SPACE AND SATELLITE SYSTEMS
UNIVERSITY OF CALIFORNIA, DAVIS

[REALOP]

[Software Overview Document]

Document ID	ADS-0020
Document Type	Engineering Subteam Overview
Version	2.0
Release Date	10/1/2024

Prepared By:	
Jacob Tkeio	jtkeio@ucdavis.edu
Charlie Kvoriak	cmkvoriak@ucdavis.edu
Version 1.0 Prepared By:	
Chris Andrade	coandrade@ucdavis.edu
Graham Bough	gwbough@ucdavis.edu

Table of Contents

Abbreviations, Acronyms, and Symbols

ADCS	Attitude Determination and Control Software
ECI	Earth-Centered-Inertial
ECEF	Earth-Centered Earth-Fixed
LVLH	Local-Vertical-Local-Horizontal
LLA	Latitude-Longitude-Altitude
NED	North-East-Down
HDD	Hard-Disk Drive
MRW	Manufactured Reaction Wheel
TLE	Two-Line Element Set
RTC	Real-Time Clock
IMU	Inertial Measurement Unit
IGRF	International Geomagnetic Reference Frame
SGP4	Simple General Perturbations Model 4
SPA	Solar Position Algorithm
NOVAS[C]	Naval Observatory Vector Astrometry Software [C language]
ISS	International Space Station
URC	Undergraduate Research Conference <3

1. Introduction

1.1 REALOP Overview

REALOP is a technology demonstration mission. We are developing our first satellite bus for use in future Earth Science missions.

On a typical mission, a satellite's primary objective is to take a picture of the Earth. To do so, it needs to determine its orientation relative to Earth and physically point itself at it before it can take a photo.

In contrast, REALOP will not have a camera. It is solely a proof of concept for hard-disk drive reaction wheels. However, it will still need to perform determination and pointing to prove that HDDs are capable of pointing a camera accurately at Earth for future missions.

So, REALOP's primary objective is to collect data that demonstrates that it can use the HDD to point at Earth. Its Attitude Determination and Control Systems are key to this objective.

1.2 Attitude Determination and Control Systems

The goal of ADCS is to determine the attitude of the satellite and control its rotation. This includes pointing in the target direction using the HDD, but also *detumbling*.

Tumbling is the initial state of the satellite after it's launched from the ISS. It doesn't know where it is or what time it is, is rotating randomly, and doesn't know which of its components are functioning. *Detumbling* is the process of canceling out this random rotation to make experimenting easier (if we have too much initial rotation, the HDD will have a much smaller noticeable effect on the overall angular velocity). It turns out detumbling requires only the magnetometer and coils!

In general, the attitude is determined using our onboard sensors: the sun sensor, magnetometer, and inertial measurement unit. The sun sensors measure the position of the sun relative to the satellite, while the magnetometer measures the magnetic field of the Earth relative to the satellite at a specific point in orbit. The IMU measures spatial acceleration and angular velocity. We know where the sun and magnetic field vectors "should be" based on our onboard models (SPA and IGRF), so we can compare our measured values with expected values to solve for our absolute orientation. This is called the TRIAD method.

Based on its absolute orientation, the satellite can determine its orientation relative to the Earth in order to point at a desired location on its surface using the HDD. When the ground station receives data that demonstrates this pointing, the mission is complete.

1.3 Software-Specific Tools

- **C language:** used for most ADCS software. This includes a development environment!
- **Git and Github:** all ADCS code is saved in Github.

All ADCS tool setup instructions can be found in the ADCS Google Drive Folder.

2. Subsystem Overview

2.1. Requirements

- Define an ADCS function that determines current attitude and commands for attitude control, broken down into several sub-functions.

- Inputs:
  - Sensor Inputs (magnetometer, sun sensor, angular velocity)
  - Orbit parameters (**TLE from uplink**)
  - Time (from **RTC after uplink**)
  - Mode (HDD, or Detumble)

- Outputs:
  - If detumbling: dipole command for magnetorquers
  - If HDD mode: disk speed command

2.2 Block Diagram

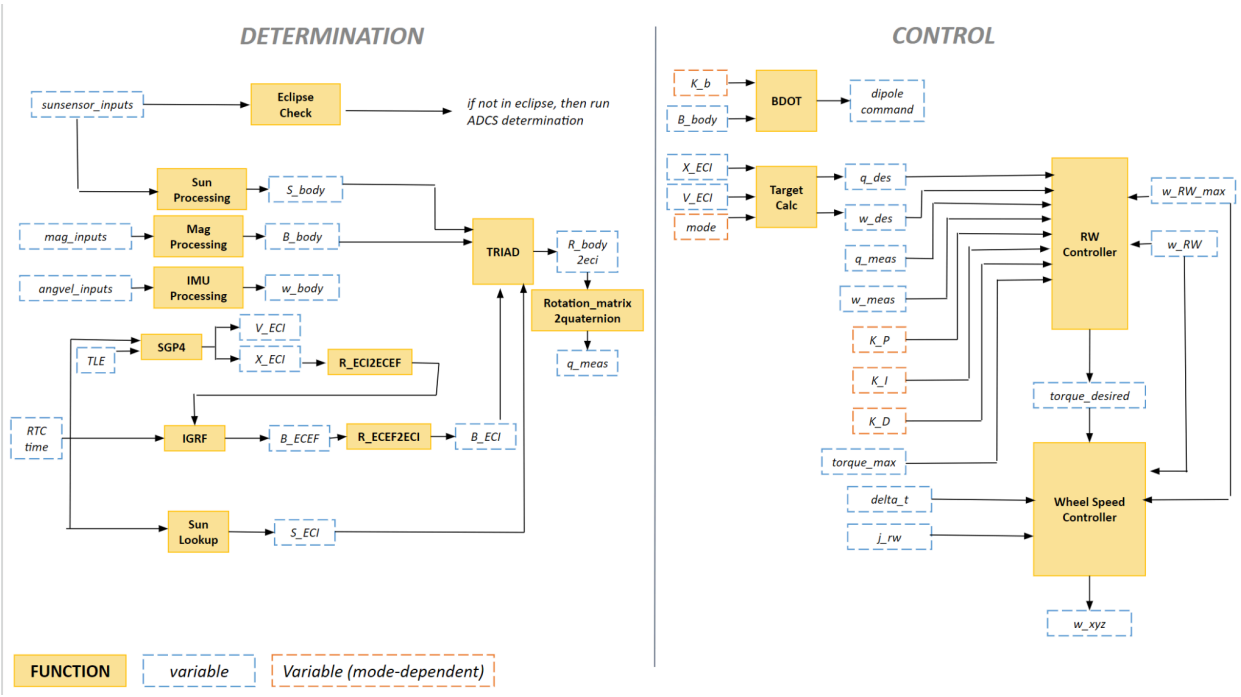

Figure 1: ADCS Software Algorithmic Block Diagram

3. Determination

Determination software uses sensor measurements and models of the sun position and Earth's magnetic field to estimate the satellite's attitude using the TRIAD algorithm. This attitude estimate can be used as feedback in the HDD control loop and as proof of mission success.

This section introduces each of the models and algorithms used for determination.

3.1 Frames of Reference

Before discussing how determination is done, it's useful to understand the reference frames that are used to define REALOP's coordinates and orientation.

Note that all nontrivial frame conversions are handled nicely by the Naval Observatory Vector Astrometry Software Library in C (NOVAS C). So we don't need to worry about the math behind the frames, just the meaning.

3.1.1 Earth-Centered-Inertial (ECI) Frame

The ECI Frame is centered at the origin of Earth and is an inertial frame of reference (non-rotating, non-accelerating frame), meaning the Earth is rotating relative to the ECI frame.

Coordinates in this frame are usually cartesian, in km from the origin. The x and y axes of the ECI Frame lie on the equatorial plane with the z axis pointing normal to this plane. The x axis points in the direction of the Vernal Equinox (a particular direction in the solar system). The y axis is constrained by the right-hand rule since the ECI Frame is a right-handed coordinate system.

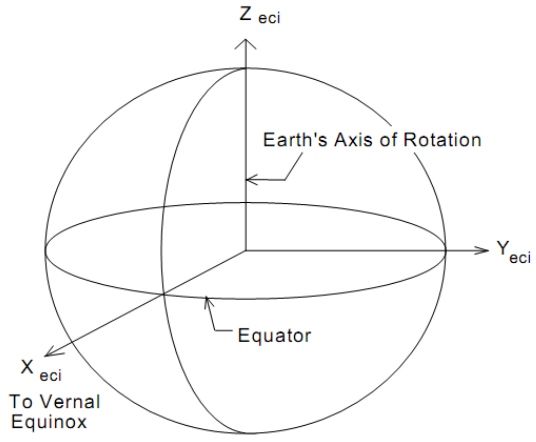

Figure 2: ECI Frame coordinate system

3.1.2 Earth-Centered-Earth-Fixed (ECEF) Frame

The ECEF Frame is centered at the origin of Earth and is a non-inertial, Earth-Fixed frame of reference (rotating, non-accelerating frame), meaning the Earth is non-rotating relative to the ECEF frame. In other words, the ECEF frame rotates with the Earth.

Coordinates in this frame are often Latitude-Longitude-Altitude (LLA) but they can also be cartesian based on the generic conversion from spherical to cartesian coordinates.

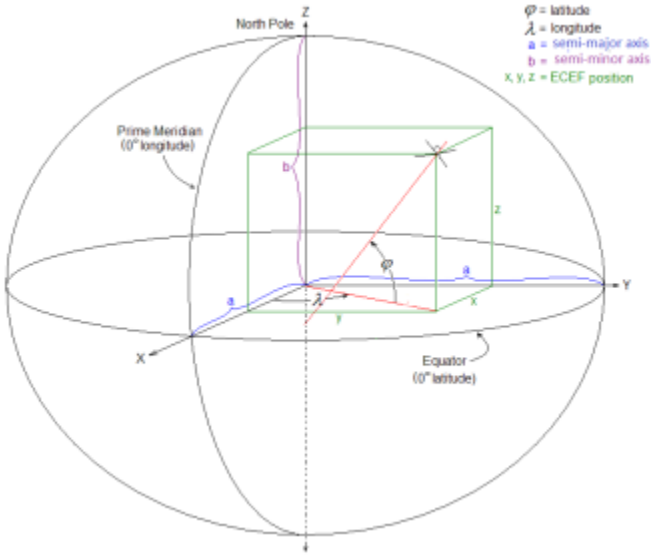

Figure 3: ECEF Frame coordinate system

3.1.3 ECI vs ECEF

See the [Wikipedia ECEF article] for an animation comparing the orbits of geosynchronous satellites in the ECI and ECEF frames.

In the ECI frame (left), geosynchronous orbits appear circular since the satellites are rotating with the Earth against the reference frame.

In the ECEF frame (right), geosynchronous orbits appear nearly stationary since the reference frame is rotating with both the Earth and the satellites.

3.1.4 Local-Vertical-Local-Horizontal (LVLH) Frame

The LVLH Frame is a coordinate frame attached to the satellite at each point in the orbit. This coordinate frame is non-inertial since it is always rotating about its z-axis such that the x-axis always points to the center of the Earth. The z-axis always points in the same direction normal to the orbital plane. Typical LVLH coordinates are cartesian North-East-Down (NED).

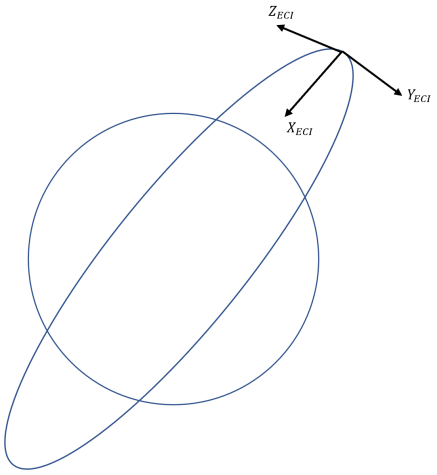

Figure 4: LVLH Frame coordinate system

3.1.5 Body Frame

The satellite Body Frame is the coordinate system made up of the principal axes of the CubeSat (usually the axes of symmetry). The Body Frame is a non-inertial frame since it rotates with the CubeSat. Coordinates in this frame are classic cartesian X-Y-Z.

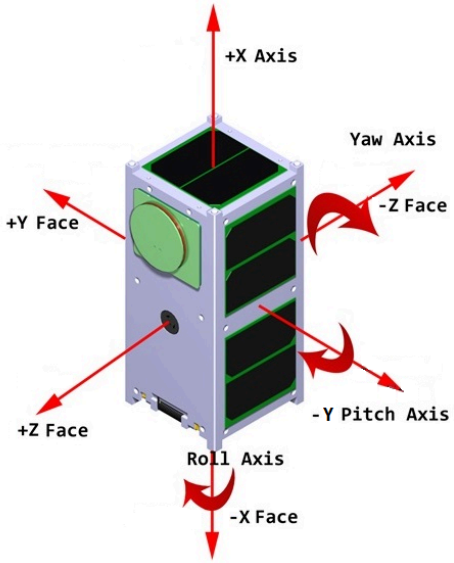

Figure 5: Satellite Body Frame coordinate system

3.2 TRIAD

TRIAD is the algorithm we use to determine the attitude of the satellite. It uses sensor measurements in the body frame and models of the Sun's position and Earth's magnetic field in the LVLH frame. By comparing the equivalent vectors in these two frames, TRIAD can solve for the orientation of the body frame relative to the LVLH frame.

See the Wikipedia page and the original 1964 paper by Harold Black for more information.

3.3 Magnetic Field Model (IGRF)

The Earth's magnetic field generally points North. However, the randomness in the motion of the Earth's core causes the magnetic field to bend and warp significantly depending on where you measure it on Earth. For example, mapmakers need to include the "magnetic declination" in their maps to make it possible to find true North using the magnetic North given by a compass. In the worst cases, a compass will point 30° East or West of true North! Additionally, the declination changes over time.

Luckily, we don't need to refer to maps to find the declination. The International Association of Geomagnetism and Aeronomy has maintained a mathematical model of the Earth's magnetic field called the International Geomagnetic Reference Field (IGRF) since 1965. risherlock's implementation of IGRF on GitHub will tell us the strength and direction of Earth's magnetic field at any position in orbit.

The current IGRF data, generation 13, will expire in 2025. Depending on REALOP's launch date and the release date of generation 14, we might need to use the World Magnetic Model (WMM) instead.

3.4 Sun Model (SPA)

The National Renewable Energy Laboratory (NREL) provides several models of the position of the Sun in the sky. We will use the Solar Position Algorithm (SPA) by Ibrahim Reda and Afshin Andreas. SPA gives us an estimate of the sun's position in the LVLH frame to 0.0003°. This model takes a struct containing the current time (in the format of several variables for year, month, day, etc.) and the latitude, longitude, and elevation of the satellite. It outputs the azimuth and elevation of the Sun, which convert freely to LVLH coordinates.

3.5 Position/Orbit Model (SGP4)

Simplified General Perturbations 4 (SGP4) is a mathematical model used to calculate the position and velocity of a satellite in orbit using the current time (from the RTC) and TLE (a description of the satellite's orbit, uplinked to REALOP after deployment). It outputs coordinates in the True-Equator-Mean-Equinox (TEME) frame that we convert to ECI, ECEF and then LLA. We are using aholinch's implementation on GitHub.

3.6 Sensor Processing

All sensor data can be processed to reduce noise and improve accuracy before being used in TRIAD or recorded. This includes:

- Offsets (derived from sensor calibration – for example, if the IMU measures an angular velocity of 1° / second when it's not spinning, then we apply an offset of -1° / second to all angular velocity measurements about that axis)
- Gains (derived from sensor calibration - for example, if the IMU measures an angular velocity of 1° / second when it's actually spinning at 5° / sec, then we multiply all angular velocity measurements about that axis by 5)
- Low pass filter (reduces noise in data)

The values for these calibrations will be determined by other ADCS subteams, but ADCS Software will implement the calibrations themselves.

Finally, the ADCS software should be able to ignore a sensor's input if we communicate to it, via uplink, that that sensor is malfunctioning. This might include deciding not to run determination at all if too many sensors are down.

3.7 Eclipse Check

If REALOP is in eclipse (blocked from the sun), then the sun sensors cannot measure the sun vector and attitude cannot be calculated. So if most of the sun sensors read a very small value, then REALOP knows it's in eclipse and won't calculate the attitude.

If determination were to run anyway, the attitude estimate would be erratic due to small fluctuations in the sun sensor readings.

4. Control

4.1 B-Dot Controller

When REALOP is first launched from the ISS, the satellite will spin out of control, or tumble, due to disturbance torques in space. These disturbances include aerodynamic drag from low earth orbit, solar radiation, gravity gradient, etc. All of these disturbances combined cause REALOP to rotate in undesired ways. Since our overall mission requires that REALOP points toward Earth, it needs a method for mitigating this rotation, or detumbling. Magnetorquer coils do just this! By running a current through the coils, REALOP generates a magnetic field that Interacts with the Earth's magnetic field to stop the satellite's rotation.

B-Dot is the algorithm used to control the magnetorquers such that the satellite eventually stabilizes and aligns with Earth's magnetic field.

Using the rotation rate of the surrounding magnetic field, B-Dot calculates a dipole command that describes how much current should run through each magnetorquer to generate a magnetic moment that will slow the satellite down. Also, a faster rotation rate results in a greater dipole command, and a slower rotation rate results in a smaller dipole command. Eventually, the satellite will stabilize such that it is not rotating about the x and y axes, and it is aligned with Earth's magnetic field. It can still be rotating about the z axis, but ideally it won't.

4.2 PID Wheel Controller

One of REALOP's experiments will spin the satellite using a feedback controller for the HDD. This is an important experiment because it applies directly to our next missions–if we were trying to point the camera at Earth, our first choice would be to use a feedback controller. Because of its simplicity and popularity, REALOP will use a Proportional-Integral-Derivative (PID) controller.

The MIT Aerospace Controls Lab has a phenomenal video on PID controllers: www.youtube.com/watch?v=4Y7zG48uHRo

While the ADCS HDD team is in charge of tuning the PID controller, the ADCS Software team is in charge of the PID controller itself, including the control math and the controller's communication with sensors and the HDD.

4.3 Ramp Wheel Controller

For some of REALOP's experiments, we want to test the performance of a non-feedback controller. This is not an unrealistic scenario because we know, theoretically, how the satellite will react to any given change in wheel speed. So, if we wanted to point at the Earth, we could generate a list of commands that should, in theory, turn the satellite towards the Earth. This paradigm is called feedforward control.

"Ramp" refers to the shape of the feedforward commands. Since we don't want to instantly command the HDD to maximum rotation, we slowly increase the commands we send until it reaches the maximum rotation, where it holds steady, and slowly decrease the commands on the way down. On a graph, we see a "ramp" shape.

4.4 Target Calculation

In the LVLH frame, the Earth is always in a fixed direction: down. Since TRIAD gives the satellite's orientation in the LVLH frame, it is trivial to solve for the direction and amount we need to rotate the satellite to point towards the Earth.

5. Testing and Validation

ADCS Software must guarantee that our software will not fail in orbit. This is a difficult task given the abundance of possible errors:

- Programming errors
- Logic errors
- Mission design errors and oversight
- Errors in external libraries
- Errors integrating with FSW and other ADCS subteams
- Incorrect assumptions
- Version control mistakes
- Hardware integration and errata
- Et cetera

To combat these errors, ADCS Software must test the software early and often. Some simple but effective methods include:

- Simulations
- Experiments
- Direct comparisons with other software
- Risk analysis

Most of the previous work on ADCS Software testing and validation is summarized in the 2024 Undergraduate Research Conference (URC) poster.
Current testing and validation efforts are listed in the ADCS Software Validation document.
These are located in the google drive under REALOP/ADCS/Software/URC 2024.

6. Integration

6.1 Virtual Intellisat

ADCS Software relies on FSW to be able to interact with REALOP's hardware. But, how does ADCS Software actually ask FSW's code to perform a given function?

A simple approach is to let ADCS Software call the hardware driver functions directly. This has several issues:

- ADCS needs to wait for FSW to implement a driver before including it in the ADCS code. Otherwise, we wouldn't know what to write in the code when we want to spin the HDD, read a sensor value, etc.
- ADCS can easily use the drivers incorrectly.
- The ADCS code needs to include headers from the FSW code to know how to call the drivers, but the FSW code needs to include headers from the ADCS code to know how to ask ADCS to perform experiments. Unless ADCS and FSW were developing their code in the same repository, which has its own issues, this is impossible or extremely difficult to manage.

The current approach is to let ADCS Software define a header called "Virtual Intellisat," named after the FSW project, *Intellisat*. In this header, ADCS Software can declare functions for any functionality it wants, like commanding the HDD, getting a sensor value, and storing data, in any format it wants. This has several advantages:

- ADCS can write these functions into a final version of the code without delay.
- FSW handles the driver code and can implement an extra layer of safety.
- The ADCS code can live entirely inside a subdirectory of the FSW code.

6.2 Building and Deploying the ADCS Software

For testing that does not require hardware, use the Makefile in the top level of the repository by calling "make" on the command line. This will generate an ADCS shared object library. In order to make an executable for testing purposes, create a "main.c" file in the repository with a main function and call "make executable." This kind of local build is useful for testing algorithms for correctness, but cannot test anything that requires hardware or a Virtual Intellisat function.

For building and deploying to the Orbital Platform (the flight computer), which requires building with the FSW software, use STM32CubeIDE. The FSW project is already configured to compile and deploy, so the only extra step is to copy in the ADCS shared object library and make sure the FSW project knows it's there. See our instructions for deploying.

7. Mission Control

ADCS Software is responsible for communicating with FSW about what data it needs from the ground and what data should get sent back to Earth. These decisions require knowledge of the entire mission, including what experiments will be run and how mission success is determined based on the data. Since no other team is explicitly responsible for this knowledge, ADCS Software is implicitly responsible. This section outlines our current plan for each part of the mission.

All of our initial knowledge is from Adam Zufall, our Graduate Advisor.

7.1 Uplink/Downlink

7.2 Experiment Design

7.3 Success Criteria

What we need from the ground, what we can uplink optionally, experiment design, data analysis and success criteria

8. Additional Resources

URC 2020: ADCS Software Overview

Pro Git E-Book