AIAA 2009-1844

Check for updates

The TableSat Platform and its Verifiable Control Software

Ella M. Atkins,[*] Jeremy Green[†]
University of Michigan, Ann Arbor, MI 48109

Jianliang Yi,[‡] Honguk Woo,[§] James Browne,[**] Aloysius Mok[††]
University of Texas, Austin, TX 78703

Fei Xie[‡‡]
Portland State University, Portland, OR 97207

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844

The TableSat single degree-of-freedom tabletop satellite platform was developed to support education and research activities in embedded software and control systems. This paper describes the TableSat system, actuated by a pair of computer fans as thrusters, and then focuses on the evolution of its software for sensor calibration and fault-tolerant control. Sensor systems include a three-axis magnetometer, core sun sensor suite, and single-axis rate gyro. The sun sensor and magnetometer together provide redundancy in pointing direction measurement, supporting limited sensor fusion and enabling the system to continue operation given redundant sensor failure situations. Baseline software was developed to support rate control and pointing control based either on magnetometer or sun sensor data. This software was encapsulated within the ObjectBench code generation system to support formal software specification. Guarantees of execution properties such as sequencing and timing constraints were established with the ObjectCheck model checking system. Design choices, evaluation processes, and results from all phases of software development, testing and verification are presented. Use of the TableSat system as an interdisciplinary research and education tool is also discussed.

I. Introduction

The TableSat platform is a one degree-of-freedom "tabletop" satellite that emulates the dynamics, sensing, and actuation capabilities required for spacecraft attitude control. Illustrated in Figure 1, TableSat is driven by two "computer fan" thrusters commanding clockwise and counter-clockwise torques, respectively, and experiences extremely low friction on its central pivot point. TableSat was developed as a classroom, demonstration, and research tool, enabling students and researchers to gain practical experience with real-world sensor and actuator systems analogous to those found in deployed Aerospace systems. Although limited to single degree-of-freedom studies, multiple guidance, navigation, and control modes continue to be developed, providing experiences ranging from rate control and (attitude) trajectory tracking to nonlinear sensor calibration and dynamics compensation.[1] The embedded software environment supports hard real-time multi-threaded execution, enabling realistic if limited-scale emulation of a spacecraft software-enabled control system.

This paper describes the TableSat hardware and software systems, focusing on the evolution of sensor calibration, control algorithm, software development, and formal verification. Sensor calibration and control law development was made challenging by significant signal disturbances in TableSat's laboratory environment, as well as actuator stiction and dynamically-changing friction properties. Calibration and control software was originally developed as simple, stand-alone functions,

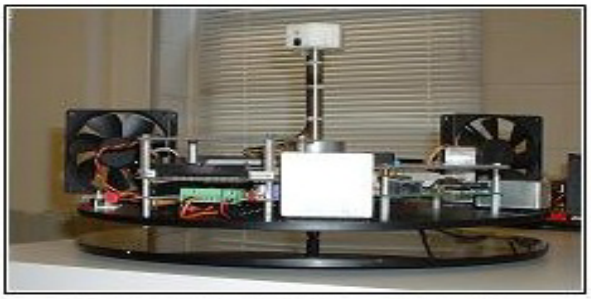


Figure 1. TableSat: The Tabletop Satellite.

[*] Associate Professor, Aerospace Engineering Dept., Associate Fell
[†] Undergraduate Research Assistant, Electrical Engineering & Comp. Sci. Dept.
[‡] Graduate Research Assistant, Computer Science Dept.
[§] Graduate Research Assistant, Computer Science Dept.
[**] Professor, Computer Science Dept.
[††] Professor, Computer Science Dept.
[‡‡] Assistant Professor, Computer Science Dept.

Copyright © 2009 by Ella M. Atkins. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

facilitating low-level debugging efforts and extensions by Aerospace students not yet familiar with object-oriented software development. This code, while functional, was verified only through repeated trials, not through formal software modeling and verification processes. As a next step, we translated the "building block" calibration and control procedures into the ObjectBench modeling and code generation package. The goal was to achieve *verifiable fault-tolerant control* of TableSat, with the intent of ultimately extending our understanding of how to synergistically model and verify software and control algorithms in the presence of hardware or software faults. In ObjectBench, state models include a nominal attitude controller for which errors are monitored over time, models of expected sensor readings, and a safe controller.[2, 3] The ObjectCheck [reference] verifier is used to convert Objectbench models to verifiable representation and to formally verify the correctness of system behaviors. For this paper we also introduce the ability to switch between an attitude controller that points based on magnetic North and a controller that points based on the direction of a single light source simulating the sun. Sensor failures are modeled in this framework, and when either a failure is explicitly detected or when the attitude does not conform to expected trends over time, the software guarantees stability through a switch to the other attitude controller or a safe 'core' controller that can either stop the platform or maintain a steady (slow) rotation rate. The formal models and software move beyond a single switch to "safe" TableSat, however, by continuing to monitor and model the system, automatically restoring nominal attitude control logic if/when the situation is appropriately modeled and remedied, as would be the case if either a transient outage or disturbance passed. Although verification on TableSat is relatively straightforward given its limited sensor suite and single degree-of-freedom design, we are confident such approaches to co-monitor and co-manage software and control system processes will be critical to manage the next generation of sophisticated and increasingly self-sufficient aircraft and spacecraft flight management systems.

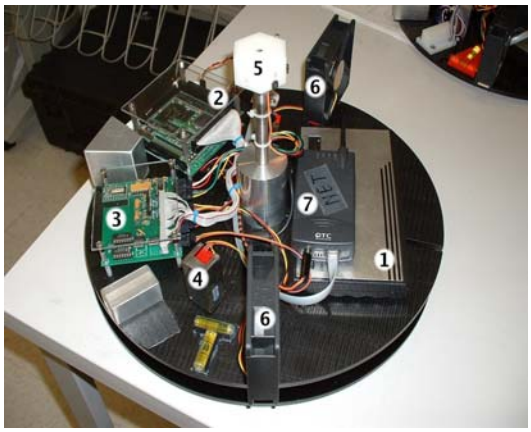
Below, we first introduce the TableSat system design, from electronics hardware to sensor systems. We then summarize the development and evaluation of baseline sensor calibration and control law development activities, describing significant and somewhat unexpected challenges associated with accurately characterizing TableSat's pointing direction in its current laboratory locale. We present the design of the ObjectBench models and associated results obtained during use of the integrated code. Finally, we discuss the use of TableSat and ObjectBench as practical and complementary pedagogical tools. TableSat provides a number of navigation, control, and sensor system challenges for Aerospace students, and use of ObjectBench and ObjectCheck to better formalize TableSat models and code has the potential to provide valuable software validation exposure for Aerospace students with dynamics and control interests. TableSat sensors and control software can provide Computer Science students with a challenging real-world experience in understanding and adapting mathematical models to challenging sensor and actuator behaviors difficult to fully characterize with discrete state models.

II. TableSat Electromechanical Systems

Figure 2 shows a top view of TableSat with pertinent components labeled. TableSat rotates in the horizontal plane about a central pivot point of low but nontrivial friction. Two unidirectional computer fans together provide positive and negative torque actuation capability. A high-precision rate gyro saturating at a 90 degree per second rate measures angular velocity. A three-axis magnetometer and set of four core sun sensor (CSS) elements measure pointing direction. An onboard 100MHz Diamond Systems Prometheus PC/104 computer running the QNX Neutrino real-time operating system communicates to a ground station via a wireless 802.11b interface. All rate gyro, magnetometer, and CSS sensors output analog signals from 0-5 Volts that are read by the computer through 16-bit analog-to-digital converters. Computer fan actuation is regulated through power-amplified 12-bit digital-to-analog output. The two computer fans each produce a tachometer signal that can be read through a switched counter-timer input on TableSat (i.e., a digital output from the computer dictates whether positive or negative-torque fan tachometer signal is sent to the counter-timer).

TableSat exhibits nonlinear dynamics, including off-axis wobble given certain actuation magnitudes and switching profiles. TableSat provides a variety of practical challenges in hardware and software for control engineers with limited experience using real physical systems. In its current lab environment, magnetometer and CSS sensor calibrations are nonlinear and dependent on location and time of day (e.g., proximity to the ferrous building frame, lighting conditions). Control system designs have ranged from rate control based on gyro readings to pointing with single or multiple sensor measurements.

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844



1. Laptop Battery
2. PC/104 Computer Stack
3. Sensor Interface Board
4. Gyroscope
5. Sun Sensors
6. Computer Fan "Thruster" Pair
7. Wireless Adapter

Figure 2: TableSat Components (Top Assembled View)

III. Baseline TableSat Software

TableSat supports two Diamond Systems PC/104 computers running the QNX Neutrino 6.2.1 operating system. A 100 MHz Prometheus board with embedded data acquisition and network support was used for the work presented in this paper, and an upgraded Athena II 400 MHz board supporting the same software also has been integrated into a second TableSat prototype otherwise identical to the first. To facilitate software development, we adopted the Diamond Systems Corporation Universal Driver (DSCUD) software to provide low-level interface to the data acquisition system. Use of DSCUD will also facilitate migration to alternate supported operating systems in the future as needed. The baseline software used for calibration and control law development was structured as single thread that initialized the hardware, opened data files, and executed a loop, nominally at 100 Hz. Each execution cycle sequentially reads all analog sensor channels, applies calibration equations to convert readings to MKS units, then computes actuator outputs based on a single reference command set by the user. A pair of baseline programs, "read_sensor" and "write_motor", are made available to monitor raw voltages and manually set motor commands. These programs are frequently used to validate or debug hardware functionality and to provide functional code upon which students can build embedded calibration, filtering, and control software.

Sensor calibration and control law development, with results described below, were performed in two phases. First, we established a baseline rate control program based on a simple proportional-integral equation using "gyro-volt" units to achieve steady rotation rates. We then established a gyro calibration by observing the periodicity of the magnetometer and CSS readings for different gyro-volt speeds. This allowed us, without additional support equipment, to calibrate the gyro from volts to angular velocity in degrees/sec. We subsequently recorded magnetometer and CSS sensor readings throughout the 0-360 degree pointing range and developed nonlinear calibration algorithms as described below in the results section. Once calibrations were in place, we developed two attitude control laws to complement our baseline rate controller. The first used magnetometer only to point in a compass direction relative to magnetic North, structured as a proportional-integral-derivative control law supported also by rate gyro data. The second attitude controller used the pointing estimate provided by the CSS along with rate gyro data in a separate proportional-integral-derivative (PID) control law. As will be discussed below, the magnetometer provides more accurate sensor data thus was used as our primary means of pointing

IV. ObjectBench Software Development

Once the baseline controller was working properly, we created a multi-threaded implementation to facilitate analysis of software execution specifies (e.g., timings) as well as control law performance. TableSat onboard software is composed of the following four threads, each of which executes as a periodic real-time task of constant frequency:

- *State Estimator* thread that reads sensors and estimates the current state of TableSat;
- *Controller* thread that applies the specified control law to compute output torque;
- *Actuator* thread that outputs computer fan voltages to achieve the commanded control torque;
- *Communication* thread that supports data/command transmission from/to an external client program.

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844

Our ObjectBench implementation focuses on anomalies or failures, not handled at all by our baseline software. Due to the atmosphere-free spacecraft environment and the low friction experienced by TableSat, uncontrolled actuation can lead to undesirable high-speed rotations that impart dangerous forces to system components. For space operations, damping rotational motion to zero, or near-zero, is viewed as a capability all spacecraft must possess at all times, even after being "safed" due to other exception(s). We have implemented a core (safe) controller for TableSat that relies only on its rate gyro to stabilize TableSat motion should other sensors or controllers be compromised. Rate control is a straightforward algorithm. A simple proportional controller approximately achieves commanded rates; augmentation with an integral term drives steady-state rate error to zero and has been demonstrated effective given TableSat's relatively high thread execution rates and small external disturbance magnitudes. Although subject to bias thus typically small but non-zero steady-state drift, rate control is not difficult to prove correct so long as the rate gyro and computer fans function properly, a simplifying assumption we make to constrain the scope of our core controller for this work. Because it uses only rate gyro data, the proportional-integral rate control law deployed as the TableSat safe core control law is impervious to changes in lighting conditions and magnetic field disturbances.

Rate control is a safe backup but cannot accurately point TableSat in any particular direction. Under normal operating conditions, TableSat incorporates inertial attitude measurements in its feedback control law. Three inertial attitude sensing strategies are possible, all relying on the gyro for angular rate estimates: 1) Magnetometer-based (North-referenced) pointing, 2) Sun sensor (CSS) pointing, and 3) Pointing over a fused estimate of magnetic field and light sensor measurements. The initial implemented controllers relied strictly on the magnetometer for pointing information, but CSS-based control is more capable in TableSat's University of Michigan locale due to ferrous building structure disturbance of magnetic field.

Failure of attitude sensors is the primary reason the risky controller will malfunction in our experiments. To conduct controlled, repeatable tests, we induce sensor failures. The CSS fails in two ways: by shutting down the single light source or by adding a second "competing" light source. Although we have identified an ideal lab location at which our magnetometer is calibrated, we can similarly induce magnetometer failures by carrying TableSat through the lab during testing. A magnetometer-based control law destabilizes when magnetic North and a building beam exert approximately equal magnetic influences. Pointing direction is marginally stable but incorrect when TableSat further approaches a ferrous building column due to the dominance of the structure's local magnetic field but with a less-localized directional signal.

Regardless of the pointing angle data source, TableSat applies baseline proportional-derivative and proportional-integral-derivative controllers over pointing direction and angular rate to achieve and maintain pointing commands. For this project, it is sufficient to command TableSat to achieve a single reference attitude or constant angular rate (likely zero for the safe controller) since we are studying resource (sensor) failure rather than logic failure. However, as an analogue to spacecraft science missions, TableSat will ultimately be extended to execute uploaded pointing command sequences of specified durations, enabling more complex analysis of the long-term interaction of safe core and risky control laws.

A. Formal Software Model

An xUML model of TableSat[2,3] was derived by refactoring the C-implemented software into logical components, extending the component set to include multiple controller components and implementing these components as classes in xUML. The xUML model of TableSat was developed in ObjectBench and verified by ObjectCheck.[4] ObjectCheck is a model checker for xUML models, which supports verification of both safety and liveness properties. The following are properties which were verified on the xUML model of the TableSat control core:

- **Key control states are reachable.** This type of properties ensures that the key control states of the system are reachable. They are in the form of *NEVER(Control_State),* which indicates that the referenced state never appears in any system execution. Note that *Control_State* can be generalized to any predicate over system states. The expected result from verifying these properties is that they do not hold on the system, which indicates that the referenced states are reachable in some system executions. If these properties do hold on the system, then there are certain control states not reachable and the control logic must be examined for the causes.
- **Bad states are unreachable.** This type of property shows that the control core stays in its predefined safety envelope through checking that certain bad states are not reachable in any system execution. They are formulated

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844

in the form of *NEVER(Unreachable_State).* Note that *Unreachable_State* can be generalized to any predicate over system states. The expected result from verifying these properties is that they hold on the system, indicating the referenced states are unreachable. If these properties do not hold on the systems, ObjectCheck returns a counter example explaining where and how the system breaks out of the desired safety envelope.

- **Scheduling fairness.** This type of property ensures tasks of the control core are fairly scheduled. They are in the form of *Repeatedly(Task_Execution_Entry_Point).* The expected result from verifying these properties is that they should hold on the system. If these properties do not hold on the system, then there is possibility of starvation in the system, i.e., some task is ready to be executed, but never gets executed. For starvation, ObjectCheck returns an infinite loop of the system that excludes the execution entry point of the task. Starvation is a major concern in embedded systems, which keeps systems from responding to certain events and breaks their real-timeness.

V. TableSat Results

Results from calibration and ObjectBench tests are presented below. All were conducted in the Autonomous Aerospace Systems Lab at the University of Michigan. TableSat has remained at a constant position in the lab chosen to maintain as much distance from steel building beams as possible. However, as will be shown below, magnetometer readings are still influenced by the building, requiring nonlinear calibration and recall of recent attitude state history. Note that the laboratory receives significant natural lighting from the West, thus all tests with sun sensors were conducted either at night or early morning. TableSat is accessible through a local network and an overhead video feed connected to the web, enabling students and research collaborators to view and control TableSat motion in real-time. The only local support requirements are that TableSat be manually powered down and tethered for recharge after approximately eight hours of use depending on thruster use, and that the sun sensor light source be activated and repositioned as needed. Over its 4+ years of use, TableSat, in its stable laboratory operating environment, has exhibited a high level of reliability. We rely on it each term for 20+ students to use for one or more projects and have thusfar experienced only network outage and very sporadic balance issues (e.g., due to a balancing weight moving during a commanded high-speed rotation).

A. Sensor Calibration

TableSat sensors have been calibrated by a number of users. Figure 3 gives shows the typical noise, in volts, seen across the TableSat sensor systems. These noise levels are consistent with expectations from our 16-bit A/D conversion system configured to be single-ended with full-scale range 0 to +5V. The rate gyro, with internal temperature compensation, gives a linear calibration from volts to rotational velocity (degrees/second), not shown here. The rate gyro, now calibrated by several classes of students over the past three years as well as by lab researchers, has demonstrated excellent repeatability and is considered TableSat's most reliable and consistent sensor. The rate gyro can be used as a stand-alone sensor for rate control, the simplest baseline controller used for our experiments. It can also be used to provide rate information in an attitude controller referencing magnetometer and/or CSS for pointing information.

A typical plot of raw magnetometer and sun sensor voltage readings over time are shown in Figure 4. Although trends are consistent and repeatable over multiple revolutions, the magnetometer readings experience a significant disturbance we believe due to the building steel frame, and the sun sensors only provide signal over a small range (~60 degrees) centered on the angle at which each sensor points directly at the sun. The magnetometer and CSS data are quite nonlinear. As shown in Figure 4, the magnetometer data contains appreciable noise, while CSS voltage levels have limited range in this test. Note that a brighter light source increases range but sensor saturation then becomes an important consideration.

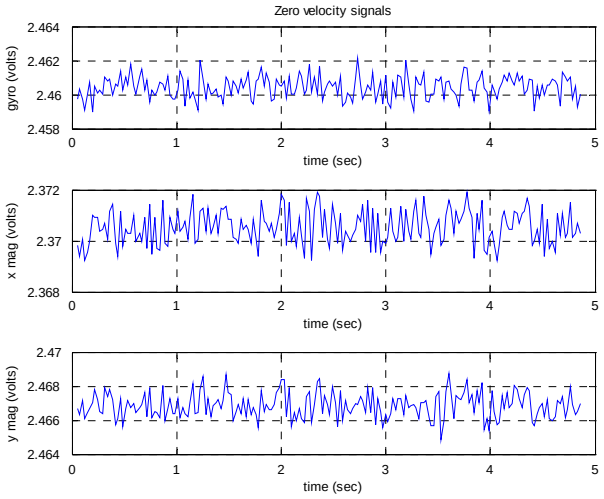


Figure 3: Sensor Noise.

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844

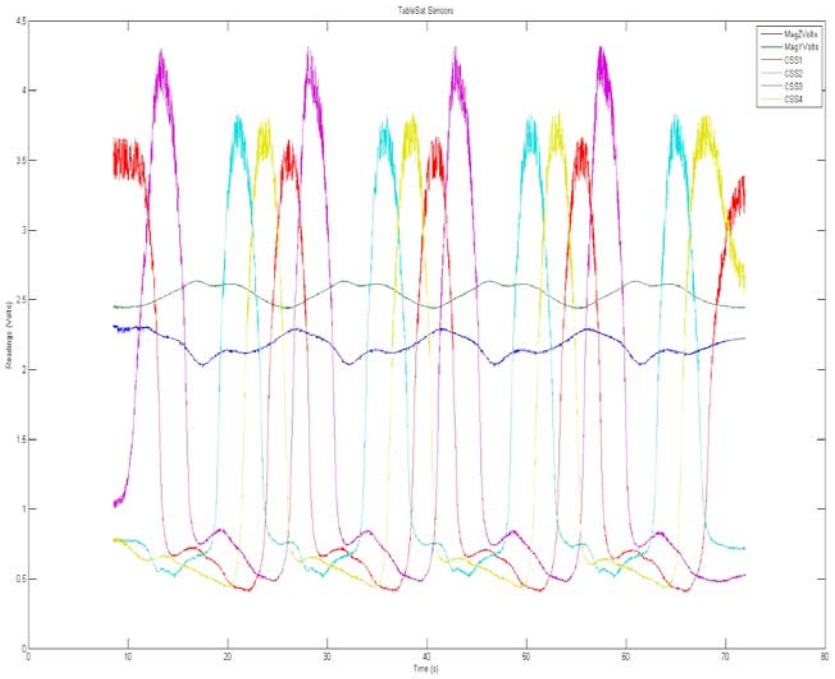


Figure 4: Raw Magnetometer and CSS Data.

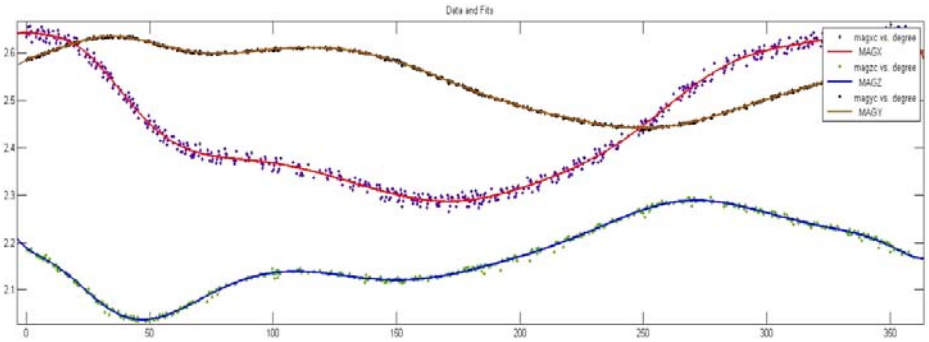


Figure 5: Example Three-axis Magnetometer Calibration.

The best magnetometer calibration obtained to-date is shown in Figure 5. Expressed as an 8th order Fourier Series, the data is well-matched to the voltage readings, which we have observed are repeatable. The software, however, must initialize the magnetometer by conducting a slow complete revolution of Tablesat to ensure unique voltage-to-heading values are captured. Data from a typical indexing sequence is illustrated in Figure 6. Once indexing is complete, the state estimate can be updated from the last known reference heading and current rate. During operation, the state estimation loop must execute with sufficient frequency to ensure the appropriate transitions are captured.

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844

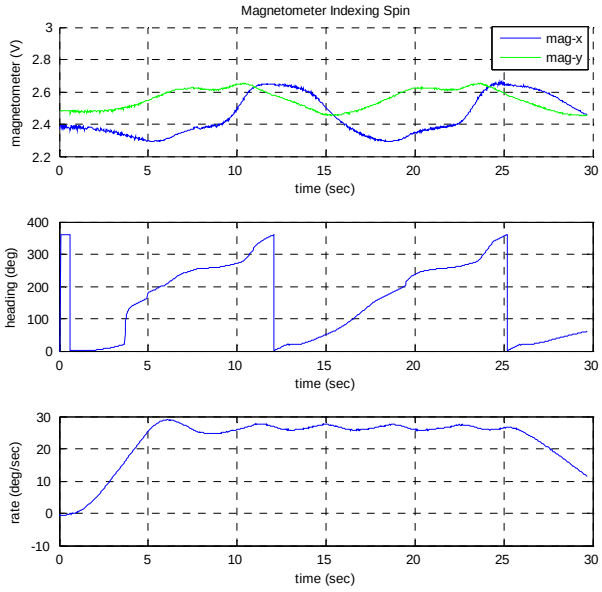


Figure 6: Magnetometer Indexing Spin Sequence (Constant Motor Output).

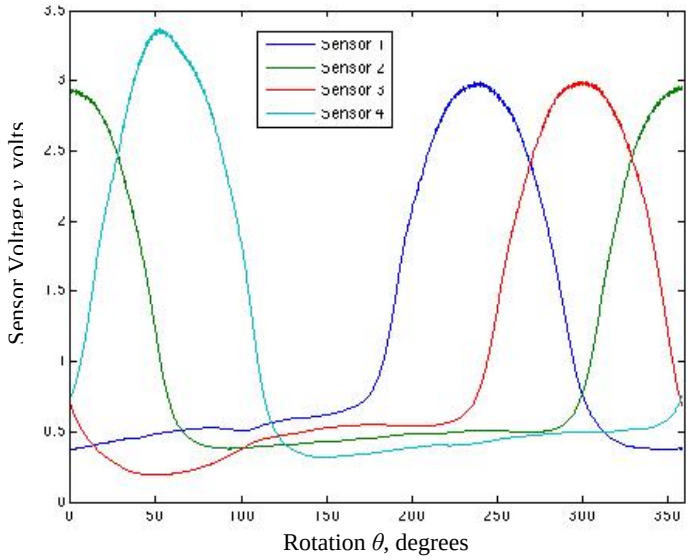


Figure 7: Example Sun Sensor Calibration Results.

Table 1: TableSat Sun Sensor Calibration Algorithm.

Priority	**Range** (degrees)	**Sensor to Read**	**Polynomial** (v=sensor voltage)
1	300-359	2	$11.3883v^3 - 55.6788v^2 + 100.7866v + 248.7776$
2	0-60	2	$-8.7188v^3 + 39.7791v^2 - 71.2196v + 92.1526$
3	240-300	1	$-9.1211v^3 + 43.8965v^2 - 82.6692v + 342.1194$
4	180-240	1	$11.4332v^3 - 57.5302v^2 + 107.5444v + 121.2973$
5	60-130	4	$-5.5257v^3 + 27.6326v^2 - 56.0133v + 143.6712$

The CSS unit was calibrated as shown in Figure 7 and Table 1. Each sensor, as described previously, is only active when within +/- 30 degrees of the light source. Five third-order polynomial equations, provided in Table 1, were derived from the CSS data to span attitudes in the range 0-130º and 180-359º. The equations use the three best sensors in the overall coverage map, although sensor three could be included as needed. Note that there was no sufficiently linear region to cover all pointing directions, thus additional sensors are needed to provide support for comprehensive attitude control. Note that pointing errors of approximately 10 degrees are observed even with our best calibration using only the CSS, thus the magnetometer baseline controller is considered the preferred option.

B. Object Bench Control Tests

The TableSat model in xUML has been generated, tested, and the stated properties formally verified. Code has been generated and the complete model has been linked to QNX and tested on TableSat. Figure 8 illustrates rate control step response from 0 to -1 rad/sec. The nontrivial transient is an unavoidable product of the relatively low saturation torque compared with system rotational inertia. Note that there is minimal overshoot due to careful gain tuning.

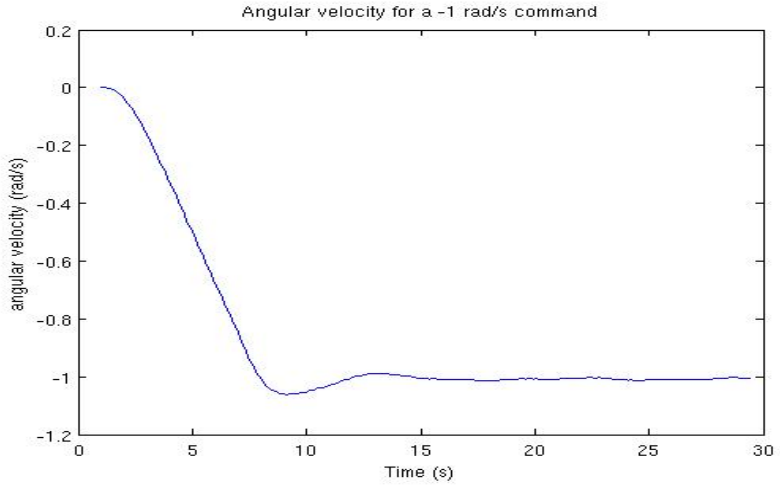


Figure 8: Safe Core Rate Control Step Response.

Attitude control law performance within ObjectBench was evaluated initially presuming sensors were functioning nominally. Figures 8-10 show typical results, one for the baseline rate controller achieving the designated rate, and two additional plots for the magnetometer-based attitude controller achieving designated reference commands that were relatively distant (Figure 9) and proximal (Figure 10) from the initial reading. Note that some overshoot exists in the attitude commands, more than was observed with a stand-alone control law operating at its maximum possible bandwidth. Results were repeatable, and the additional overshoot we believe could be mitigated by further increasing the rate at which the state estimation / control threads execute.

Figure 11 shows a sample TableSat execution sequence from the ObjectBench-generated code in which magnetometer sensor failures are simulated through a random process. We also assume the magnetometer can be "repaired", thus allow the magnetometer to be re-used should it again begin functioning correctly. As shown in the Figure, an initial self-calibration activity is executed, followed by a nominal attitude pointing maneuver. This maneuver is cut short by a simulated attitude sensor failure (magnetometer), prompting reversion to the safe controller. The safe controller examines the attitude sensor output while maintaining a nonzero steady rotation speed. A re-calibration (and software-based restoration of the functional magnetometer readings) then enables the system to shift back to its nominal attitude control mode, after which it settles onto a 145 degree reference, followed by the "final" commanded 150 degree pointing direction.

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844

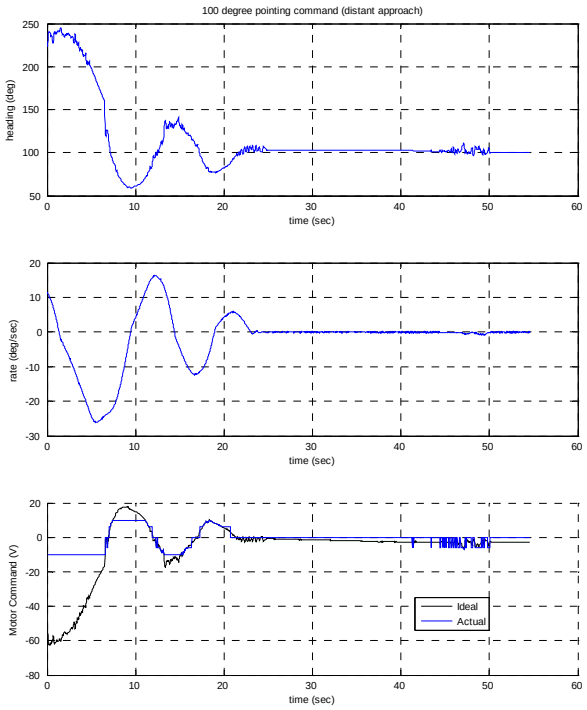


Figure 9: Attitude Control with 100 degree Reference (large initial offset case).

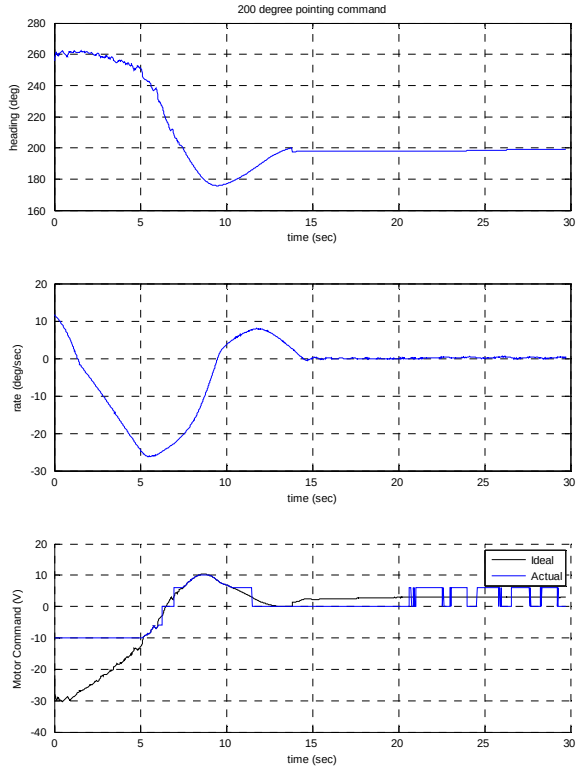


Figure 10: Attitude Control with 200 degree Reference (small initial offset case).

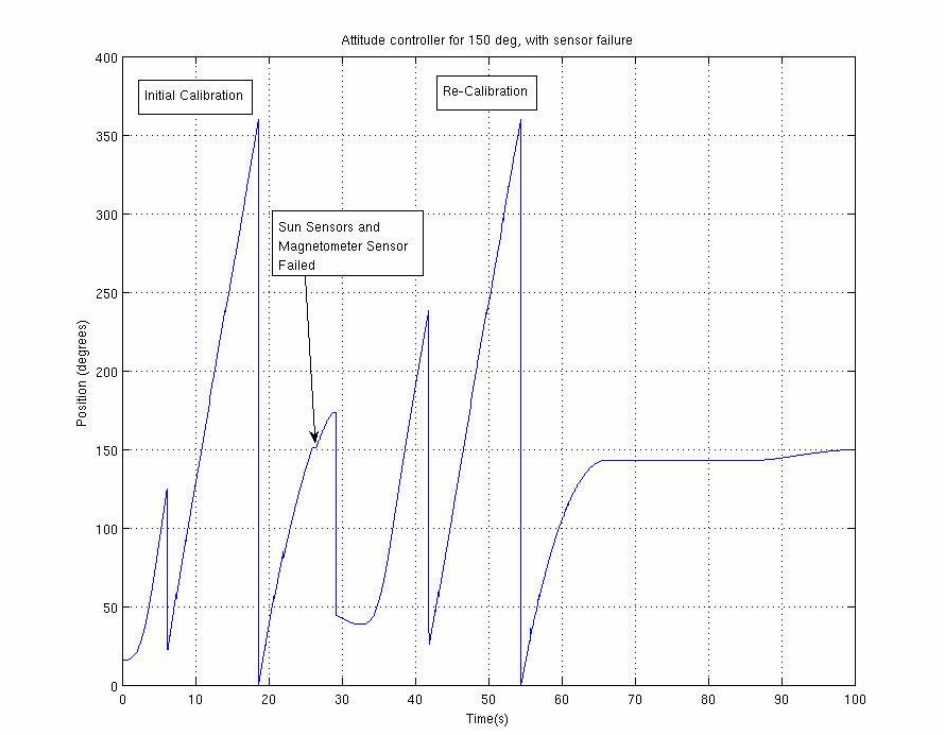


Figure 11: TableSat Heading with a Simulated Sensor Failure/Recovery Sequence.

VI. TableSat for Aerospace Engineering Education

TableSat has been adopted for use in a variety of courses and independent study research projects. First used at the University of Maryland,[§§] students in an embedded systems course developed the first sensor calibration equations as part of a senior Aerospace embedded systems course, after which one of these students continued TableSat work through an MS thesis.[1] Since then, TableSat has been used at least annually for demonstrations and assignments in introductory Aerospace control and embedded system courses at the University of Maryland and University of Michigan.

Co-author E. Atkins has utilized TableSat extensively in courses. Capable engineering freshmen, self-selecting an accelerated programming course, are challenged to create a baseline rate controller for TableSat, learning the basics of sensor calibration, A/D and D/A conversion, and proportional-integral-derivative control. This same project is used in a senior-level Flight Software Systems course, but as a single assignment rather than a final project. In the senior-level course, students are also asked to complete a challenging final project. Each year, at least two student groups have elected TableSat, attempting and succeeding at the challenging tasks of calibrating the magnetometer and/or sun sensors and subsequently using them in the best attitude control laws they can devise. To-date, Aerospace students working with TableSat have used straightforward procedural code in their efforts. We hope the work with TableSat presented in this paper will migrate into the Flight Software course, supporting better fundamental education of software engineering principles through design, implementation, and verification processes. We believe this will better prepare students to enter a workforce in which computer science and Aerospace control experts will frequently collaborate. We hope to promote the complementary process as well, specifically better exposing computer science students to the mathematics behind embedded controls in the context of software modeling and verification. .

[§§] Co-author Ella Atkins, in collaboration with Robert Sanner, designed and constructed the first PC/104-based TableSat when both were at the University of Maryland, College Park.

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844

VII. Conclusions and Future Work

This paper has presented the TableSat tabletop satellite platform, a one-degree of freedom satellite simulation platform conceived, developed, and persistently upgraded to support a variety of education and research goals. This paper describes the TableSat system hardware and software, detailing the evolution of software from baseline calibration and control modes through integration within the ObjectBench software modeling and verification tools suite. Sensors and control laws have been shown to function adequately during nominal operation, and application of ObjectBench and ObjectCheck facilitate fault detection and control system reconfiguration in cases where sensors such as the magnetometer or CSS fail.

Though TableSat has become a capable and reliable tool for research and education, future extensions can appreciably enhance performance in multiple directions. First, the University of Michigan laboratory environment still challenges the robustness of our magnetometer and sun sensor calibration algorithms. We have developed a magnetometer initialization and calibration procedure that works well, but still constrains TableSat's maximum angular velocity to a speed that guarantees we will detect transitions between each calibration region. By increasing magnetometer sampling rate to its maximum of ~1 kHz for our PC/104 board, we anticipate no significant issues in accurate magnetometer attitude determination when driving TableSat at speeds up to rate gyro saturation. The core sun sensor system currently in place can be extended straightforwardly by adding two additional sensor elements to better handle the region in which the four existing sensors point away from the light source. Although not performed to-date, magnetometer and CSS data can also be fused into a single estimate provided light source direction is accurately known with respect to magnetic North. The ObjectBench software implementation has matured to the point where we can switch between controllers, and it can accommodate future improvements to our attitude control law suite as they become available. The remaining challenge is to more fully extend fault management logic to incorporate detailed models of sensor faults, a task tightly coupled with improvements to the CSS and magnetometer calibration procedures themselves.

TableSat will continue to provide an invaluable educational embedded programming experience for students ranging from college freshmen to graduate students. Although entry-level students will remain focused on sensor calibration and PD control with maximally-simple software, we hope to use the more sophisticated ObjectBench implementation to educate upper-level students not only in embedded system control but also in the software development and verification processes required before deploying any software in practice. Tighter integration of control systems and software engineering is becoming increasingly critical throughout the engineering of today's complex Aerospace systems. Our collaborative efforts with TableSat represent a small step in this direction.

VIII. Acknowledgements

The authors would like to thank NASA Goddard Space Flight Center, led by David Mangus, for conceiving of the TableSat concept, initial prototype, and funding initial TableSat system development at the University of Maryland. The authors also thank Robert Sanner and Melissa Vess for their initial work to develop and test the sensor calibration and control software documented in Melissa Vess' thesis[1], as well as University of Michigan undergraduate Jeffrey Leath for his tireless efforts to calibrate and control TableSat in the Autonomous Aerospace Systems Laboratory despite its suboptimal magnetic field and lighting characteristics. This research was supported in part by the National Science Foundation under "Collaborative Research: SoD-TEAM: A Feedback-Based Architecture for Highly Reliable Embedded Software." Grant 0613665 to the University of Texas at Austin, Grant 0650049 to the University of Michigan and Grant 0613930 to Portland State University. We also acknowledge QNX Software Systems for support through open-source and educational licensing programs.

IX. References

[1] M. F. Vess, *System Modeling and Controller Design for a Single Degree of Freedom Spacecraft Simulator*, Master's Thesis, Aerospace Engineering, Univ. of Maryland, 2005.

[2] H. Woo, J. Yi, J. Browne, A. Mok, E. Atkins, F. Xie, "Design and Development Methodology for Resilient Cyber-Physical Systems," *First Intl. Workshop on Cyber-Physical Systems (WCPS2008),* Beijing, June 20, 2008.

[3] H. Woo, J. Yi, J. Browne, A. Mok, F. Xie, E. Atkins, C.-G. Lee, "Incorporating Resource Safety Verification to Executable Model-based Development for Embedded Systems," *Real-time Application Symposium (RTAS),* IEEE, St. Louis, MO, April 2008.

[4] F. Xie, V. Levin and J. C. Browne, *ObjectCheck: A Model Checking Tool for Executable Object-Oriented Software System Designs*, Proc. of FASE, 2002.

Downloaded by University of Michigan - Duderstadt Center on October 24, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2009-1844