(Preprint) AAS 25-778

REAL-TIME TESTING OF SATELLITE ATTITUDE CONTROL WITH A REACTION WHEEL HARDWARE-IN-THE-LOOP PLATFORM

Morokot Sakal[*], George Nehma[*], Camilo Riano-Rios[†], and Madhur Tiwari[†]

ABSTRACT

We propose the Hardware-in-the-Loop (HIL) test of an adaptive satellite attitude control system with reaction wheel health estimation capabilities. Previous simulations and Software-in-the-Loop testing have prompted further experiments to explore the validity of the controller with real momentum exchange devices in the loop. This work is a step toward a comprehensive testing framework for validation of spacecraft attitude control algorithms. The proposed HIL testbed includes brushless DC motors and drivers that communicate using a CAN bus, an embedded computer that executes control and adaptation laws, and a satellite simulator that produces simulated sensor data, estimated attitude states, and responds to actions of the external actuators. We propose methods to artificially induce failures on the reaction wheels, and present related issues and lessons learned.

INTRODUCTION

Reaction Wheel (RW) arrays are crucial means for attitude control on many satellites due to their ability to precisely execute the control actions required for attitude maneuvers via exchange of angular momentum.[1] As such, it is a highly critical subsystem that incorporates levels of redundancy in the case of actuator faults. Extensive research has been made to enhance the capabilities of attitude controllers so that they minimize the hardware cost and computational load as much as possible whilst being able to guarantee tracking.[2–5] Methods such as Sliding Mode Control (SMC),[6] Model Predictive Control (MPC),[7,8] neural networks[9] as well as a variety of adaptive controllers[10–14] have been proposed for attitude control, each with varying capabilities and degrees of success. The advantage of adaptive controllers is their ability to compensate for uncertaities in the system dynamics, whilst maintaining stability.

In our previous work,[15] we proposed a method to simultaneously learn the health of a satellite's RWs and attitude tracking under scenarios of failing or degraded RW(s). Our method involves a Lyapunov-based adaptive controller with an integral concurrent learning (ICL)-based adaptive update law that ensures convergence of the estimated health of the RWs once a finite excitation condition is met. This controller was shown to guarantee exponential convergence of error states and RW health estimates. We demonstrated via MATLAB/Simulink-based numerical simulations that the controller correctly estimated the health of each RW and performed the alternating attitude tracking reference required by the mission. We presented simulations with arrays of 4 and 6 RWs, a varying number of degraded RWs and varying levels degradation. A Software-in-the-Loop (SIL) test of the

[*]Ph.D. Student, Aerospace, Physics and Space Sciences Department, Florida Institute of Technology
[†]Assistant Professor, Aerospace, Physics and Space Sciences Department, Florida Institute of Technology

controller was conducted to test the real-time capability with embedded hardware (NVIDIA Jetson Nano), achieving comparable results.

The promising SIL experiments have prompted us to continue with a Hardware-in-the-Loop (HIL) test to validate the adaptive controller in more realistic conditions, including hardware limitations, sensor noise, and communication latency that are difficult to replicate in simulation. In addition, we aim to use this controller to demonstrate and validate our envisioned modular testbed, currently under development, at the Space Vehicle and Robotics (SVR) lab at the Florida Institute of Technology.

HIL testing is a vital aspect in the development of real, flight ready controllers and actuator-driver systems as it is a mission critical system that can afford little to no failures. The sim-to-real gap is large and poses a number of issues when developing control systems, the greatest of which is the ability to model the behavior of the actuating system in simulation as close to the real behavior as possible. RW dynamics, although can be approximated in simulation, are often hard to be precisely emulated due their physical complexity and interaction with motor drivers, sensors, inner control loops, among others. Hence, the need to verify the health estimation capability of a controller using real hardware is of great importance.

The contributions of this paper are two fold:

- We develop a Hardware in the Loop testing architecture to verify actuator health estimation performance of the adaptive controller with real RWs.
- We highlight the problems encountered during HIL testing of a RW array, and provide solutions and discussion on each.

This paper is organized as follows. First, we provide an overview of the adaptive controller that is being tested in this HIL set up. Then we present the architecture design of the testbed. The next sections describe the experimental setup and procedures to integrate each component to realize the test. Finally, we discussed the results and concluded with future work.

ADAPTIVE ICL CONTROLLER

Since the focus of this paper is to highlight and analyze the HIL testing for this adaptive controller, this section only briefly overviews the design and architecture of our controller. For further, more detailed explanation and derivation of the controller and its stability analysis, we refer the reader to our previous paper.[15]

The Equations of Motion for the attitude of a spacecraft with N RWs are given as

$$J\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times (J\boldsymbol{\omega} + J_{RW}G\boldsymbol{\Omega}) + G\Phi\boldsymbol{u} \tag{1}$$

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}\left[\left(1 - \boldsymbol{\sigma}^{\boldsymbol{T}}\boldsymbol{\sigma}\right) I_3 + 2\sigma^{\times} + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^{\boldsymbol{T}}\right]\boldsymbol{\omega}, \tag{2}$$

where $\boldsymbol{\omega} \in \mathbb{R}^3$ is the spacecraft angular velocity expressed in the body coordinate system, $\boldsymbol{\sigma} \in \mathbb{R}^{\mathbf{3}}$ is the vector of Modified Rodrigues Parameters (MRP) that represent the orientation of the spacecraft with respect to the inertial frame, $\boldsymbol{\Omega} = [\Omega_1, \Omega_2, \cdots, \Omega_N]^T \in \mathbb{R}^N$ is a vector containing the N RW angular velocities, $\boldsymbol{u} = -J_{RW}\dot{\boldsymbol{\Omega}} = [u_1, u_2, \cdots, u_N]^T \in \mathbb{R}^N$ is the control input that represents the torque applied by each RW, $J \in \mathbb{R}^{3\times3}$ is the total inertia matrix, $J_{RW} \in \mathbb{R}_{>0}$

is the inertia of the flywheels about their spin axis, $\Phi = diag\{\phi_1,\ \phi_2,\ \cdots,\ \phi_N\} \in \mathbb{R}^{N\times N}$ is the uncertain RW health matrix, $G = \{\hat{\boldsymbol{s}}_\mathbf{1}, \hat{\boldsymbol{s}}_\mathbf{2}, \cdots, \hat{\boldsymbol{s}}_\boldsymbol{N}\} \in \mathbb{R}^{3\times N}$ is the RWA configuration matrix, and $\hat{\boldsymbol{s}}_\boldsymbol{i} \in \mathbb{R}^3$ is the direction of the i^{th} RW's spin axis expressed in the body coordinate system. The matrix $I_m \in \mathbb{R}^{m\times m}$ represents an identity matrix of dimension $m \times m$, and $a^\times \in \mathbb{R}^{3\times 3}$ is a the skew-symmetric matrix built with the vector $\boldsymbol{a} = [a_1, a_2, a_3]^T \in \mathbb{R}^3$.

The designed auxiliary control law $\boldsymbol{u_d}$ that stabilize the spacecraft attitude is

$$\boldsymbol{u_d} = \boldsymbol{\omega} \times (J\boldsymbol{\omega} + J_{RW}G\boldsymbol{\Omega}) + J\tilde{R}\dot{\boldsymbol{\omega}}_\boldsymbol{d} - J\tilde{\omega}^\times \tilde{R}\boldsymbol{\omega_d} + 4JB^{-1}\left[-\frac{1}{4}\dot{B}\tilde{\boldsymbol{\omega}} - \alpha\dot{\boldsymbol{\sigma}}_\boldsymbol{e} - K\boldsymbol{r} - \beta\boldsymbol{\sigma_e}\right], \quad (3)$$

where $\beta \in \mathbb{R}_{>0}$ is a constant control gain, $\alpha \in \mathbb{R}^{3\times 3}$ is a symmetric, positive definite control gain matrix, $\tilde{R} \in \mathbb{R}^{3\times 3}$ is a rotation matrix between the spacecraft desired and body frames, $\boldsymbol{\sigma_e} \in \mathbb{R}^\mathbf{3}$ is the error MRP and $\boldsymbol{r} = \dot{\boldsymbol{\sigma}}_\boldsymbol{e} + \alpha\boldsymbol{\sigma_e} \in \mathbb{R}^3$ is a modified error state.

The torque commands to be sent to the RWs, $\boldsymbol{u}$, are recovered as

$$\boldsymbol{u} = \left(G\hat{\Phi}\right)^\dagger \boldsymbol{u_d}, \quad (4)$$

where $(\cdot)^\dagger$ is the Moore-Penrose pseudo-inverse of $(\cdot)$ and $\hat{\Phi}$ can be obtained by numerically integrating the adaptation law given by:

$$\dot{\hat{\boldsymbol{\theta}}} = \text{proj}\left\{\frac{1}{4}\Gamma Y^T\left(J^{-1}\right)^T B^T\boldsymbol{r} + \Gamma K_1 \sum_{i=1}^{N_s} \mathcal{Y}_i^T\left(J\boldsymbol{\omega}(t) - J\boldsymbol{\omega}(t-\Delta t) + \boldsymbol{\mathcal{U}_i} - \mathcal{Y}_i\hat{\boldsymbol{\theta}}\right)\right\}, \quad (5)$$

where $\hat{\boldsymbol{\theta}} \in \mathbb{R}^N$ is the estimate of the vector containing RWs' health factors $\boldsymbol{\theta} = [\phi_1, \phi_2, \cdots, \phi_N]^T$, $\Gamma,\ K_1,\ \in \mathbb{R}^{N\times N}$ are constant, positive definite adaptation gain matrices, $\boldsymbol{\mathcal{U}_i} \in \mathbb{R}^3$ and $\mathcal{Y}_i \in \mathbb{R}^{3\times N}$ are integrals of input-output data terms, and the matrix $B \in \mathbb{R}^{3\times 3}$ is a matrix used in the MRP kinematics.[16]

TESTBED ARCHITECTURE DESIGN

Figure 1 illustrates the overall architecture of the HIL testbed developed for the experiment. This architecture extends the already tested and proven SIL setup with ROS2 middleware used as for communication between computers in a local network.[17] The HIL testbed includes a Satellite Simulator implemented in Matlab/Simulink running on a Linux computer, an NVIDIA Jetson Nano embedded computer hosting various software and algorithms undertest, and the RW testbed, which consists of hardware components including four Maxon EC 60 flat brushless motors[18] emulating RWs, each controlled by a digital position controller EPOS4 Compact 50/5CAN[19]communicating via USB/CAN bus, and an artificial fault injection mechanism used to simulate various RW fault scenarios.

As shown in Figure 1, the ROS2 nodes in the HIL test setup utilize the publisher/subscriber model. Satellite Simulator Node (SatSimNode) runs the simulation of the satellite dynamics, environment, actuator models, and sensor models. The sensor models, including sun sensors, magnetometers, and gyroscopes, are modeled by adding noise and realistic sampling rates. The noisy sensor outputs are processed by the attitude determination block, which employs an Extended Kalman Filter

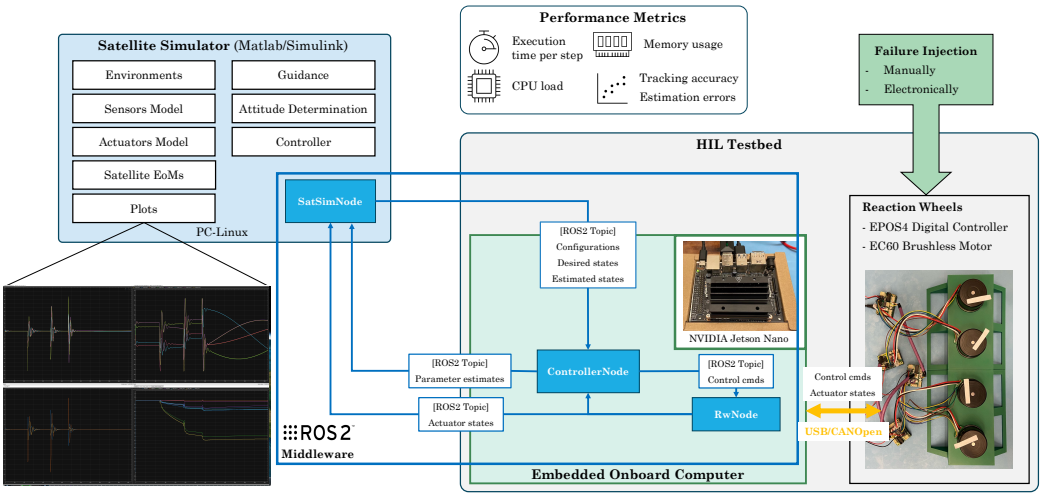


Figure 1: Overview of Testbed Architecture.

(EKF) based on multiplicative quaternion[20] to estimate the attitude states, i.e., satellite inertial angular velocity $\boldsymbol{\omega}$, and quaternion $\rightarrow$ MRPs $\boldsymbol{\sigma}$. The SatSimNode provides desired attitude states, satellite configuration parameters, and estimated states data. It also receives the controller outputs and actuator states data, i.e., RW angular velocity and current measurements back from the testbed.

The embedded computer operates two ROS2 nodes, each designated for a specific function. Using the estimated attitude states, desired spacecraft states, and satellite configuration, the ControllerNode calculates the required torque command. Depending on the test configuration, the torque command is converted to angular velocity or current commands. The ControllerNode publishes this command to the network via a ROS2 topic. The RwNode then subscribes to this command topic and forwards the commands to the digital controller using a USB/CANopen interface. Simultaneously, the RwNode acts as a publisher and delivers angular velocity and current measurement as the actuator states back to the network, which is received by both the ControllerNode and SatSimNode. The final step is to create the closed-loop HIL system and ensure synchronization between the simulation model and the physical hardware.

The testbed is built with a modular design. It is possible to swap in new actuators, sensors, or embedded computers with minimal changes to the overall architecture. It is flexible to make the transition between testing modes, from pure simulation, also called Model-in-the-Loop (MIL), to SIL, or HIL modes. This architecture is scalable to support future upgrades, such as air-bearing tests, satellite integration and testing campaigns, and comprehensive end-to-end system tests.

EXPERIMENTAL SETUP

Figure 2 shows the experimental setup used to validate the controller and its RW health estimation capability with actual hardware. The NVIDIA Jetson Nano is connected to one digital motor controller via USB interface, and the remaining motor drivers are interconnected through a CAN bus, using the CANopen protocol. A dedicated power supply unit is installed to power the electronics. The Jetson Nano communicates with MATLAB/Simulink wirelessly in a local network and exchanges data using the ROS2 middleware.

Performing a HIL test introduces a number of complexities that are either ignored or not present

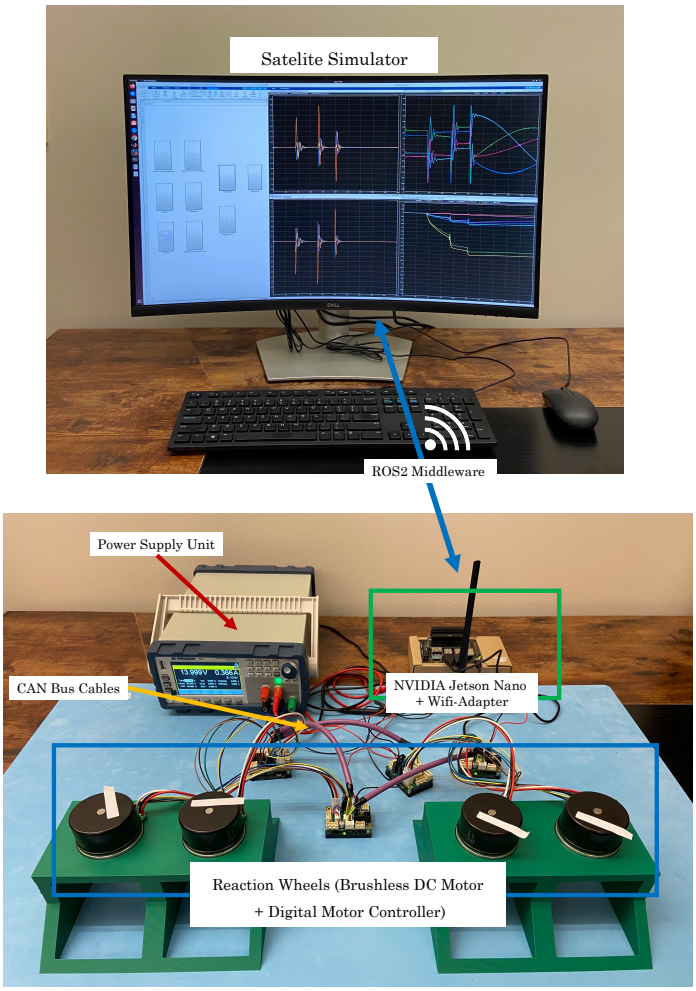


Figure 2: HIL Experimental Setup

in simulation, one of which is the assumption that all states of the RW are available to the controller, or whether the inner controller loop of the motor controller is closed via current or angular velocity. These considerations greatly impact the design of the controller and whether a real-time HIL test is feasible or representative.

As a first proof of concept, we artificially induce failure through manipulation of the control command being sent to each RW, as is common practice in many recent literature.[21–30] Testing full failure of the RW is simple, the power stage of the failing RW can be disconnected, whilst other failures can be induced by manipulating the amount of effort that is sent as command to the motor controller with respect to the effort output by the main attitude controller. This is normally a scaling factor that is applied to the required control effort.

EXPERIMENTAL PROCEDURE

The process of transferring a completely simulated system to one where the actuators and sensors are real hardware connected to the integrated, embedded controller and state estimator is a large jump to complete altogether. As such, we break down the process of implementing the physical

hardware in the loop in multiple steps to ensure accuracy and reliability. The first step, as seen in our previous work[15] included the SIL testing of the controller executed on an embedded computer. In order to completely verify the HIL tests, the breakdown of how we integrated certain components is shown in Figure 3. After the SIL test, which verifies that the embedded computer can run the control algorithm accurately and efficiently as compared to the simulation, we begin by adding the reaction wheels, with their CAN bus motor drivers and integrated sensors in the loop.

Model ⟷ **Physical**

Test Config.	SatSim	Jetson Nano		RWs Hardware
		ControllerNode	RwNode	
MIL	✓	X	X	X
SIL	✓	✓	X	X
HIL-a	X	X	✓	✓
HIL-b	✓	X	✓	✓
HIL-c	✓	✓	✓	✓

Figure 3: Experimental Procedures

HIL-a: Testing Reaction Wheel Response

Current Commands In order to create the most realistic simulated model of the RW and motor driver set up, control torque commands calculated by the adaptive controller are passed through a transfer function that is designed to mimic the behavior of the motor driver and RW operating in closed-loop torque control. As such, the first test is to pass the control commands, converted to current via the torque constant for the brushless motor,[18] from the simulator to the RW and motor drivers and then observe their behavior in comparison to the simulated RWs. Hence, the simulation propagates the equations of motion (EOM) and RW dynamics, with the torque commands being fed in parallel through the ROS2 network to the real RW and motor drivers, but maintaining the control-loop closed with the simulated RWs. The built-in current sensor reports back the actual current and Hall-effect sensor measures angular velocity of each RW.

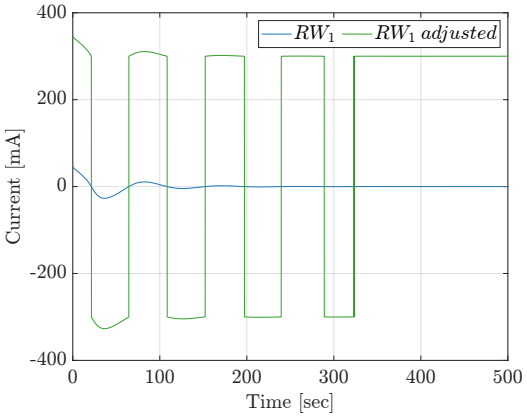
As the simulation runs, we compare the output torques and angular velocities between the simulated and real RWs to assess the differences between them. Parameters of the RW such as inertia, mass and torque constant were taken from the datasheet of the brushless motors and included in the simulated RWs.

The RW motor drivers have two modes of operation, torque (current) command and angular velocity command. In order to reduce complexity and added calculations that may introduce unwanted behaviors, we decided to control the RW's based on a given torque command. In order to do this on the RW hardware, a current command would be delivered to the motor drivers which in turn would use their internal PID feedback controller to track this command. We derive the current command from the torque required by the controller

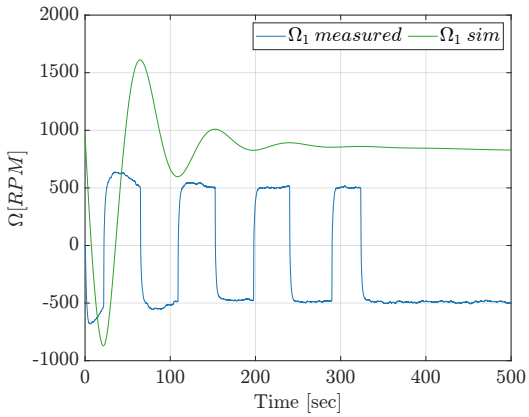
$$I_{cmd} = K_t \cdot \tau_{cmd} \tag{6}$$

where, K_t is the torque constant of the RW motor, given in the datasheet.

As seen in Figure 4a, the primary issue with this method of control commands is that the RWs have a current deadband of around 300mA, throughout which any current commanded would result in no output from the RWs. This is not an uncommon issue but in our case, because the desired torque commands were generally small and would eventually converge to zero, a majority fell within this deadband, as such the RWs were unable to actuate. A common solution to this issue is to kickstart the RWs when the simulation starts to break through the deadband, and although this momentarily helped alleviate the deadband issue, the motors quickly became unresponsive. Another solution that proved unsuccessful was to append the command signal with the sign of the command multiplied by the deadband region, $I_{cmd} = I_{cmd} + \text{sign}(I_{cmd}) \cdot 300$. As evident in Figures 4b this resulted in large jumping in the RW angular velocity and did not track the simulated angular velocity closely. As such, our next solution to this problem was to convert the output torque commands of the controller into angular velocity commands.



(a) Controller current commands and deadband adjusted current commands sent to RWs.



(b) Angular Velocity of Simulated RW and real RW with deadband

Figure 4: Current commands and angular velocity plots showing the deadband issue when generating torque commands from the adaptive controller.

Velocity Commands In order to convert the torque commands that are output from the adaptive controller into velocity commands that are acceptable commands to the RW motor drivers, the following steps must occur. First the commanded simulated torque for each RW is saturated by the max torque that is possible by each RW, given in its respective datasheet. Then, the torque command is divided by the inertia of the wheel to obtain angular acceleration

$$\dot{\Omega}_{cmd} = \frac{\tau_{cmd}}{J_{RW}}, \tag{7}$$

and the velocity command obtained using the forward-Euler integration algorithm which proved to be sufficient. Velocity saturation was also applied to ensure commands within the motor specs. These velocity commands can be tracked by the RWs and do not suffer from the deadband issue.

To verify the correct operation of the RWs with the velocity commands generated from the simulator, a similar experiment was performed where the main control loop is still closed with the simulated RWs that receive torque commands, and the corresponding computed velocity commands were also sent to the physical RWs. Figure 5 demonstrates that although the measured velocity signal includes a small amount of noise, its profile closely follows that of the simulated RW.

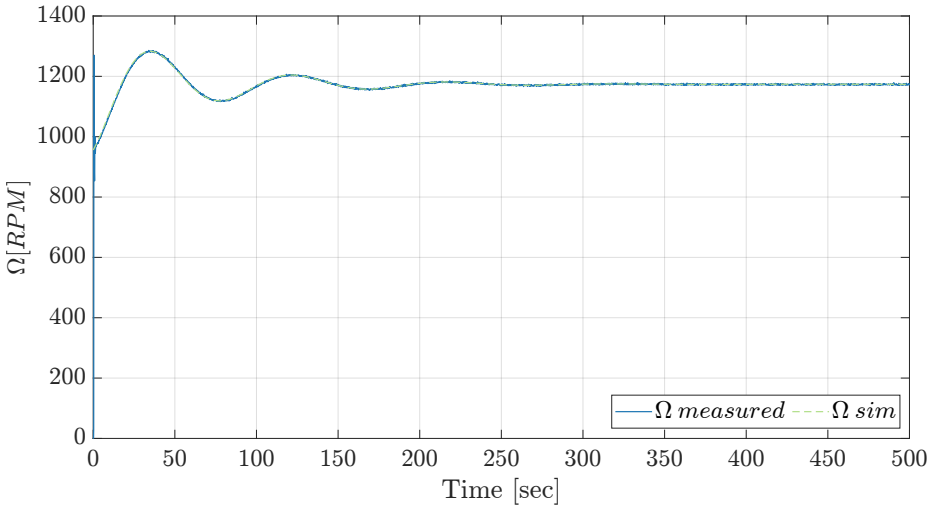


Figure 5: Comparison of simulated and physical RWs' angular velocities under angular velocity command.

HIL-b: Adding Physical RWs in the Main Control Loop

Once it was verified that the simulated and real RWs were behaving similarly for given control commands, the next step in achieving a complete HIL test was to feed the angular velocities and angular accelerations from the real RWs into the simulation, as required to propagate the EoMs as in Equation (1). This was a major step in the process as it introduced a number of issues that were required to be addressed to bridge the sim-to-real gap.

Dealing with noisy measurements from the RWs' angular velocities and currents, especially velocity measurements near zero speed, a known issue related to Hall-effect sensors,[?] added to the need of computing the corresponding angular accelerations to be able to propagate the EoMs, became an important issue. We attempted two approaches to compute the angular acceleration (a) by directly converting the RWs' current measurements to angular acceleration and (b) through numerical differentiation of the velocity measurements, then using low-pass filters to smooth out the signals. For the approach (a), we found a similar issue earlier with the deadband of the current measurement, which resulted in an incorrect mapping between current and angular acceleration. For approach (b), it introduced a phase shift (delay) in the signal, which is quite significant to be fed into the EoMs, resulting in an unstable system.

To mitigate the issue with noisy sensor data, we fed the calculated velocities and angular accelerations into the satellite's EoMs instead of actual accelerations obtained from measurements. As a result, current measurements were excluded from the setup, and only velocity measurements were used to interface with the simulator. Because the control and adaptation laws under test only require estimated satellite states and RWs' angular velocity measurements as inputs (as shown in

Equations (3), and (5)), and EoMs propagation is only required for numerical simulation and not for real satellite operation, the proposed data exchange remains representative.

However, the disconnection between the measured (experienced) RWs' angular velocities, and the calculated angular accelerations being fed into the EoMs, prevented us from physically inducing RW failures or degradation. In order to physically induce failure on a RW, an angular acceleration profile consistent with the measurement would be required. The need of angular acceleration is only present in this specific HIL test that requires EoMs propagation. However, in a more comprehensive test (e.g., tests involving a spherical air-bearing testbed), no artifacts would be required to construct RW angular acceleration and the framework proposed here remains fully applicable for physically induced failures. Finally, to address the increased noise around zero RW speed, RWs were spun-up to an initial $\mathbf{\Omega} = [100, -100, -100, 100]$ rad/s. With these adjustments, we achieved a closed-loop operation between the SatSim and the physical RWs.

HIL-c: Adding Embedded Computer in the Loop

In this stage, we aim to ensure that the controller is on the embedded hardware. This involved two additional tasks: (a) To check that the controller still provides correct output even when it receives the noisy velocity measurements, and (b) To ensure that the controller node correctly implements failure induction logic, i.e., computes the correct RW commands to be sent to the RW.

Up to this point, RWs are still being commanded by the Satellite Simulator. As an intermediate step we executed the controller block on the Satellite Simulator and, in parallel, on the Jetson Nano embedded computer to assess their performance. Although the ControllerNode on the Jetson Nano had already been validated during the SIL stage, the main difference now is the noisy data measurements from the physical RWs, and newly incoporated satellite state estimates from an attitude determination EKF.

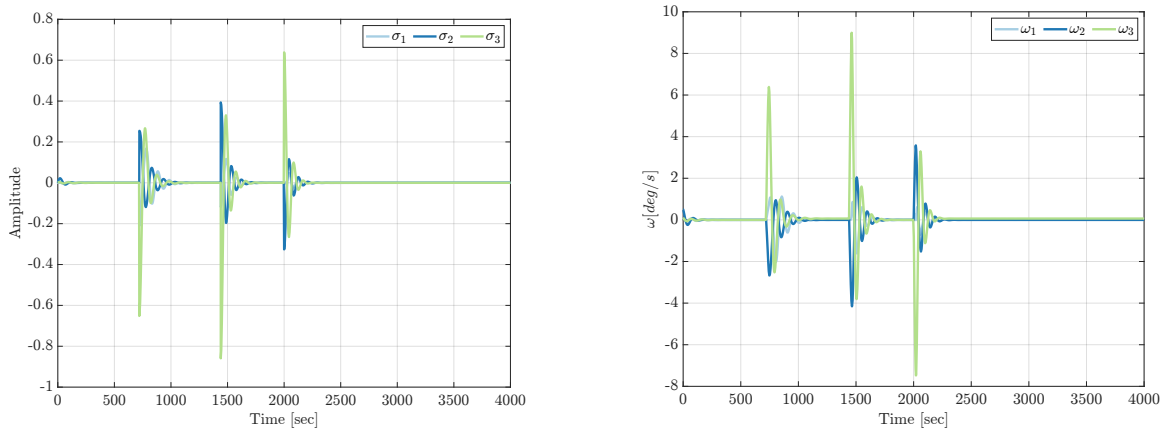
The estimated states from the EKF (computed in Satellite Simulator) and real-time RW angular velocity measurements were sent to the ControllerNode on the Jetson Nano, and its outputs are transmitted back to the SatSimNode for comparison and verification. Once validated, we replace SatSim's RW commands with the commands calculated on the Jetson Nano to fully execute the control computations on the embedded computer.

To validate the controller in this final setup, we employed the same test scenario from our previous work. In this scenario, the satellite begins by aligning itself with the Earth-Centered Inertial (ECI) frame and then alternates between its initial orientation and nadir-pointing three times. The simulation scenario lasts for 4000 seconds, during which the satellite switches its orientation every 12 minutes, then maintains nadir-pointing after 2000 seconds. The only changes we made to the simulation parameters were to increase the degraded wheel factor from 0 to 0.5 to induce a partial failure instead and reduce K_{ICL} gain by an order of magnitude, i.e., $K_{ICL} = 1$. The gain adjustment was necessary to avoid demanding torque commands that exceed the limits of the RWs, which initially caused the simulator and the ControllerNode on the Jetson Nano to stop once the ICL term was activated. This highlights the gap in sim-to-real, where sometimes control torques that are feasible in simulation do not mean that they can be replicated in a real test.

RESULTS

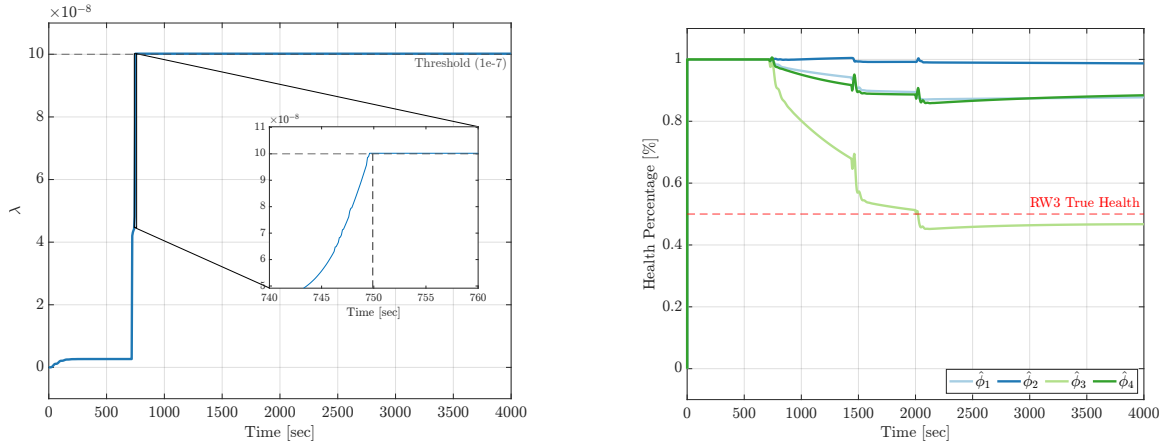
The satellite was able to follow its guidance commands throughout the simulation as evident in Figure 6a and Figure 6b. Figure 7 shows the RW health estimation performance, i.e., $\hat{\boldsymbol{\theta}}$. The

lambda λ plot represents the verifiable excitation level due to the input-output data accumulated by the system.[15] Although the lambda λ reached the defined threshold value earlier than in purely simulated tests, the health estimation $\hat{\boldsymbol{\theta}}$ can be seen to converge to its true value. Because we reduce the gain K_{ICL} to avoid the simulation from stopping, it affects the convergence rates, taking more time to converge. As the satellite perform more maneuvers, the action of the gradient-based term (i,e., first term in Equation (5)) helped move it closer to its true value faster.



(a) Error MRP (b) Body Angular Velocity

Figure 6: Performance of the Attitude Tracking Accuracy.



(a) Lambda (b) Estimated RW Health

Figure 7: RW Health Estimation with RW hardware in the loop.

Two consistent behaviors with prior results were observed. The estimation accuracy for degraded RW#3 is better than for the non-degraded wheels (steady-state error) and that there was overshoot on the estimation for RW#3. The steady state error was partly due to the numerical integration errors from the forward-Euler integration algorithm used to compute the estimated value $\hat{\boldsymbol{\theta}}$ via integration of Equation (5), as well as the fact that in the development of the controller some parts of the simulation model, such as attitude perturbations were neglected, leading to a disparity between the two.

Figure 8 shows the angular velocity measurements from the RW. Due to internal low-pass filtering implemented in the digital motor controller, the measurement noise was observed only at low-speed regions. Despite the measurement noise, it has minimal impact on the performance of the attitude tracking and estimation of the controller, as seen in earlier plots.

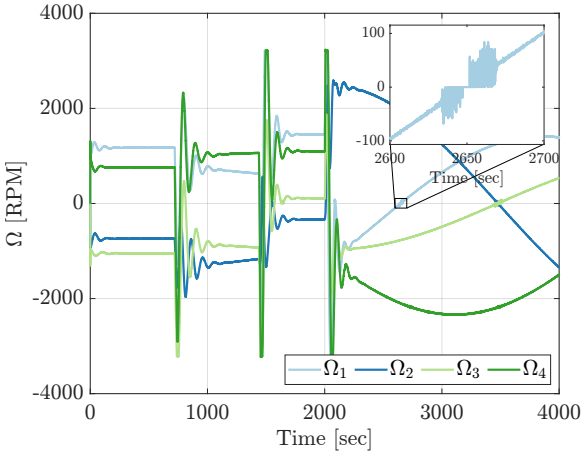


Figure 8: RW Angular Velocity Measurement.

Table 1 reports the performance of the Jetson Nano embedded computer during the HIL test. The controller executes the loop in 3.86 ms on average with a worst case of 6.62 ms, which is well below the 100 ms of the control loop limits. The average CPU load across the four cores is 8.76%, and the maximum value reached 36.70% at the start of the HIL test, which is an expected behavior during the system initialization. Memory usage appears to show a high value of 86.13% on average. However, when we conducted an additional measurement of memory usage while the system is idle, it reported similar value which suggest that there may be some background services running. The high memory usage value may not representative of what actually being consumed by the ROS2 nodes themselves and additional test and analysis are needed to get better understanding of the memory usage by the nodes.

Table 1: Jetson Nano Performance Metrics.

Metric	Execution Time (ms)	CPU Load (%)	Memory Usage (%)
Average	3.86	8.76	86.13
Maximum	6.62	36.70	87.40
Std Dev	0.20	2.90	0.26

FUTURE WORK

Given the results and lessons learned in this HIL testing, the avenues for future work are highly promising. The next immediate step in validating the adaptive controller in a simulation setup that mimics its true mission profile as close as possible is to transition the RW hardware setup from a flat test bench array to one where the RWs are placed in the design configuration array inside a mock satellite on an rotational air bearing. This setup would alleviate the problematic issues caused by the EoM in the simulation, allowing for a more comprehensive and thorough analysis of the controller.

With this setup another possibility for future work includes moving the EKF that is currently hosted on the simulation computer along with the attitude determination node onto the embedded computer, as it would be on a real mission. This testing would ensure all necessary communication and computations that would be required by the embedded computer would be possible without causing too much computational burden.

Finally, with the EoM now being omitted from the simulation setup, we would then be able to physically induce failures to the RW again to an even more realistic scenario to test the learning capability of the controller. With this component of the HIL test, performed on the air bearing testbed, full validation of the adaptive controller would be achieved.

CONCLUSION

In this paper, we validated an adaptive satellite attitude controller capable of estimating the health level of its RWs through the HIL test that includes an embedded computer and physical RW hardware with artificially induced failure. Using the HIL testbed, we demonstrated that the proposed adaptive controller performed considerably well under hardware constraints, with noisy measurements from the sensor. In addition, through the test, we have established a baseline validation of our envisioned modular HIL testbed design that is flexible and scalable. Despite several issues encountered during the development of the HIL testbed, the lessons learned are valuable as we move towards the next step involving a full air-bearing testbed experiment and other comprehensive end-to-end system tests in the future. This work will contribute to advancing the state-of-the-art ground hardware testing of fault-tolerant spacecraft attitude controllers.

ACKNOWLEDGMENT

The authors would like to thank Cole Schumacher for his contribution to the initial tests and integrations of the RWs to the HIL testbed.

REFERENCES

[1] F. L. Markley, R. G. Reynolds, F. X. Liu, and K. L. Lebsock, "Maximum Torque and Momentum Envelopes for Reaction Wheel Arrays," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1606–1614, 10.2514/1.47235.

[2] H. Hassrizal and J. Rossiter, "A survey of control strategies for spacecraft attitude and orientation," *2016 UKACC 11th international conference on control (CONTROL)*, IEEE, 2016, pp. 1–6.

[3] M. Y. Ovchinnikov and D. Roldugin, "A survey on active magnetic attitude control algorithms for small satellites," *Progress in Aerospace Sciences*, Vol. 109, 2019, p. 100546, https://doi.org/10.1016/j.paerosci.2019.05.006.

[4] M. N. Hasan, M. Haris, and S. Qin, "Fault-tolerant spacecraft attitude control: A critical assessment," *Progress in Aerospace Sciences*, Vol. 130, 2022, p. 100806, https://doi.org/10.1016/j.paerosci.2022.100806.

[5] S. Ahmed khan, Y. Shiyou, A. Ali, S. Rao, S. Fahad, W. Jing, J. Tong, and M. Tahir, "Active attitude control for microspacecraft; A survey and new embedded designs," *Advances in Space Research*, Vol. 69, No. 10, 2022, pp. 3741–3769, https://doi.org/10.1016/j.asr.2022.02.020.

[6] A. Li, M. Liu, X. Cao, and R. Liu, "Adaptive quantized sliding mode attitude tracking control for flexible spacecraft with input dead-zone via Takagi-Sugeno fuzzy approach," *Inf. Sci.*, Vol. 587, Mar. 2022, pp. 746–773.

[7] M. Mirshams and M. Khosrojerdi, "Attitude control of an underactuated spacecraft using tube-based MPC approach," *Aerospace Science and Technology*, Vol. 48, 2016, pp. 140–145, https://doi.org/10.1016/j.ast.2015.09.018.

[8] P. Iannelli, F. Angeletti, and P. Gasbarri, "A model predictive control for attitude stabilization and spin control of a spacecraft with a flexible rotating payload," *Acta Astronautica*, Vol. 199, 2022, pp. 401–411, https://doi.org/10.1016/j.actaastro.2022.07.024.

[9] S. Mokhtari, A. Abbaspour, K. K. Yen, and A. Sargolzaei, "Neural Network-Based Active Fault-Tolerant Control Design for Unmanned Helicopter with Additive Faults," *Remote Sensing*, Vol. 13, No. 12, 2021, 10.3390/rs13122396.

[10] C. Riano-Rios, R. Bevilacqua, and W. E. Dixon, "Differential drag-based multiple spacecraft maneuvering and on-line parameter estimation using integral concurrent learning," *Acta Astronautica*, Vol. 174, 2020, pp. 189–203, https://doi.org/10.1016/j.actaastro.2020.04.059.

[11] R. Sun, C. Riano-Rios, R. Bevilacqua, N. G. Fitz-Coy, and W. E. Dixon, "CubeSat Adaptive Attitude Control with Uncertain Drag Coefficient and Atmospheric Density," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 2, 2021, pp. 379–388, 10.2514/1.G005515.

[12] X. Xie, T. Sheng, Y. Zhang, J. Wang, and X. Chen, "Adaptive Fault-Tolerant Attitude Control for Rigid Spacecraft With Disturbances and Uncertainties," *2023 China Automation Congress (CAC)*, IEEE, Nov. 2023, pp. 596–600.

[13] S. M. Sadigh, A. Kashaninia, and S. M. M. Dehghan, "Adaptive sliding mode fault-tolerant control for satellite attitude tracking system," *Adv. Space Res.*, Vol. 71, Feb. 2023, pp. 1784–1805.

[14] C. Wang, L. Guo, C. Wen, Q. Hu, and J. Qiao, "Event-Triggered Adaptive Attitude Tracking Control for Spacecraft With Unknown Actuator Faults," *IEEE Trans. Ind. Electron.*, Vol. 67, Mar. 2020, pp. 2241–2250.

[15] G. Nehma, C. Riano-Rios, M. Sakal, and M. Tiwari, "Adaptive Controller for Simultaneous Spacecraft Attitude Tracking and Reaction Wheel Fault Detection," https://camilori.com/wp-content/uploads/2025/07/RW_health_estimation___Journal_version.pdf, 2025. Manuscript under review at *Journal of Spacecraft and Rockets*.

[16] H. Schaub and J. Junkins, *Analytical Mechanics of Space Systems*. American Institute of Aeronautics and Astronautics, 4th ed., 2018.

[17] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, Vol. 7, No. 66, 2022, p. eabm6074, 10.1126/scirobotics.abm6074.

[18] Maxon Motor AG, "EC 60 flat Ø60mm, brushless, 100W (Part No. 647691) Datasheet," https://www.maxongroup.com/maxon/view/product/647691, July 2025. [Online; accessed 25-Jul-2025].

[19] Maxon Motor AG, "EPOS4 Compact 50/5CAN Digital Position Controller (Part No. 541718) Datasheet," https://www.maxongroup.com/maxon/view/product/control/Positionierung/EPOS-4/541718, July 2025. [Online; accessed 25-Jul-2025].

[20] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*. Boca Raton, FL: Chapman & Hall/CRC, 2 ed., Oct. 26 2011, 10.1201/b11154.

[21] I. Sadeghzadeh and Y. Zhang, *A Review on Fault-Tolerant Control for Unmanned Aerial Vehicles (UAVs)*, 10.2514/6.2011-1472.

[22] P. R. Yanyachi, A. Mamani-Saico, X. Wang, and B. Espinoza-García, "Development of an Air-Bearing Testbed for Nanosatellite Attitude Determination and Control Systems," *IEEE Access*, Vol. 12, 2024, pp. 168864–168876, 10.1109/ACCESS.2024.3497722.

[23] M. Post, J. LI, and R. Lee, *Nanosatellite Air Bearing Tests of Fault-Tolerant Sliding-Mode Attitude Control with Unscented Kalman Filter*, 10.2514/6.2012-5040.

[24] N. P. Nguyen and S. K. Hong, "Sliding Mode Thau Observer for Actuator Fault Diagnosis of Quadcopter UAVs," *Applied Sciences*, Vol. 8, No. 10, 2018, 10.3390/app8101893.

[25] B. Ghalamchi, Z. Jia, and M. W. Mueller, "Real-Time Vibration-Based Propeller Fault Diagnosis for Multicopters," *IEEE/ASME Transactions on Mechatronics*, Vol. 25, No. 1, 2020, pp. 395–405, 10.1109/TMECH.2019.2947250.

[26] D. Vey and J. Lunze, "Experimental evaluation of an active fault-tolerant control scheme for multirotor UAVs," *2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)*, 2016, pp. 125–132, 10.1109/SYSTOL.2016.7739739.

[27] N. P. Nguyen, N. Xuan Mung, and S. K. Hong, "Actuator Fault Detection and Fault-Tolerant Control for Hexacopter," *Sensors*, Vol. 19, No. 21, 2019, 10.3390/s19214721.

[28] M. Saied, B. Lussier, I. Fantoni, C. Francis, H. Shraim, and G. Sanahuja, "Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5266–5271, 10.1109/ICRA.2015.7139933.

[29] R. Avram, X. Zhang, and J. Muse, "Quadrotor Sensor Fault Diagnosis with Experimental Results," *Journal of Intelligent & Robotic Systems*, Vol. 86, 04 2017, 10.1007/s10846-016-0425-1.

[30] Z. Cen, H. Noura, T. Susilo, and Y. Younes, "Robust Fault Diagnosis for Quadrotor UAVs Using Adaptive Thau Observer," *Journal of Intelligent & Robotic Systems*, Vol. 73, 01 2014, 10.1007/s10846-013-9921-8.