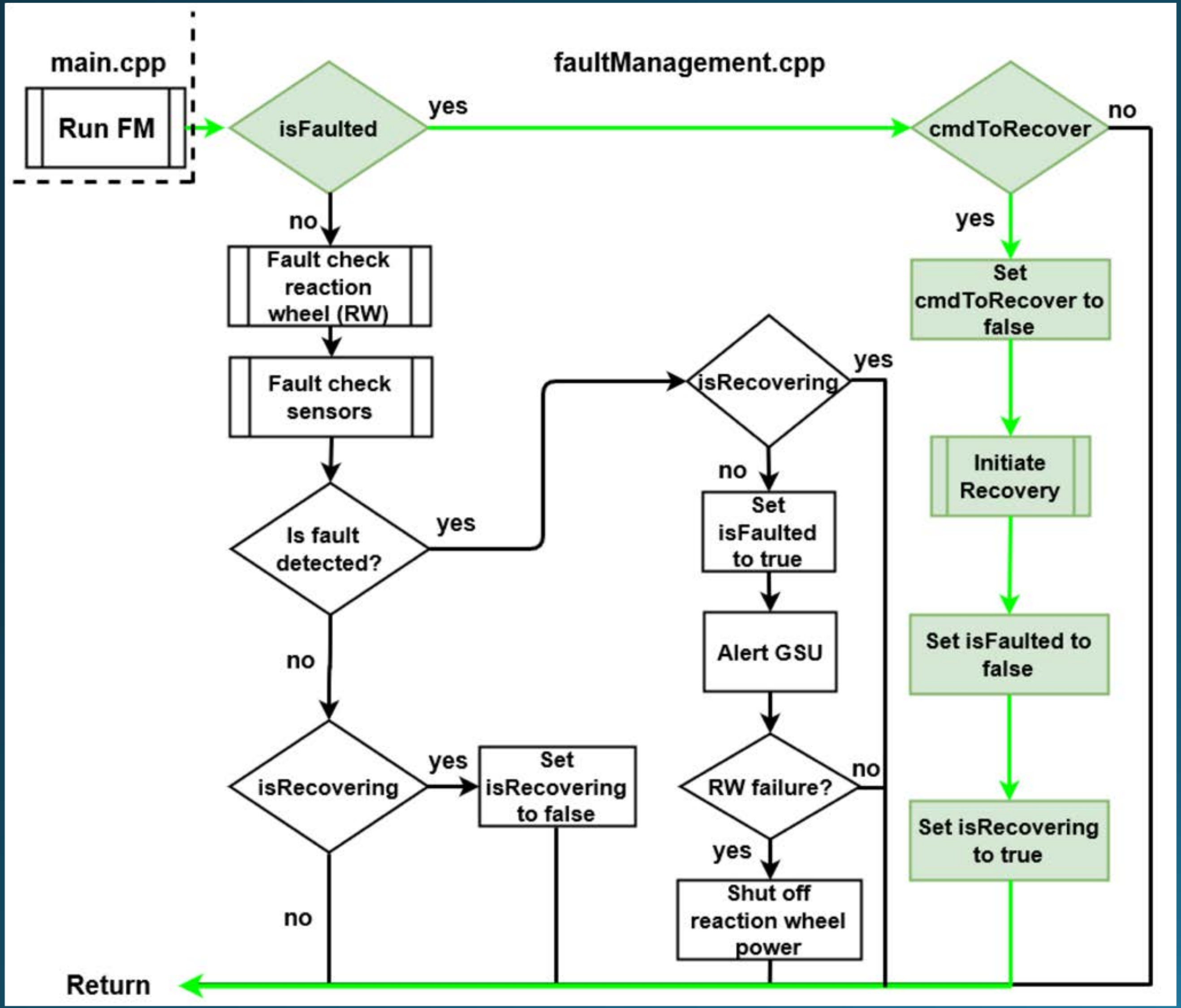


Fault Management Flow Chart: Waiting for Ground Station

Possible MockSat operational states are:

1. Nominal operation
2. Faulted
3. Waiting for Ground Station Unit (GSU)
4. Initiate Recovery Sequence
5. Recovering
6. Recovered

Design Requirements and Satisfaction – Fault Management

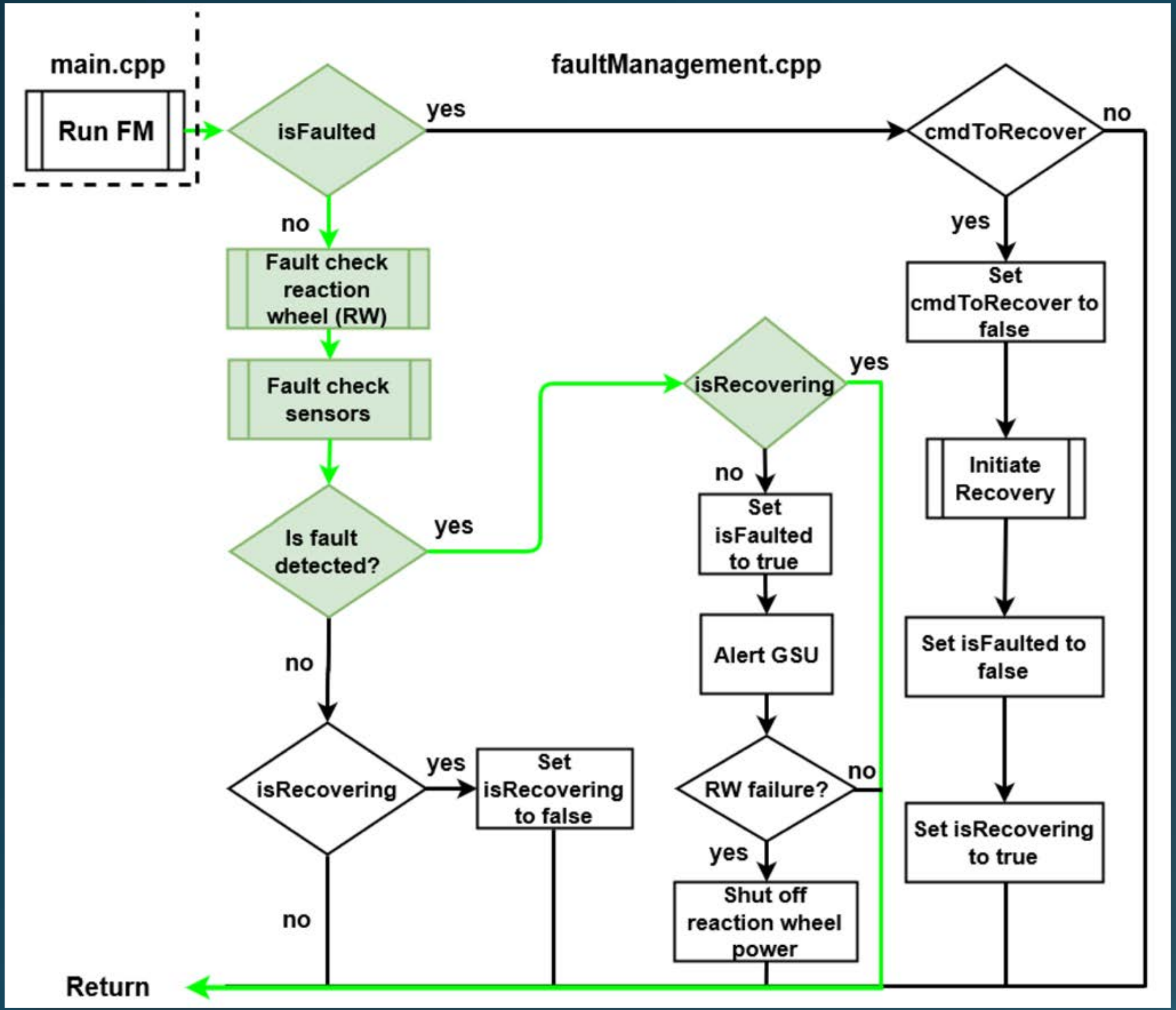


Fault Management Flow Chart: Initiate Recovery Sequence

Possible MockSat operational states are:

1. Nominal operation
2. Faulted
3. Waiting for Ground Station Unit (GSU)
4. Initiate Recovery Sequence
5. Recovering
6. Recovered

Design Requirements and Satisfaction – Fault Management

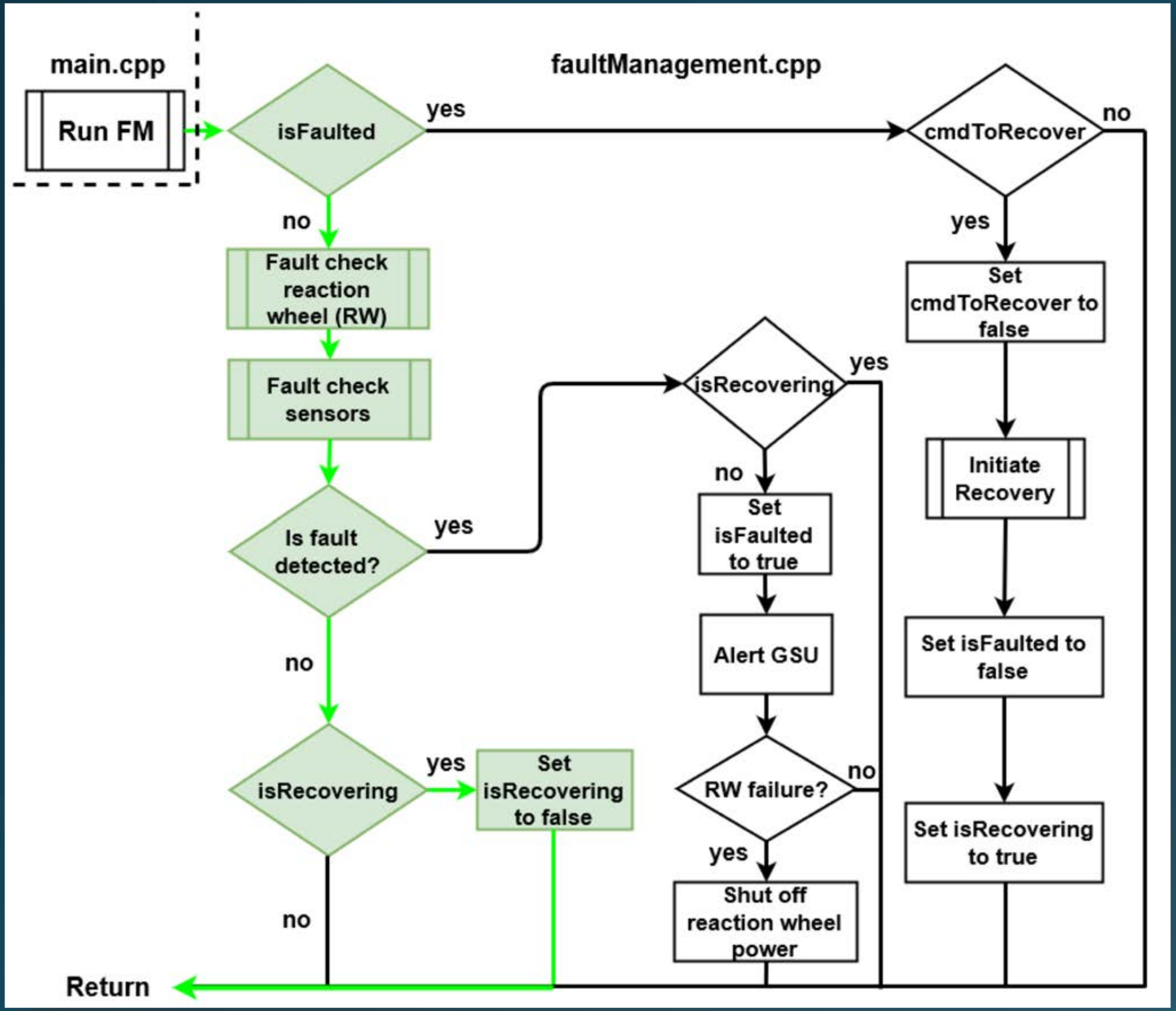


Fault Management Flow Chart: Recovering

Possible MockSat operational states are:

1. Nominal operation
2. Faulted
3. Waiting for Ground Station Unit (GSU)
4. Initiate Recovery Sequence
5. Recovering
6. Recovered

Design Requirements and Satisfaction – Fault Management

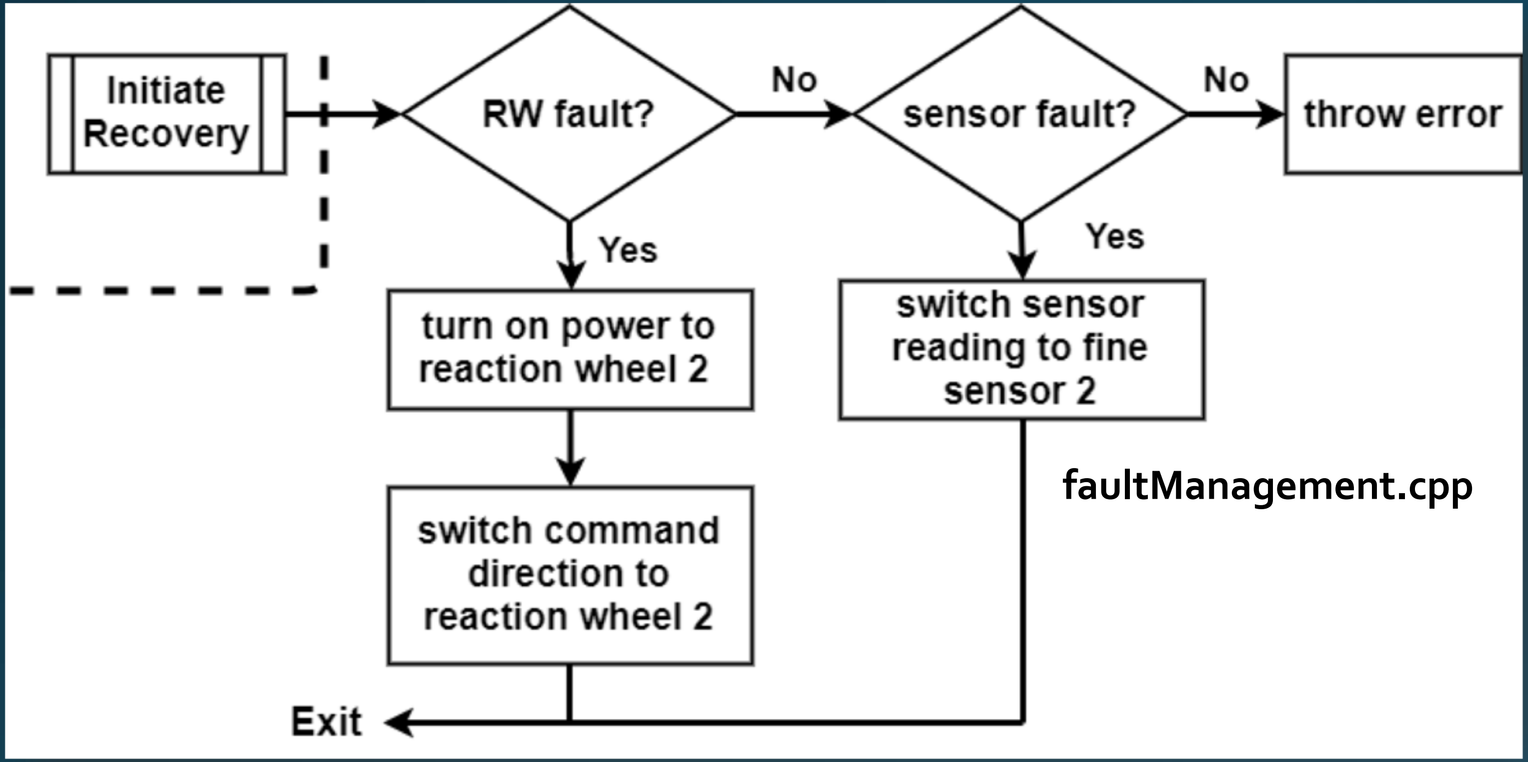


Fault Management Flow Chart: Recovered

Possible MockSat operational states are:

1. Nominal operation
2. Faulted
3. Waiting for Ground Station Unit (GSU)
4. Initiate Recovery Sequence
5. Recovering
6. Recovered

Design Requirements and Satisfaction – Fault Management



faultManagement.cpp

Recovery Flowchart

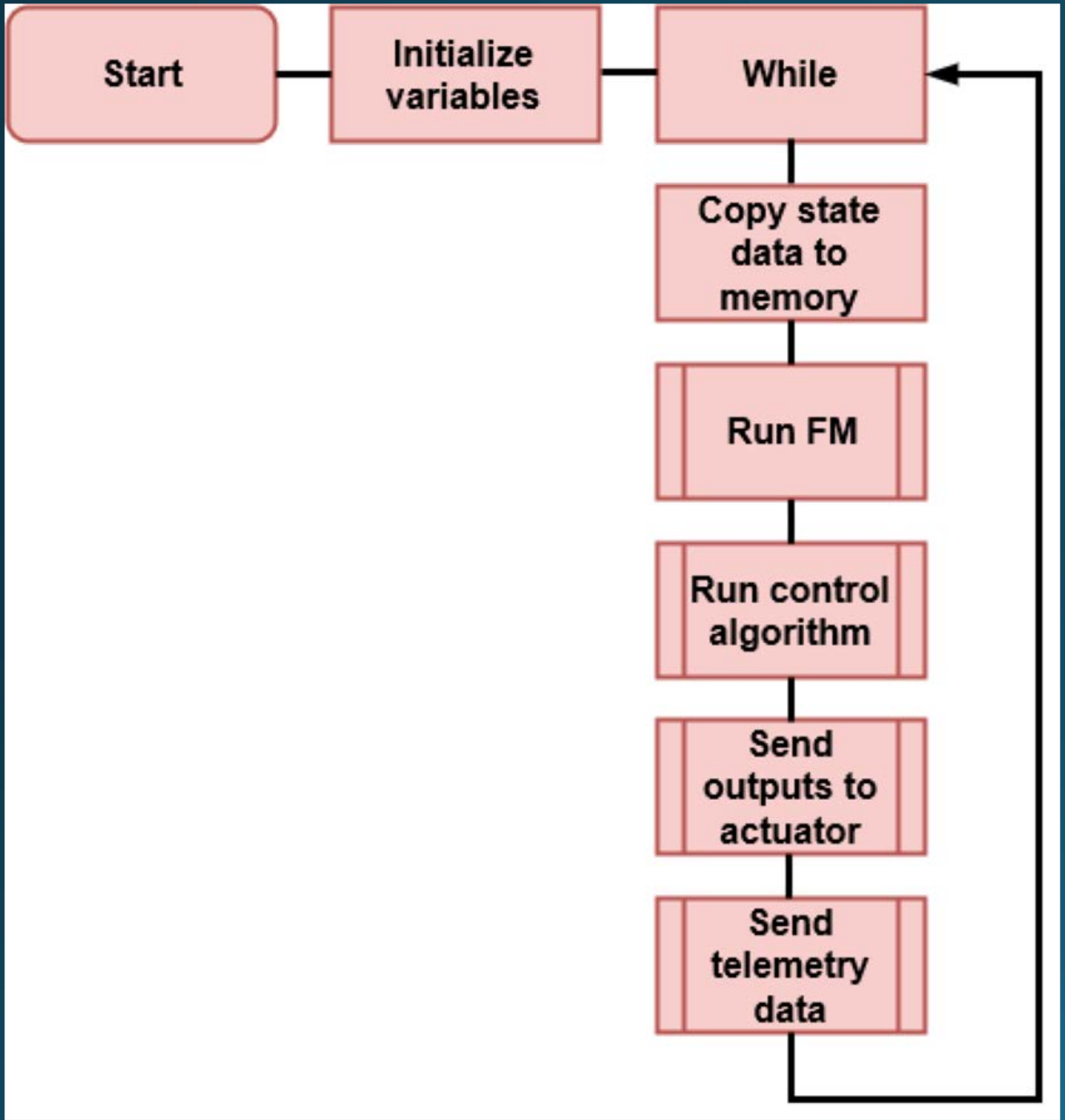
RW Recovery

1. Upon fault detection, shut off power to primary RW
2. Enter a safe mode until primary RW slows to ensure consistent dynamics
3. GSU initiates command to recover
4. Switch power and control to secondary RW

Sensor Recovery

1. Upon fault detection, enter safe mode and wait for GSU command
2. GSU initiates command to recover
3. Switch control to secondary Pixy

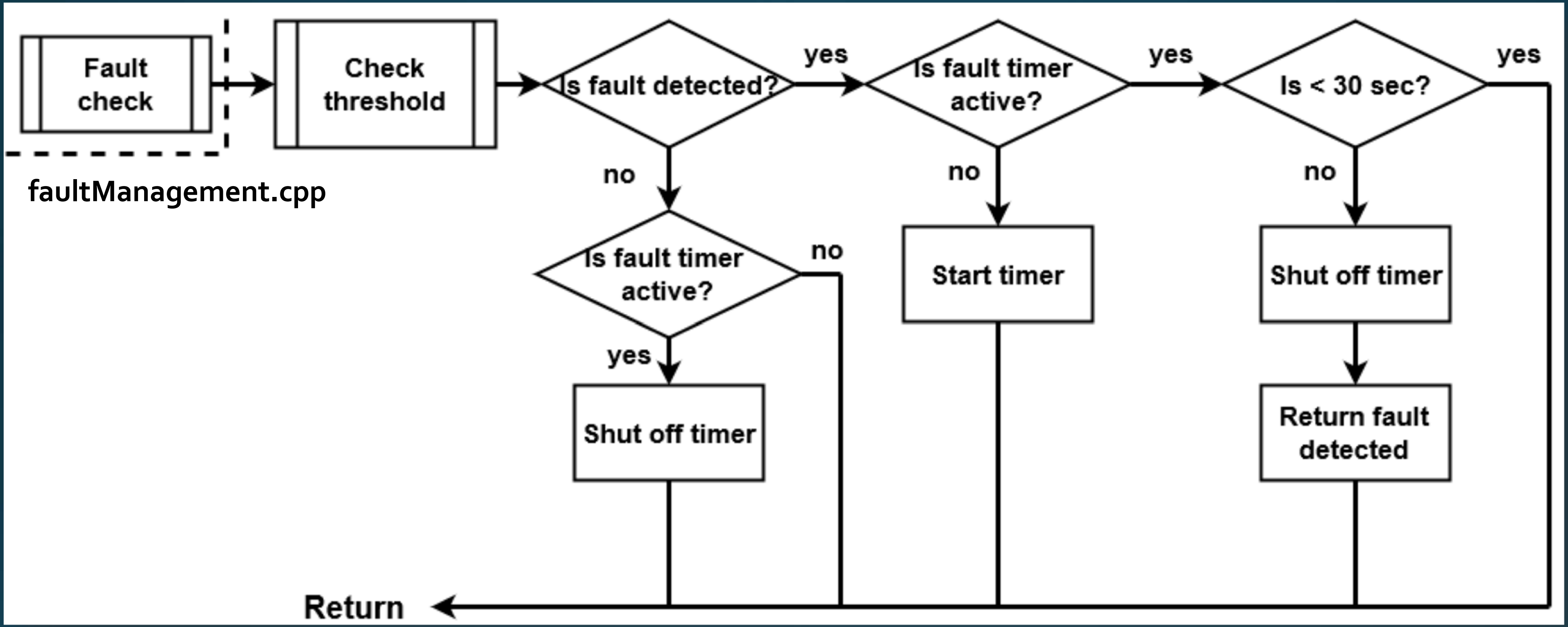
Software: Main Loop



Main.cpp logic flowchart

- Main loop for software operation, runs indefinitely
- Copy current state from all sensors at the beginning of each iteration to ensure data consistency across a loop iteration

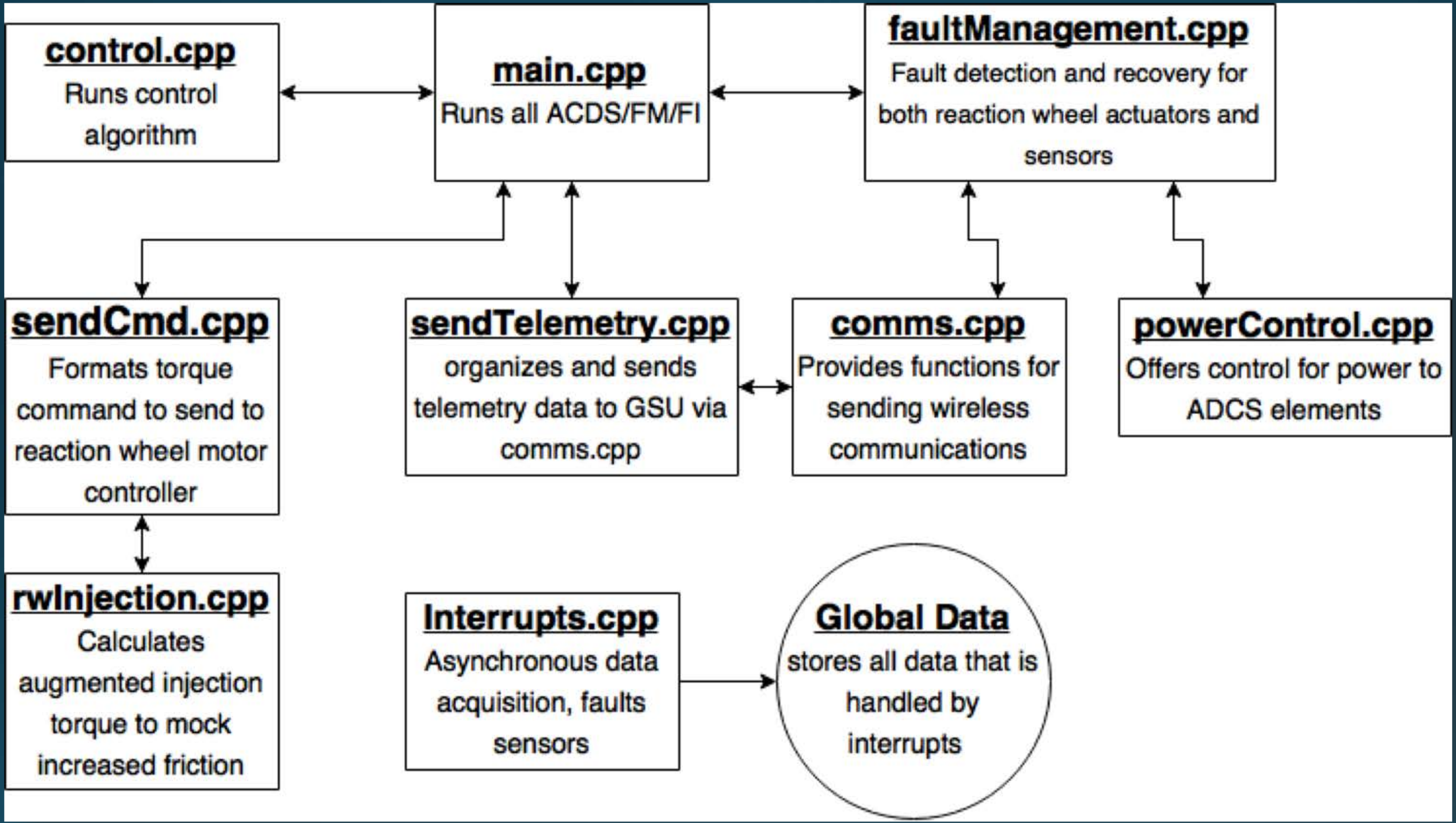
Design Requirements and Satisfaction – Fault Management



Fault checking algorithm flowchart

- Only detect persistent faults
- Use same detection method for RW and fine sensor
- This allows for code re-use, ideally to other systems

Design Solution – Software: Class Diagram



control.cpp
Runs control algorithm

main.cpp
Runs all ACDS/FM/FI

faultManagement.cpp
Fault detection and recovery for both reaction wheel actuators and sensors

sendCmd.cpp
Formats torque command to send to reaction wheel motor controller

sendTelemetry.cpp
organizes and sends telemetry data to GSU via comms.cpp

comms.cpp
Provides functions for sending wireless communications

powerControl.cpp
Offers control for power to ADCS elements

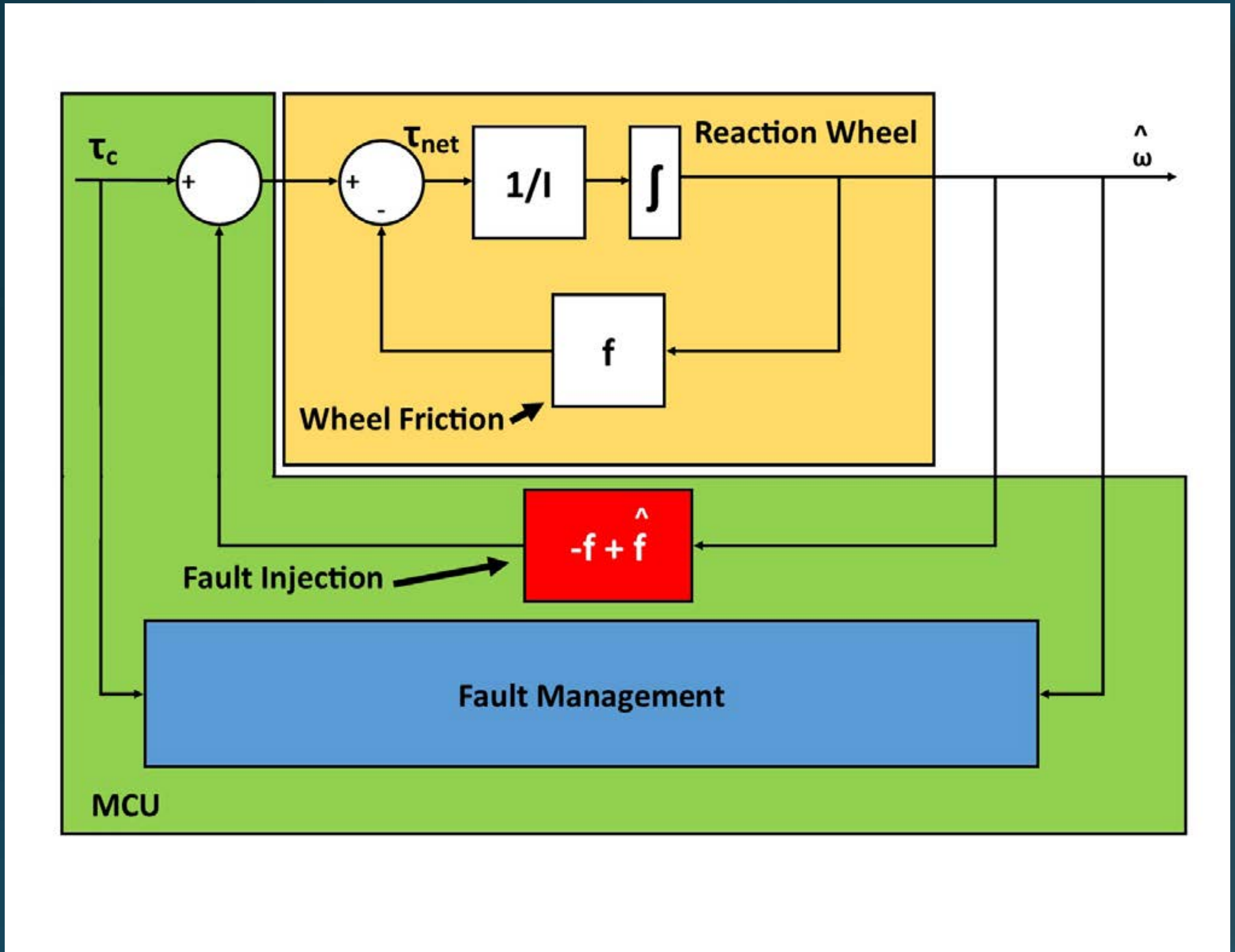
rwInjection.cpp
Calculates augmented injection torque to mock increased friction

Interrupts.cpp
Asynchronous data acquisition, faults sensors

Global Data
stores all data that is handled by interrupts

Class diagram showing major classes/programs, their functions, and their interactions with other software modules

Fault Injection: Reaction Wheel Friction



FBD for induced reaction wheel friction

- Friction is a common and near inevitable fault in the reaction wheels of space systems
- Fault injection system creates apparent friction in software only
  - This DOES NOT *physically* increase the friction in the reaction wheel, but rather it makes the fault management ADCS systems "see" increased friction
- Injects fault into reaction wheel by:
  - Subtracting off nominal friction
  - Adding induced friction function
- Nominal Friction function:

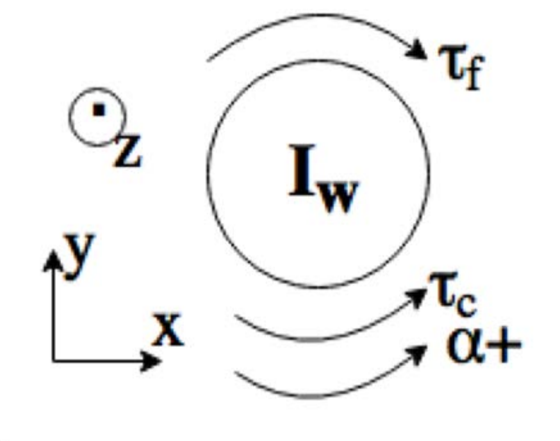
$$\tau_f = f(\omega)$$

- Induced friction function:

$$\hat{\tau}_f = \hat{f}(\omega)$$

Fault Management: Reaction Wheel Friction

Model:



Reaction wheel dynamics

- Governing Equation:

$$\sum \tau = I\alpha \text{ (1-DOF)}$$

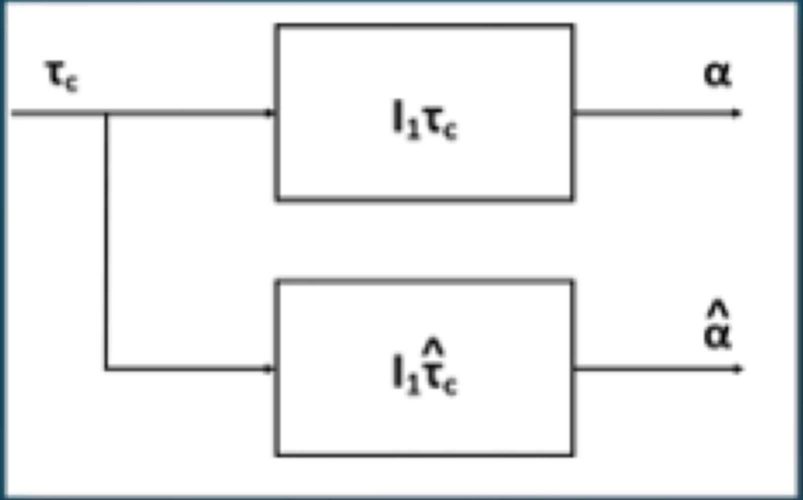
- Nominal friction:

$$\tau_c - \tau_f = I\alpha$$

- Induced Friction:

$$\tau_c - \hat{\tau}_f = I\hat{\alpha}$$

Detection:



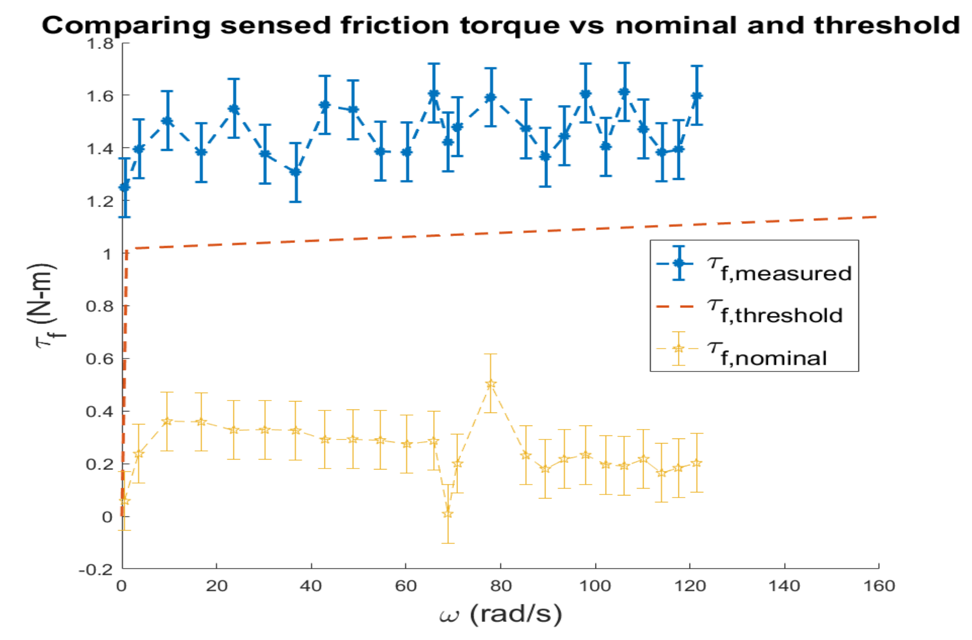
Comparison between model and actual

- Fault Management Process:
  1) Read output wheel speed
  2) Calculate induced friction from governing equation

$$\hat{\tau}_f = I\hat{\alpha} - \tau_c$$

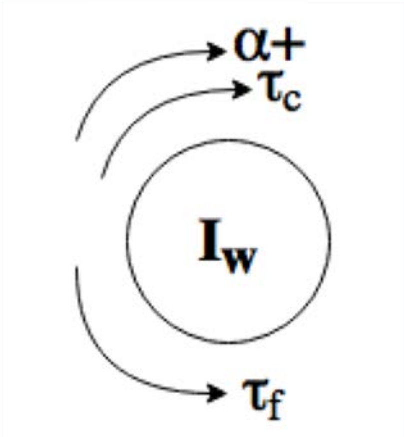
  3) Compare vs model. If friction is above threshold value, then fault exists in system

Feasibility Example:



- ASEN 3200 spin module data used to create nominal friction function
- Induced friction function used to inject fault
- Modeled using governing equation and Matlab's ode45 solver
- Friction in system is greater than threshold value, therefore this is feasible

CPE – Fault Injection and Management System: Actuators



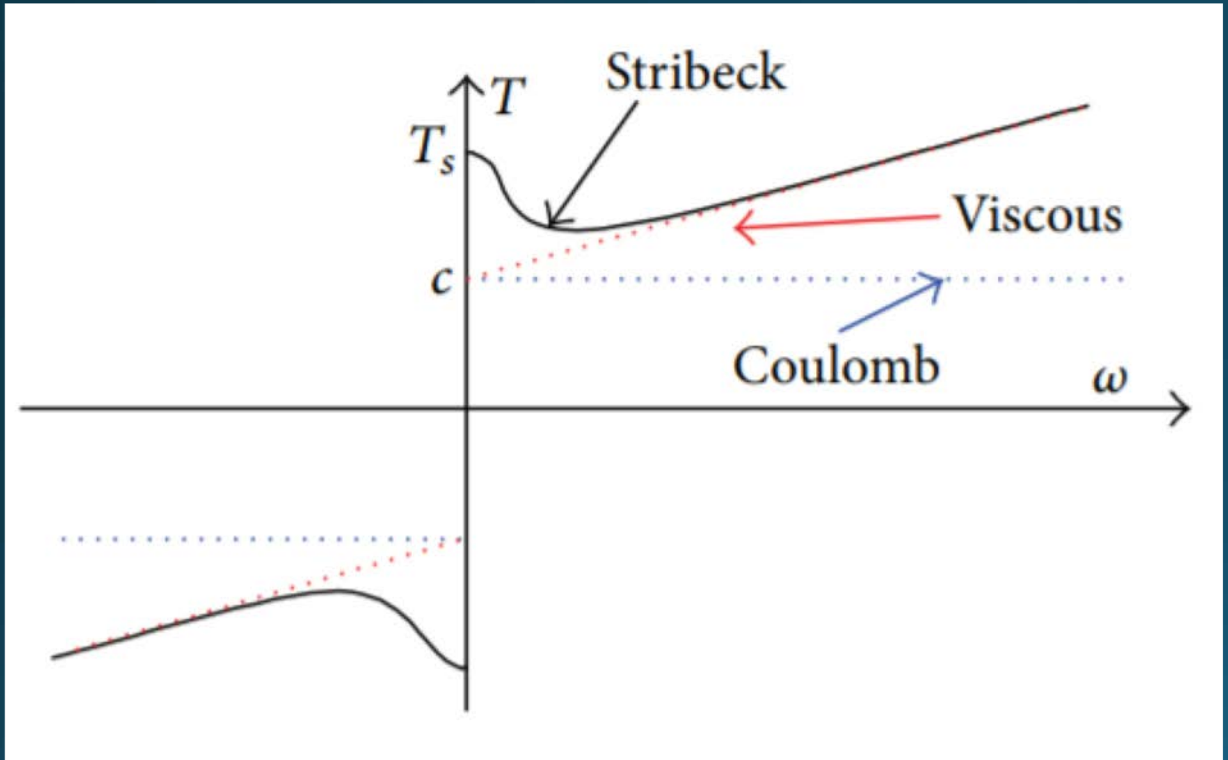
Reaction wheel dynamics

$$\tau_{net} = \text{Net torque on reaction wheel}$$
$$\tau_c = \text{Commanded torque}$$
$$\tau_f = \text{Torque due to friction, nominal}$$
$$I_\omega = \text{Reaction wheel moment of inertia}$$
$$\alpha_\omega = \text{Reaction wheel angular acceleration}$$

$$\tau_{net} = \tau_c - \tau_f = I_\omega \alpha_\omega$$

$$\tau_f = f(\omega)$$

CPE – Fault Injection and Management System: Characterizing Reaction Wheel Friction



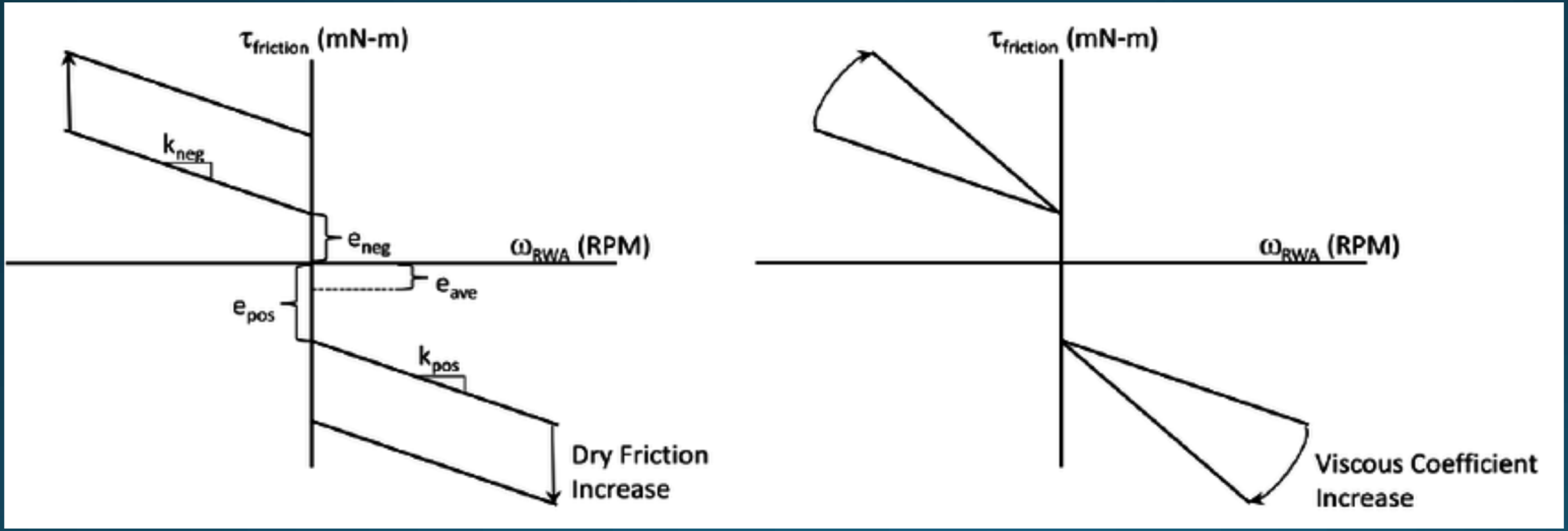
Representative friction torque curve of reaction wheel

- Friction in reaction wheels is combination of Viscous, Coulomb, with some initial Stribeck friction near angular velocities of zero

Source: Carrara, Vlademir, and Hélio Koiti Kuga. "Estimating Friction Parameters in Reaction Wheels for Attitude Control." *Mathematical Problems in Engineering*, vol. 2013, 12 May 2013.

CPE – Fault Injection and Management System: Reaction Wheel Friction Failure

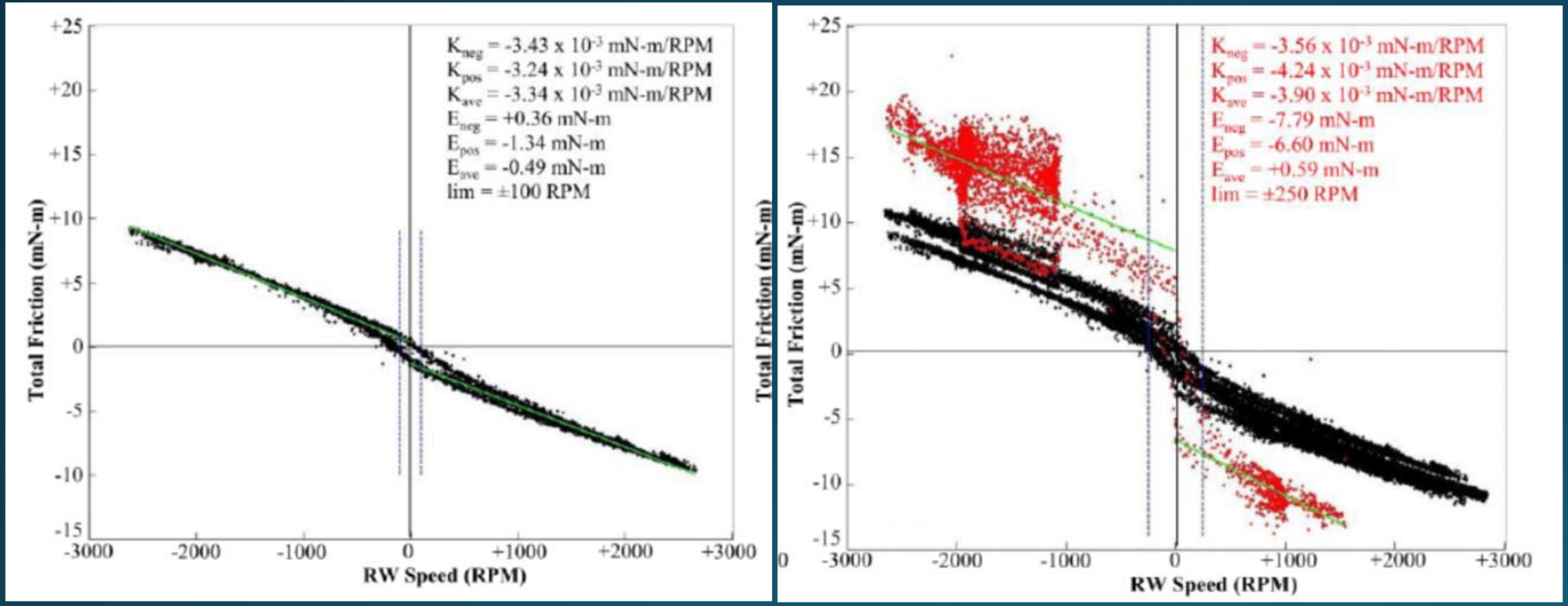
- Hard failures in reaction wheels are caused by an increase in Coulomb friction.



Left: Increase in Coulomb friction. Right: Increase in Viscous friction

Source: Hacker, Johannes M, et al. *Reaction Wheel Friction Data-Processing Methodology and On-Orbit Experience*. AIAA, Aug. 2014.

CPE – Fault Injection and Management System: Reaction Wheel Friction Failure



Left: Nominal Friction Data. Right: Increase in Coulomb friction causing hard failure

- Actual on-orbit data of failing reaction wheel
- Hard failure occurs at 5 mN-m above nominal, with nominal static friction of 0.85 mN-m
- Use this scaling for fault detection threshold in our system.

Source: Hacker, Johannes M, et al. *Reaction Wheel Friction Data-Processing Methodology and On-Orbit Experience*. AIAA, Aug. 2014.

CPE – Fault Injection and Management System: Actuator Management

- Fault management has access to commanded torque as well as reaction wheel angular velocity at discrete time steps. Calculate angular acceleration of the wheel by:

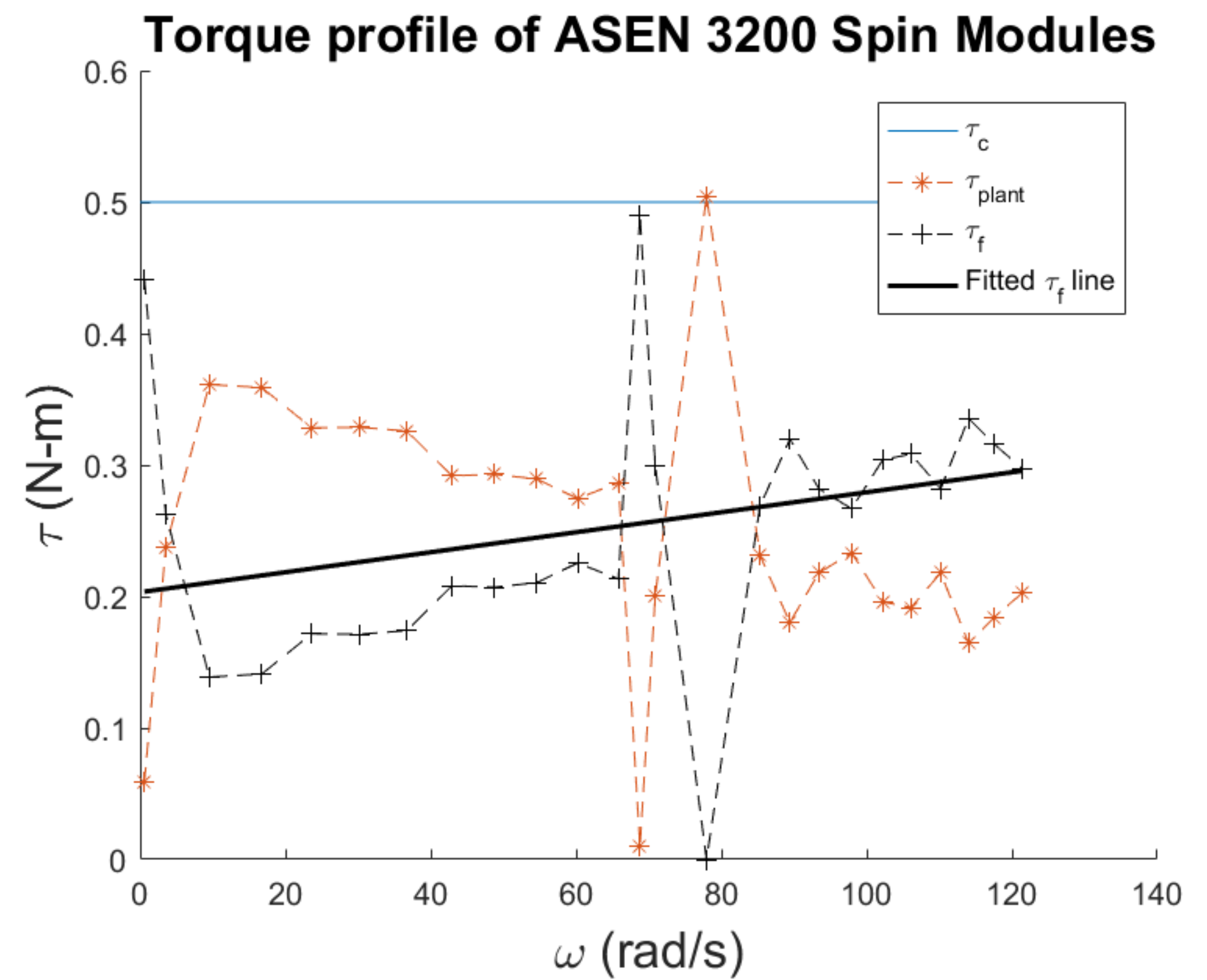
$$\hat{\alpha}_\omega = \frac{\Delta\omega}{\Delta t}$$

- Then, calculate the system friction by:

$$\hat{\tau}_f = \tau_c - I_\omega \hat{\alpha}_\omega$$

- This is then compared versus a threshold friction torque of 4 times the nominal static friction torque present in the reaction wheel.
- If the system friction calculated by fault management is above this threshold value, characterize as a fault

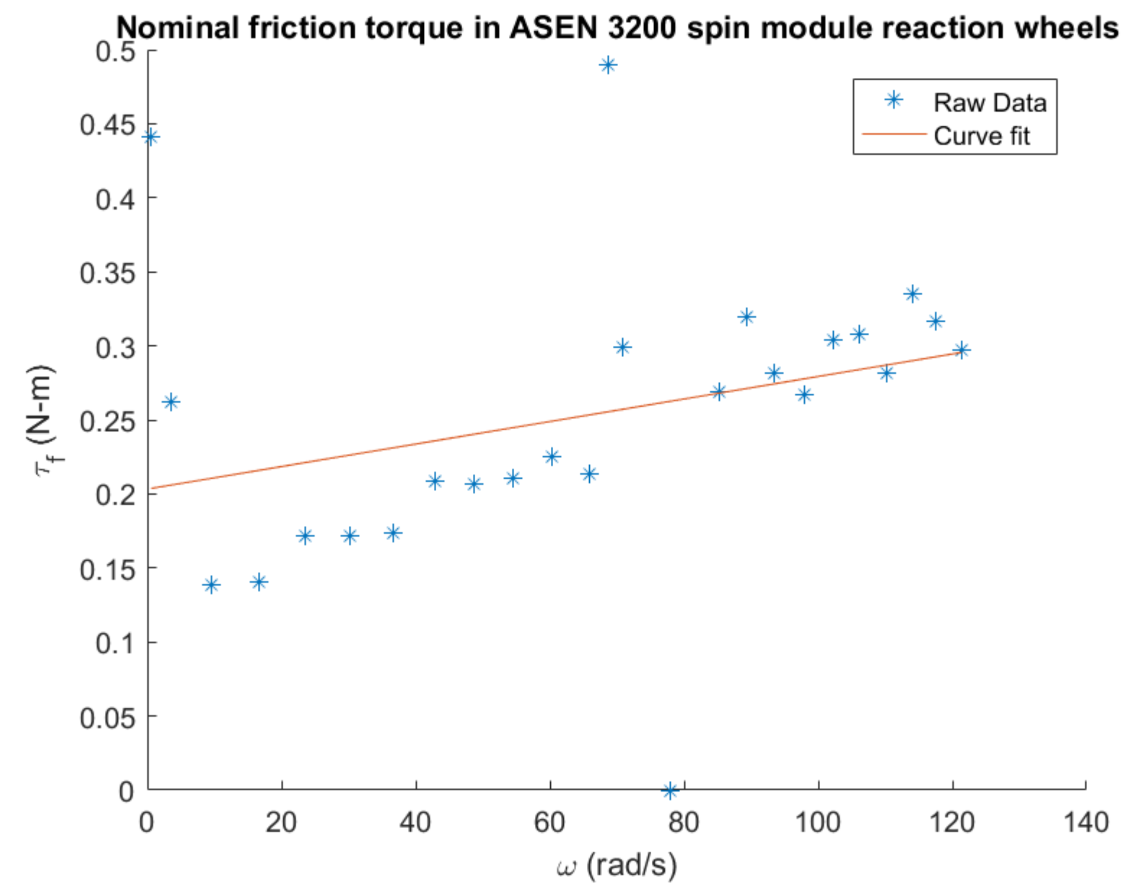
CPE – Fault Injection and Management System: Example Analyzing 3200 Reaction Wheels

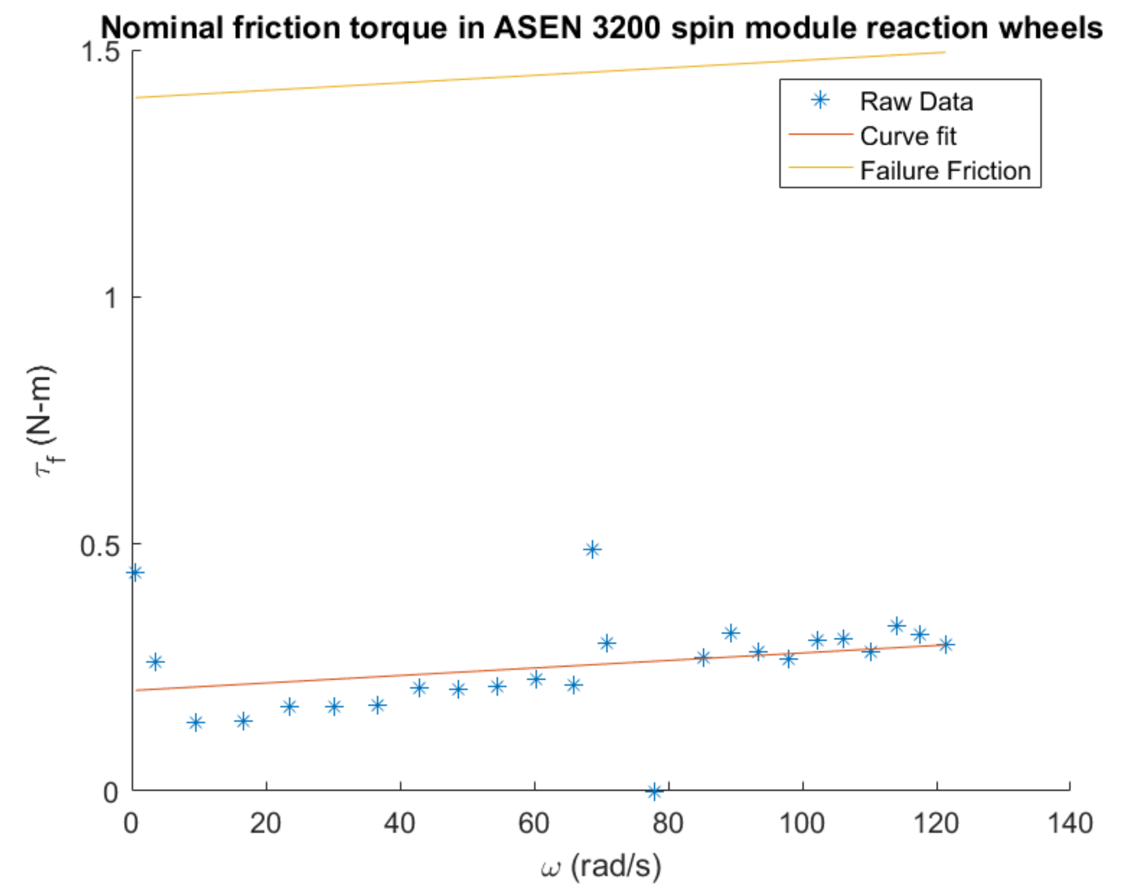


- Used data from ASEN 3200 to examine nominal friction in this system.
- Constant commanded torque of 0.5 N-m
- Data file contained time stamps every 0.1 s with commanded torque and wheel speed.
- From this data, the friction torque present as a function of angular velocity was calcucated.
- Then, a linear fit of this data was made to determine an approximate nominal friction torque as a function of angular velocity.

CPE – Fault Injection and Management System: Analyzing 3200 Reaction Wheels

- Triggering a fault – Example using ASEN 3200 Spin Module data





Requirements Flow-down

1. The TestTable shall allow for two degrees of freedom in translation and one degree of freedom in rotation in a low friction environment.
   1.1. The TestTable shall allow for unrestricted rotation of the MockSat about its axis normal to the plane of the TestTable.
   1.2 The TestTable shall allow for translation along two orthogonal axes within a designated portion of the plane of the TestTable surface
      1.2.1 The TestTable shall utilize a station-keeping mechanism to restrict the translation of the MockSat to less than 1.0 inch.
   1.3 The TestTable shall support the weight of the MockSat whilst providing a reduced friction surface.
      1.3.1 The total rotational friction between the MockSat and the TestTable during nominal operation shall be no greater than $1.5\ \frac{lbm \cdot in^2}{s}$.
   1.4 The TestTable shall comply with OSHA Two-Man Lift Criteria
      1.4.1 The TestTable shall occupy a volume no greater than 72 x 72 x 28 inches.
      1.4.2 The TestTable shall weigh no more than 100 pounds

Requirements Flow-down

2. The MockSat shall be equipped with an attitude determination and control system (ADCS) that replicates the 0.04 Hz bandwidth response of the GOES-16 satellite to within 10%.
    - 2.1. The MockSat shall be equipped with two reaction wheels for rotational control.
        - 2.1.1. The MockSat reaction wheels shall be scaled/tuned to simulate the response of GOES-16 about its max MOI.
        - 2.1.2. The MockSat reaction wheels shall be capable of responding to user fault injection.
    - 2.2. MockSat shall have a sensor to provide rotational data.

Requirements Flow-down

3. The MockSat shall have the ability to maintain a controlled attitude relative to a point of reference within ±2.5°.
    3.1. The MockSat shall be equipped with a sensor array to determine its orientation.
        3.1.1. The MockSat shall have a coarse sensor to provide a wide field of view and get fine sensor in range.
        3.1.2. The MockSat shall have a fine sensor to determine attitude with an accuracy of ±2.5°.
        3.1.3. The MockSat shall maintain pointing accuracy for no less than 30 seconds.

Requirements Flow-down

4. The system shall have the ability to introduce a fatal operating fault in either the MockSat's primary reaction wheel, the coarse orientation sensor, or the fine orientation sensor (but not more than one fault at a time).
    - 4.1. The fault injection system shall not cause permanent damage to the ADCS system
    - 4.2. The fault injection system shall wait for user command from the ground station to initiate fault injection.
        - 4.2.1. The ground station unit shall allow the user to initiate a choice of reaction wheel fault, coarse sensor fault, or fine sensor fault.
            - 4.2.1.1. The fault injection system shall create a sensed increase in friction torque of 5.5 times the natural coulomb friction in the reaction wheel.
                - 4.2.1.1.1. The fault shall be injected as a feedback loop living on the microcontroller.
            - 4.2.1.2. The coarse and fine sensor shall be injected with a fault capable of introducing an error as a position bias.
        - 4.2.2. The ground station unit shall be able to send a command for fault initiation to the fault injection system.
    - 4.3. The fault injection system shall be able to be deactivated by user command.
        - 4.3.1. The ground station unit shall allow the user to deactivate the fault injection system
        - 4.3.2. The ground station unit shall be able to send a command to deactivate the fault injection system.

Requirements Flow-down

5. The MockSat flight control software shall recover from a fatal operating fault in either the MockSat's primary reaction wheel or the fine orientation sensor (but not more than one fault at a time) by regaining normal operation.
    - 5.1. There shall exist in software a fault management system to handle fault detection and identification.
        - 5.1.1. The fault management system shall have the ability to detect a fatal operating fault from the reaction wheel.
        - 5.1.2. The fault management system shall have the ability to detect a fatal operating fault from the coarse attitude sensor.
        - 5.1.3. The fault management system shall have the ability to detect a fatal operating fault from the fine attitude sensor.
        - 5.1.4. The fault management system shall be independent of the fault injection system existence.
        - 5.1.5. The fault management system shall classify the location of the fault (either reaction wheel, coarse attitude sensor, or fine attitude sensor).
        - 5.1.6 The fault management system shall recover nominal operation of the satellite in the presence of a fault.
            - 5.1.5.1. The fault management system shall be able to communicate with the power regulation board.
            - 5.1.5.2. The fault management system shall be able to control power to the primary reaction wheel.
            - 5.1.5.3. The fault management system shall be able to control power to the secondary reaction wheel.
            - 5.1.5.4. The fault management system shall be able to switch sensing to a secondary attitude sensor.
        - 5.1.6. The fault management system shall alert the ground station operator that a fatal fault has occurred.
            - 5.1.6.1. The fault management system shall be able to alert the ground station operator to the type of fault that has occurred.
            - 5.1.6.2. The fault management system shall be able to communicate with the Ground Station Unit

References

[1] Chapel, J. (2014). *Guidance, Navigation, and Control Performance for the GOES-R Spacecraft. Porto, Portugal. ESA*

[2] Rozelle, D. *The Hemispherical Resonator Gyro: From Wineglass to the Planets*

[3] Hubble Space Telescope Hot and Cold Pixels. Space Science Telescope Institute. http://www.stsci.edu/hst/nicmos/performance/anomalies/hotcoldpix.html