KiboCUBE Academy

Lecture 25

Introduction to CubeSat On-board Software and Simulation Environment

The University of Tokyo

Department of Aeronautics and Astronautics

Associate Professor Dr. Satoshi Ikari

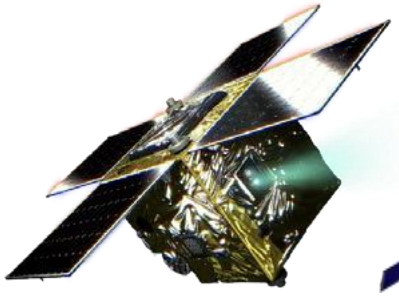
This lecture is NOT specifically about KiboCUBE and covers GENERAL engineering topics of space development and utilization for CubeSats.
The specific information and requirements for applying to KiboCUBE can be found at:
https://www.unoosa.org/oosa/en/ourwork/psa/hsti/kibocube.html

JAXA

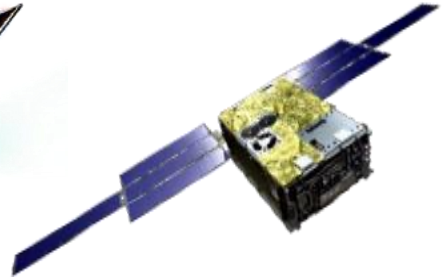
UNISEC University Space Engineering Consortium JAPAN

United Nations Office for Outer Space Affairs ★ Japan Aerospace Exploration Agency
Kibo CUBE

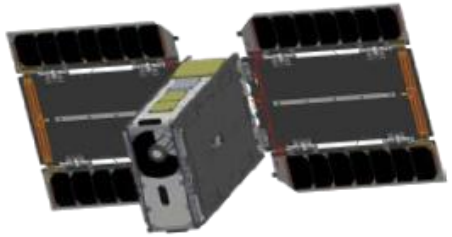
Lecturer Introduction



PROCYON(2014)



EQUULEUS(2022)



Sphere-EYE 1(2023)

AOCS Module

Satoshi Ikari, Ph.D.

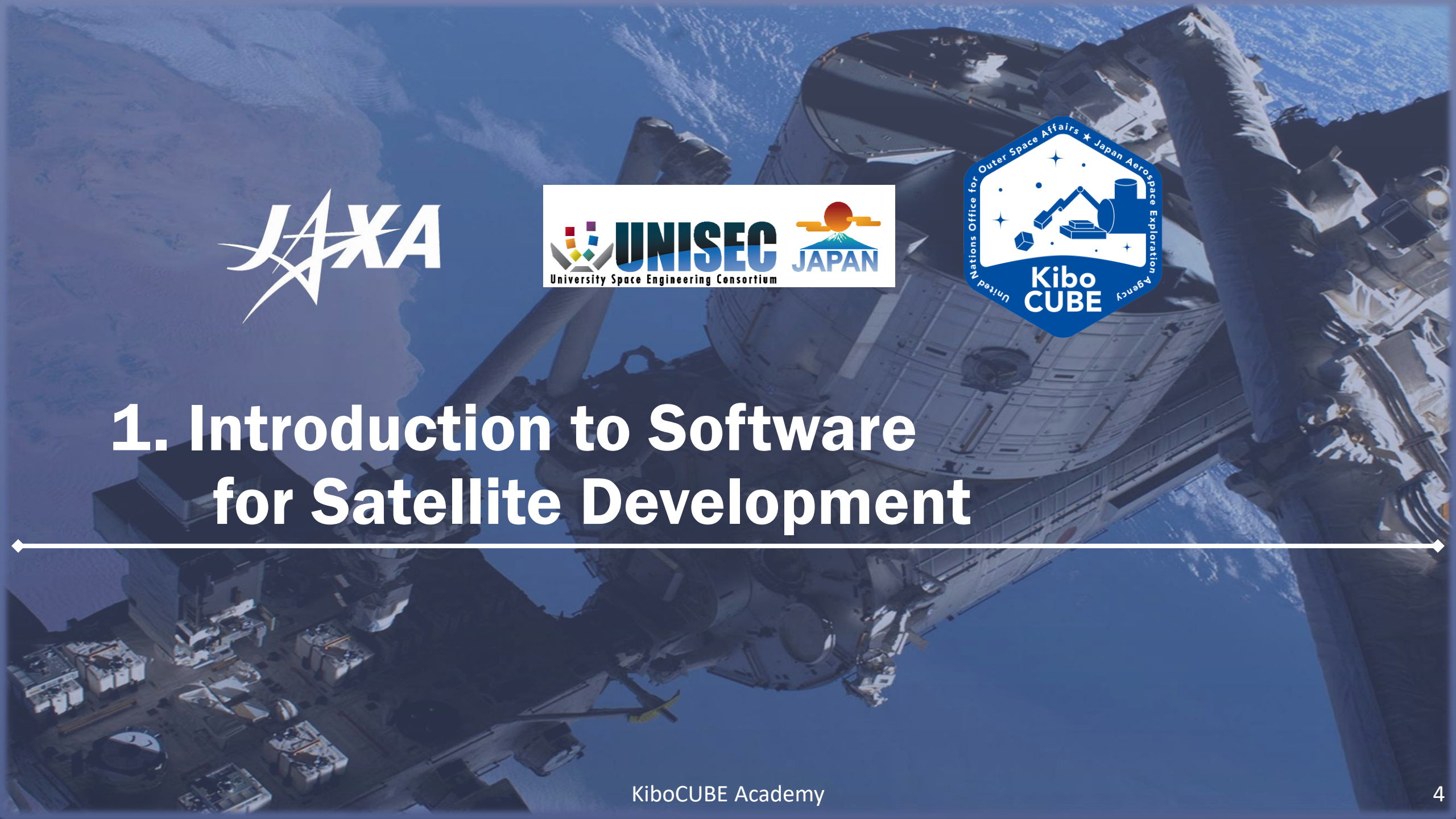
Position:

2017 – 2023 Assistant Professor, Department of Aeronautics and Astronautics, The University of Tokyo
2022 – 2024 Guest Researcher, German Space Operation Center, DLR
2024 – Associate Professor, Department of Aeronautics and Astronautics, The University of Tokyo

Research Topics:

Micro/nano-satellites, Astrodynamics, Formation Flying, GNSS, Attitude Determination and Control, and Numerical Simulation

Contents



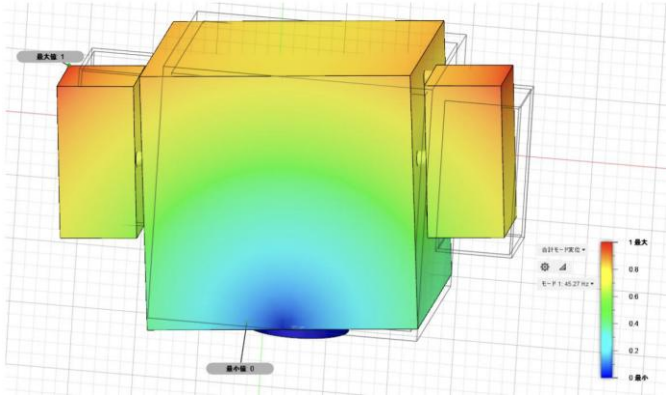
1. Introduction to Software for Satellite Development

1. Introduction to Software for Satellite Development

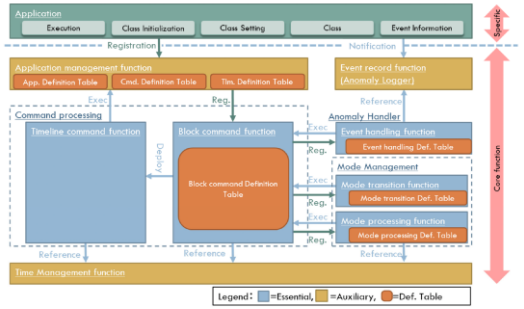
1.1. Software used in Satellite Development

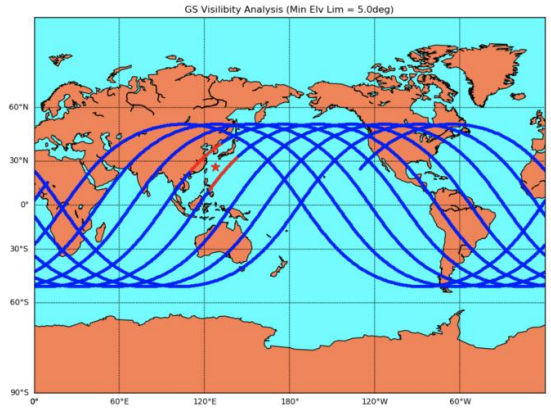
- Software plays a crucial role in space development activities.
- Software used in satellite development:
  - CAD, Structure and Thermal Analysis
  - Electrical Power Analysis
  - Astrodynamics Simulators
  - Flight Software
  - Ground Station Software
  - Telemetry Data Analysis
  - Mission Data Analysis

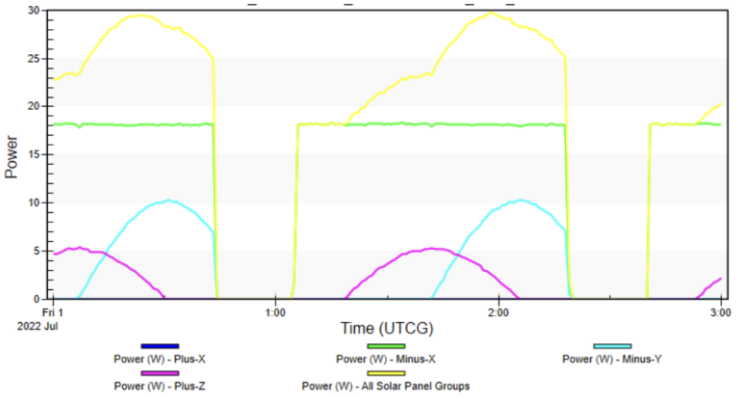
In Mission analysis
and Satellite Design phase



In Development, Test,
and On-orbit Operation phase







1. Introduction to Software for Satellite Development

1.2. Importance of Software in Satellite Development

- Software plays a crucial role in space development activities.
- In particular, **Flight Software** is most important for the success of satellite projects.
  - "*Software is responsible for **3 to 33 % of failures**, with most values close to **10%**.*" [1]
- Examples of software failure in space projects
  - Viking-1 (1975, 5 billion Y2024 USD): "*Due to an error in the reprogramming of the software controlling the battery charging cycle, the configuration of the antenna alignment was overwritten. The mission was ended by losing the communication.*" [1]
  - ASTRO-H(2016, 500 million Y2024 USD): "*A Software error had spun the satellite so rapidly that parts, including the solar panels, tore off.*" [1]
- Many micro/nano-satellites also have software failures leading to mission failures.

[1] Christian R. Prause, and et al., "Fatal Software Failures in Spaceflight", Encyclopedia, 4, 936-965, 2024.

1. Introduction to Software for Satellite Development

1.2. Importance of Software in Satellite Development

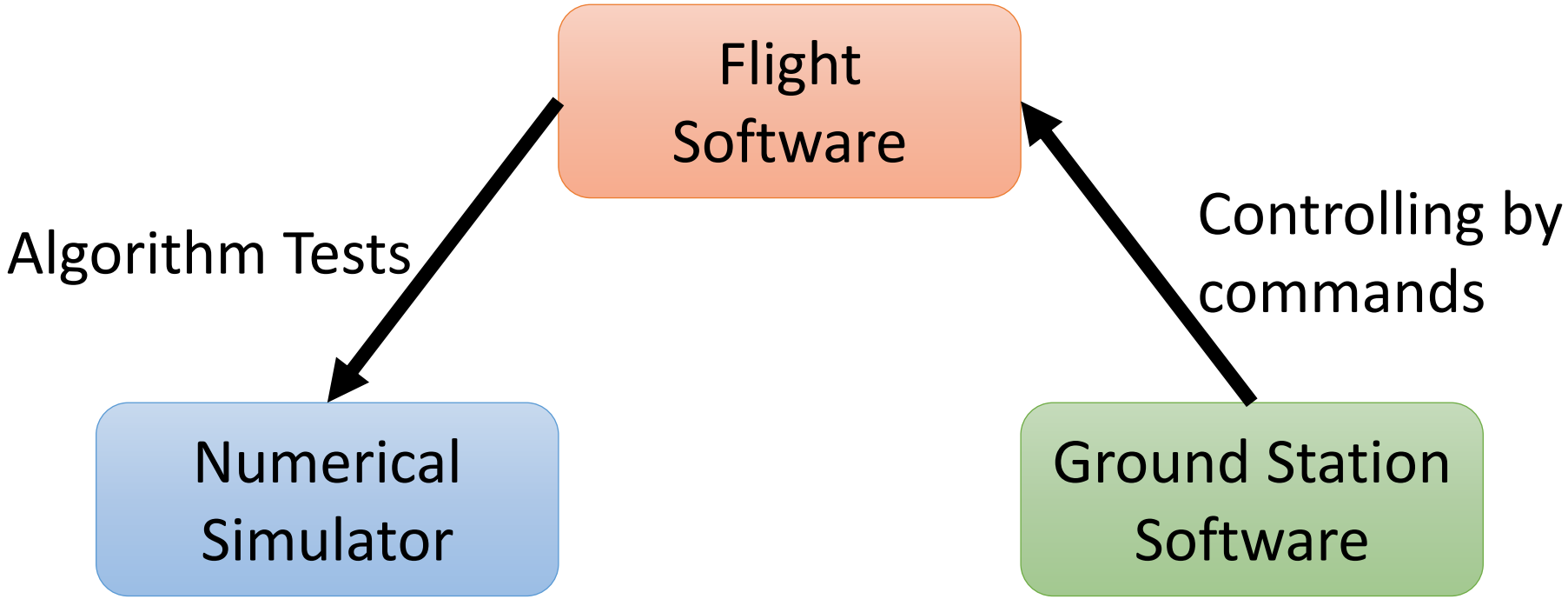
- Software testing is essential, but often gets neglected in spacecraft development.
  - *"Spaceflight technology is historically and also traditionally located **in the area of mechanical engineering**. This means that **software engineering is seen as a small auxiliary discipline**."* [1]

- Satellites are largely **non-repairable**. Software is typically the only element that is **repairable**.
  - Upload tuned parameters, debugged software, and new applications to change or fixed the satellite behaviour.

[1] Christian R. Prause, and et al., "Fatal Software Failures in Spaceflight", Encyclopedia, 4, 936-965, 2024.

1.2. Importance of Software in Satellite Development

- To develop a reliable flight software, we need reliable numerical simulator for testing the flight software.
- To correctly operate the flight software, we need a reliable ground operation software.



1.3. Scope of the Lecture

- This lecture focuses on Flight Software, Numerical Simulator, and Ground Station Software.
  - We will learn the purpose and the overview of these software.
- This lecture also covers Software Management.
  - We will learn how to develop reliable software with a team.
- Finally, we examine an Example of Software Suite for Satellite Research and Development.
  - We will learn about Open-Source Software developed by the University of Tokyo.

2. Flight Software

2. Flight Software

2.1. Purpose of Flight Software

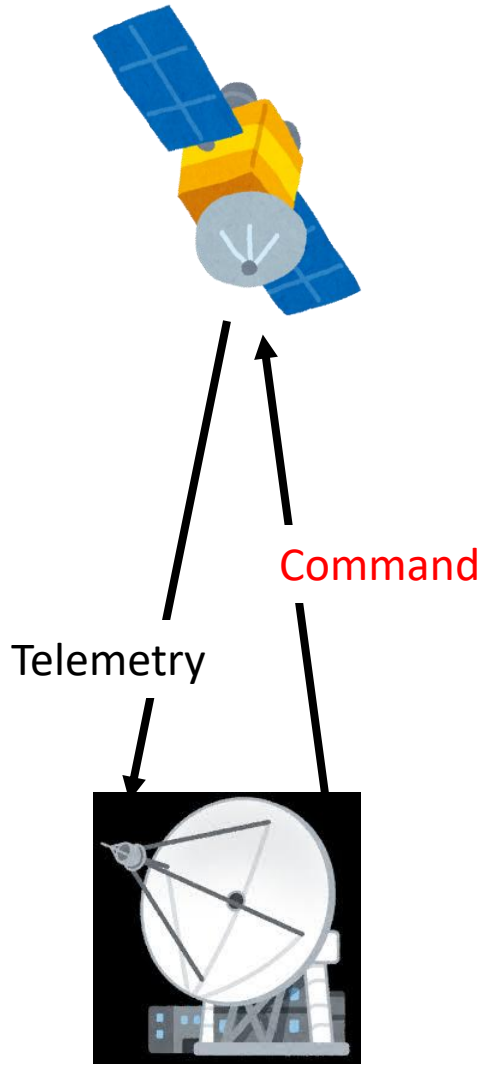
- Flight software is responsible to **achieve the required functions** of the satellite with the **available computer resources**.
- Typical roles of flight software:
  - Telemetry generation and command execution for communication with ground.
  - Storing telemetry and mission data into memory.
  - Storing scheduled command.
  - Periodical processing of implemented applications.
  - Execution of autonomous applications to maintain a healthy satellite bus system.
  - Execution of autonomous applications for mission payload operation.
- Constraints of flight software
  - Limited computer power and memory.
  - Need to consider radiation effect (Reset, Reboot of CPUs)

	ONGLAISAT MOBC (Fig. 2)	ONGLAISAT AOBC	ONGLAISAT TOBC (Fig. 2)	MAGNARO OBC[6)]
CPU	Renesas Electronics SH-2A	Microchip Technology PIC32MX7	Microchip Technology PIC32MX7	STMicroelectronics STM32F4
Clock	200 MHz	80 MHz	30 MHz	90 MHz (2U) / 45 MHz (1U)
ROM	2.5 MiB internal ROM	512 KiB internal ROM	512 KiB internal ROM	2 MiB internal ROM
RAM	128 KiB internal RAM, 8 MiB external SRAM	128 KiB internal RAM	128 KiB internal RAM	384 KiB internal RAM, 500 KiB external SRAM
NVRAM	2 MiB MRAM	512 KiB FRAM	–	524 KiB MRAM, 131 KiB EEP-ROM
Storage	2 GiB NAND flash memory	–	–	16 GB SD card
Interface	UART (RS422, LVTTL), CCSDS (LVTTL), GPIO (LVTTL), ADC	UART (RS422, RS485, LVTTL), SPI, I2C, GPIO (LVTTL), ADC	UART (LVTTL), I2C, GPIO (LVTTL), ADC	UART, SPI, I2C, GPIO, ADC, DCMI

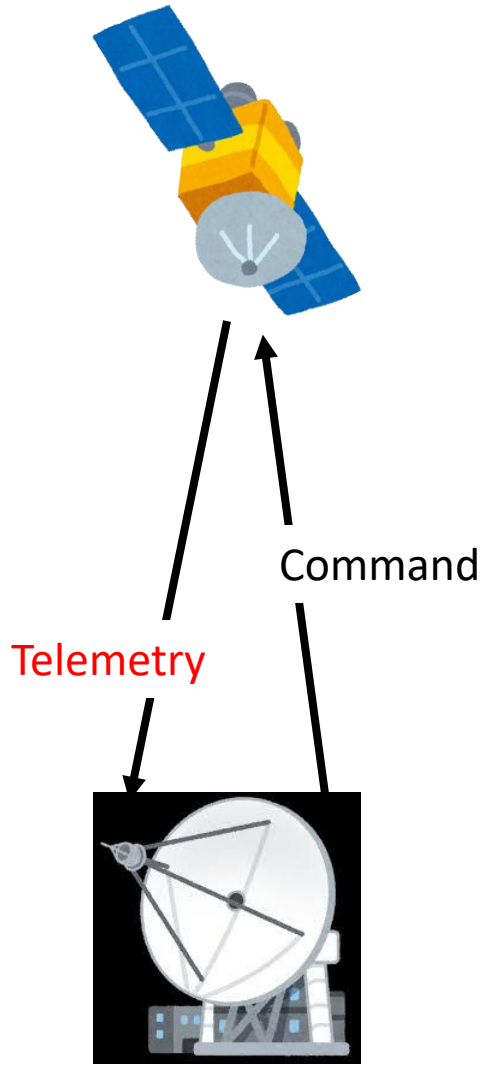
Example of computer resource for CubeSats[2]

[2] R. Suzumoto, and et al., “Improvement of C2A (Command-Centric Architecture) Reusability for Multiple Types of OBCs and Development of Continuous Integration Environment for Reliability of Flight Software”, Journal of Evolving Space Activities, 2023.

- Command
  - Command is data sent from a ground station to a satellite to control the satellite behaviour.
  - Types of commands
    - Realtime command: A command executed soon after the satellite receives it.
    - Scheduled command: A command to be executed at the specified time
- Scheduling execution timing of command is an essential feature of the flight software.
- Safety and Security are important for command handling
  - Ignore commands for other satellites
  - Ignore commands from unexpected operators
  - Ignore commands with invalid parameters



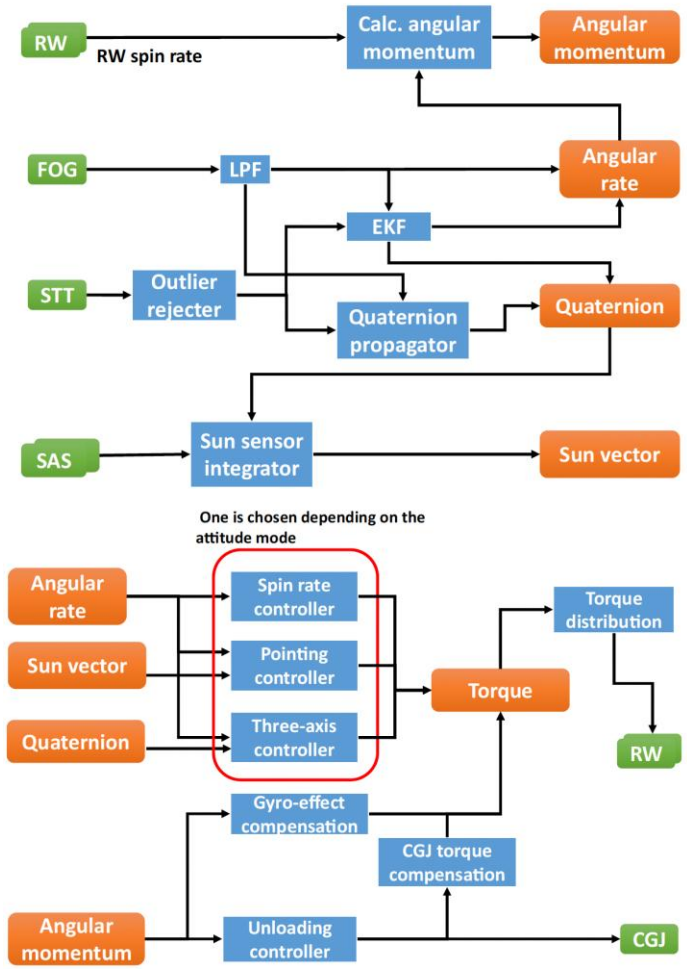
- Telemetry
  - Telemetry is data sent from a satellite to ground stations.
  - Types of telemetry
    - Housekeeping data (HK)
      - Basic status information of satellite components to monitor the health of the satellite. HK is usually periodically and continuously sent to ground stations during contact.
    - Stored data
      - Information stored in the satellite memory when the satellite does not communicate with a ground station.
    - Mission data
      - Data acquired from the mission instruments, such as telescopes.
- Scheduling downlink timing of telemetry data is an essential feature of the flight software.
- Encryption is required if you want to keep your data secret.



- Sensor Data Handling
  - Set sensor configurations
  - Get sensor observation data
- Filtering and Attitude Determination
  - Remove noise on observation data
  - Estimate needed states (e.g. Kalman Filter)
- Target Attitude Calculation
  - Calculate target attitude
- Attitude Control
  - Calculate control torque to stabilize attitude (e.g. PID control)
- Actuator Control
  - Set actuator configurations
  - Get actuator information

Same with other components.
Current sensor, thermometers, and etc.

Same with other components.
RF devices, switches, and etc.



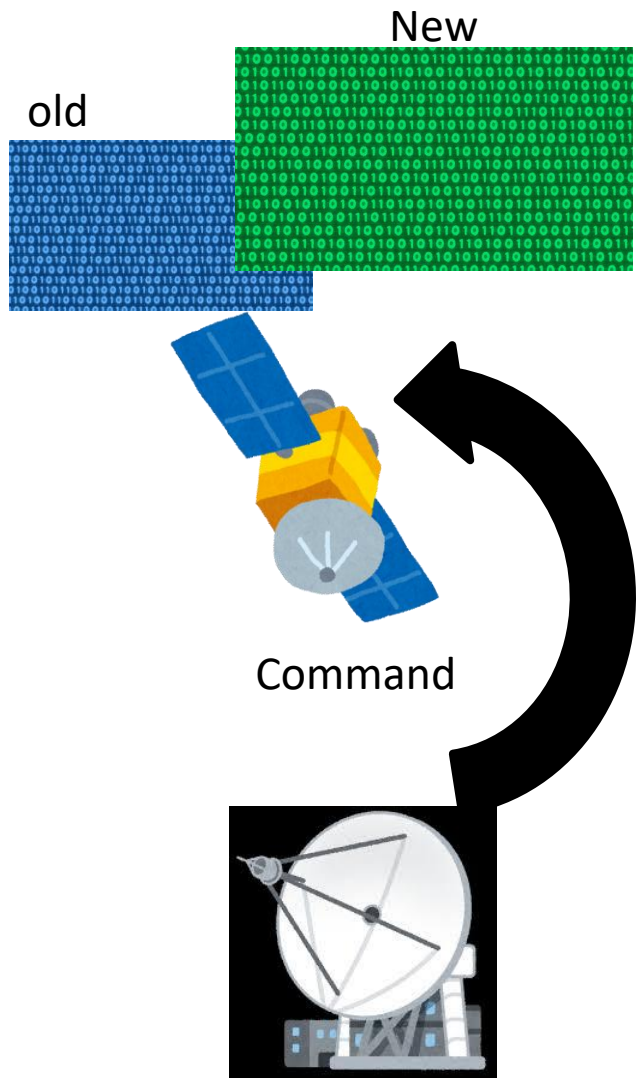
Block Diagram of Attitude Control[3]

[3]S. Ikari, and et al., “Attitude Determination and Control System for the PROCYON Micro-Spacecraft”, Trans. JSASS, 2017

2. Flight Software

2.4. Important Features for Satellites: e.g., On-orbit reprogramming

- Satellites are largely **non-repairable**. Software is typically the only element that is **repairable**.
- To keep the repairability of software, an on-orbit reprogramming feature is essential.
- Types of reprogramming
  - Full reprogramming: Rewrite all program memory
  - Partial reprogramming: Rewrite partial regions of program memory
- Risk of reprogramming
  - Takes an amount of time for uplink
  - Data loss during uplink
  - Failure due to bugs in the new software
  - Critical areas need to be maintained and not rewritten.



2.5. Limitation from Hardware

- Limited computer power and memory
- Need to consider the radiation effect
  - Reset and Reboot of CPUs is essential
  - Watchdog timers or mutual monitoring to detect hang-up of CPUs
- Programming language is also limited
  - Traditionally, C, C++ are used.
  - Recently, Python, Rust are started to use.

	ONGLAISAT MOBC (Fig. 2)	ONGLAISAT AOBC	ONGLAISAT TOBC (Fig. 2)	MAGNARO OBC[6)]
CPU	Renesas Electronics SH-2A	Microchip Technology PIC32MX7	Microchip Technology PIC32MX7	STMicroelectronics STM32F4
Clock	200 MHz	80 MHz	30 MHz	90 MHz (2U) / 45 MHz (1U)
ROM	2.5 MiB internal ROM	512 KiB internal ROM	512 KiB internal ROM	2 MiB internal ROM
RAM	128 KiB internal RAM, 8 MiB external SRAM	128 KiB internal RAM	128 KiB internal RAM	384 KiB internal RAM, 500 KiB external SRAM
NVRAM	2 MiB MRAM	512 KiB FRAM	–	524 KiB MRAM, 131 KiB EEP-ROM
Storage	2 GiB NAND flash memory	–	–	16 GB SD card
Interface	UART (RS422, LVTTL), CCSDS (LVTTL), GPIO (LVTTL), ADC	UART (RS422, RS485, LVTTL), SPI, I2C, GPIO (LVTTL), ADC	UART (LVTTL), I2C, GPIO (LVTTL), ADC	UART, SPI, I2C, GPIO, ADC, DCMI

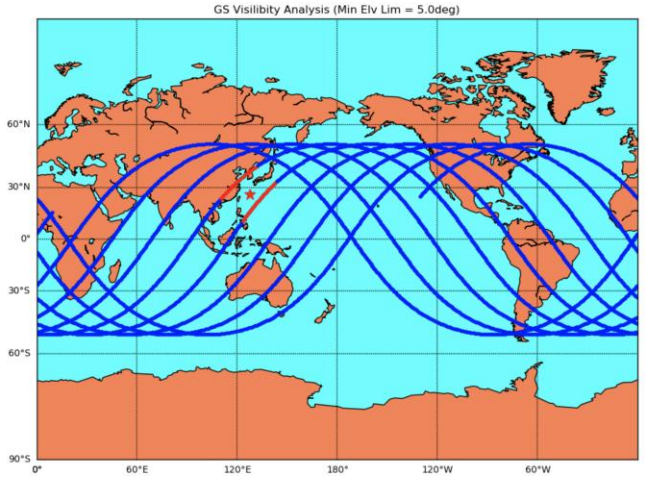
Example of computer resource for CubeSats[2]

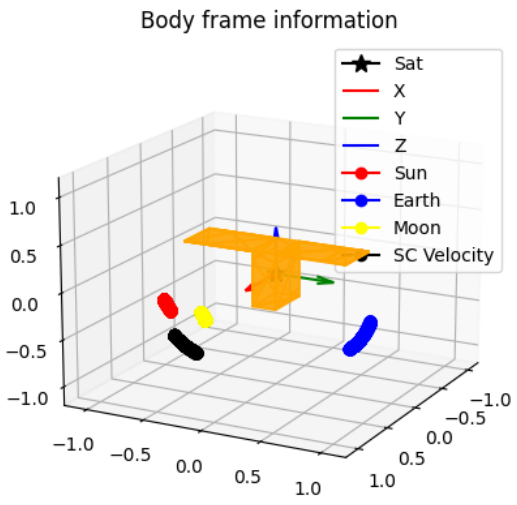
3. Numerical Simulator

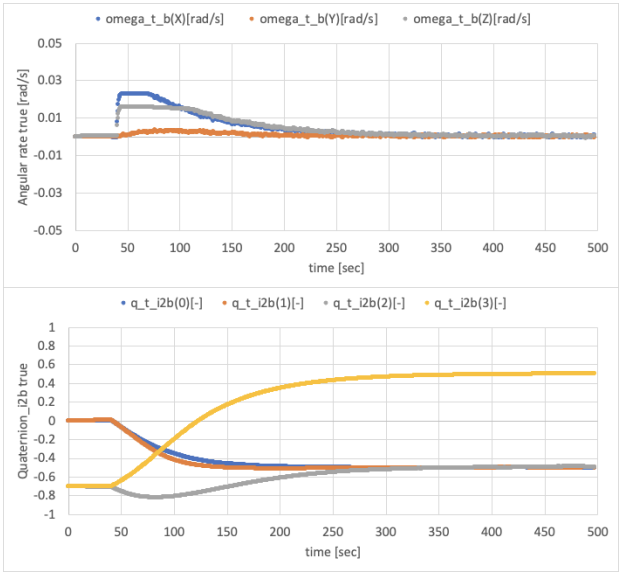
3. Numerical Simulator

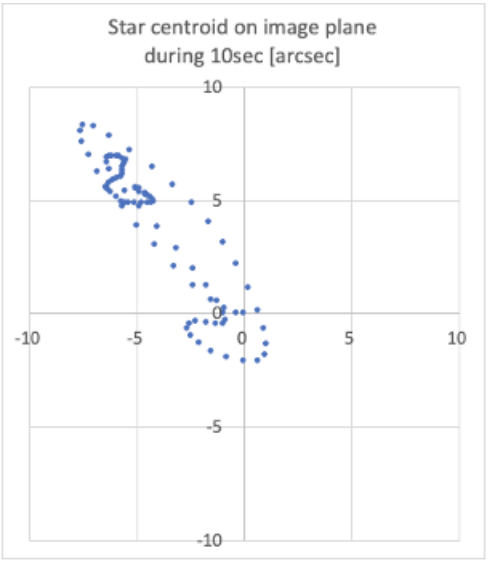
3.1. Purpose of a Numerical Simulator

- Mission design and Analysis
  - Orbit and Attitude calculation
  - Power balance analysis
  - Thermal analysis
  - Communication Visibility Analysis
  - Lifetime and Deorbit time Analysis
- Flight Software Verification
  - On-board autonomous functions
  - Attitude Determination and Control Algorithm
  - Orbit Control Algorithm
- Operation Training
- Operation Command Validation





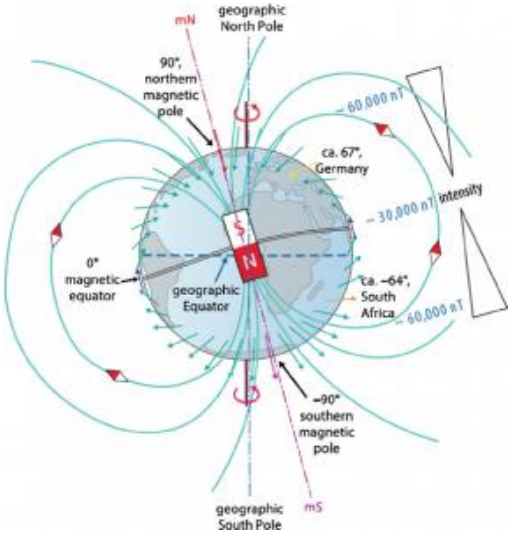




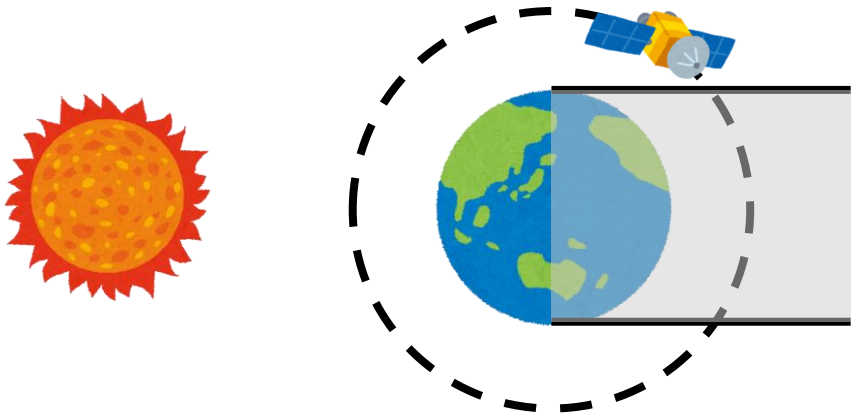
3. Numerical Simulator

3.2. Space Environment Simulation

- Understanding the space environment is essential for:
  - Thermal / Power Balance / Communication Visibility Analysis
  - Calculation of Orbital Disturbances
  - Calculation of Sensor Observation data
- Position of planets / Rotation of planets
- Position of stars
- Geomagnetic Field
- Atmosphere
- Eclipse Calculation
- Artificial Infrastructure
  - Position of GNSS satellites
  - Position of Data Relay satellites



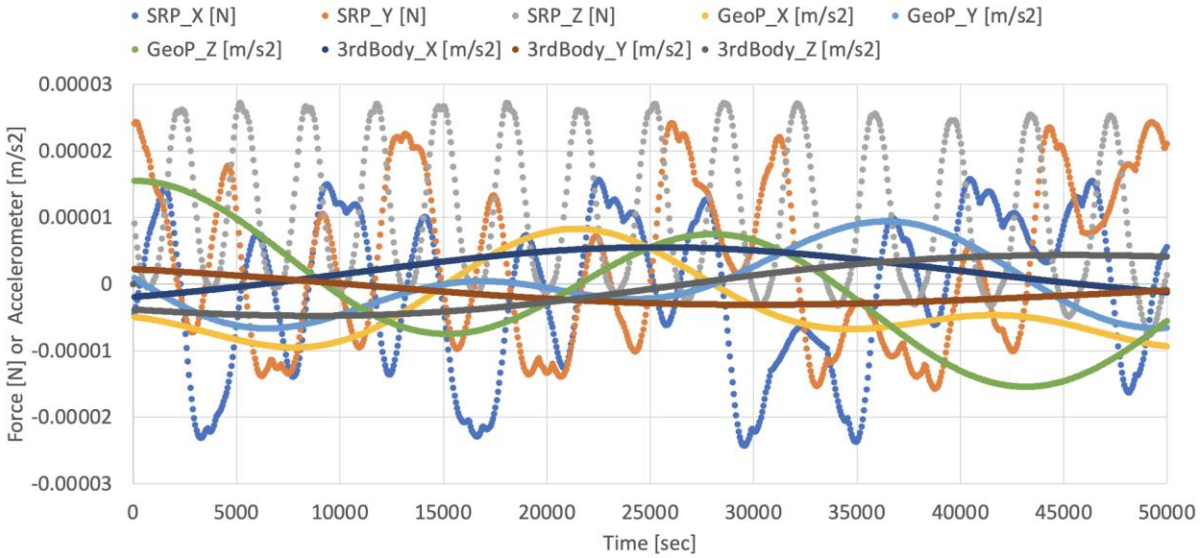
Ref: https://physicsopenlab.org/2019/06/10/geomagnetic-field/



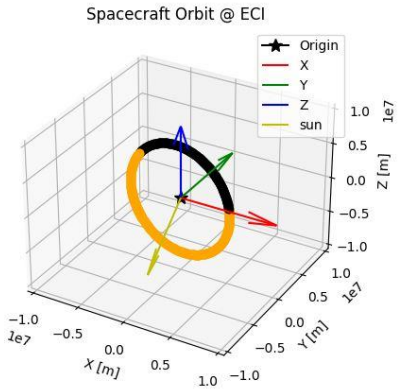
3. Numerical Simulator

3.3. Satellite Dynamics Simulation

- Satellite dynamics simulations are essential for satellite position and attitude calculation
- Disturbances
  - High-order geopotential
  - Third body gravity
  - Magnetic Disturbances
  - Solar Radiation Pressure
  - Air Drag
- Satellite Orbit calculation / propagation
  - Analytical calculation: Kepler, SGP4
  - Numerical Integration: Runge-Kutta method
- Satellite Attitude propagation
  - Numerical Integration: Runge-Kutta method



Example of calculated Disturbances

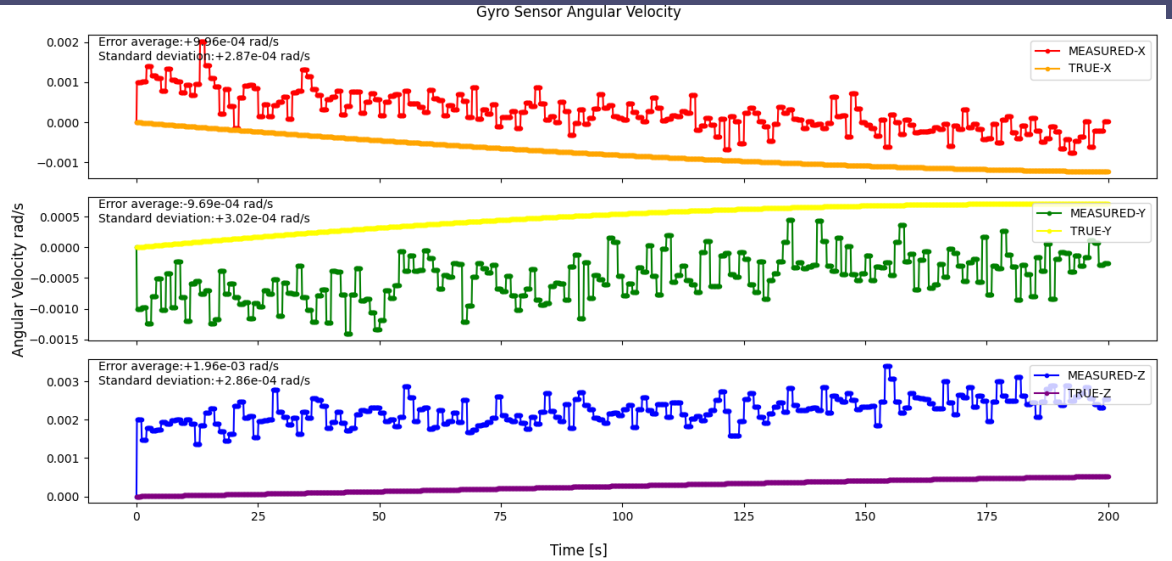


Orbit motion in ECI frame

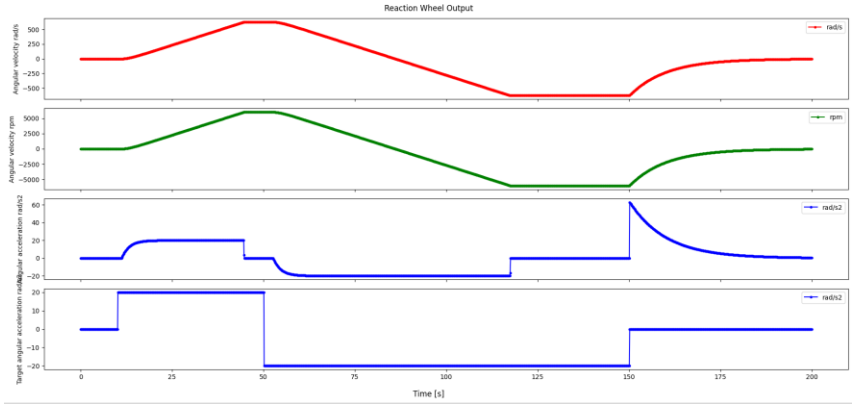
3. Numerical Simulator

3.4. Satellite Components Emulation

- Satellite components emulation is essential for
  - Realistic test for satellite attitude determination and control algorithm
- Electrical Power
  - Switch State
  - Power Consumption
- Thermal
  - Heat generation
  - Thermal Conductance
- Communication
  - Telemetry and Command of the Component
- Others
  - Noise on sensing data
  - Noise on actuation torque
  - Mounting position and coordinate
  - Failure emulation



Example of gyro sensor output



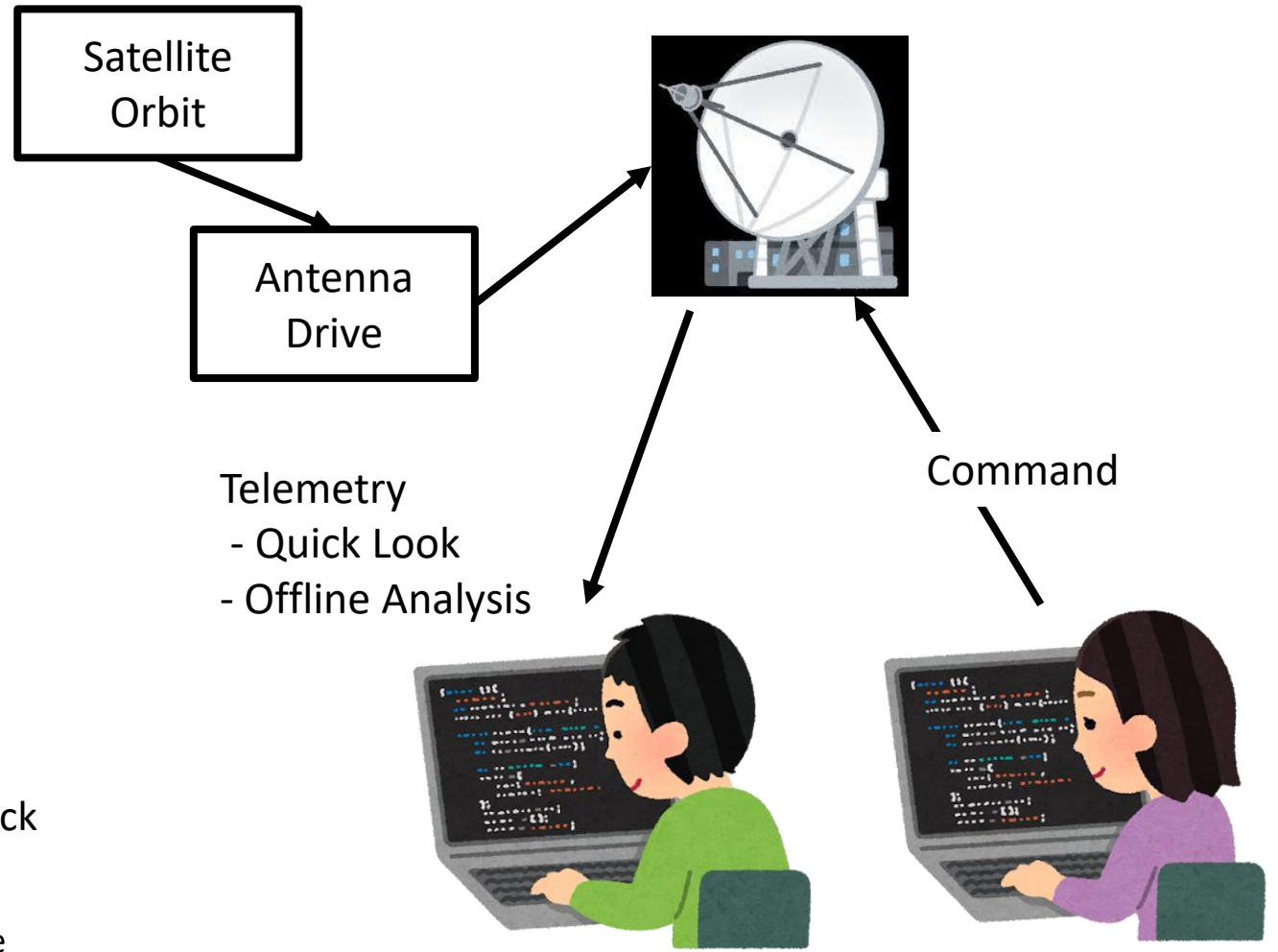
Example of Reaction Wheel

4. Ground Operation Software

4. Ground Operation Software

4.1. Purpose of Ground Operation Software

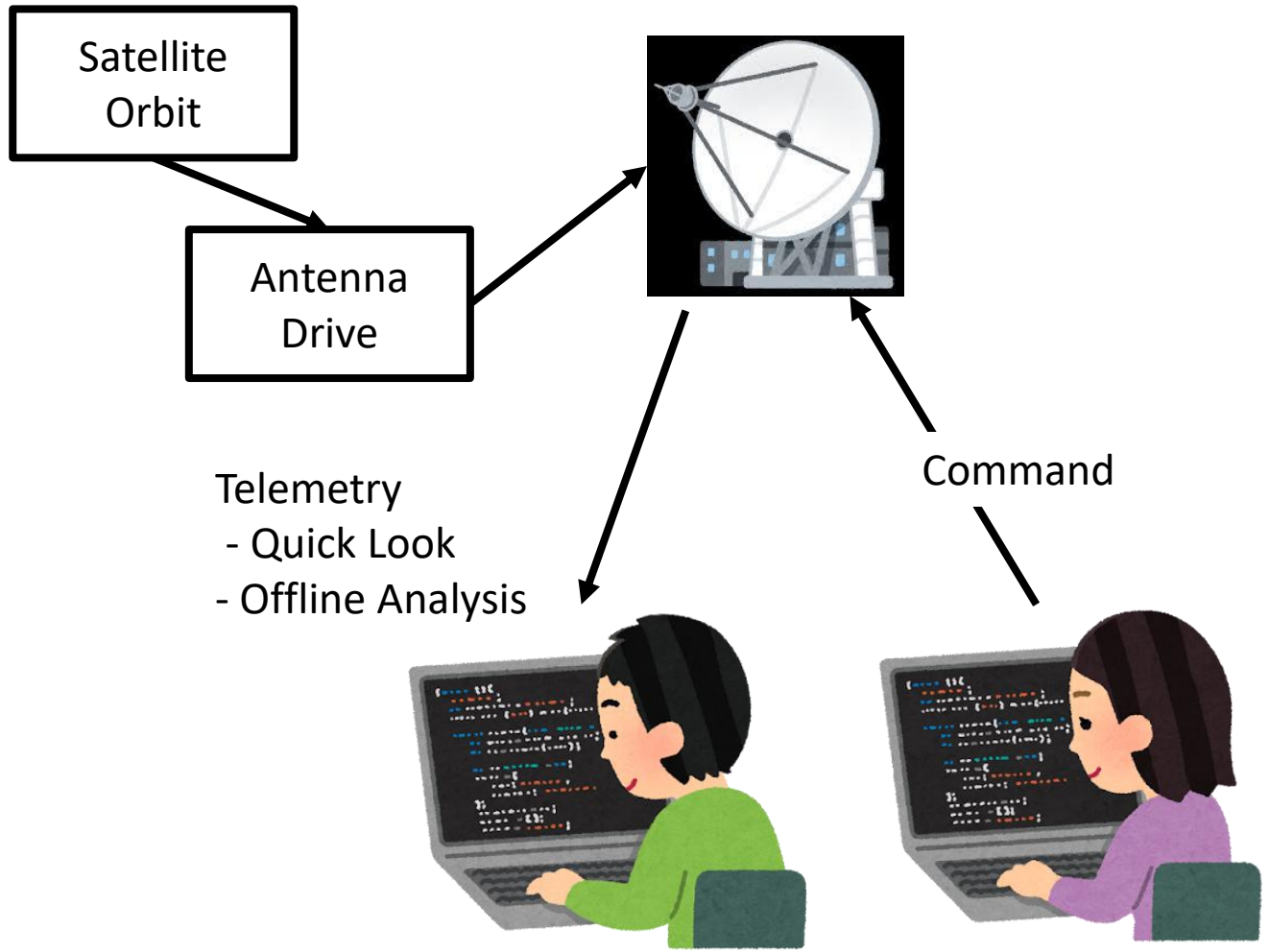
- Operation Planning
  - Determine AOS and LOS time with satellite orbit information.
  - Orbit Determination for future operation planning.
- Ground Station Antenna Direction Control
  - Elevation/Azimuth calculation from satellite orbit.
  - Generate signal to control the antenna direction
- Telemetry Quick Look with GUI
  - Display real-time telemetry in good GUI to quickly check the satellite status.
- Command Sending
  - Send command to a satellite.
- Data Storage
  - Store telemetry and command data in a database
- Telemetry Analysis
  - Offline analysis of telemetry to get mission data or to check health monitoring of the satellite.



*AOS: Acquisition of Signal, LOS: Loss of Signal, GUI: Graphical User Interface

4.2. On-ground Antenna Driving Software

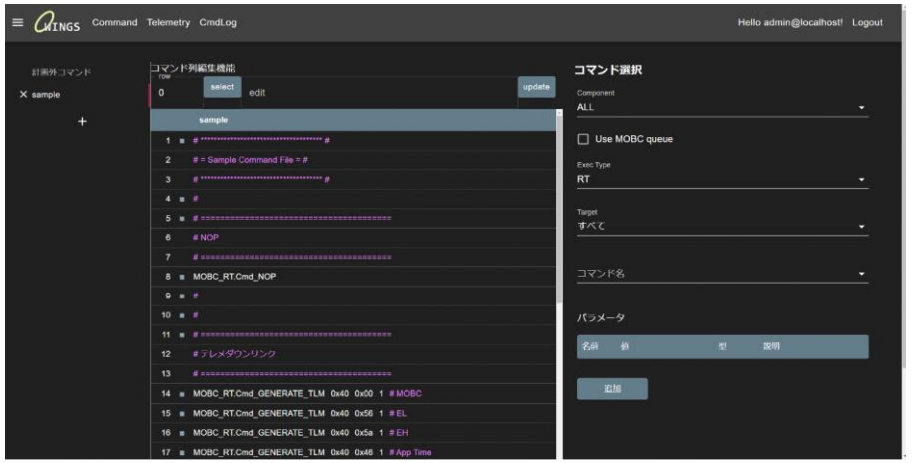
- Purpose
  - Control the antenna direction (Elevation/Azimuth) to point the satellite direction.
- Sequence
  - Get satellite orbit information
    - e.g. TLE from space-track
  - Calculate satellite position in ECEF frame
  - Calculate antenna pointing target direction (Elevation/Azimuth)
  - Control motors on ground station antenna



4. Ground Operation Software

4.3. Front End: Command and Telemetry Handling GUI

- The GUI (Graphical User Interface) is very important to improve operational efficiency and reduce the frequency of operational errors.
- Command Handling GUI
  - List of sending command
  - File input for scheduled command list
  - Command selection for emergency operation
- Telemetry Quick Look GUI
  - List of received telemetry data and its value
  - Time history graph, state indication with colour, and other graphs are effective to quickly understand the satellite status



Example of command GUI

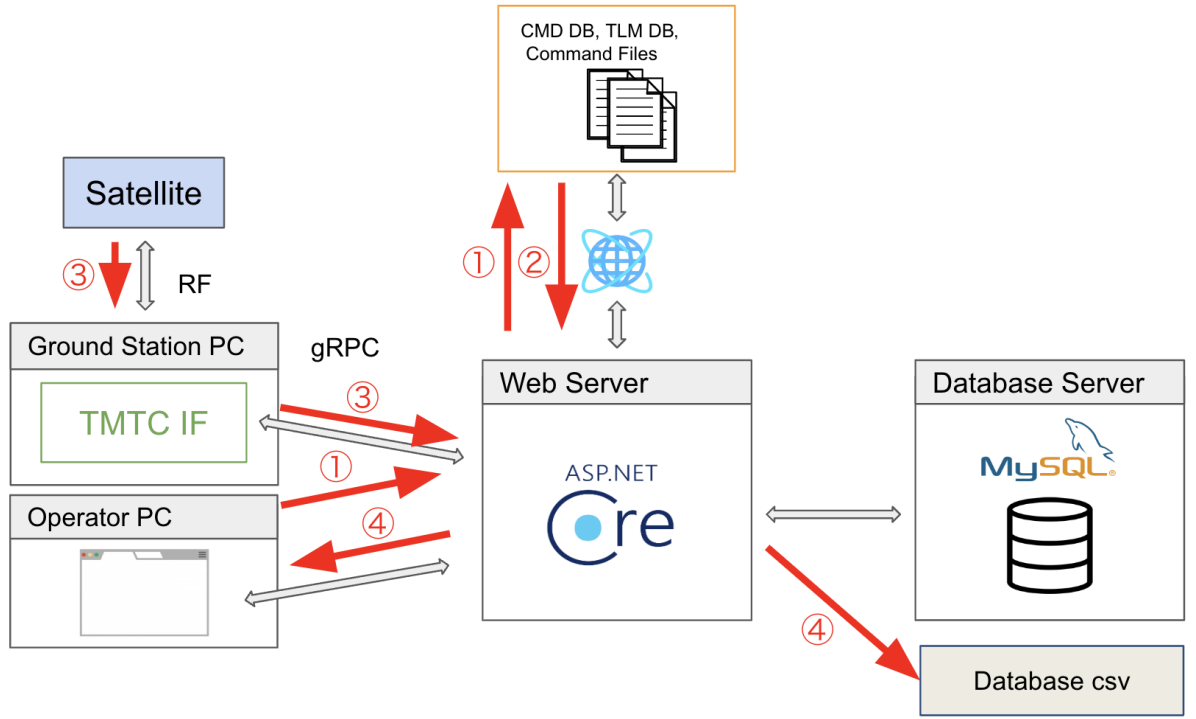
Example of command GUI

4. Ground Operation Software

4.3. Front End: Command and Telemetry Handling GUI

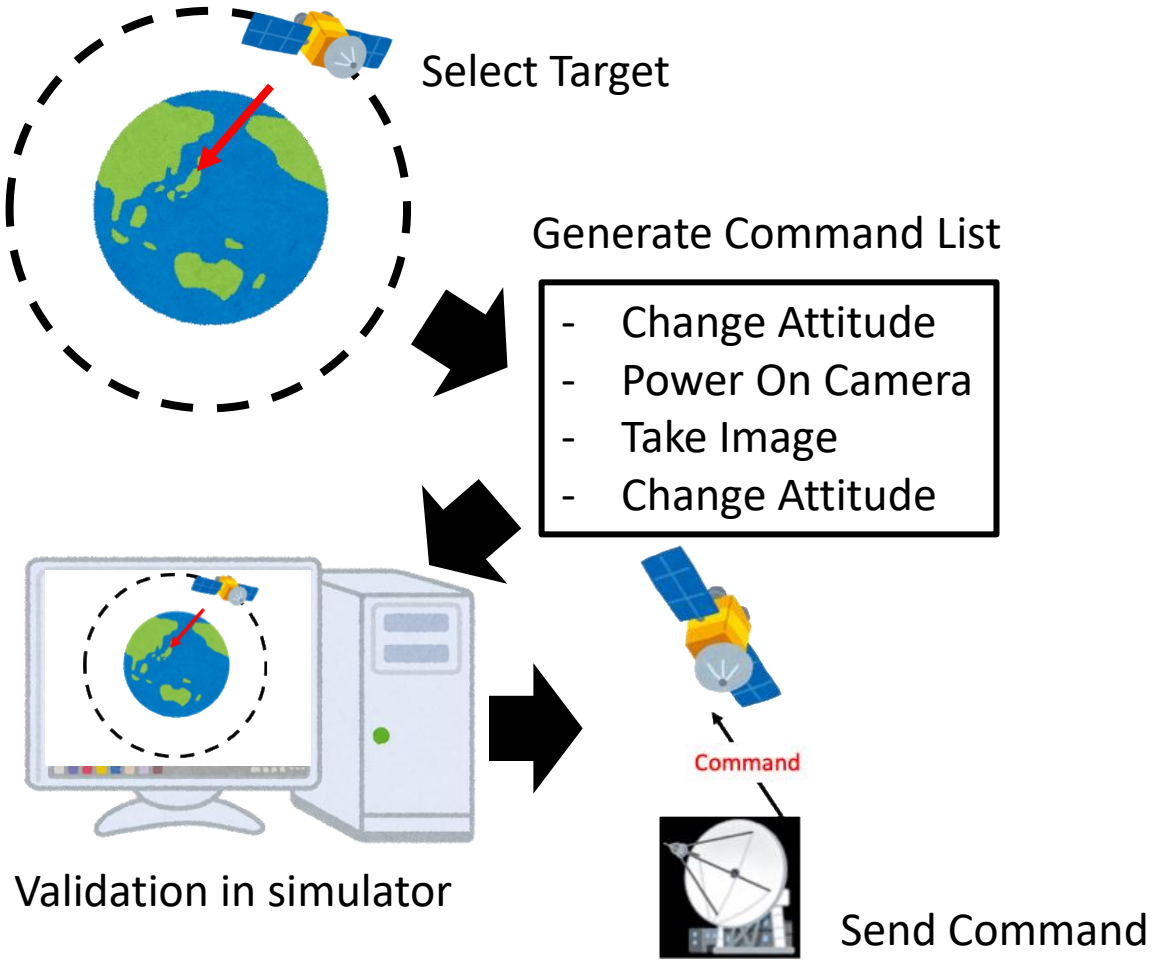
Examples of telemetry Quick Look GUI with Grafana

- The received telemetry data is the most important artifact of the satellite mission and must be kept in strict confidence.
- The telemetry data including housekeeping data, and mission data should be stored in storage servers.
- The transmitted command information should also be stored on the server to verify what kind of satellite operations were performed.



Example of Database Server Connection

- When operating a large number of CubeSats simultaneously, it is essential to automate the operation.
- Example of Automatic Operation
  Automatic Command Generation before the operation path
  1. The target imaging point is selected by users.
  2. The target attitude of the satellite to get the image is calculated and a maneuver plan is generated.
  3. Command list to realize the maneuverer and imaging is generated.
  4. The command list is validated with the numerical simulator by checking the power generation, thermal analysis, attitude angular velocity, and so on.
  5. When the command list is feasible, the commands send to the satellite in the uplink path.



5. Software Management

5. Software Management

5.1. Purpose of Software Management

- To develop reliable software
- To develop reusable software
- Cooperative development with team members
- To realize the above requirements, the following features are important
  - Readability
  - Modularity
  - Coding rules
  - Peer review
  - Test, test, test!

- Coding Rules
- Naming rule
  - How to decide files, variables, and functions name
  - Abbreviation rule
- Coding Format
  - Indent style
  - Length of a line
- Popular Coding Rules
  - Google C++ Style Guide
  - WebKit Code Style Guidelines
  - Google Python Style Guide

Recommend to use formatter.

snake_case
UPPER_SNAKE_CASE
kebab-case
PascalCase

```
Vector<4> q;
Vector<4> quaternion_i2b;
```

Allman style

```
while (x == y)
{
    something(); somethingelse();
}
```

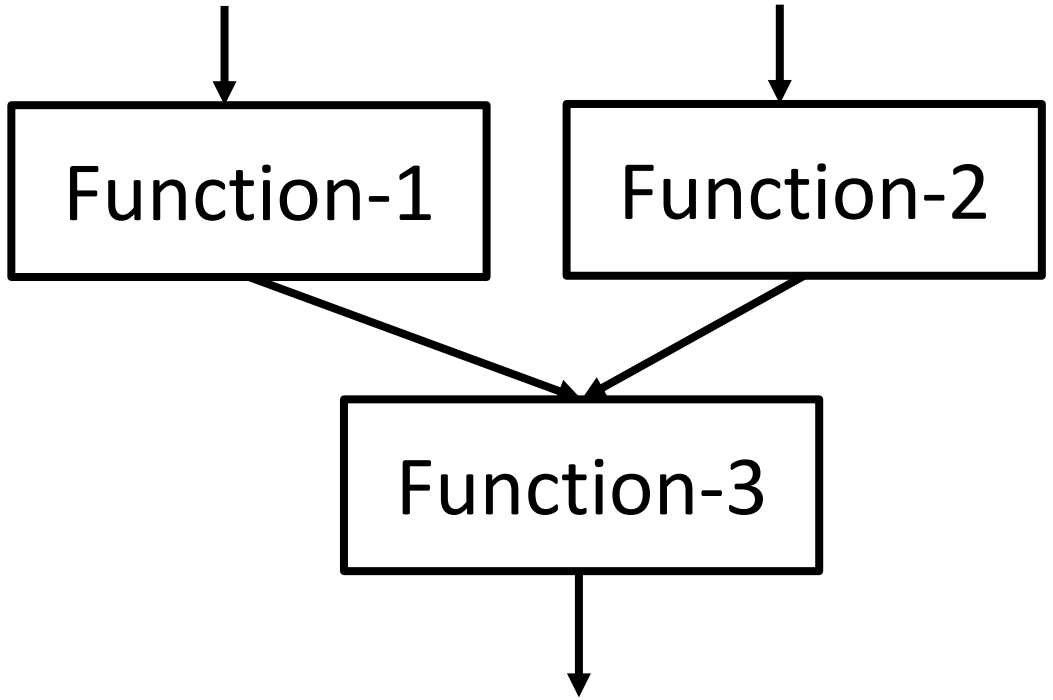
K&R style

```
while (x == y){
    something(); somethingelse();
}
```

- Define the interface between functions
  - To clarify the division of work

  -> Modularity, Team development

- Input and Output of functions
  - Meaning of arguments and return
  - Range of arguments and return
    - Return error when the value over the range
  - Dimension, unit, frame definition of variables



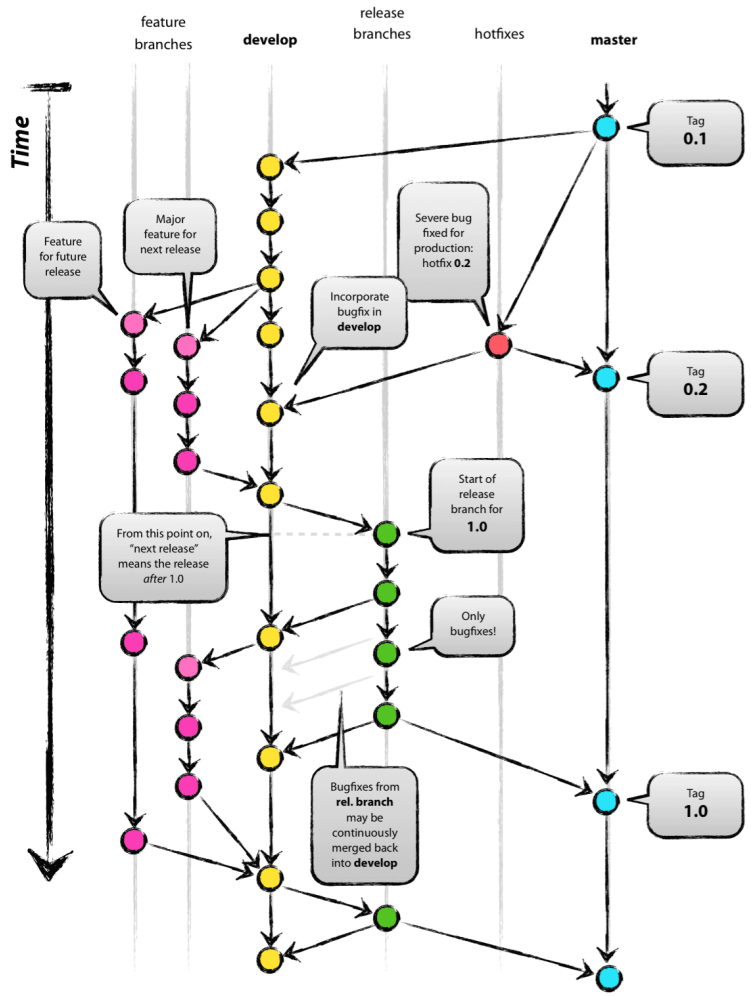
Is the angle in unit rad or degree?

Is the sun vector in the ECI frame or the body fixed frame?

5. Software Management

5.3. Version Management

- Software is easy to edit
- Software version control is essential to track who made what changes and when.
  - For effective collaboration and ensuring the stability and consistency of code across development stages.
- Git
  - A distributed version control system that enables efficient tracking of changes.
- GitHub
  - A cloud-based platform that uses Git for hosting, managing, and collaborating on repositories
  - Additional features: issue tracking, pull requests, and Actions

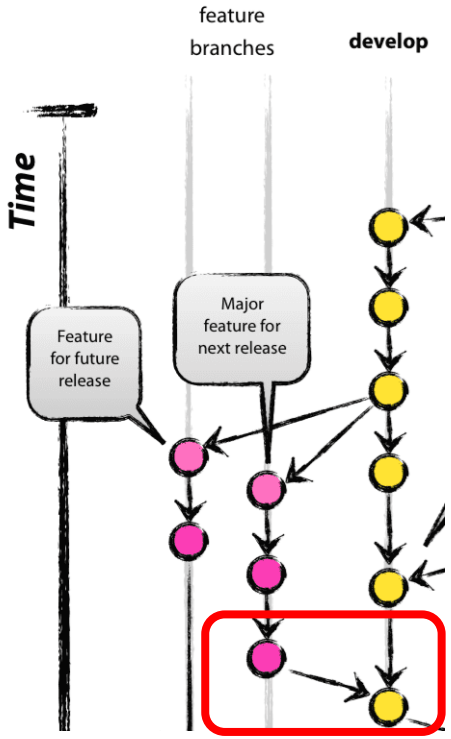


https://nvie.com/posts/a-successful-git-branching-model/

5. Software Management

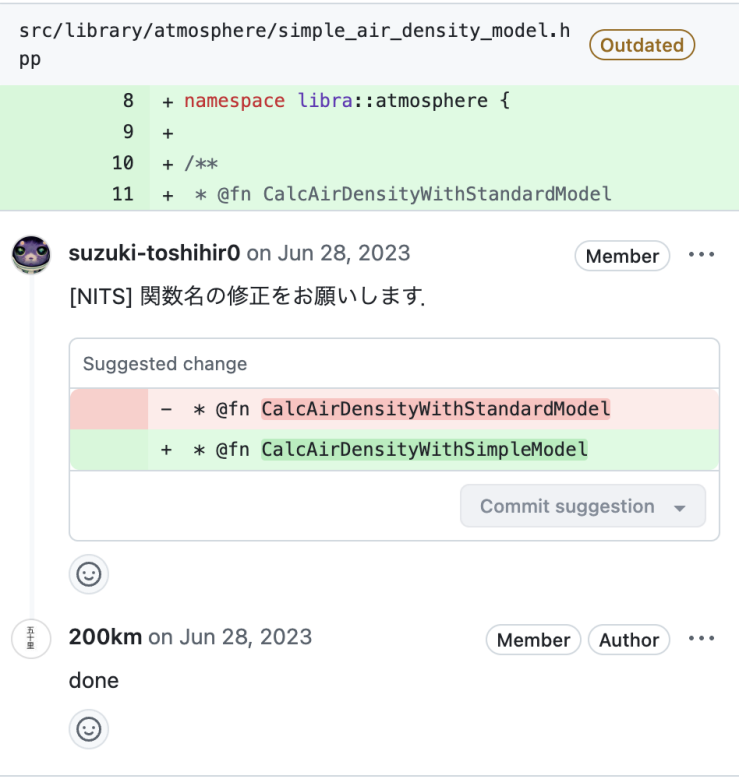
5.4. Peer Review

- Source codes should be reviewed by other team members before merging to the main branch.
  - To ensure quality, identify bugs, and improve maintainability
- GitHub has peer review system as Pull Request.
- Review by automatic bots (GitHub Actions)
  - Check code format or run formatter.
  - Build check, Unit test, Output test
- Review by team members
  - Check the code is readable, naming is readable.
  - Carefully check for potential bugs.
  - The developer should add test result to clarify the adding source code is correct.



Pull Request with peer review

https://nvie.com/posts/a-successful-git-branching-model/



Example of Peer Review in GitHub

- What has not been tested does not work well !!
- Unit Test
  - Verification of the functionality of individual components or modules in isolation to ensure they work as intended.
  - Any code, no matter how small, should be tested.
  - Unit tests can be automatically done when someone makes a Pull Request.
- Integration Test
  - Verification of the interactions between integrated modules or components to ensure they work together as expected.
  - It is necessary to consider what kind of testing needs to be done on a project-by-project basis.

```
[----------] 6 tests from EpochTime
[ RUN      ] EpochTime.ConstructorNominal
[       OK ] EpochTime.ConstructorNominal (0 ms)
[ RUN      ] EpochTime.ConstructorWithCalender
[       OK ] EpochTime.ConstructorWithCalender (0 ms)
[ RUN      ] EpochTime.TimeAdd
[       OK ] EpochTime.TimeAdd (0 ms)
[ RUN      ] EpochTime.TimeSubtract
[       OK ] EpochTime.TimeSubtract (0 ms)
[ RUN      ] EpochTime.EqualOperator
[       OK ] EpochTime.EqualOperator (0 ms)
[ RUN      ] EpochTime.ComparisonOperator
[       OK ] EpochTime.ComparisonOperator (0 ms)
[----------] 6 tests from EpochTime (0 ms total)

[----------] 3 tests from GpsTime
[ RUN      ] GpsTime.ConstructorNominal
[       OK ] GpsTime.ConstructorNominal (0 ms)
[ RUN      ] GpsTime.ConstructorWithEpochTime
[       OK ] GpsTime.ConstructorWithEpochTime (0 ms)
[ RUN      ] GpsTime.ConstructorWithDateTime
[       OK ] GpsTime.ConstructorWithDateTime (0 ms)
[----------] 3 tests from GpsTime (0 ms total)

[----------] Global test environment tear-down
[==========] 118 tests from 18 test suites ran. (143 ms total)
[  PASSED  ] 118 tests.
```

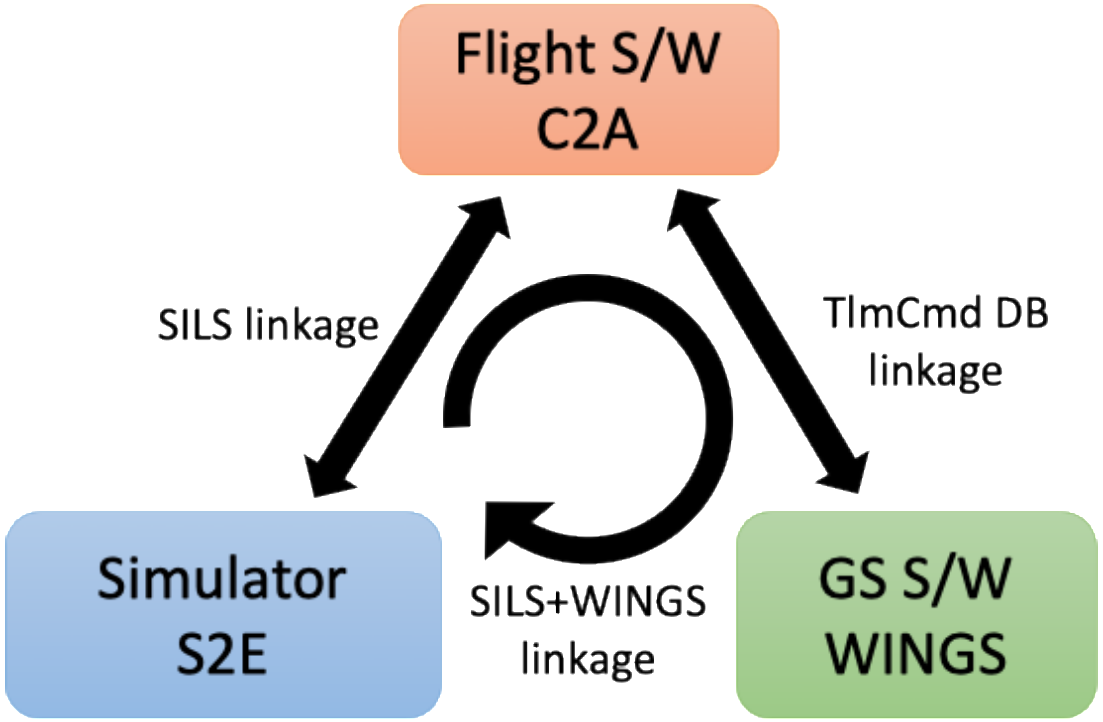
Example of Unit Test

6. Examples of Software Suite for Satellite Research and Development

6. Example of Software Suite for Satellite Research and Development

6.1. Overview of ISSL Open-Source Software Project

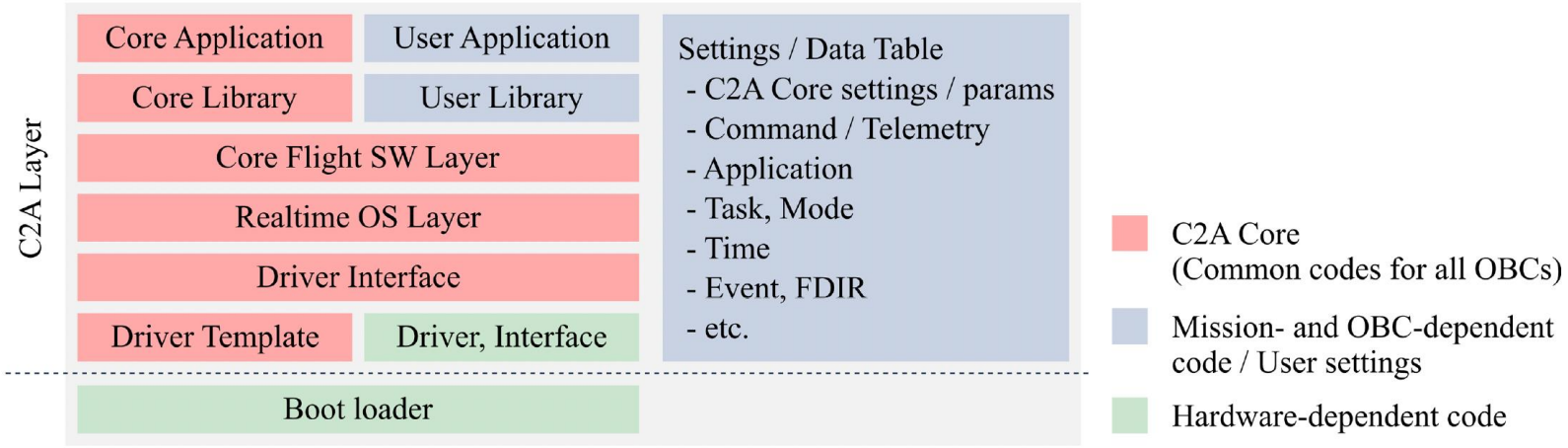
- ISSL (Intelligent Space Systems Laboratory) at the University of Tokyo has released the OSS (Open-Source Software) suite for satellite research and development.
  - Please visit GitHub: https://github.com/ut-issl
- The OSS Suite includes
  - C2A (Command Centric Architecture)
    - Highly Flexible Flight Software Architecture
  - S2E (Spacecraft Simulation Environment)
    - High Fidelity Astrodynamics Numerical Simulator
  - WINGS (Web-based INterface Ground-station Software)
    - Web based Ground Station Software
- S2E, C2A, and WINGS can be executed independently, yet they seamlessly connect to enhance spacecraft mission analysis and operational experiences.



6. Example of Software Suite for Satellite Research and Development

6.2. Flight Software: C2A (Command Centric Architecture)

- ISSL has been developing the C2A since 2013.
- C2A has been developed with a focus on providing high reusability and flexible on-orbit reconfiguration capability. A major feature of C2A is its ability to describe the behaviour of the satellite by command. This software architecture has enabled the short-term development of the software and on-orbit reconfiguration to modify the functionality of the satellite.
- We already used the C2A for several micro/nano-satellites we developed on orbit.
  - Hodoyoshi-3, 4, PROCYON, EQUULEUS, Sphere-EYE 1, ONGLAISAT and other satellites



Overview of C2A

[4] R. Suzumoto, S. Ikari, and et al., “Open-source Software Suite for Small Satellites: C2A, S2E, WINGS”, SSC, 2022

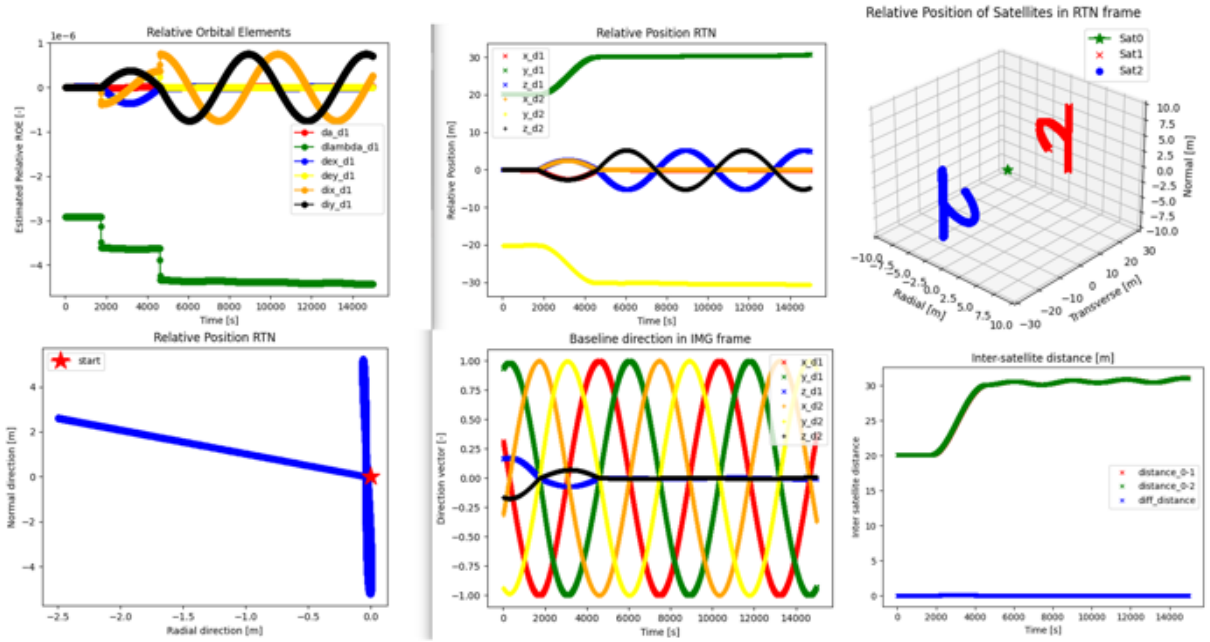
For more details on C2A, Please visit the following GitHub repositories.

- c2a-core
  - The core code of C2A
  - https://github.com/ut-issl/c2a-core
- c2a-aobc
  - A user side code of C2A to implement attitude control algorithm

6. Example of Software Suite for Satellite Research and Development

6.3. Numerical Simulator: S2E (Spacecraft Simulation Environment)

- ISSL has been developing the S2E since 2016.
- S2E has the following features to realize high-fidelity astrodynamics simulation.
  - Space Environment
    - Celestial body position information with SPICE
    - Rotation of the Earth and the moon
    - Geomagnetic field with IGRF
    - Air density with NRLMSISE-00
    - GNSS satellite position
    - Eclipse by the Earth
  - Satellite Dynamics
    - Attitude
      - Propagation with 4th order Runge-Kutta
      - Predefined controlled attitude
    - Orbit
      - Kepler motion without any disturbances
      - Propagation with 4th order Runge-Kutta
      - SGP4 with TLE

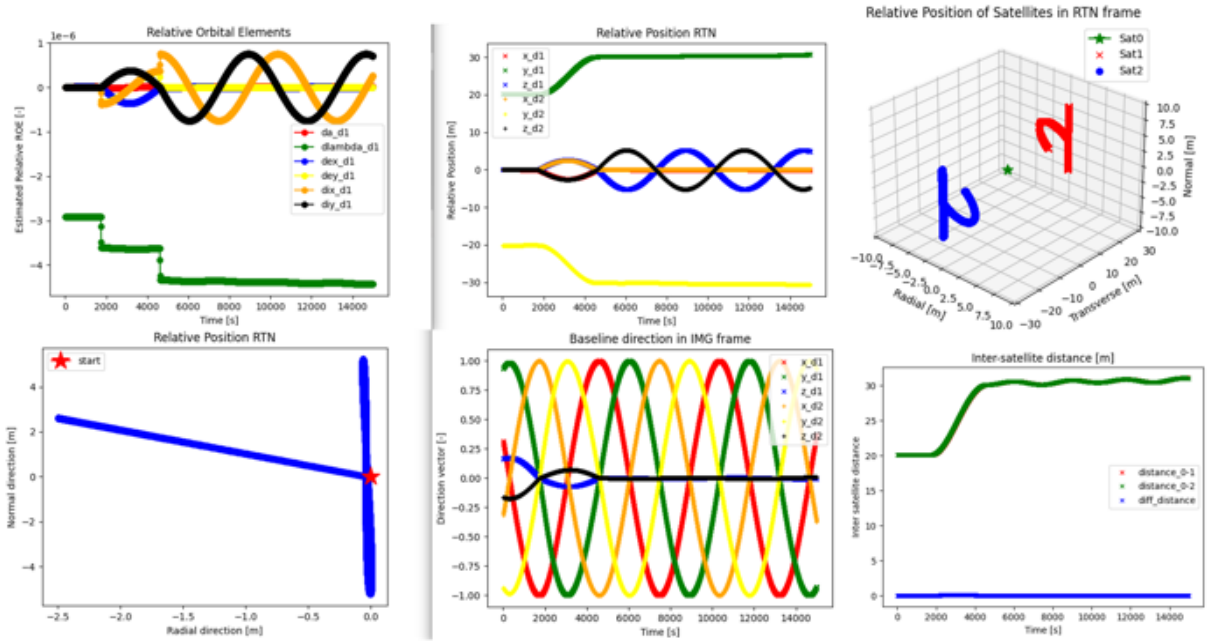


Example of S2E Output
(Formation Flying Satellite Analysis)

6. Example of Software Suite for Satellite Research and Development

6.3. Numerical Simulator: S2E (Spacecraft Simulation Environment)

- ISSL has been developing the S2E since 2016.
- S2E has the following features to realize high-fidelity astrodynamics simulation.
  - Disturbance
    - Geopotential (up to 360 degrees)
    - Third body gravity
    - Air drag force and torque (Multi-surface model)
    - Solar radiation pressure force and torque (Multi-surface model)
    - Magnetic disturbance torque
    - Gravity gradient torque
    - Micro-vibration of RWs
    - Flexible structure vibration (implementing

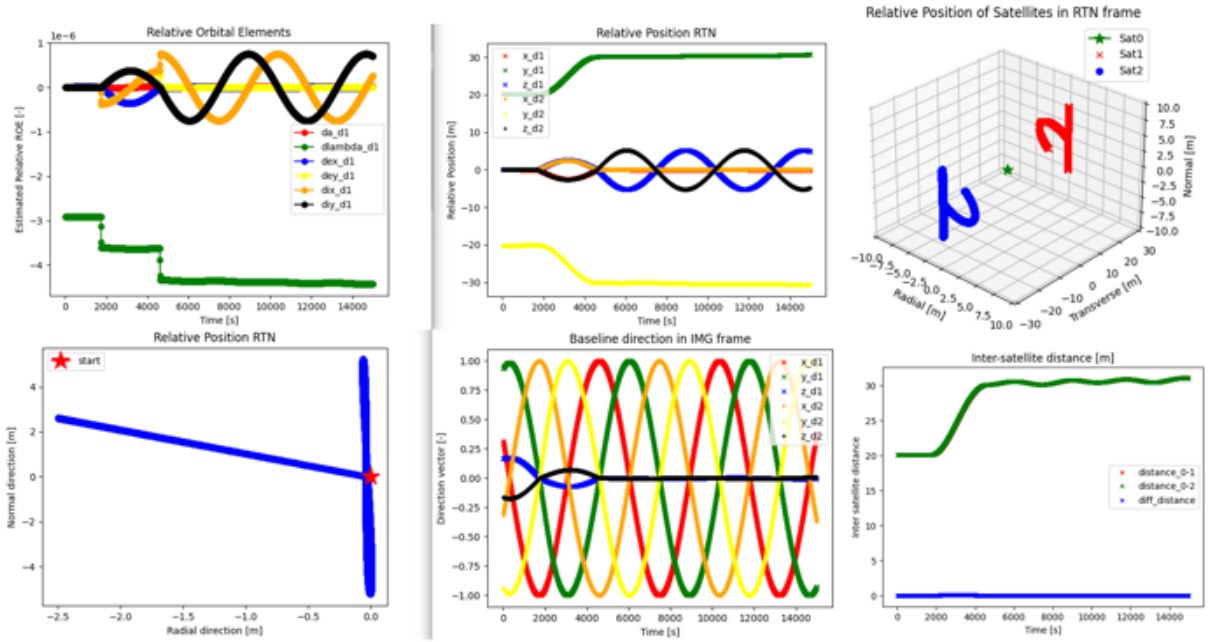


Example of S2E Output
(Formation Flying Satellite Analysis)

6. Example of Software Suite for Satellite Research and Development

6.3. Numerical Simulator: S2E (Spacecraft Simulation Environment)

- ISSL has been developing the S2E since 2016.
- S2E has the following features to realize high-fidelity astrodynamics simulation.
  - Component Emulation
    - Position and Coordinate definition
    - Noise and delay of measure or control signals
    - Telemetry and command for the components
    - Power switch control
  - Others
    - Thermal Analysis
    - Monte-Carlo simulation
    - Multiple satellites simulation
    - Communication with PC's COM ports for HILS

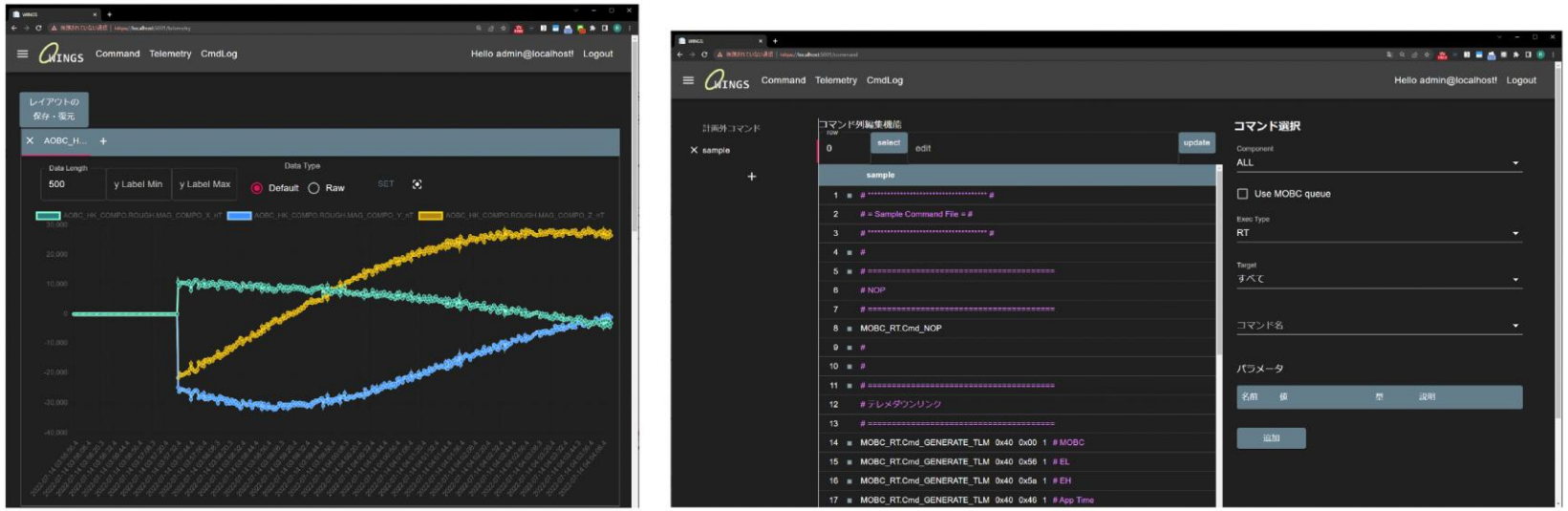


Example of S2E Output
(Formation Flying Satellite Analysis)

6. Example of Software Suite for Satellite Research and Development

6.4. Ground Operation Software: WINGS (Web-based INterface Ground-station Software)

- ISSL has been developing the WINGS since 2020.
- WINGS has been developed for use in satellite operations, ground tests, and component tests. The front-end and back-end of WINGS are implemented separately, and the front-end calls REST APIs to connect the back-end.
- Therefore, the user interface can be extended to be more user-friendly by modifying only the front-end. Furthermore, by calling the APIs via scripts such as Python, it is easy to perform automated testing of components and build an automated operating system.



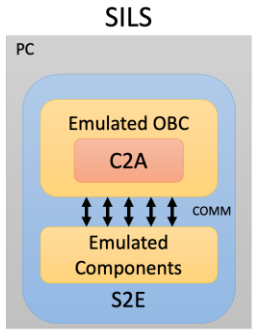
GUI of WINGS

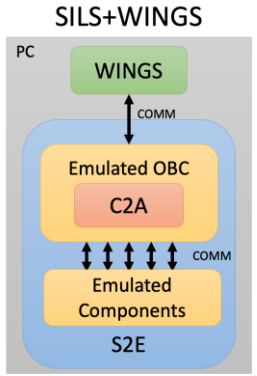
[4] R. Suzumoto, S. Ikari, and et al., “Open-source Software Suite for Small Satellites: C2A, S2E, WINGS”, SSC, 2022

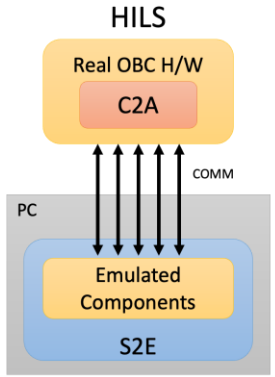
6. Example of Software Suite for Satellite Research and Development

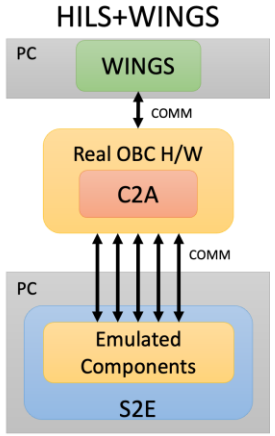
6.5. Example of usage of the ISSL OSS suite in Satellite Development

- The ISSL OSS suite widely covers many satellite research and development activities, from education to real satellite operations.
- SILS (Software In the Loop Simulation)
  - C2A+S2E : The simplest testing configuration without control from outside. The automatic onboard algorithm like attitude control is tested.
  - C2A+S2E+WINGS: Users can uplink commands to the flight software and downlink telemetry from the software.
- HILS (Hardware In the Loop Simulation)
  - C2A and WINGS naturally work with hardware like OBC and ground station equipment. S2E can also work with hardware by connecting it to the PC USB ports, and users can make a HILS system.









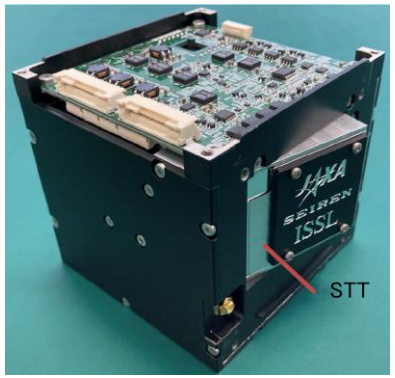
Situations	Activities	Combination
Research & Education	Education of Astrodynamics	S2E
	Control Algorithm Research	S2E (+ C2A)
	Automatic Operation System Research	SILS + WINGS
	Experience of Satellite Operation	SILS + WINGS
Satellite Mission Design	Orbit Design, Power Consumption Analysis, Communication Analysis	S2E
	Attitude Control Accuracy Analysis	S2E + C2A
Satellite Development	Component communication test	Component + WINGS
	Satellite communication test	OBC + WINGS
	Flight S/W development	SILS (+WINGS)
Satellite Verification	Flight SW test	HILS + WINGS
	Operation Training	SILS (or HILS) + WINGS
Satellite Operation	Orbit Prediction, Visibility analysis	S2E
	Verification of operation commands	SILS (or HILS) + WINGS

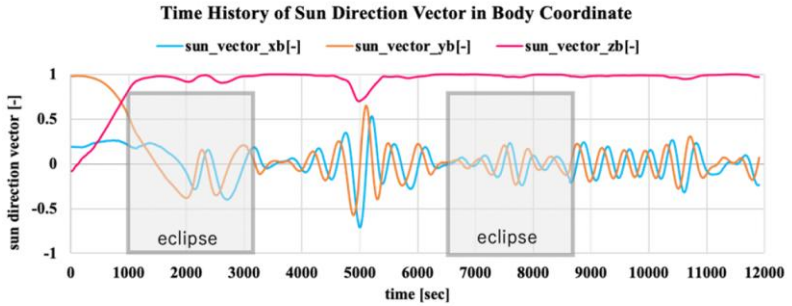
6. Example of Software Suite for Satellite Research and Development

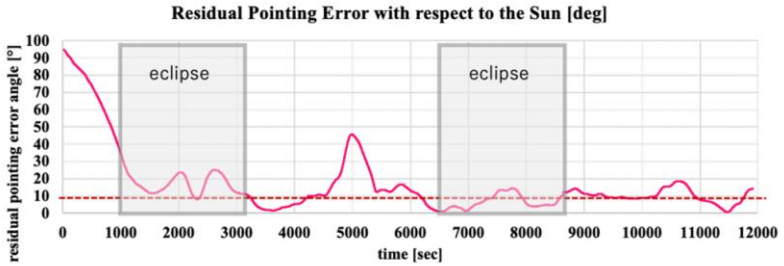
6.5. Example of usage of the ISSL OSS suite in Satellite Development

- We have developed a 1U-size AOCS module that includes the full feature of three-axis attitude control.
- We effectively used the OSS suite during the AOCS module's onboard software development and verification process.
- The customized S2E (S2E-AOBC) and C2A (C2A-AOBC) for the AOCS module are also published as OSS.
- The flight software C2A-AOBC was verified with SILS and HILS configurations .

S. Ikari, and *et al.*, "Development of Compact and Highly Capable Integrated AOCS Module for CubeSats", Journal of Evolving Space Activities, vol. 1, ID 63, 2023.







Attitude Control with Magnetorquer

7. Conclusion

7. Conclusion

- Software used in satellite development was introduced. The importance of software development to improve the success rate of satellites was explained.
- Objectives and overview of flight software, numerical simulation, and ground station software were discussed. Key features of each software were explained.
- An overview of software management methods were introduced. The software management method is essential to realize reliable software development with team members. It is directly connected with the success rate of satellites.
- As examples of flight software, numerical simulation, and ground station software, ISSL OSS (Open-Source Software) Suite was introduced. The details of the independent software was explained, and the connection between the software was also introduced.

Thank you very much.

[Disclaimer]
The views and opinions expressed in this presentation are those of the authors and do not necessarily reflect those of the United Nations.