Development and Application of Web-based Interface Ground station Software (WINGS) for the Operation of SPHERE-1 EYE

Tomoki Mochizuki, Yosuke Kawabata, Takeshi Matsumoto, Shunsuke Shimomura, Akihiro Ishikawa,
Ryu Funase, Shinichi Nakasuka
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan; +81-80-4669-8479
mochizuki@space.t.u-tokyo.ac.jp

Kota Kakihara
ArkEdge Space Inc., Japan

ABSTRACT

The evolution of small satellites as both technological demonstrators and operational platforms has been significant. However, these projects often face constraints like limited budgets and stringent timelines, coupled with growing mission complexities. To address these challenges, the Intelligent Space System Laboratory (ISSL) at the University of Tokyo developed Web-based INterface Ground station Software (WINGS), an open-source ground station software with a modular architecture that includes front-end, back-end, and database components, as well as a telemetry and telecommand interface (WINGS-TMTC-IF). The paper highlights the application of WINGS in the SPHERE-1 EYE mission, showcasing its capability to customize and rapidly deploy satellite operations effectively, and compares its features and advantages against other ground station software. It concludes with operational insights and the lessons learned from the SPHERE-1 EYE mission, emphasizing the critical functionalities of ground station software in supporting small satellite operations.

INTRODUCTION

In recent years, small satellites have been used not only as technology demonstration platforms, but also for practical mission execution. However, small satellite projects often struggle with resource constraints, including tight budgets and schedules, while facing increasing mission complexity. In this environment, the importance of robust satellite software development cannot be overstated. The Intelligent Space Systems Laboratory (ISSL) at the University of Tokyo has initiated three open-source projects at the ISSL Organization on GitHub[1]: an on-board flight software C2A (Command Centric Architecture), a satellite simulator S2E (Spacecraft Simulation Environment), and a ground station software WINGS (Web-based INterface Ground station Software)[2].

In the realm of satellite development and operation, the process typically involves various phases, including individual component testing, integrated testing of multiple components, comprehensive testing of the fully assembled satellite, and satellite operations. Furthermore, Software-In-the-Loop Simulation (SILS) and Hardware-In-the-Loop Simulation (HILS) are conducted to verify the health of the satellite system before writing the developed program to the On-Board Computers (OBCs). Given these requirements, there is a substantial demand for ground station software that can be universally applied across all these phases. However, until recently, there were few examples of such satellite development and operational software being available as open source. The ISSL has addressed this gap by developing WINGS, an OSS (Open-Source Software) that can be used in any phase of the satellite lifecycle, from individual component testing to ground testing, SILS and HILS tests, and operational deployment, simply by writing the necessary driver code.

This paper presents a detailed exploration of the functionalities of WINGS, with particular emphasis on its comprehensive feature set, modular architecture, and versatility. In addition, it compares WINGS to other ground station software, highlighting its uniqueness and advantages. The paper also highlights the successful application of WINGS in the SPHERE-1 EYE mission collaborating with Sony Group Corporation and its contributions to the mission's operational success. Finally, the paper provides valuable insights and lessons learned about essential ground station software functionalities based on the SPHERE-1 EYE operational experience.

SOFTWARE FEATURES REQUIRED FOR THE SMALL SATELLITE DEVELOPMENT

In the context of small satellite development, operational software must possess the capability to display real-time

telemetry and transmit commands to the satellite. Additionally, it is desirable for the software to exhibit certain features that are beneficial for the development and operation of small satellites within limited timeframes and budgets. These features include applicability across various phases of the satellite development process, support for multiple communication protocol, and connectivity to SILS and HILS testing environment.

Ensuring the functional integrity of components such as the MOBC (Main On-Board Computer), attitude control modules, transceivers, and PCDU (Power Conditioning and Distribution Unit) necessitates the interpretation of data emitted by each component and the transmission of commands to verify their behavior. Although different software can be used for each phase, having software that can cater to all these phases significantly enhances the efficiency of satellite development. Thus, it is desirable to have the telemetry-command system applicable across various phases, including individual component testing, comprehensive ground testing of the satellite, and actual satellite operations.

During individual component testing, UART communication protocol is commonly used, while tests involving satellite transceivers typically employ UDP or TCP/IP communication protocol with modulators and demodulators. Thus, software that supports diverse communication protocols is particularly advantageous. Furthermore, integrated tools for SILS and HILS testing are crucial for pre-launch testing and validation, simulating the satellite's behavior and interaction with the ground station to ensure that all systems function correctly before deployment.

FEATURES OF OTHER GROUND STATIONS

Ground Station Systems Developed by the Georgia Institute of Technology

From 2018, the Georgia Institute of Technology needed to manage the operations of several satellite projects scheduled for launch as displayed in Table 1[3]. Since the transceiver frequencies used by each satellite varied, it is required to operate three different ground stations to accommodate these varying frequencies. Therefore, the Geogia Institute of Technology established a ground station architecture sharing the same connection interface. For a mission operation software package, they utilized quantumCMD, a ground station software developed by KRATOS, to visualize command transmission and telemetry.

quantumCMD software offers various functionalities including telemetry monitoring, measurand limit checks and alarming, command generation and formatting, transmission and tracking, procedure scripting, real-time user interface, operations automation, and plotting and trending[9]. However, it only provides the satellite operation environments and does not provide single component testing environments, HILS testing environment, and SILS testing environment.

Table 1: Overview of the small satellites developed by Georgia Institute of Technology[3-8].

Mission	Frequency Band (Uplink/Downlink)	Launch Year
RANGE	UHF/UHF	2018 (Scheduled in 2017)
Prox-1	UHF/S-band	2018 (Scheduled in 2017)
ARMADILLO	VHF/UHF	2019 (Scheduled in 2017)
RECONSO	UHF/UHF	Cancelled (Scheduled in 2018)
TARGIT	UHF/UHF	2021 (Scheduled in 2018)

Generic Spacecraft Test and Operations Software (GSTOS) developed by JAXA

Generic Spacecraft Test Operations Software (GSTOS) is designed to support the verification and operation of both spacecraft components and complete spacecraft systems. GSTOS is part of a comprehensive suite of tools developed to streamline the spacecraft development process, reduce the complexity of preparing test procedures, and enhance the efficiency of test execution[10]. GSTOS has essential elements required for satellite development and operation including a telemetry packet display view, a telemetry graph display view, an automated command transmitting function, an automated telemetry health-checking function. However, because the application of this software is limited to JAXA missions, it is required to use another ground station software in other satellite projects.

Integrated Test and Operations System (ITOS) developed by NASA

Integrated Test and Operation System (ITOS) is a ground station software to provide comprehensive command and telemetry solution for board and box development, instrument and component development, spacecraft development, spacecraft or component simulations, flight software development, integration and testing of the components, and mission operations[11]. ITOS appears to employ nearly all essential elements required for satellite development. However, its application is predominantly restricted to NASA missions, making it difficult to utilize this software for satellite testing and operations in a straightforward and accessible manner in other projects.

OVERVIEW OF WINGS SYSTEM

Among the aforementioned ground station systems, some are capable of supporting various phases of satellite development and satellite operations. However, these systems are often proprietary software or restricted to specific satellite projects. In response, the ISSL developed a ground station system as an OSS, ensuring that it is accessible for use across different phases of satellite development and operations by anyone. The ground station system developed by the ISSL comprises two main components: WINGS[12] and WINGS-TMTC-IF[13]. The following sections will provide a detailed explanation of WINGS and WINGS-TMTC-IF.

WINGS

WINGS is designed as a web application to enable connections from multiple users and multiple tabs simultaneously. This design allows users to view telemetry data and send commands using just a web browser, without the need for specialized software. Developed using ASP .NET Core (6.0, Web API, C#), it leverages Microsoft's cross-platform web application development environment, ensuring compatibility with Windows, macOS, Linux, and Docker using the same codebase. The architecture separates the frontend and backend, with the frontend interacting with the backend through HTTP APIs. This separation allows the APIs to be accessed not only via a web browser but also through Python code or other means, making the system operational without a browser.

The home screen of WINGS serves as the main operation interface as shown in Figure 1. By selecting a target component in the "Component" section, users can choose various components such as MOBC and AOBC (SECONDARY_OBC). Additionally, users can create drivers that align with the telemetry and command formats of the components they wish to develop. By compiling the telemetry and command lists into CSV files in a specified format, it becomes possible to use WINGS for command transmission and telemetry visualization.

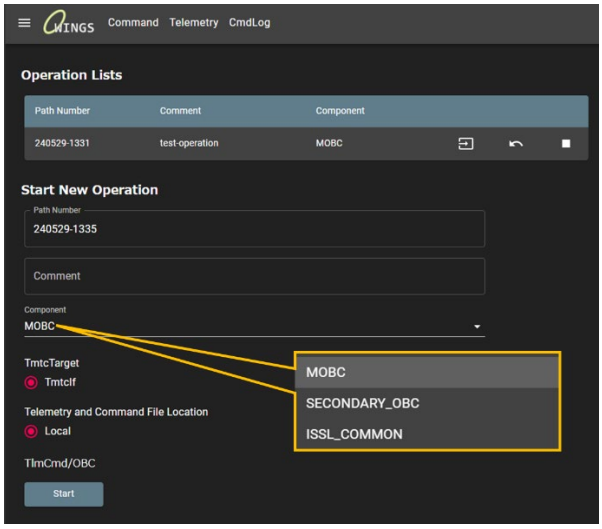

Figure 1: Home screen of WINGS.

The telemetry packet visualization screen, although lacking in customization options, automatically displays telemetry columns based on the telemetry database CSV files as shown in Figure 2. This feature is particularly useful during phases of component development when the telemetry array undergoes significant changes, allowing for the creation of a telemetry visualization screen at low cost, thereby providing substantial convenience. It also provides the capability to graphically plot the time history of individual telemetry data points.

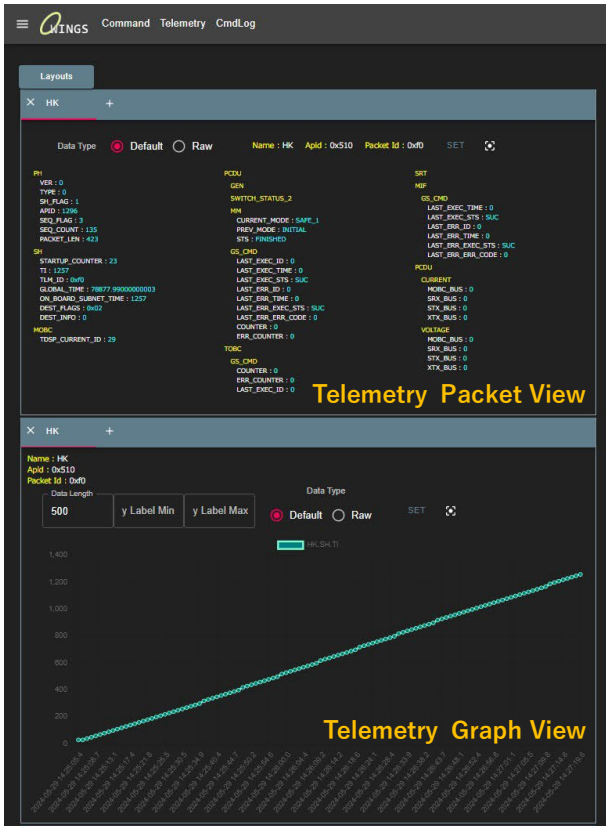

Figure 2: Telemetry visualization screen of WINGS.

The command transmission screen is equipped with features for sending both preplanned command sequences and unplanned commands. Preplanned command sequences can be loaded from files with a ".ops" extension. Additionally, the screen includes a health check function to verify the satellite's behavior

any time by executing health check command. The current functionalities available in WINGS are summarized in Table 2.

WINGS utilizes the same telemetry and command database used in the onboard software C2A developed by ISSL. In C2A, telemetry and command lists are managed as CSV files, which are then used for automatic software generation. This integration allows for seamless usage of telemetry and command data across both onboard and ground station systems.

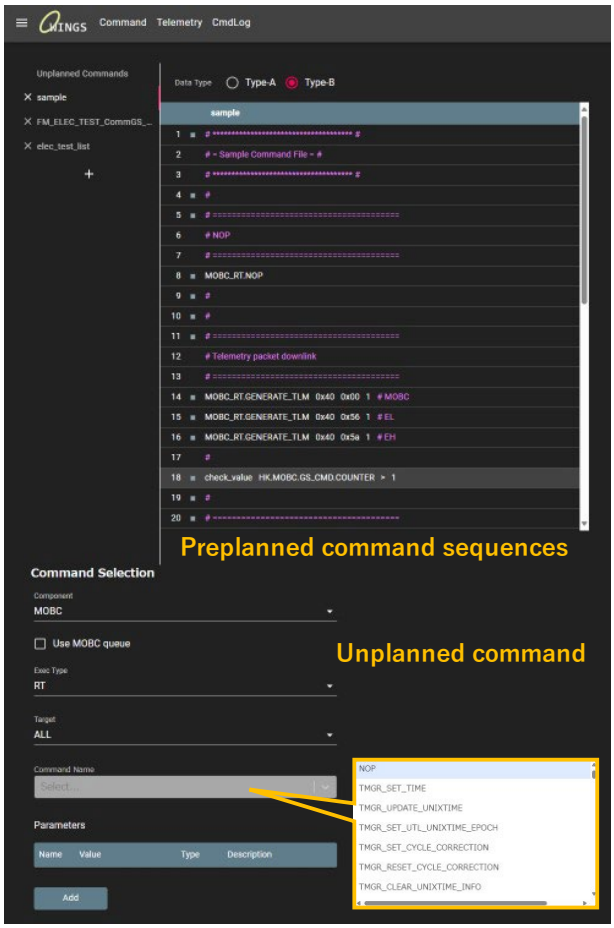

Figure 3: Command visualization screen of WINGS.

Table 2: Functionalities available in WINGS.

Functionality	Description
call	Function to call another .ops file.
wait_sec	Function to specify how many seconds to wait before executing next command written in the next line.
let	Function to define internal variables, which helps verify the execution of the health check of the satellite.
get	Function to get the current value of the telemetry or defined variable using “let” function.
check_value	Function to check whether the specific condition is satisfied.
wait_until	Function to wait until the specific condition is satisfied. This function can add a timeout option to escape from “while” loop.

WINGS-TMTC-IF

WINGS-TMTC-IF handles the absorption of hardware-dependent interfaces, facilitating data transmission and reception between WINGS and the hardware. This includes managing UART communication protocols with each component or satellite, as well as TCP/IP and UDP communication protocols with modulators and demodulators.

To facilitate communication with components using WINGS-TMTC-IF, the user must initially select the target component for communication. Subsequently, the operational procedure developed in WINGS is selected, followed by specifying the communication port and address. Once these parameters are configured, telemetry data sent to WINGS-TMTC-IF is displayed in binary format on the WINGS-TMTC-IF interface and then transmitted to WINGS to visualize in browsers. Conversely, commands issued from WINGS are transmitted to the designated component through WINGS-TMTC-IF. Figure 4 illustrates the telemetry-telecommand interface view of WINGS-TMTC-IF.

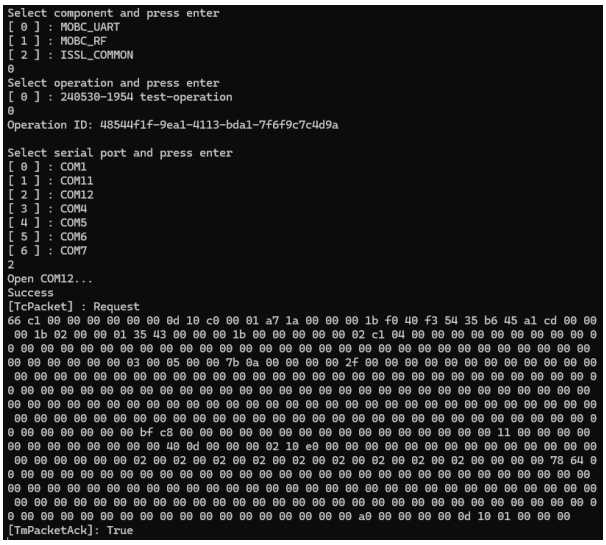

Figure 4: Telemetry-telecommand interface view of WINGS-TMTC-IF.

APPLICATION OF WINGS SYSTEM TO THE OPERATION OF SPHERE-1 EYE

One notable feature of WINGS, as previously mentioned, is the independent programming of its frontend and backend, which allows for flexible customization of the

ground station system. In practice, for the operation of SPHERE-1 EYE, a ground station system was constructed using selected functionalities of WINGS in conjunction with other OSS, and operations were conducted under this customized system. Figure 5 illustrates the comparison between the ground station systems used during the development and operational phases of SPHERE-1.

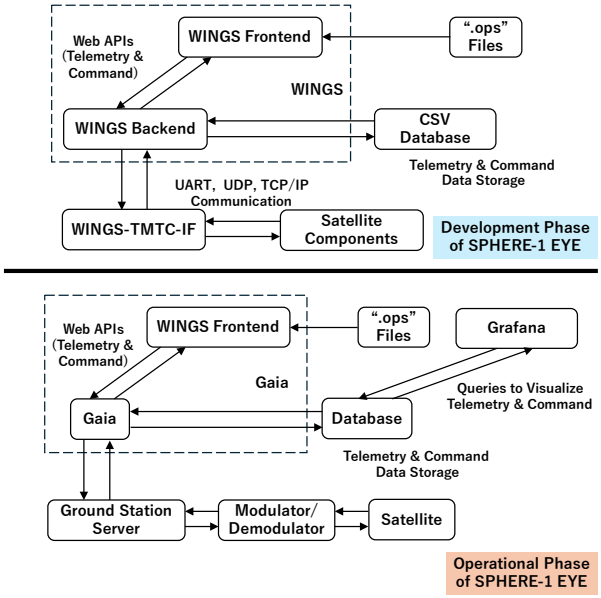

Figure 5: Comparison of the ground station system of SPHERE-1 EYE in the development phase and the operational phase.

In the satellite operation system for SPHERE-1 EYE, Gaia[14], an OSS developed by ArkEdge Space Inc.[15], was used in place of the WINGS backend. Additionally, an OSS named Grafana[16] was employed as a tool for visualizing telemetry and transmitted command sequences. This configuration allowed for the selective use of visualization screens from both WINGS and Grafana, depending on the specific operational requirements.

Grafana's visualization interface offers extensive customization options for displaying telemetry data, excelling at rendering trend graphs, pie charts, bar graphs, and other visualizations in parallel on a single screen as shown in Figure 6. Conversely, WINGS is particularly adept at providing a list view of telemetry packets. Additionally, WINGS supports command transmission and telemetry health checks through its command transmission interface. Consequently, for SPHERE-1 EYE operations, command transmission, satellite health checks, and packet-level telemetry listings are managed using WINGS, while comprehensive telemetry monitoring screens are created using Grafana.

Figure 6: Sample view of Grafana [17] .

Lessons Learned from the Development and Operation of SPHERE-1 EYE

During the development phase of the SPHERE-1 EYE, the telemetry data arrays sent from components such as MOBC and AOBC changed dynamically. In these phases, WINGS appeared to be highly useful as it automatically generated telemetry screens from CSV files with the telemetry lists, thereby reducing the effort required to reconfigure the telemetry layout for satellite development. Additionally, the capability to verify satellite health allowed for the identification of many satellite anomalies and software malfunctions through the WINGS system.

In contrast, during the satellite operation phase, the telemetry data arrays became static, and the focus shifted from automatic generation of telemetry screens to the ability to customize telemetry displays in order to show desired data on a single screen. To address this need, the ISSL utilized the Grafana for the operation of SPHERE-1 EYE.

Based on the experience of the development and operation of the satellite, the combined use of these two telemetry visualization tools allowed for effective management of various issues arising during satellite operations. This suggests that it is advantageous to use WINGS system from the development phase in conjunction with an additional tool that compensates for its limitations, thereby enhancing overall operational efficiency.

CONCLUSION

In this paper, we presented the design of WINGS system, an open-source ground station software system composed of WINGS and WINGS-TMTC-IF. This system was developed with the aim of creating a telemetry reception and command transmission system capable of supporting a wide range of phases, from satellite development to satellite operations. In practice, the WINGS frontend has been used in the operation of SPHERE-1 EYE and has significantly contributed to the

satellite's operation. However, since WINGS does not meet the customization needs for telemetry required during satellite operations, it is used in conjunction with the OSS: Grafana. The combined use of these two telemetry monitoring tools has enabled the smooth operation of SPHERE-1 EYE.

ACKNOWLEDGEMENT

The authors would like to thank all the project members for their contributions to the development and operation of the SPHERE-1 EYE, and construction of the ground station software system for SPHERE-1 EYE.

REFERENCES

[1]. GitHub Organization of Intelligent Space Systems Laboratory, The University of Tokyo, https://github.com/ut-issl, (accessed on May 22, 2024).

[2]. R. Suzumoto, S. Ikari, K. Kakihara, *et al*, "Open-source Software Suite for Small Satellites: C2A (Command Centric Architecture), S2E(Spacecraft Simulation Environment), and WINGS (Web-based Interface Ground-station Software)", 36th Small Satellite Conference, Utah, August, 2022.

[3]. Thomas Choi, Terry H. Stevenson, and E. Glenn Lightsey. "Reference Ground Station Design for University Satellite Missions with Varying Communication Requirements". AIAA SciTech Forum, 55th AIAA Aerospace Sciences Meeting, 2017.

[4]. Georgia Tech, "The Ranging And Nanosatellite Guidance Experiment (RANGE)", https://cshaft.gtri.gatech.edu/missions-2/range/, (accessed on May 27, 2024).

[5]. Georgia Tech, "Georgia Tech Satellite Successfully Launched Into Space", https://news.gatech.edu/news/2019/06/25/georgia-tech-satellite-successfully-launched-space, (accessed on May 27, 2024).

[6]. D. McDonald, "Recovery of a Lost Satellite: The ARMADILLO Mission", 36th Small Satellite Conference, SSC22-VII-04, Utah, August, 2022.

[7]. Nanosats Database, "RECONSO Satellite", https://www.nanosats.eu/sat/reconso, (accessed on May 27, 2024).

[8]. Space System Design Laboratory, "The Tethering And Ranging Mission Of The Georgia Institute Of Technology (TARGIT)", https://ssdl.gatech.edu/research/projects/tethering-and-ranging-mission-georgia-institute-technology-targit, (accessed on May 27, 2024).

[9]. KRATOS, "Command & Control for Small Satellites: quantumCMD". Home > Products > Space > Satellites > Command & Control > quantumCMD, https://www.kratosdefense.com/-/media/k/pdf/s/sa/ds-160-quantumcmd.pdf, (accessed on May 27, 2024).

[10]. K. Matsuzaki, T. Kominato, S. Okunishi, and Y. Miyano. "SIB2/GSTOS Tools for Preparing Test Procedure - PlanEditor and SIB2Viewer". Transactions of the Japan Society for Aeronautical and Space Sciences, AerospaceTechnology Japan, vol. 12, no. ists29, pp. Tt_19-Tt_24, 2014.

[11]. ITOS Group "Integrated Test and Operations System (ITOS)", https://itos.gsfc.nasa.gov/, (accessed on May 27, 2024).

[12]. GitHub Organization of Intelligent Space Systems Laboratory, The University of Tokyo, "WINGS", https://github.com/ut-issl/wings, (accessed on May 31, 2024).

[13]. GitHub Organization of Intelligent Space Systems Laboratory, The University of Tokyo, "WINGS-TMTC-IF", https://github.com/ut-issl/wings-tmtc-if, (accessed on May 31, 2024).

[14]. GitHub Organization of ArkEdge Space Inc., "Gaia", https://github.com/arkedge/gaia, (accessed on June 2, 2024).

[15]. ArkEdge Space Inc., https://arkedgespace.com/en, (accessed on June 2, 2024).

[16]. GitHub Organization of Grafana Labs, "Graphana", https://github.com/grafana/grafana, (accessed on June 2, 2024).

[17]. Grapana Labs, "Dashboard anything. Observe everything.", https://grafana.com/grafana/, (accessed on June 2, 2024).