Spring Final Review

Lockheed Martin
LLAMAS Team

Dalton Anderson
Daniel Greer
Ben Hutchinson
Kent Lee

Andrew Levandoski
Andrew Mezich
Samuel O’Donnell
Zach Reynolds

Kristyn Sample
Corwin Sheahan
Pol Sieira
Zack Toelkes

LLAMAS
Satellite Acds fault MAnagement System

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

<table>
<thead>
<tr>
<th>FR 1</th>
<th>The TestTable shall allow for two degrees of freedom in translation and one degree of freedom in rotation in a low friction environment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR 2</td>
<td>The MockSat shall be equipped with an ADCS that replicates the 0.04 Hz bandwidth response of the GOES-16 satellite to within ±10%.</td>
</tr>
<tr>
<td>FR 3</td>
<td>The MockSat shall have the ability to maintain a controlled attitude relative to a point of reference within ±2.5°.</td>
</tr>
<tr>
<td>FR 4</td>
<td>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).</td>
</tr>
<tr>
<td>FR 5</td>
<td>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).</td>
</tr>
</tbody>
</table>
Concept Of Operations (CONOPs)

- Test Initiation
  - MockSat initializes and begins searching for target.

- Nominal Operation
  - MockSat has acquired target and tracks motion to within $\pm2.5^\circ$.

- 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 $\pm2.5^\circ$. 

CONOPS (accelerated speed)
Design Description
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: \(~9\text{lb} \rightarrow ~24\text{lb}\).

- 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.
MockSat Final Design

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XBee Wireless Transmitter/Receiver</td>
<td></td>
</tr>
<tr>
<td>Fine FOV Optical Sensor (x2)</td>
<td>Fine Field Of View (FOV) = 0.05°/pixel</td>
</tr>
<tr>
<td>Coarse FOV Optical Sensor (x2)</td>
<td>Coarse Field Of View (FOV) = 0.16°/pixel</td>
</tr>
<tr>
<td>Reaction Wheel (x2)</td>
<td>1DOF Rotation: One degree of rotational freedom to track target</td>
</tr>
<tr>
<td>Power Conditioning Board</td>
<td></td>
</tr>
<tr>
<td>LiPo Battery</td>
<td></td>
</tr>
<tr>
<td>Mass Balance</td>
<td>• Center of Mass: Fine tuning of center of mass for better rotational dynamics</td>
</tr>
<tr>
<td>Reaction Wheel Motor Controller (x2)</td>
<td></td>
</tr>
<tr>
<td>Arduino DUE</td>
<td>• Communication: Using UART to relay data from ground station to MockSat</td>
</tr>
<tr>
<td>MOI about Axis of Rotation</td>
<td>• Avionics: Controller for pointing and switching of components</td>
</tr>
<tr>
<td>Total Weight</td>
<td>11.4 lbm</td>
</tr>
<tr>
<td>MOI</td>
<td>188.6 lb*in²</td>
</tr>
<tr>
<td>Height</td>
<td>5.9 in</td>
</tr>
<tr>
<td>Width</td>
<td>12 in</td>
</tr>
<tr>
<td>Height</td>
<td></td>
</tr>
</tbody>
</table>
Design Description – Pixy Operation

Pixy functions based upon a hue detection algorithm

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 ($\vartheta$) can be determined by the following equation:

$$\vartheta = \arctan \left( \frac{2 \tan \left( \frac{\theta}{2} \right)}{\Xi} \right)$$

Pixy is capable of utilizing different lenses.

**Coarse Sensor**
- $\theta_c = 75^\circ$
- $\vartheta = 0.16^\circ$/pixel

**Fine Sensor**
- $\theta_f = 20^\circ$
- $\vartheta = 0.05^\circ$/pixel
**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 increased friction in the reaction wheel.
- Faulted torque is then converted to a PWM signal and sent to the motor controller.

**Digital Camera (Pixy)**
- Takes $\Delta \theta$ calculated from the number of pixels between the reference target centroid and the center of the camera FOV*.
- $8^\circ$ is added to $\Delta \theta$ 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
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 gm²  
New MOI: 0.047 gm²
Critical Project Elements
• 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.

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
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
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 lbm-in²·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
Results:
• $\mu = 5.105 \text{ lbm\cdot in}^2\cdot \text{sec}^{-1}$, 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, $\mu \leq 1.5 \text{ lbm\cdot in}^2\cdot \text{sec}^{-1}$
• 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.
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 ±10%

Importance:
Satisfying FR 2, 0.04 Hz within ±10%
**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 target's 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 Δθ output by the Pixy and adding a bias of 8° before feeding Δθ to the control law. This is only done if commanded by GSU.
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)
**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)*
**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)
**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
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

<table>
<thead>
<tr>
<th></th>
<th>Model</th>
<th>Experimental</th>
</tr>
</thead>
<tbody>
<tr>
<td>kp</td>
<td>0.0056</td>
<td>9.3160</td>
</tr>
<tr>
<td>ki</td>
<td>5.2724E-6</td>
<td>0.8003</td>
</tr>
<tr>
<td>kd</td>
<td>1.3279</td>
<td>46.7969</td>
</tr>
</tbody>
</table>

**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
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.
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
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
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.

<table>
<thead>
<tr>
<th>Risk Matrix – Mitigated Risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Likely</td>
</tr>
<tr>
<td>Likely</td>
</tr>
<tr>
<td>Possible</td>
</tr>
<tr>
<td>Unlikely</td>
</tr>
<tr>
<td>Very Unlikely</td>
</tr>
<tr>
<td>Negligible</td>
</tr>
<tr>
<td>Minor</td>
</tr>
<tr>
<td>Moderate</td>
</tr>
<tr>
<td>Significant</td>
</tr>
<tr>
<td>Severe</td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td>Acceptable</td>
</tr>
<tr>
<td>Marginal</td>
</tr>
<tr>
<td>Unacceptable</td>
</tr>
</tbody>
</table>

1. Lack of torque resolution
2. Fault Management Implementation
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.
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
  • “2π5” 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
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
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
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
    o 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
    o 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:**
- $\mu = 5.105 \text{ lbm-in}^2\text{-sec}^{-1}$, 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, $\mu \leq 1.5 \text{ lbm-in}^2\text{-sec}^{-1}$
- Value necessary for control system tuning
Reaction Wheel Torque Test
Reaction Wheel Torque Test

**Reaction Wheel Speed vs Time for $\Delta \Theta = -3.45$ deg**

- Measured
- Predicted by Commanded Torque

**Reaction Wheel Torque vs Time for $\Delta \Theta = -3.45$ deg**

- Measured Torque
- Commanded Torque

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

![Graph of Reaction Wheel Speed vs Time for Δθ = 5.38 deg](image1)

![Graph of Reaction Wheel Torque vs Time for Δθ = 5.38 deg](image2)

- Avg. Commanded Torque: 2.64 mNm
- Avg. Measured Torque: 2.546288 mNm
- Std. Dev. Measured Torque: 123.36 nNm
Reaction Wheel Torque Test

Reaction Wheel Speed vs Time for $\Delta \theta = 6.34$ deg

- Measured
- Predicted by Commanded Torque

Reaction Wheel Torque vs Time for $\Delta \theta = 6.34$ deg

- Measured Torque
- Commanded Torque

Avg. Commanded Torque: 3.10 mNm
Avg. Measured Torque: 2.980497 mNm
Std. Dev. Measured Torque: 117.84 pNm
Reaction Wheel Torque Test

- Reaction Wheel Speed vs Time for $\Delta \Theta = -6.68$ deg
  - Measured
  - Predicted by Commanded Torque

- Reaction Wheel Torque vs Time for $\Delta \Theta = -6.68$ deg
  - Measured Torque
  - Commanded Torque

  Avg. Commanded Torque: -3.28 mNm
  Avg. Measured Torque: -3.086598 mNm
  Std. Dev. Measured Torque: 114.38 mNm
Reaction Wheel Torque Test

**Reaction Wheel Speed vs Time for \( \Delta \Theta = 0.10 \) deg**

- Measured
- Predicted by Commanded Torque

**Reaction Wheel Torque vs Time for \( \Delta \Theta = 0.10 \) deg**

- Measured Torque
- Commanded Torque

Avg. Commanded Torque: 0.05 mNm
Avg. Measured Torque: 0.319684 mNm
Std. Dev. Measured Torque: 102.41 nNm
## Risk Assessment

<table>
<thead>
<tr>
<th>Likelihood</th>
<th>Very Likely</th>
<th>Likely</th>
<th>Possible</th>
<th>Unlikely</th>
<th>Very Unlikely</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Negligible</td>
<td>Minor</td>
<td>Moderate</td>
<td>Significant</td>
<td>Severe</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Likelihood</th>
<th>Very Likely</th>
<th>Likely</th>
<th>Possible</th>
<th>Unlikely</th>
<th>Very Unlikely</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Negligible</td>
<td>Minor</td>
<td>Moderate</td>
<td>Significant</td>
<td>Severe</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Severity</th>
<th>Acceptable</th>
<th>Marginal</th>
<th>Unacceptable</th>
</tr>
</thead>
</table>

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

### Risk Matrix

- **Very Likely**: Red
- **Likely**: Yellow
- **Possible**: Green
- **Unlikely**: Yellow
- **Very Unlikely**: Green

- **Negligible**: Green
- **Minor**: Green
- **Moderate**: Yellow
- **Significant**: Red
- **Severe**: Red
Risk Assessment – Mitigation

<table>
<thead>
<tr>
<th>Risk: Lack of torque resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause:</strong> Motors do not have adequate torque resolution</td>
</tr>
</tbody>
</table>

Risk Mitigation

<table>
<thead>
<tr>
<th>Action: Proper motor selection, accurate torque characterization</th>
<th>Success Criteria: Pointing requirements satisfied</th>
</tr>
</thead>
</table>

Old Risk Level: High

New Risk Level: Marginal

Risk Mode: Technological
<table>
<thead>
<tr>
<th><strong>Risk:</strong> Fault Management Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause:</strong> Tailoring a consistent specific response to a generalized suite of hardware</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Risk Mitigation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action:</strong> Create a fault management architecture that attempts to solve the modularity aspect</td>
</tr>
<tr>
<td><strong>Success Criteria:</strong> End up with a fault management architecture that is applicable to other projects</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Old Risk Level:</strong> High</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New Risk Level:</strong> Marginal</td>
</tr>
<tr>
<td><strong>Risk Mode:</strong> Technological</td>
</tr>
</tbody>
</table>
### Risk: ADCS Integration

**Cause:** Breakdown of communication between any of the ADCS components

**Effect:** ADCS loss of control

### Risk Mitigation

<table>
<thead>
<tr>
<th>Action</th>
<th>Success Criteria</th>
<th>Old Risk Level</th>
<th>New Risk Level</th>
<th>Risk Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Careful system integration and understanding of communication protocols</td>
<td>ADCS shares and responds to data as anticipated</td>
<td>Marginal</td>
<td>Acceptable</td>
<td>Technological</td>
</tr>
</tbody>
</table>

**Old Risk Level:** Marginal

**New Risk Level:** Acceptable

**Risk Mode:** Technological
Risk Assessment – Post-Mitigation

<table>
<thead>
<tr>
<th>Likelihood</th>
<th>Very Likely</th>
<th>Likely</th>
<th>Possible</th>
<th>Unlikely</th>
<th>Very Unlikely</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Negligible</td>
<td>Minor</td>
<td>Moderate</td>
<td>Significant</td>
<td>Severe</td>
</tr>
<tr>
<td>1. Lack of torque resolution</td>
<td>2. Fault Management Implementation</td>
<td>3. ADCS Integration</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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: TestTable

<table>
<thead>
<tr>
<th>Design Options</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
</table>
| **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

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Weight</th>
<th>Air Table</th>
<th>Ice Table</th>
<th>Air Bearing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Duration</td>
<td>35%</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Cost</td>
<td>10%</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Manufacturing Required</td>
<td>15%</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Heritage</td>
<td>5%</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Simplicity</td>
<td>15%</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Logistics</td>
<td>20%</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100%</strong></td>
<td><strong>4.55</strong></td>
<td><strong>2.6</strong></td>
<td><strong>1.45</strong></td>
</tr>
</tbody>
</table>

### 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

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

- 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

- 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

- **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.
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
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
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
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
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
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

**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

---

Recovery 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
Class diagram showing major classes/programs, their functions, and their interactions with other software modules.
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:**
- Governing Equation:
  \[ \sum \tau = I \alpha (1-\text{DOF}) \]
- Nominal friction:
  \[ \tau_c - \tau_f = I \alpha \]
- Induced Friction:
  \[ \tau_c - \hat{\tau}_f = I \hat{\alpha} \]

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

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

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

\[ \tau_f = f(\omega) \]
CPE – Fault Injection and Management System: Characterizing Reaction Wheel Friction

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

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

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

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.

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

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

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.
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 calculated.

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 \( \frac{lbm-in^2}{s} \).
   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.
3. The MockSat shall have the ability to maintain a controlled attitude relative to a point of reference within $\pm 2.5^\circ$.
   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 $\pm 2.5^\circ$.
       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.

5.1.6.1. The fault management system shall be able to communicate with the power regulation board.
5.1.6.2. The fault management system shall be able to control power to the primary reaction wheel.
5.1.6.3. The fault management system shall be able to control power to the secondary reaction wheel.
5.1.6.4. 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

