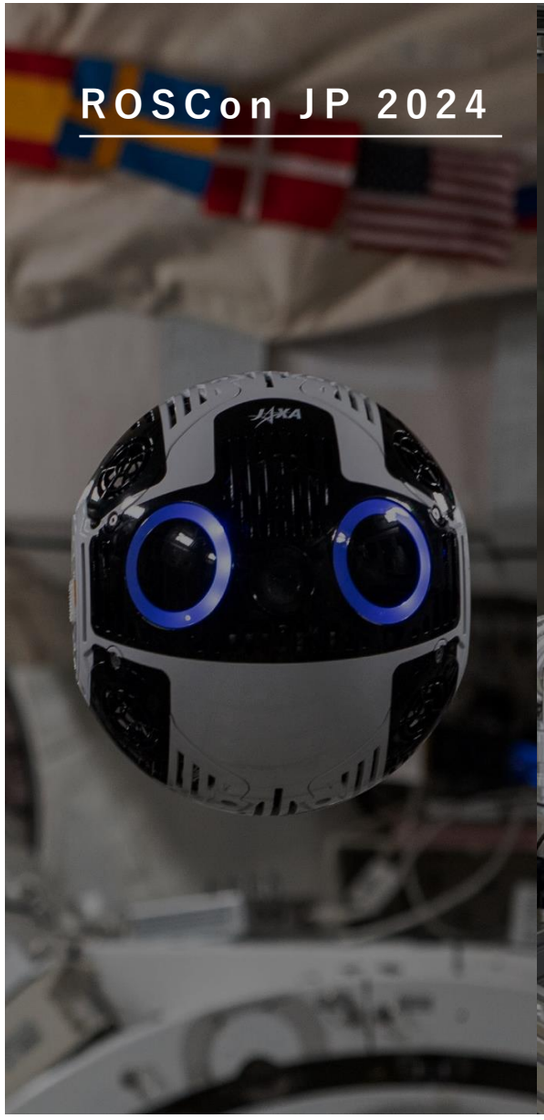
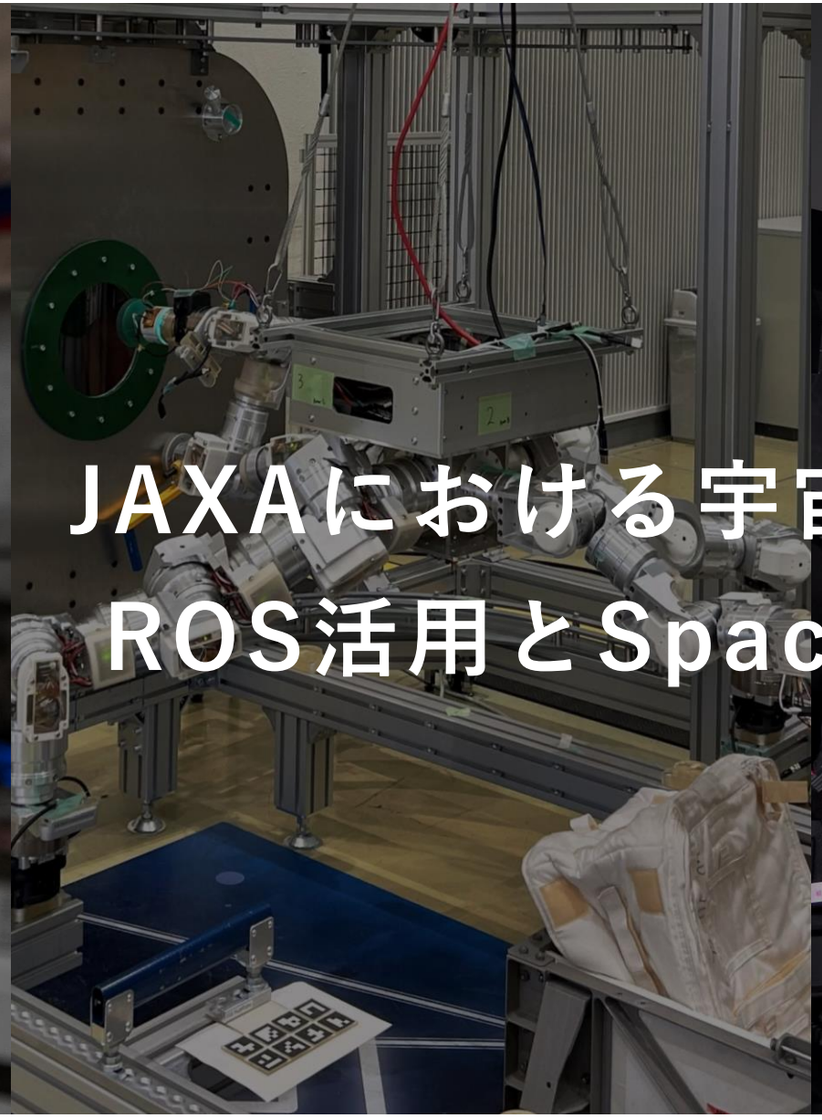
ROSCon JP 2024





JAXAにおける宇宙機開発へのROS活用とSpace ROS開発

宇宙航空研究開発機構（JAXA）
研究開発部門

池田 勇輝, 西下 敦青, 加藤 裕基

JAXA Explore to Realize

自己紹介

池田 勇輝 (Ikeda Yuki)

■ 所属

・JAXA 研究開発部門 第一研究ユニット(本務)
・JAXA 研究開発部門 商業デブリ除去実証フェーズII
　プロジェクトチーム(併任)

■ 経歴

・2017年~2023年：知能ロボティクスの研究開発(大学・大学院)
・2023年~ 現　在：宇宙ロボティクスの研究開発(JAXA)

■ 現在の業務

・商業デブリ除去実証プロジェクト(CRD2)における技術支援
・デブリの捕獲・除去技術の研究開発
・軌道上サービス実証プラットフォーム(SATDyn)の研究開発
・宇宙ロボット向けROS(Space ROS)の研究開発

@YukiIkeda0810
https://github.com/jaxa

JAXA（宇宙航空研究開発機構）

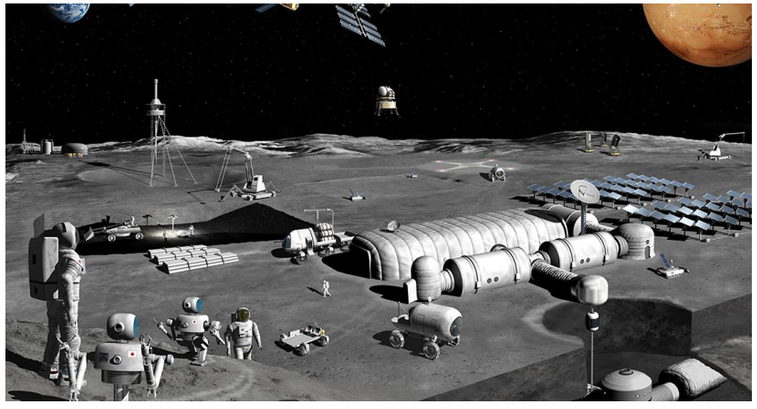
宇宙航空研究開発機構
Japan Aerospace Exploration Agency

・宇宙と空を活かし、安全で豊かな社会を実現します
・私たちは、先導的な技術開発を行い、幅広い英知と共に生み出した成果を、
　人類社会に展開します

輸送システムの研究開発と運用


人工衛星の研究開発と運用


有人宇宙活動に関する研究開発と運用


人類の活動圏を拡げる国際的な宇宙探査


太陽系と宇宙の起源の解明に向けた探査


航空科学技術の研究開発

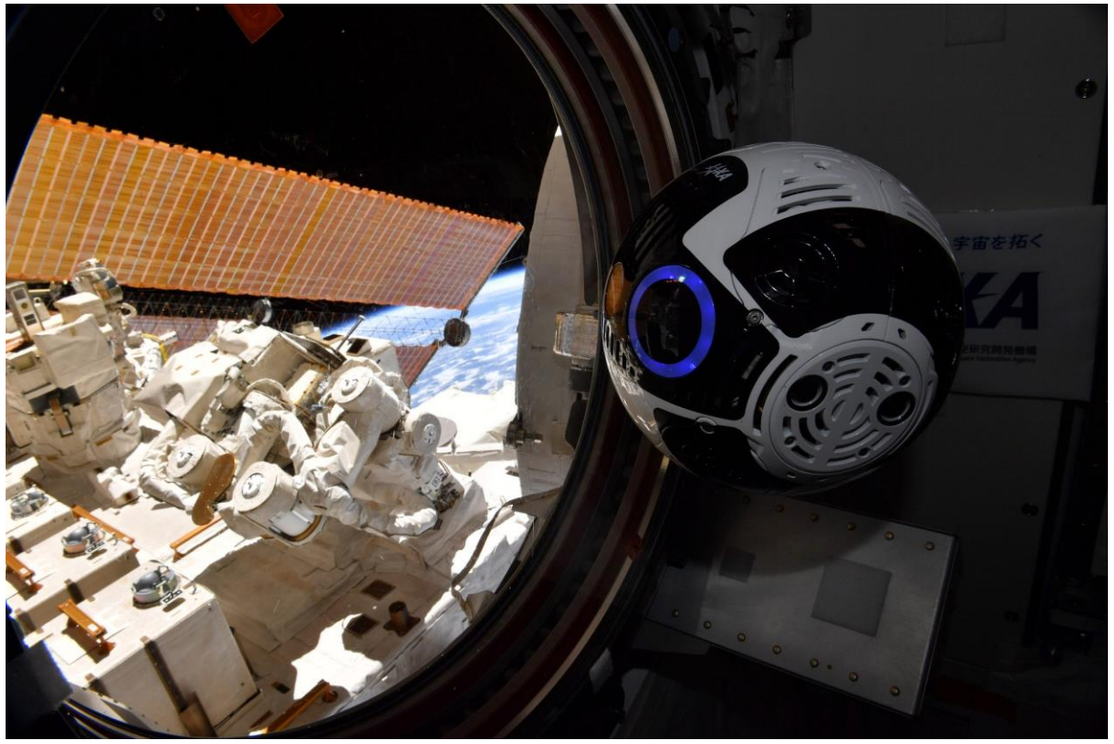
本講演における目的と重要性

【本講演における目的と重要性の説明】

・JAXAの宇宙開発におけるROSの活用事例の紹介

・Space ROSの紹介とJAXAでの取り組み(宇宙開発への参入ハードルの低減と効率化)

・地上ロボットへの応用 (地上で宇宙機並みに信頼性・安全性を担保)



宇宙環境について

■過酷な宇宙環境について

・宇宙環境の特徴として「真空」や「放射線」がある
・真空では熱の対流が起こらず，地上よりも温度変化が大きくなる
・宇宙では，高エネルギーの放射線が降り注ぎ，電子部品にビット反転や短絡故障を引き起こすことがある(万能な対策は存在しない)
・宇宙機においては，この環境に耐えるハードウェアが必要

宇宙環境のイメージ

■ 宇宙開発におけるソフトウェア

・ソフトウェアにおいては，過酷な宇宙環境でハードウェアが確実に動作する信頼性と安全性が求められる
・この信頼性と安全性には，耐障害性や安全機能(緊急検知やアプリケーション監視)，リアルタイム性などが含まれるが，これらの要件は，自動車，航空機，産業用ロボット，災害救助ロボット，医療ロボット，サービスロボットなど，地上の技術にも求められるものである
・ハードウェアと異なり，宇宙機のソフトウェアは，本質的には地上ロボットと同じ課題を抱えている

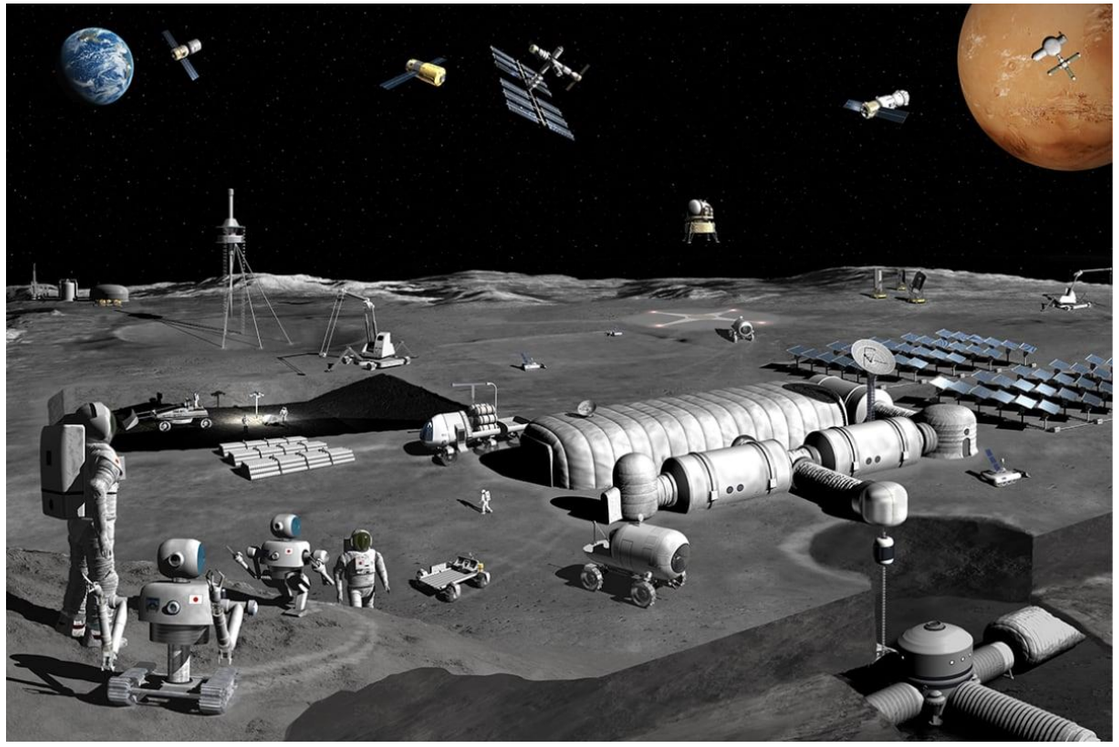
重要

地上でも宇宙機並みの信頼性・安全性が求められる要求に応えられる

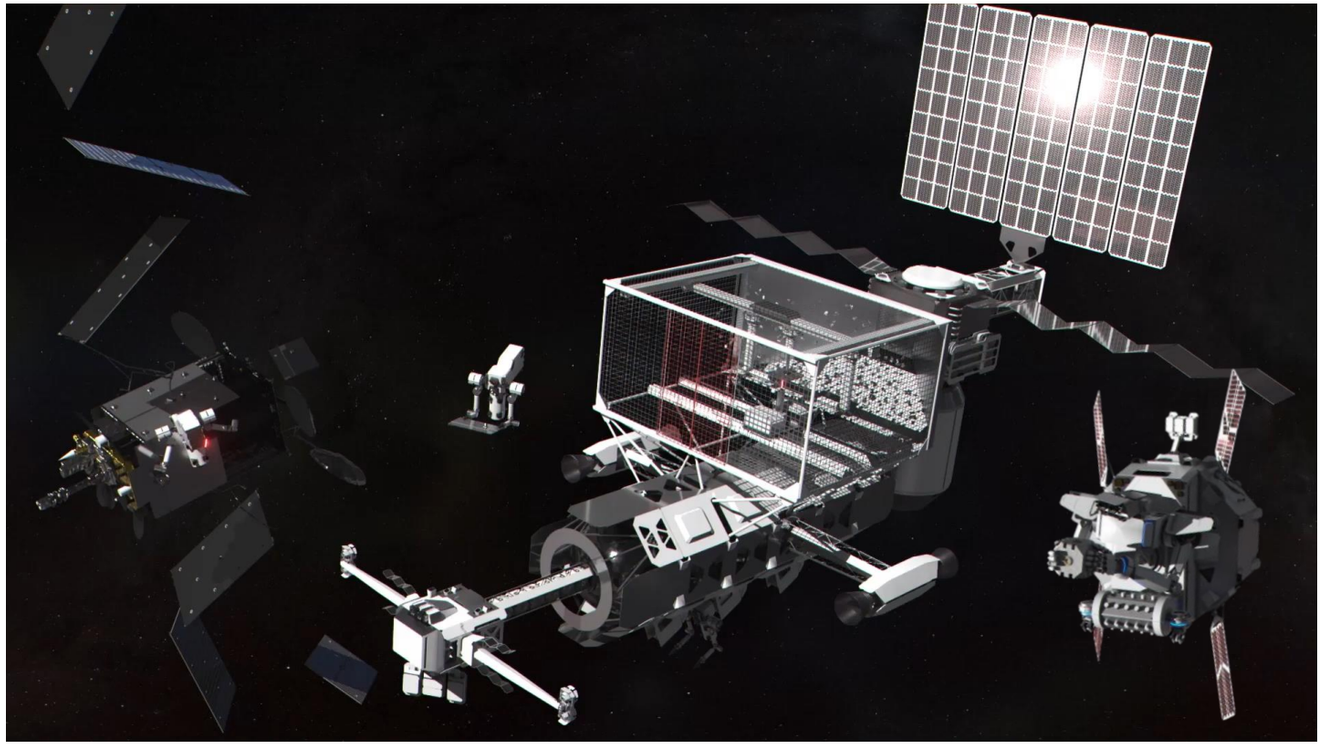
宇宙開発におけるロボット技術（1/2）

■ 将来の宇宙ミッション

・JAXAでは，月惑星探査(持続可能な月面活動，火星の有人探査など)や，軌道上サービス(スペースデブリの捕獲・除去，人工衛星の保守・修理，宇宙構造物の建設など)の実現に向けた取り組みを行っている
・大規模ミッションにおいては，自動化・自律化を目的としたロボット技術の活用が重要視される
・宇宙開発が国の事業から商業へと広がる中，より一層，開発の効率化と実行力の向上が求められる



月惑星探査の将来像

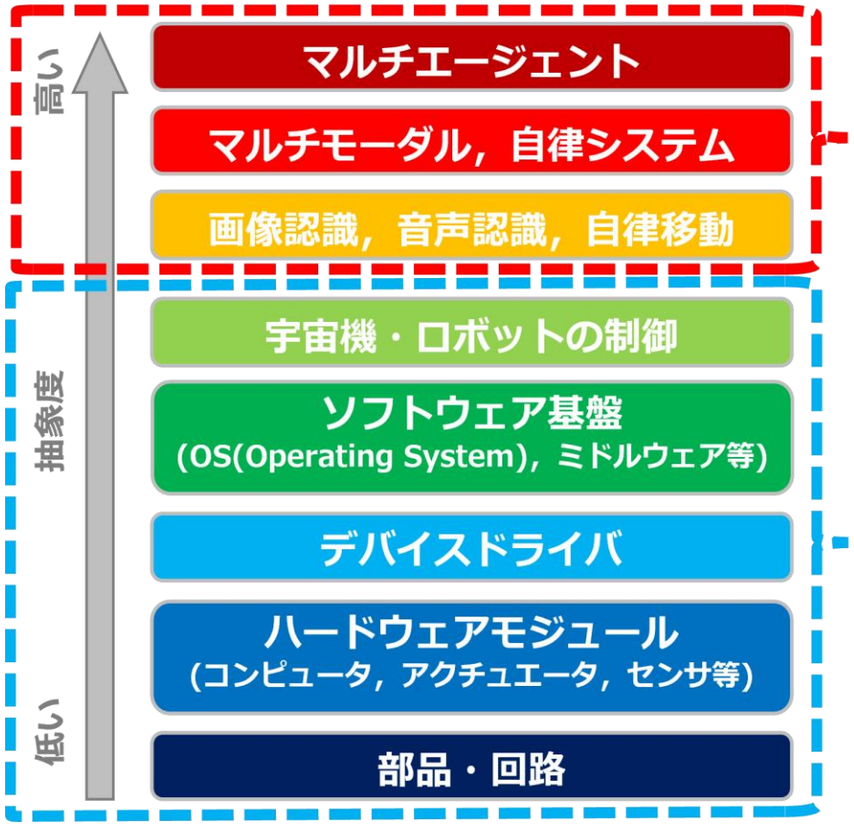


軌道上サービスの将来像

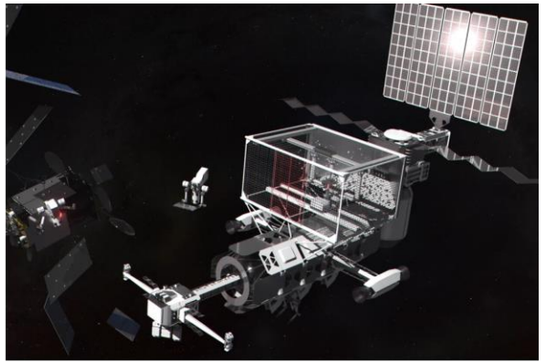
宇宙開発におけるロボット技術（2/2）

■ 宇宙開発における課題

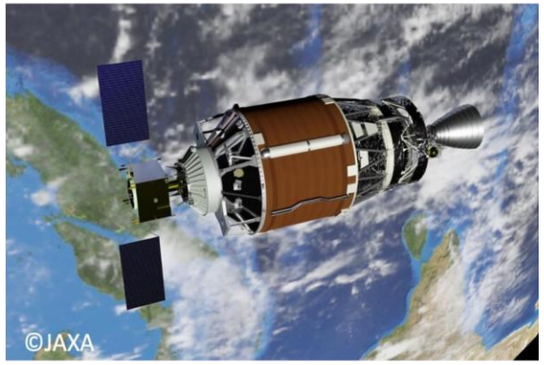
・現在の宇宙開発は，過酷な環境下で確実にミッションを成功させるために，高い信頼性と安全性が求められ，各ミッションに対してオーダーメイドでの組み込み開発が主流
・ミッションの大規模化でシステムの複雑化が進み，開発に膨大な期間とコストがかかる
・これに対して，既存技術の汎用化・抽象化を行い，それを基盤とした新しい技術を積み重ねていくことが有効
・この点でROSが優れている(豊富なライブラリ，ツール，世界規模のコミュニティ等)



宇宙機・宇宙ロボットにおける技術の抽象化



将来の軌道上サービスのイメージ



現在の軌道上サービスのイメージ

目的

ROSを宇宙機・宇宙ロボットに組み込みたい！！

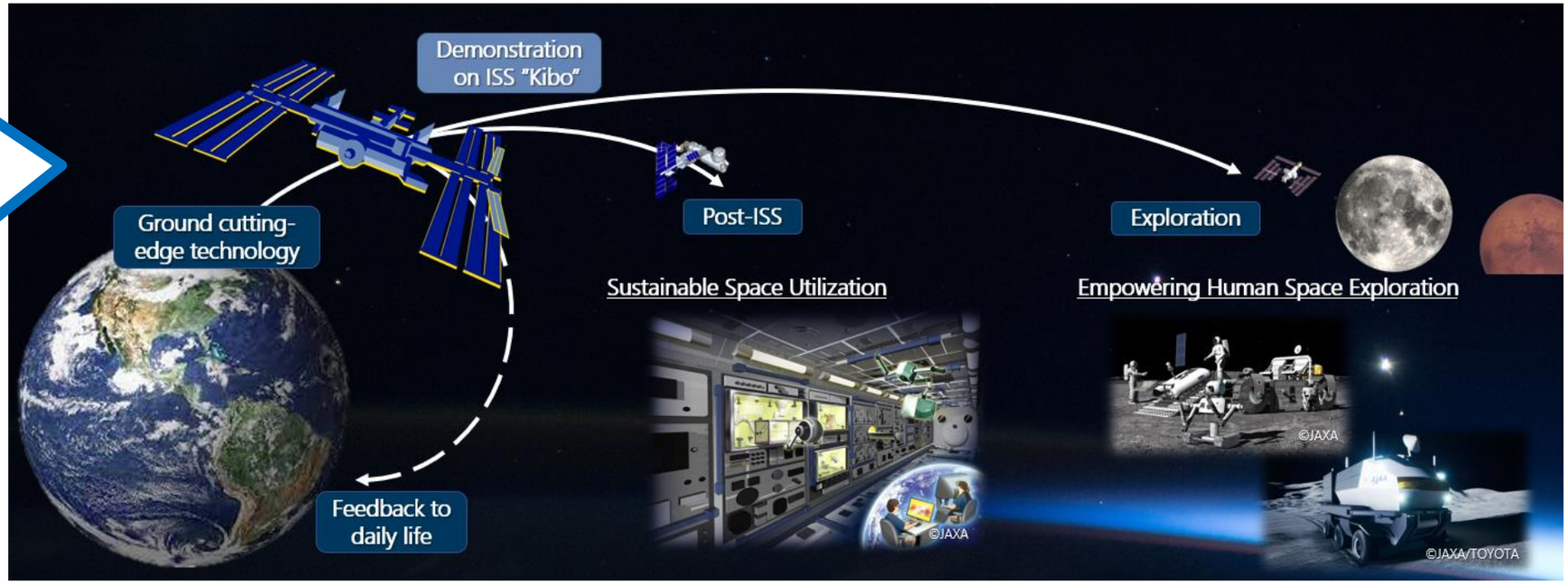
宇宙開発でのROSの活用事例(1/6)

■ 宇宙ロボットの宇宙飛行士支援

・JAXA では，宇宙の中でも比較的地上の環境に近いISS(国際宇宙ステーション) にて，ROSを活用している
・ここでは，ISS船内に滞在する宇宙飛行士の作業の自動化・自律化(宇宙飛行士の支援，無人での運用/準備)を目的としたIVR(船内ロボット)の開発している
・ISSは，人が居住できる環境であり，真空や放射線といった宇宙特有の環境問題が少ないことや，宇宙飛行士によるサポートが受けられることから，最新技術の実証の場として優れている

※ 別途，人が居住しているISS での高い安全要求を満たすことや，微小重力下での動作を考慮する必要がある

ROS



宇宙開発でのROSの活用事例(2/6)

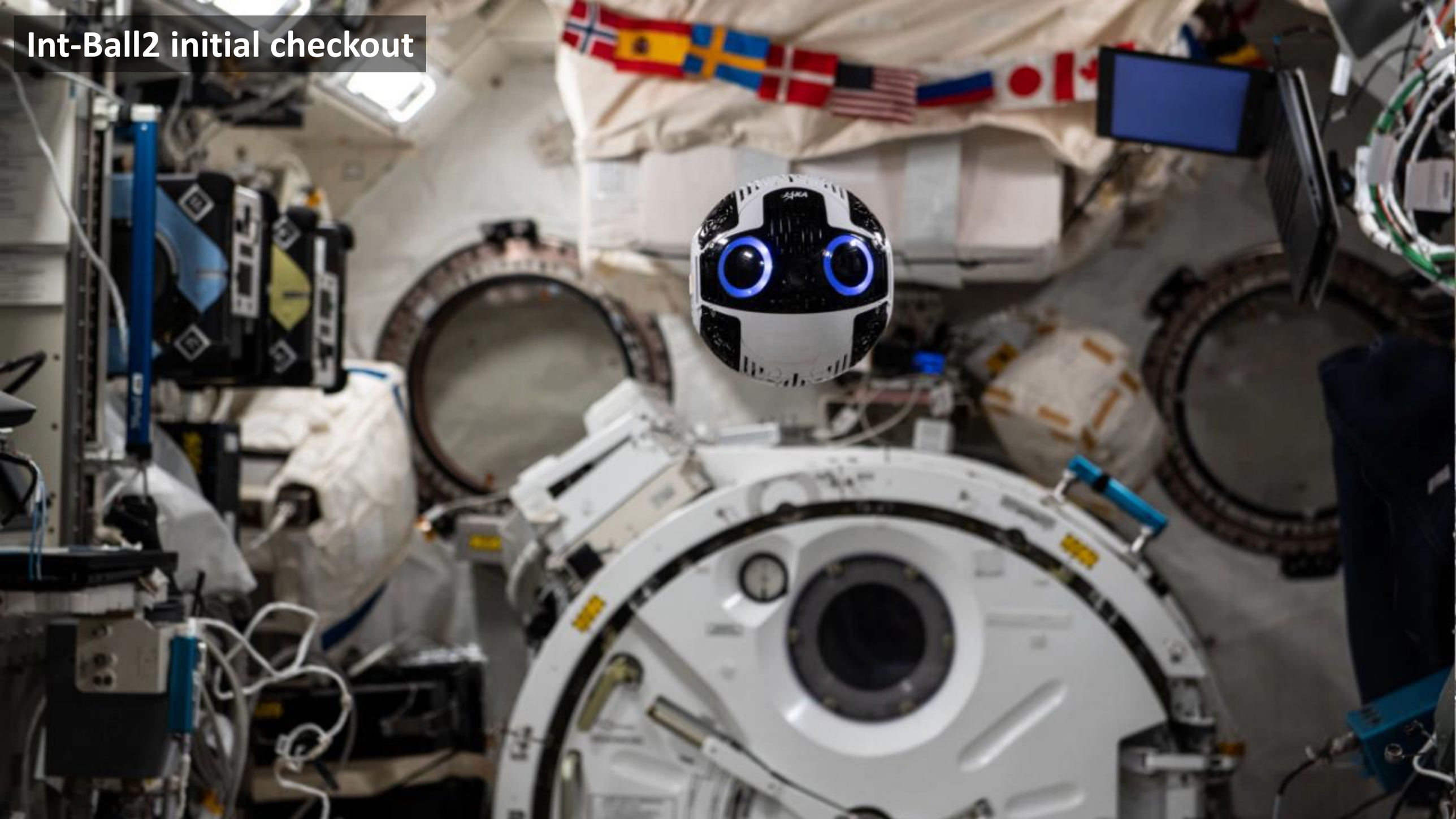
■ Int-Ball2の研究開発

・IVR のFF(Free Flyer)であるInt-Ball2 は，撮影機器のセットアップの自律化や，宇宙ロボットの技術実証プラットフォームを目指して開発されたロボットである (2号機)

・技術実証プラットフォームとして，JAXA 内外のユーザに使ってもらいやすいソフトウェアアーキテクチャにするために，ROS を導入している

Seiko P. Yamaguchi et al., 2024 iSpaRo, WedS10T1.4.

Int-Ball2 initial checkout

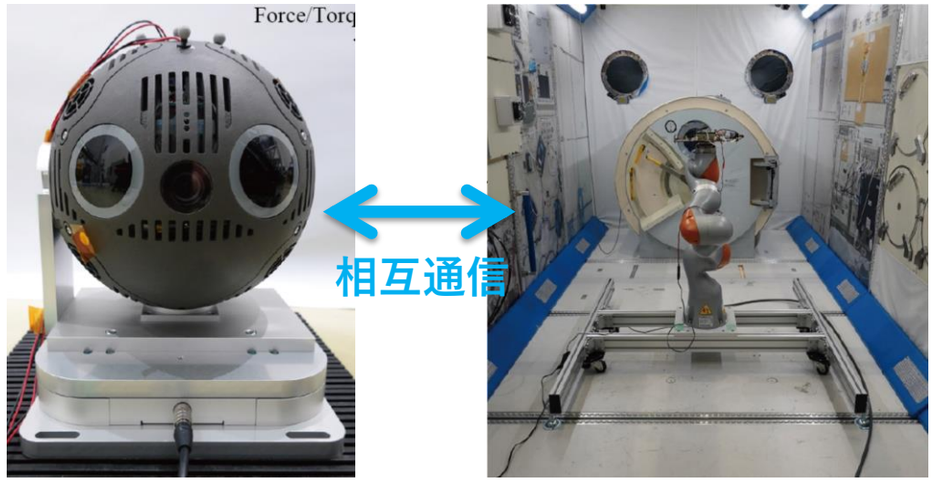


宇宙開発でのROSの活用事例(3/6)

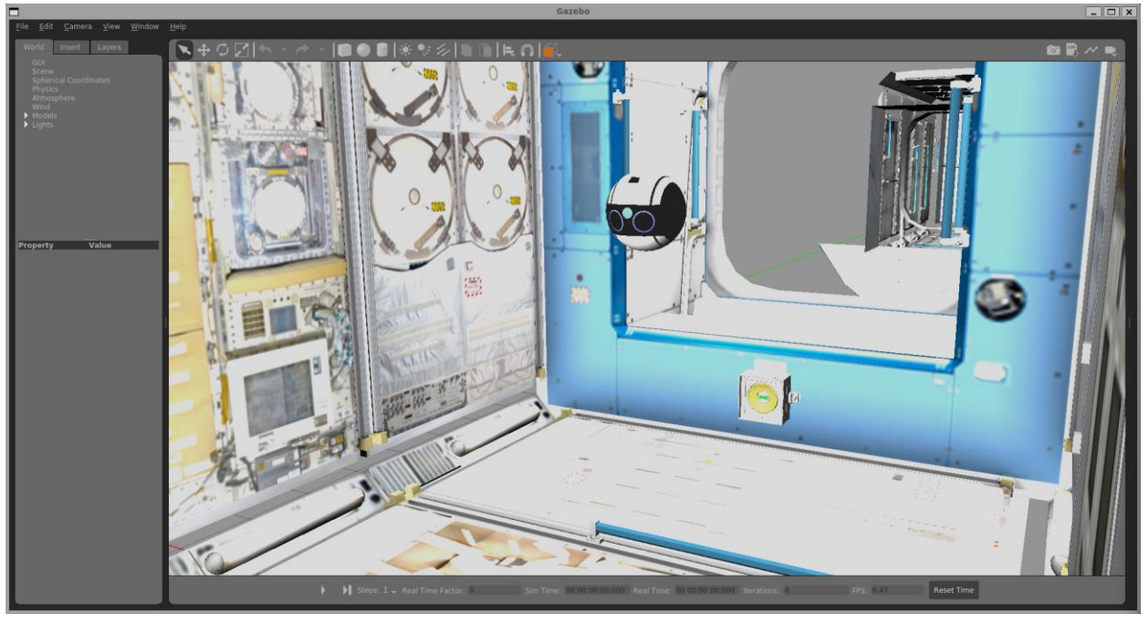
■ Int-Ball2の地上試験

- 地上で無重力空間を再現するために，様々な地上試験を実施
- 空気浮上装置を用いた2次元平面上での試験，マニピュレータを用いた3次元空間でのHILS試験，Gazeboシミュレータを用いた3次元空間上での試験を行っている

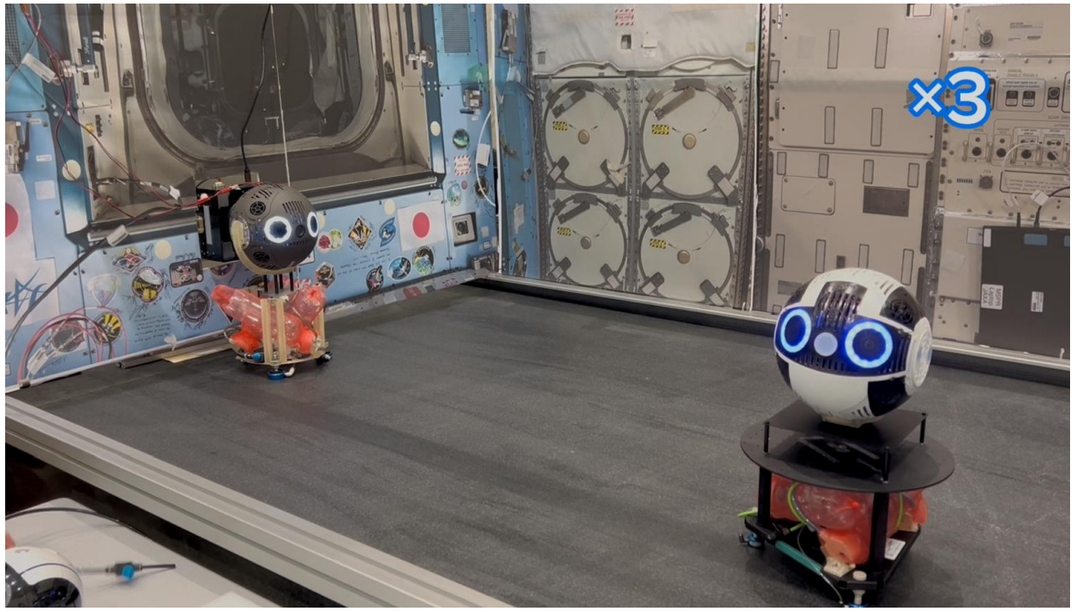
Int-Ball2のGazeboはJAXA GitHub上で公開を検討中
https://github.com/jaxa



マニピュレータを用いたHILS試験
D. Hirano et al., 2023 ICRA, doi:10.1109/ICRA48891.2023.10161499.



Int-Ball2のGazeboシミュレータ



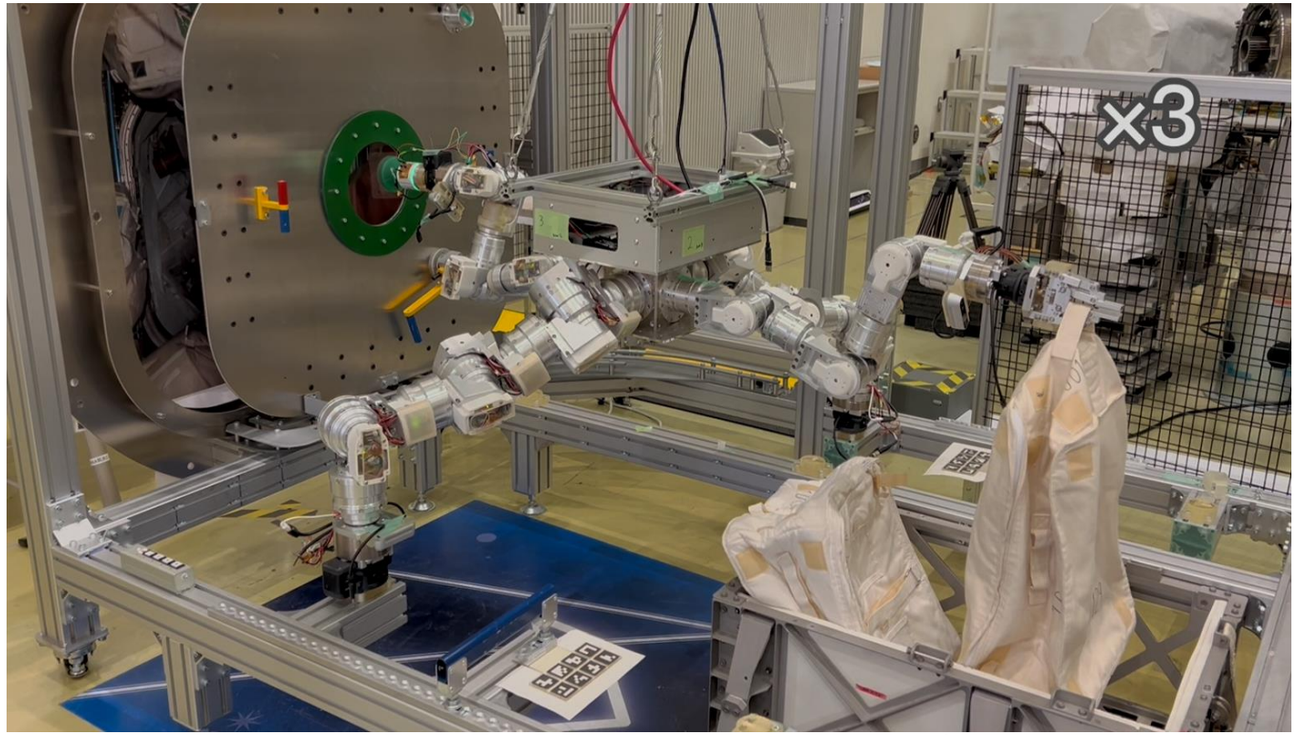
空気浮上装置を用いた試験

宇宙開発でのROSの活用事例(4/6)

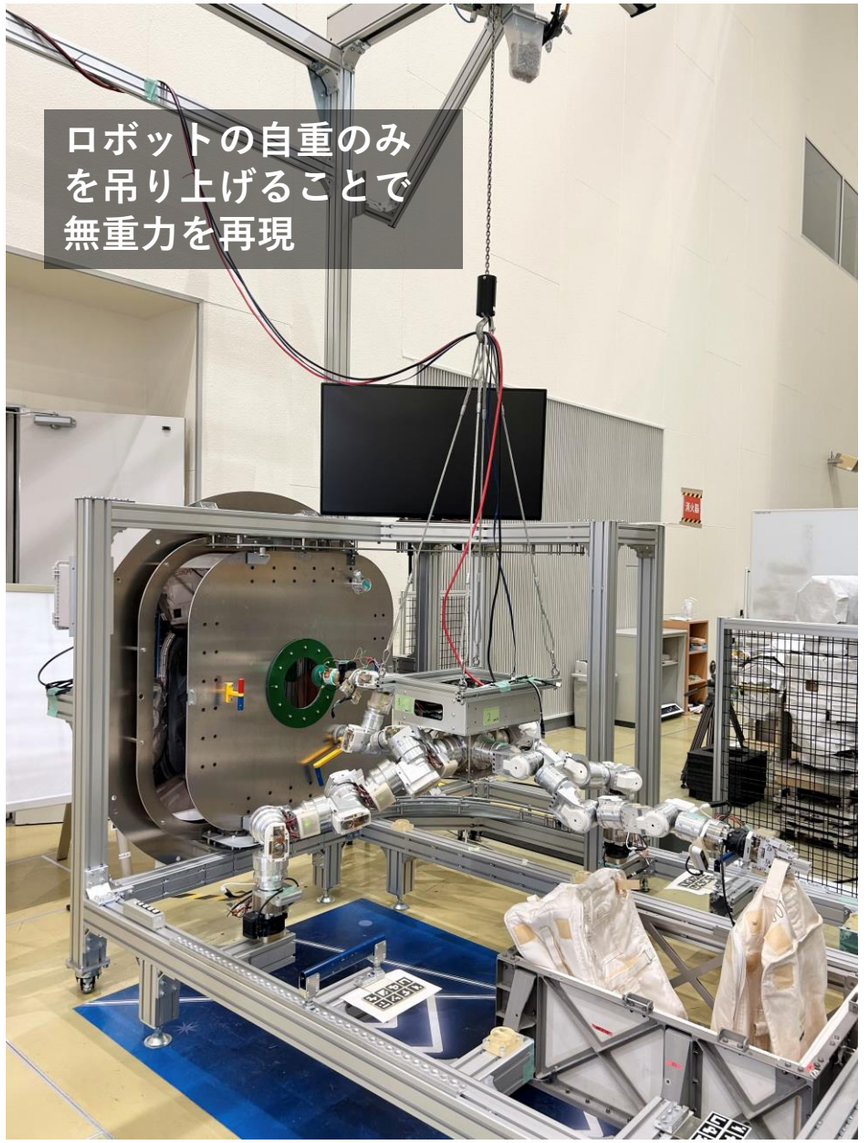
■ PORTRSの研究開発

・IVR のMM(Mobile Manipulator)であるPORTRSは，ISS や将来の月周回有人拠点(Gateway)に届けられた荷物運搬等の自律化を目指して開発されたロボットである

・複数のマニピュレータでの複雑な作業に，ROS パッケージであるMoveItを用いたマニピュレーションの開発を行っている

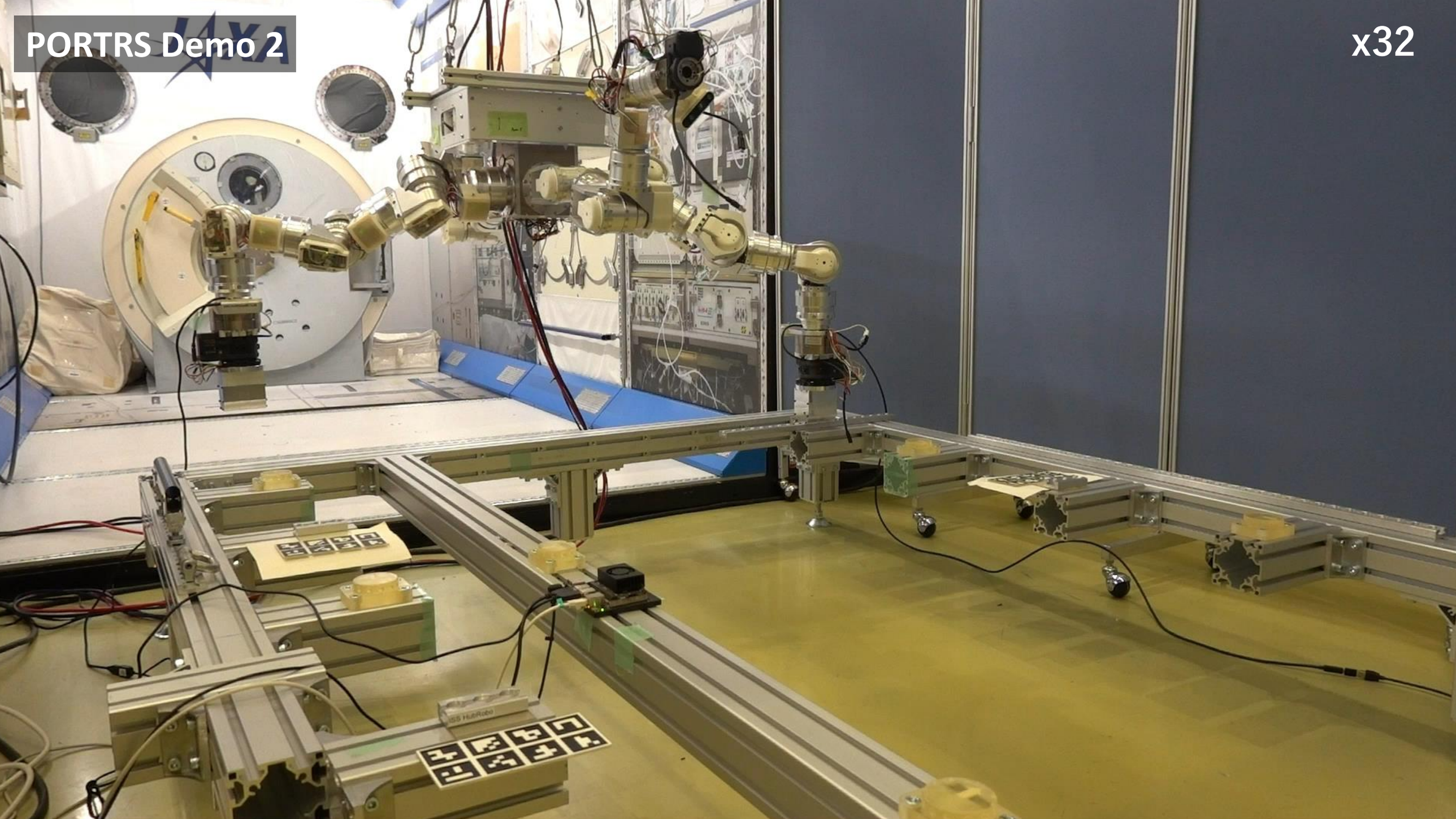


PORTRSのマニピュレーションデモ



PORTRSの試験設備

S. P. Yamaguchi *et al.*, *2024 AERO*, doi: 10.1109/AERO58975.2024.10521315.

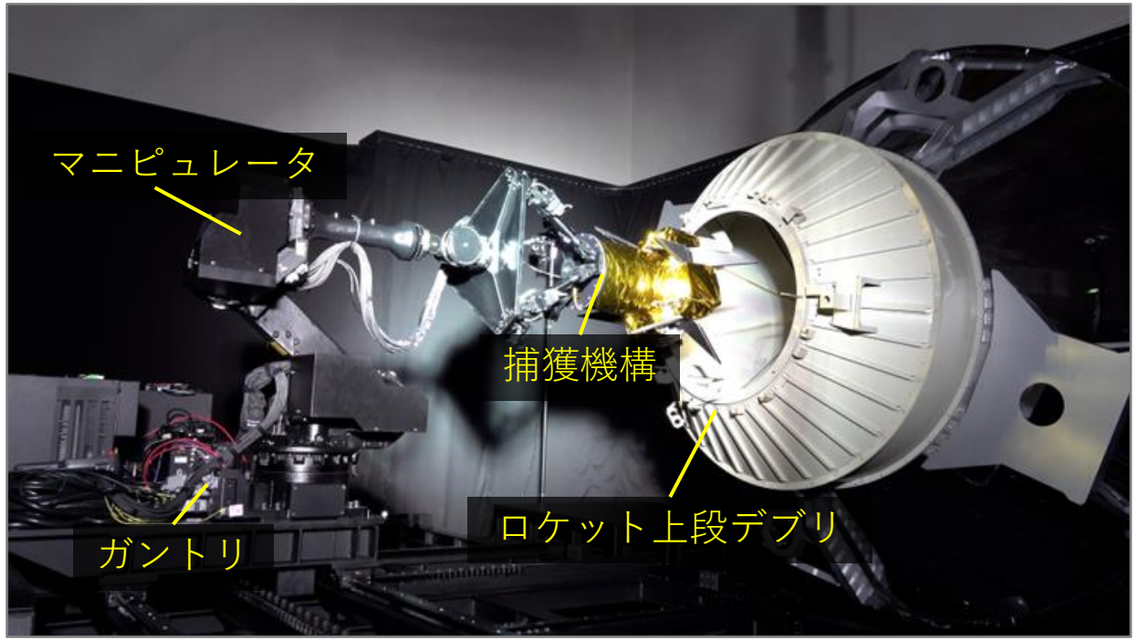

PORTRS Demo 2
x32

宇宙開発でのROSの活用事例(5/6)

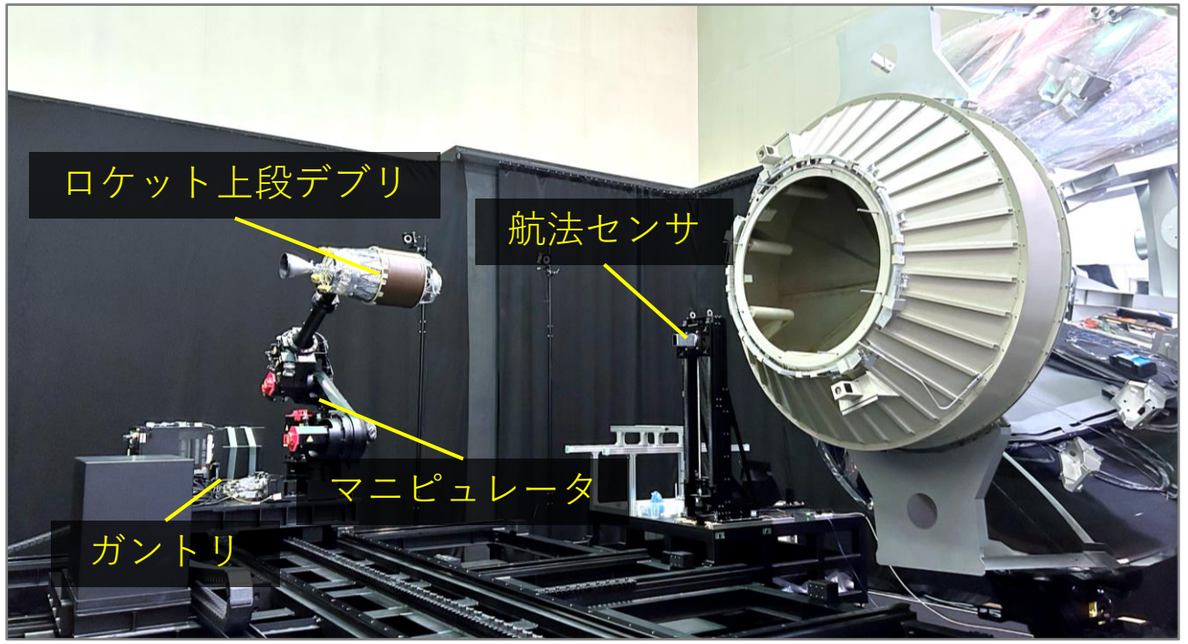
■ 軌道上サービス技術実証プラットフォーム (SATDyn) の研究開発

※ 様々なケースでの統合試験が可能

・宇宙ミッションの確実な成功のためには，地上で特殊な宇宙環境をどれだけ再現して試験検証できるかが重要
・高難度で世界でも確立した地上試験技術が存在しない軌道上サービス(デブリ除去)の実現に向けSATDynを開発
・宇宙機やデブリのダイナミクスをリアルタイムにシミュレーションし，その位置・姿勢をマニピュレータとガントリによって再現することで，軌道上サービスの統合試験・検証が可能
・ROS の可視化ツールであるRViz やロギング機能であるrosbag などによりデバッグが容易であることや，各機能のモジュール化により新規機能の開発や削除がしやすいことを，活かして開発を進めている



SATDynのマニピュレータに捕獲機構を取り付けた試験

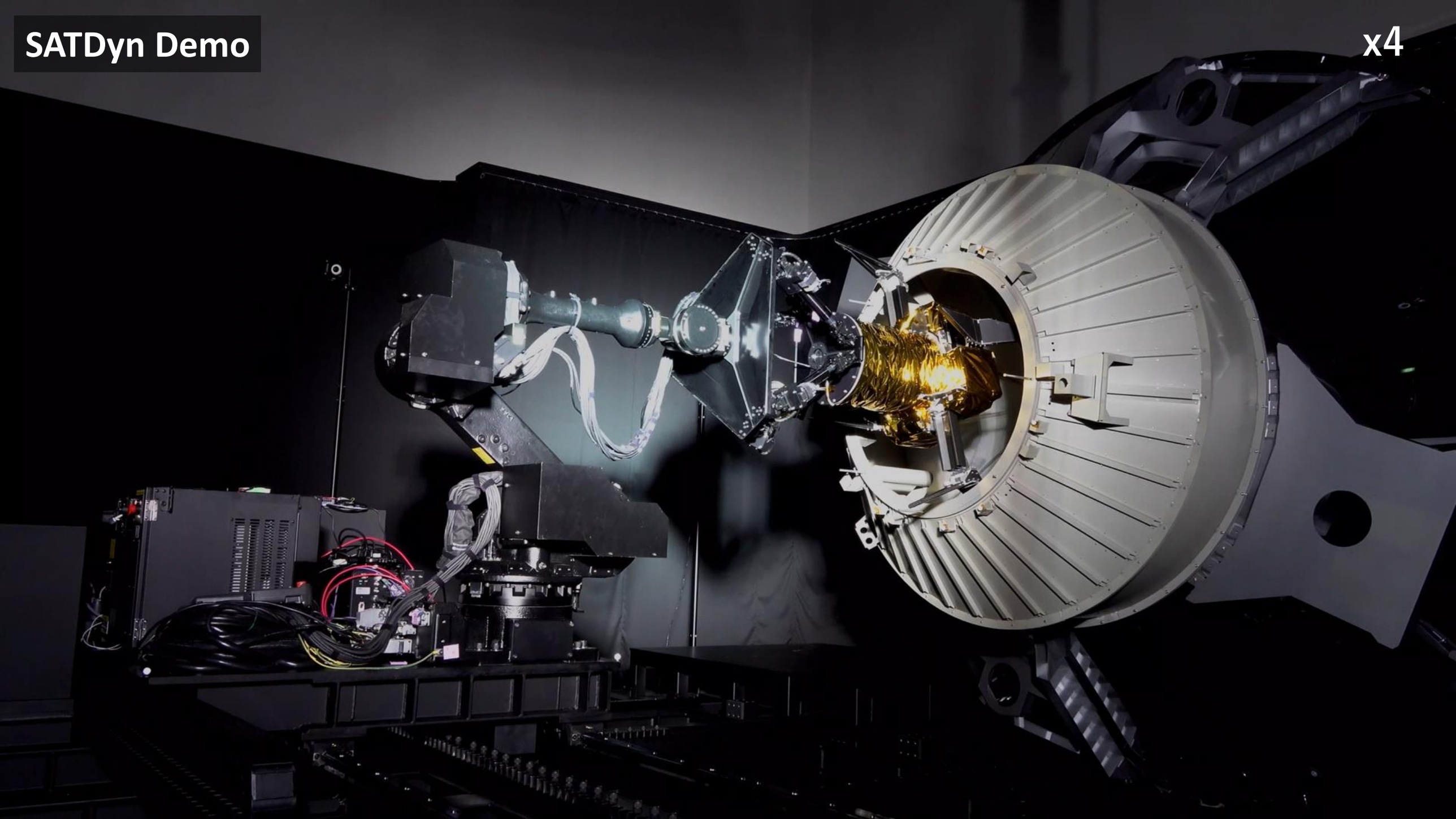


SATDynのマニピュレータにロケット上段を取り付けた試験

H. Okamoto et al., *2022 AERO*, doi: 10.1109/AERO53065.2022.9843677.

SATDyn Demo

x4



宇宙開発でのROSの活用事例(6/6)

SATDynでの地上試験を活用した宇宙ミッション

■ 商業デブリ除去実証 (CRD2 : Commercial Removal of Debris Demonstration)

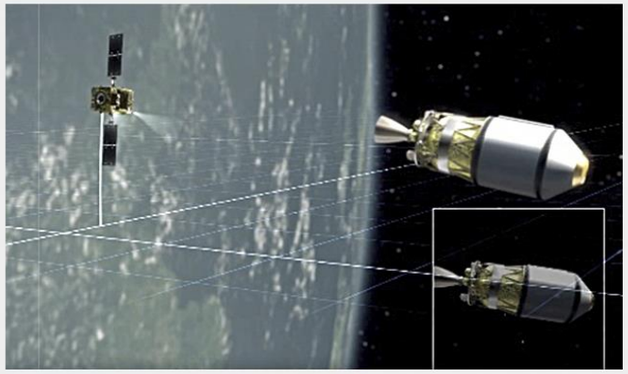
・深刻化するスペースデブリ問題に対して，除去効果が大きく，技術的に高度な大型デブリの除去を世界で初めて実施 (Phase Ⅰ と Ⅱ の2段階で実施).

・スペースデブリ対策に関する世界的な議論の先導と，市場の開拓を目指す.

2023年度打ち上げ予定

Phase Ⅰ

キー技術実証



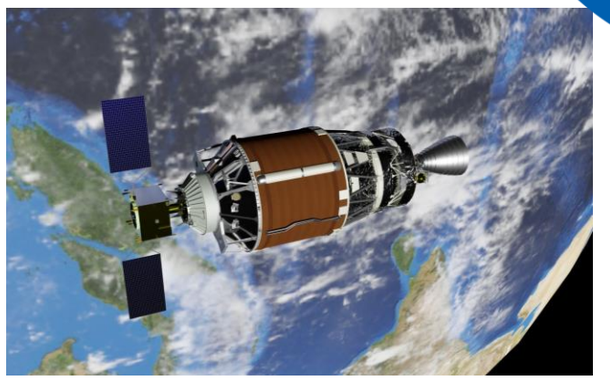
・非協力的ターゲットへのランデブ、近傍制御、映像の取得

©Astroscale Japan Inc.

2026年度以降打ち上げ

Phase Ⅱ

デブリ除去実証



・非協力的ターゲットへのランデブ、近傍制御、映像の取得

・大型デブリ（ロケット上段）の除去

©JAXA

CRD2プロジェクト(Phase Ⅰ & Ⅱ) 商業デブリ除去実証(CRD2) 公式Webページ

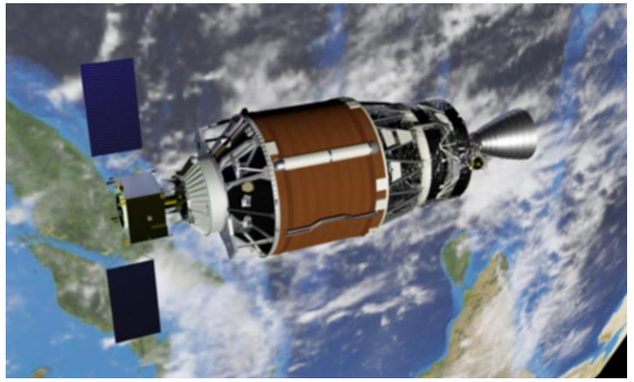
Space ROSの開発(1/4)

■ 宇宙開発でのROSの活用における課題

・ROS のメリットを宇宙開発に活かすことで,
開発期間やコストの大幅な削減に成功している
・しかし，宇宙ロボットでのROSの採用例は,
宇宙飛行士が居住・サポートできるISS 船内のみ
・ISS 船外の過酷な宇宙環境でROS を用いるには,
高い信頼性と安全性が求められるが，地上のROSは
これらの考慮が不足しており，直接の適用は困難
・ROSは，ロボットの制御，マニピュレーション,
自律移動等に特化したアプリケーションが多く,
信頼性と安全性に関するロボットの健康状態(通信や
電力等)を維持・管理する機能は別途開発が必要

変形型月面ロボット

有人与圧ローバー



デブリ除去衛星

火星衛星探査機

過酷な宇宙環境でのロボットミッション例

目的

宇宙機・宇宙ロボット向けのROSであるSpace ROSの開発！

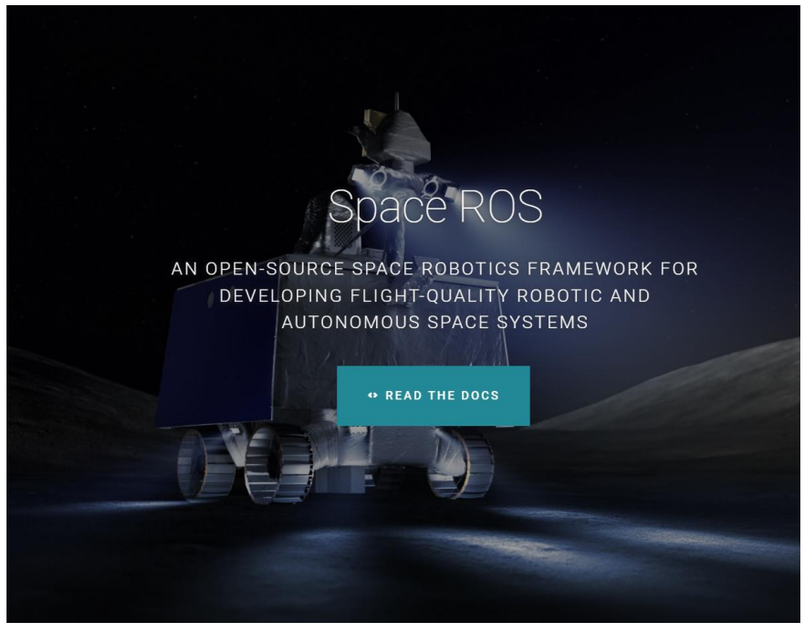
地上でも宇宙機並みの信頼性・安全性が求められる要求にも応えられる！

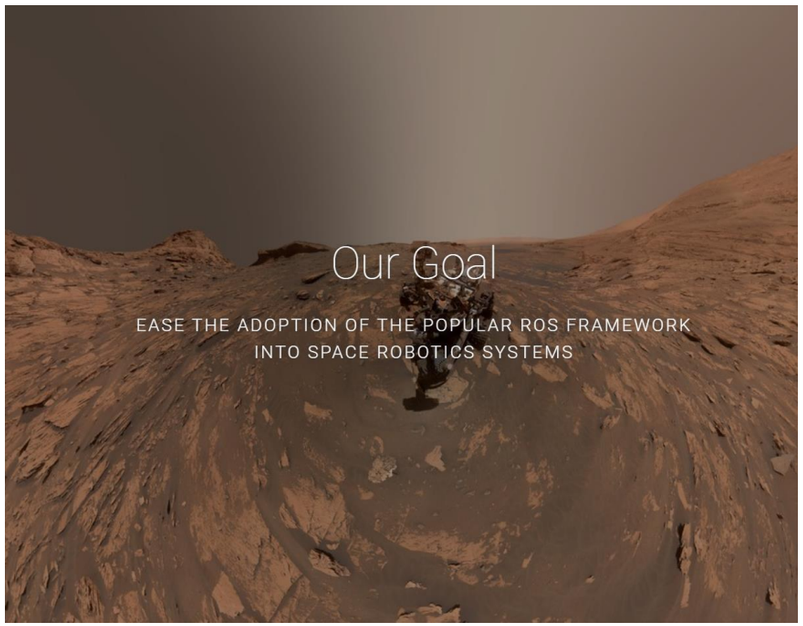
Space ROSの開発(2/4)

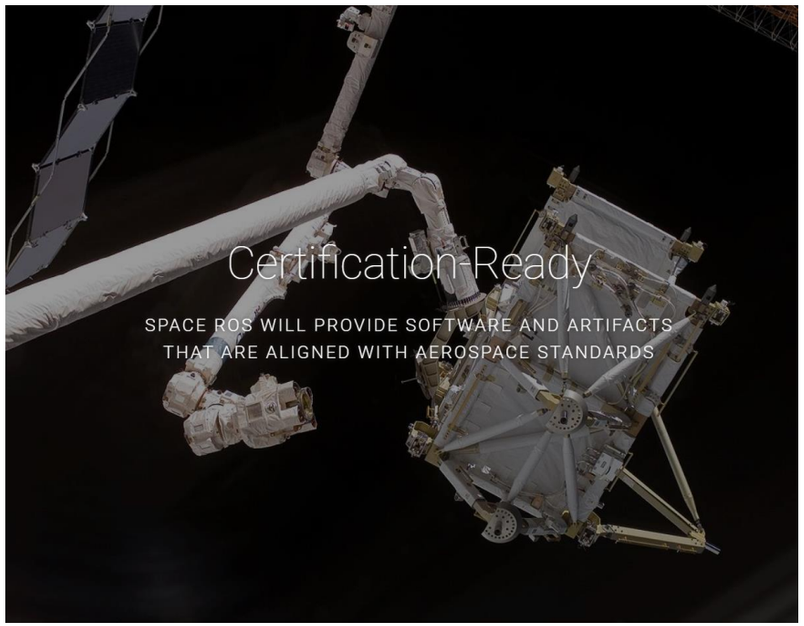
■ Space ROS

・Space ROSは，地上ロボット向けに開発されたROSを
宇宙機・宇宙ロボット向けに応用したOSSプラットフォーム
(Open Robotics，NASA，PickNik社が開発を主導)

・ROSを宇宙環境に導入する際の課題に対して，高い信頼性と安全性を実現する航空宇宙の規格に準拠した
プラットフォームを構築することで，宇宙機・宇宙ロボットでのROSリソースや，ROSと同様の柔軟性と
拡張性を活かしたソリューションをOSS として提供することを目指す







Probe, A. et al., *2023 AIAA*, doi:10.2514/6.2023-2709.　　Space ROS公式Webページ

Space ROSの開発(3/4)

■ Space ROSとROS 2との違い

・Space ROSは，ROS 2を基盤に宇宙ミッションに必要なリアルタイム機能や信頼性の強化を目指している

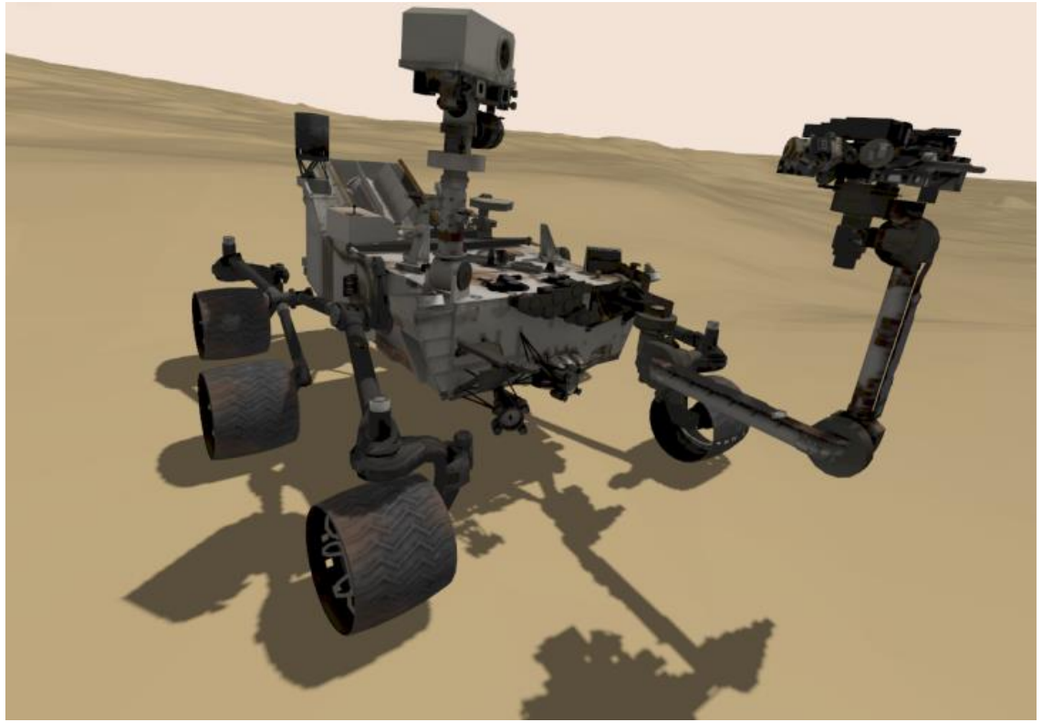
項目	説明
ビルドとインフラ	ROS 2 と異なるSpace ROS 独自のWeb サイトやドキュメントサイト，GitHub，Docker イメージ，CI/CD ツールであるEarthly などを通じた，情報提供とビルドをサポート
コード解析ツール	NASA が開発した静的解析ツールCobra, IKOSなどをSpace ROS に統合した，コード品質の向上と認証支援. 解析結果はSARIF 形式で出力されるため，ダッシュボードでの可視化が可能
要件ツールとプロセス	要件管理ツールDoorstopとNASAが開発した要件抽出ツールFRET を使用し，要件の管理やトレーサビリティ，形式化をサポート．航空宇宙の規格を満たす品質管理が可能
宇宙特有の機能	C++ メモリアロケータやETS，Mars Rover やCanadarm のシミュレーションを含むデモアプリケーションを提供．宇宙で使用されているRTOS(RTEMS)との互換性や，ROS 2 とNASA のOSSフライトソフトウェアcFS との接続をサポート

Space ROSの開発(4/4)

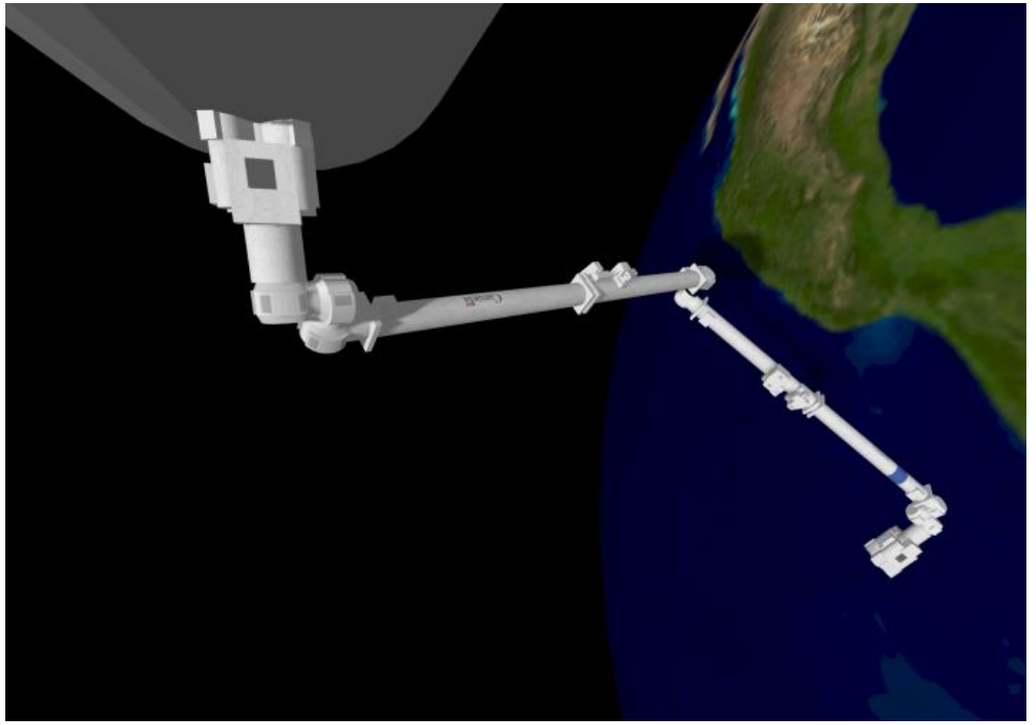
■ Space ROS開発の現状

・現時点では，Mars Rover やCanadarm のシミュレーションなどが利用可能になっているものの，
未だ開発段階であり，実際の宇宙機・宇宙ロボットで使用できるソフトウェアは存在していない

・実際の宇宙機・宇宙ロボットで動作することを考えると，本質的には地上でも考えられる課題である
信頼性と安全性の高いミドルウェアを提供できるかが焦点となる



Mars RoverのGazeboシミュレータ



Canada ArmのGazeboシミュレータ

Space ROS公式Webページ

JAXAでのSpace ROSの開発(1/4)

■ RACS2 (cFSとROSを繋ぐインターフェースソフトウェア)

- JAXAでは，Space ROS登場以前から，宇宙機でROSを動作させる方法についての検討を行ってきた
- そのために，NASAが開発したOSSのフライトソフトウェアであるcFSとROS を接続するRACS(ROS and cFS System)を考えた

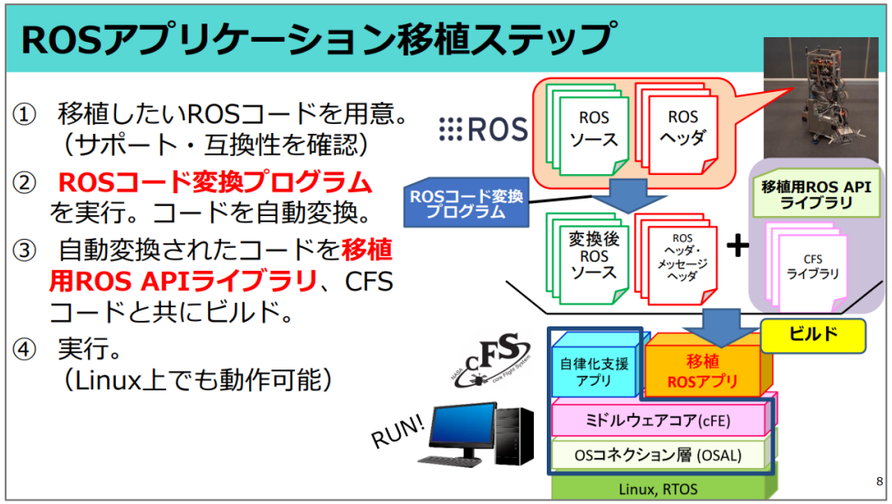
※ 2018年のROSCon JPにて発表

cFS(core Flight System)は，宇宙機の制御技術や通信技術，運用技術等を汎用化・抽象化したアプリケーションで構成されており，用途に応じた機能の追加や再利用が容易．NASAの様々なミッションへの搭載実績もある

RACS を用いたcFSとROSの双方向通信を実現することで，

高い信頼性と安全性を確保しつつ，ROSの豊富なライブラリ，ツール，パッケージ，および世界的なエコシステムを活用したソフトウェアプラットフォームを構築する

※ 2018年のROSCon JPでの発表からRACSのROS 2対応版も開発
(ROS 2は組み込みシステム(非Linux OS)で動作するため，従来の変換機能は不要)



2018年 ROSCon JPの発表 斎藤 達彦ら, ROSCon JP 2018.

H. Kato et al.,2021 AERO, doi: 10.1109/AERO50100.2021.9438288.

H. Kato et al.,2023 FSW, pp. 1-8.

JAXAでのSpace ROSの開発(2/4)

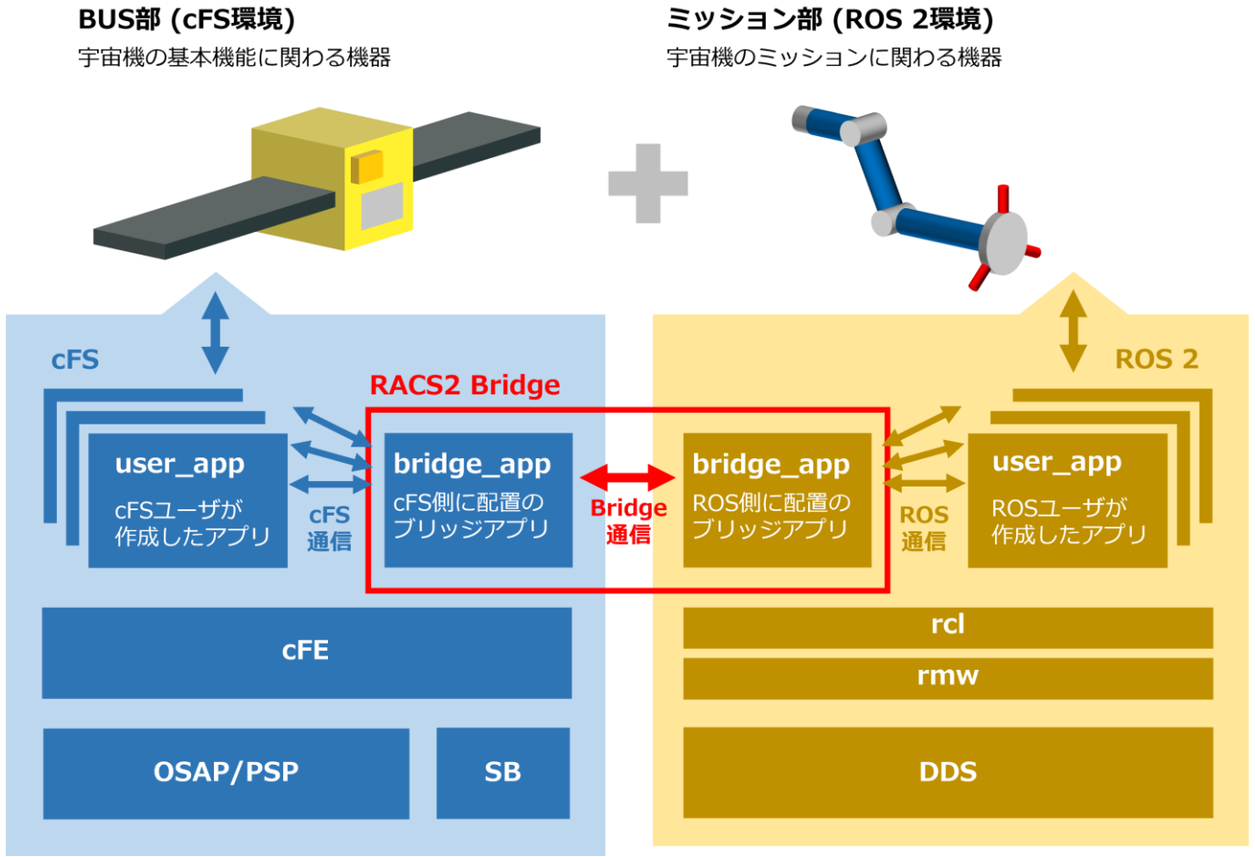
■ 宇宙機・宇宙ロボットの構成

・宇宙機は大きく分けてBUS部とミッション部で構成される
・BUS部は，C&DH(コマンド系)，EPS(電力系)，ACS(姿勢制御系)，推進系，TCS(熱制御系)等の宇宙機の基本機能や健康状態(通信や電力など)の維持・管理に関わる機器を備える
・ミッション部は，宇宙機がミッションを遂行するにあたって必要な機器を備える

■ RACS2の構成

・BUS部は宇宙機への搭載実績もあり，高い信頼性と安全性を持つcFSを用いて実装
・ミッション部はロボットにおけるソリューションが豊富なROS を用いて実装
・cFS 環境のBUS 部とROS環境のミッション部のそれぞれでブリッジ用のRACS2のアプリケーションを用いる
・cFS とROSを接続することで，信頼性と安全性を維持しつつ，ROSの利点を活かしたソリューションを実現する

RACS2はJAXA GitHub上で公開済み（更新も予定）
https://github.com/jaxa



RACS2 の構成イメージ

JAXAでのSpace ROSの開発(3/4)

■ RACS2のデモンストレーション

・Space ROSのデモであるMars RoverのGazeboシミュレータ上と，ROSの実機ロボットであるTurtleBot3 Waffle Pi (+ OpenMANIPULATOR-X) 上でRACS2の動作デモを実施

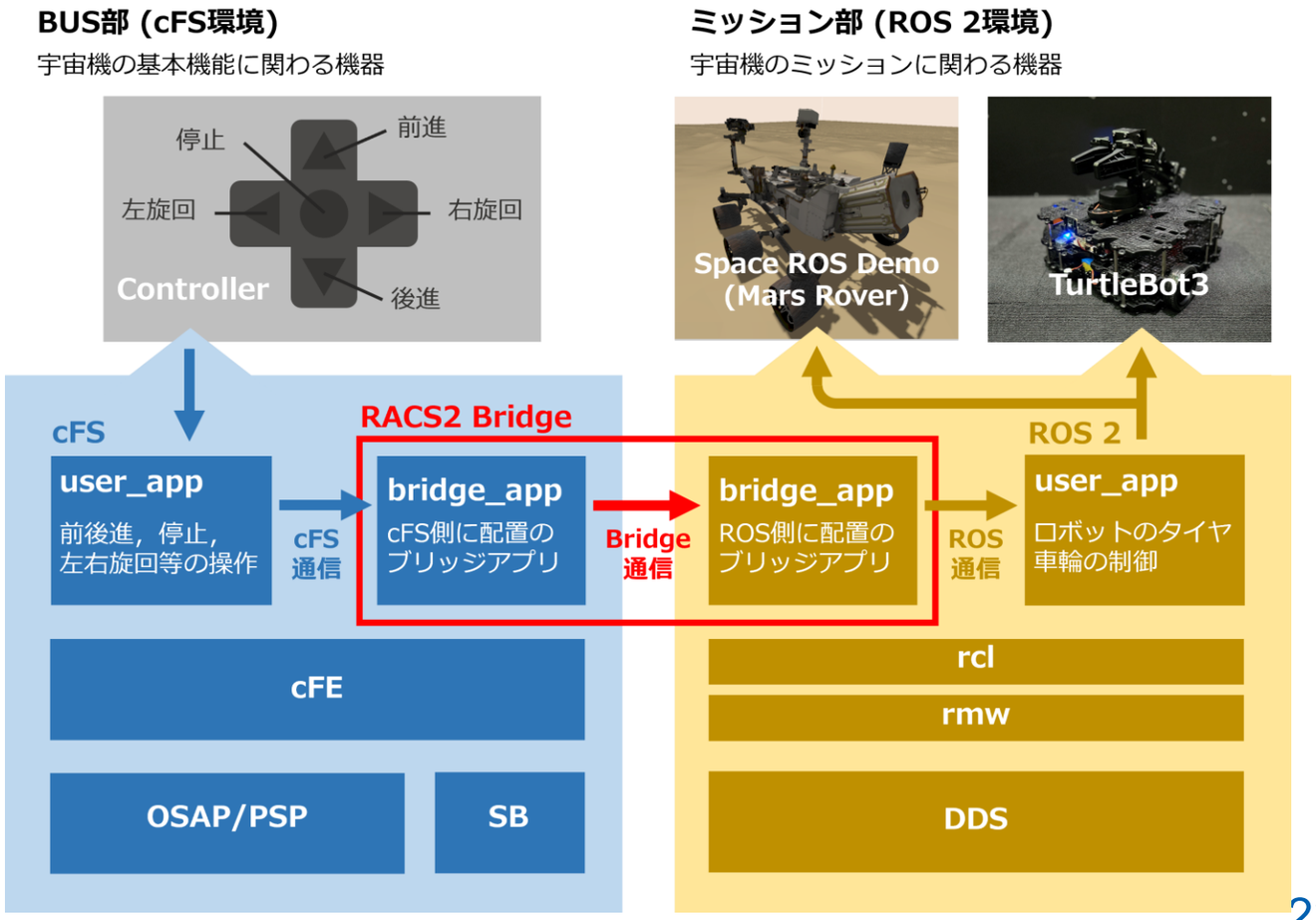
■ RACS2デモンストレーションの構成

・どちらも宇宙機・宇宙ロボットのBUS 部とミッション部に分けて動作するソフトウェアアーキテクチャを想定

・デモでは，cFS(BUS 部)から，ROS 2(ミッション部) に対して，操作コマンドを送ることで，Mars RoverまたはTurtleBot3の移動を行う

※ デモにおいては，BUS部とミッション部はシステムとしては異なるものの，同一のコンピュータ(Ubuntu)上で動作させている

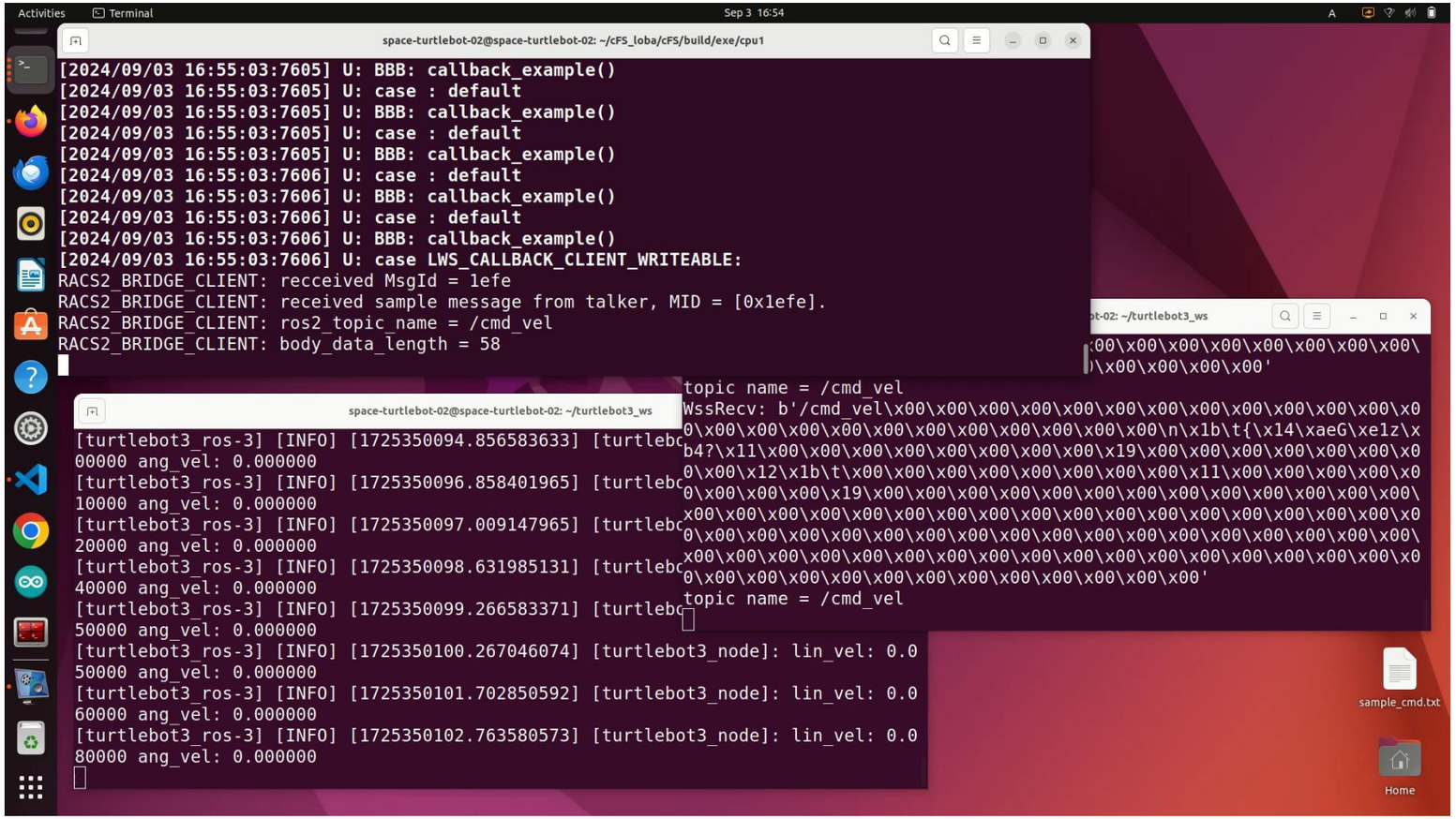
RACS2デモはJAXA GitHub上で公開予定
https://github.com/jaxa



RACS2デモの構成イメージ

JAXAでのSpace ROSの開発(4/4)

本デモは，Mars RoverまたはTurtleBot3が動くだけの地味なデモンストレーション．．．

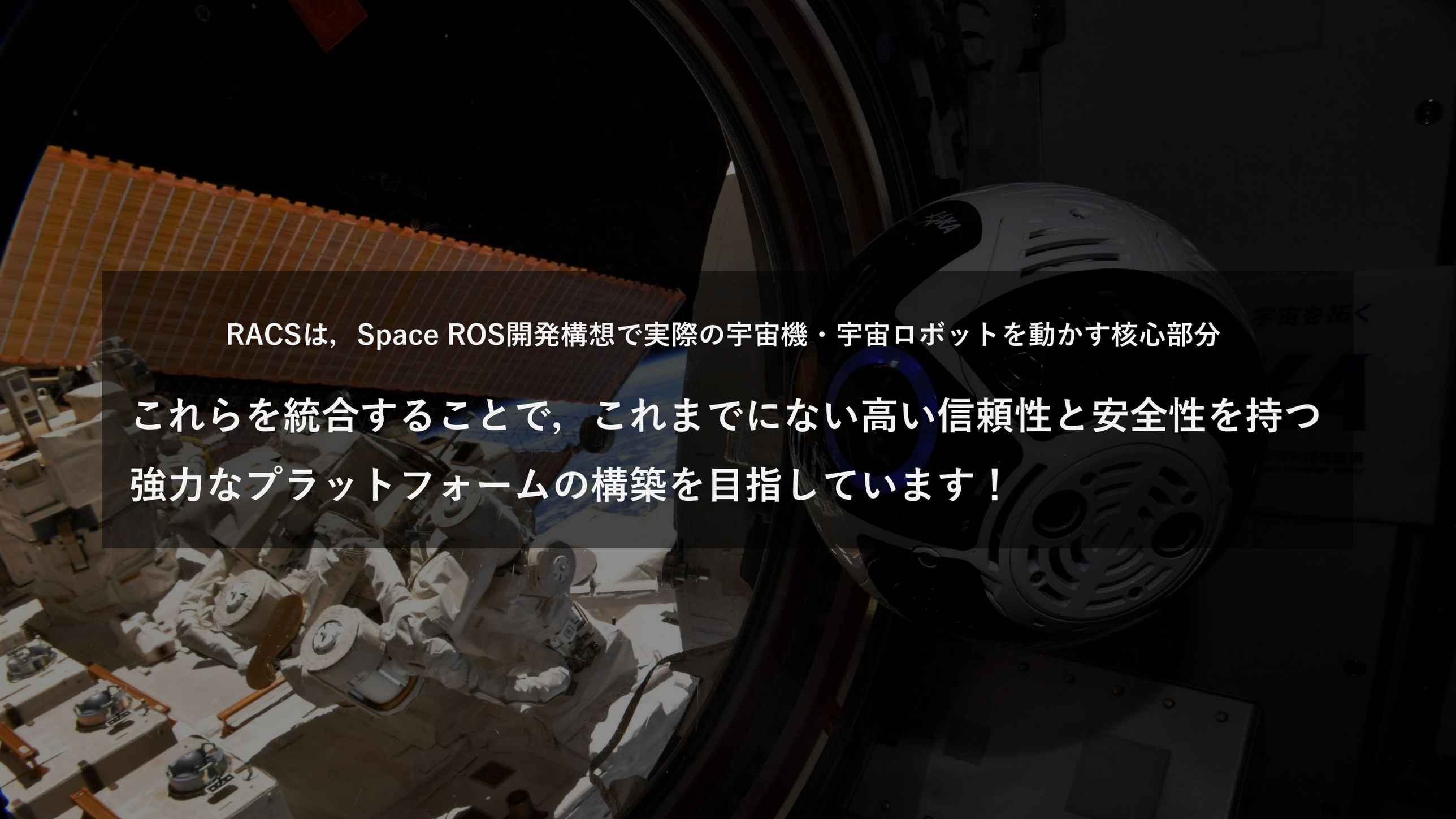


RACS2デモの動作イメージ

ただし，

実際の宇宙機・宇宙ロボットで動くソフトウェア構成で動いている！！

家庭でも簡単に宇宙機・宇宙ロボットに搭載可能なソフトウェア開発が可能



RACSは，Space ROS開発構想で実際の宇宙機・宇宙ロボットを動かす核心部分

これらを統合することで，これまでにない高い信頼性と安全性を持つ強力なプラットフォームの構築を目指しています！

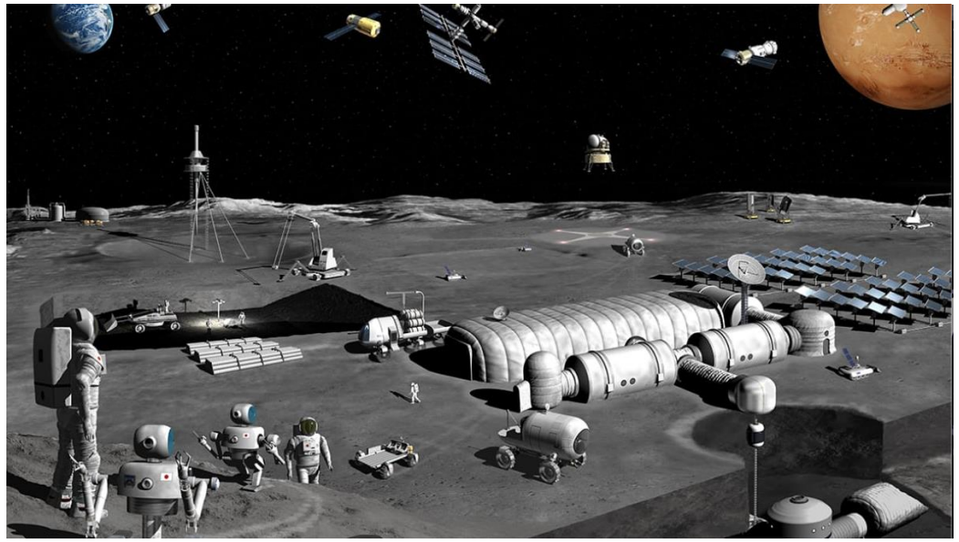
ま と め

■ まとめ

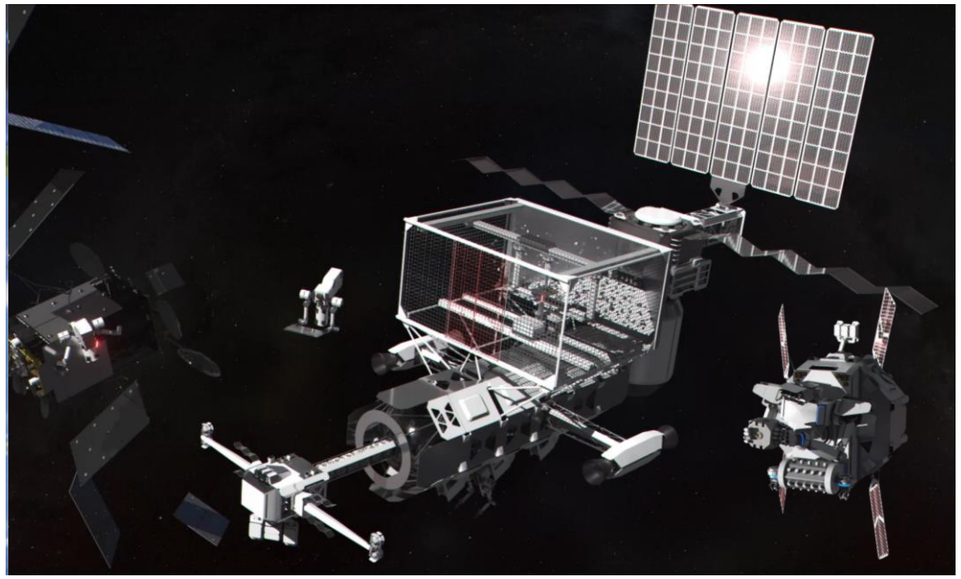
・宇宙機システムの大規模化に伴い，ROSの導入が有効
・ただし，宇宙機のソフトは，高い信頼性と安全性が必要であり，ROSをそのまま採用するのは困難
・Space ROSは，ISS船外でも使えるような，より高い信頼性と安全性を持つプラットフォームを目指している
・JAXAが開発したRACSは，宇宙環境においても，高い信頼性と安全性を維持しつつ，地上で蓄積されたリソースの活用が可能であることを示した
・Space ROSとRACSが統合することで，より強力なプラットフォームが構築できる
・地上でも宇宙機並みの安全性・信頼性が欲しいという需要に応えられる

ROSユーザの皆さんの宇宙開発への参画，
地上技術への応用を期待しています！

※ 一緒に盛り上げていけたら嬉しいです！



月惑星探査の将来像



軌道上サービスの将来像

ROSCon JP 2024

補足資料：RACS2ソフトウェア要求仕様書

ROS2 and cFS System Bridge (RACS2 Bridge) Software Requirements Specification

Explore to Realize

RACS2ソフトウェア要求仕様書
ROS2 and cFS System Bridge (RACS2 Bridge) Software Requirements Specification

■ 1.0 INTRODUCTION

□ 1.1 PURPOSE

This document describes the software requirements for the “ROS2 and cFS System Bridge” (RACS2 Bridge). RACS2 bridge has already been made open source on the JAXA github (https://github.com/jaxa/racs2_bridge), and this document is intended to publicly communicate its design intention and requirements.

本文書は「ROS2 and cFS System Bridge」（以下RACS2 Bridge)のソフトウェア要求仕様を記す。RACS2 Bridgeは既にJAXA github上においてオープンソース化されており（https://github.com/jaxa/racs2_bridge）、本文書はその設計思想と要求仕様を公に伝えることを目的とする。なお、日本語訳には細心の注意を払うが、英文と日本語文間で齟齬があった場合は英語版を優先する。

□ 1.2 PROJECT SCOPE

RACS2 is a collection of software that message communication between Robot Operating System 2 (ROS2) nodes and the Core Flight System (cFS). cFS, developed by NASA, is a platformindependent reusable software framework and a collection of reusable software applications, which includes the Core Flight Executive (cFE), a framework component of connecting flight software modules. ROS2 is the second generation of Robot Operating System, which comprises software libraries and tools designed for creating robot applications. It includes drivers, advanced algorithms, and robust developer resources, all available as open-source components for developer’s robotics projects. RACS2 is intended to allow the users, including flight software developers and ground roboticists, to easily develop new applications for space robotics.

RACS2 は、Robot Operating System 2 (ROS2) ノードと Core Flight System (cFS) 間のメッセ ージ通信を行うソフトウェア群である。cFS は、NASAが開発したプラットフォームに依存し ない再利用可能なソフトウェア フレームワークであり、再利用可能なソフトウェア アプリケ ーションを促進するものとなる。これには、フライト ソフトウェア モジュールを接続するフ レームワーク コンポーネントである Core Flight Executive (cFE) が含まれる。ROS2 は、ロボ ット アプリケーションを作成するために設計されたソフトウェア ライブラリとツールで構成 される第 2 世代の Robot Operating System となる。ドライバ、高度なアルゴリズム、強力な 開発者リソースが含まれており、すべて開発者のロボット プロジェクト用のオープン ソース コンポーネントとして利用可能である。RACS2 は、宇宙機向けソフトウェア開発者や地上ロ ボット技術者などのユーザーが、宇宙ロボット向けの新しいアプリケーションをできるだけ簡 単に開発できるようにすることを目的としている。

RACS2ソフトウェア要求仕様書
ROS2 and cFS System Bridge (RACS2 Bridge) Software Requirements Specification

RACS2 consists of two components. The first component is the ROS2 and cFS System Bridge (RACS2 Bridge), which connects ROS topics and cFS messages at the application layer. The second component is the RACS2 extended DDS, which connects ROS2 and cFS at the DDS layer. The conceptual diagram of RACS2 is shown in Figure 1.1. The main purpose of both RACS2 Bridge and RACS2 Extended DDS is to allow cFS and ROS2 to coexist. RACS2 Bridge has a communication protocol layer and a communication hardware layer on both the cFS and ROS2 sides, and can be used to exchange messages between cFS and ROS2 while utilizing them as is. On the other hand, RACS2 Extended DDS is a DDS used in ROS2 with a communication software layer in addition to the communication protocol layer and communication hardware layer, and can be implemented in a system with the freedom of design to communicate with cFS and ROS2 all at once.The RACS2 Bridge and RACS2 extended DDS are independent elements, and it is intended that one or both of them be used as necessary.This document will refer to the RACS2 Bridge in its scope.

RACS2は２つの要素から構成される。１つ目の要素はROS2 and cFS System Bridge (RACS2 Bridge)で、ROS topicとcFS messageがアプリケーション層で接続される。２つ目の要素は RACS2 Extended DDSで、ROS2とcFSをDDS layerで接続する。その概念図をFigure 1.1に示 す。使用用途としては、RACS2 Bridge もRACS2 Extended DDSもcFSとROS2の共存が大目的 である。RACS2 BridgeはcFS、および、ROS2側でそれぞれ通信プロトコル層、および、通信 ハードウェア層を有し、そのまま生かした中でcFSとROS2間のメッセージのやり取りを行う用 途が考えられる。一方で、RACS2 Extended DDSは通信プロトコル層、通信ハードウェア層に 加え、通信ソフトウェア層をROS2で使用されるDDSとしcFSとROS2一括にまとめて通信する 設計の自由度を持つシステムにおいて実装される用途として考えられる。RACS2 BridgeとRACS2 extended DDSは独立した要素であり、RACS2提供者は、必要に応じ てその片方、および、両方が使用されることを意図している。本文書は、RACS2 Bridgeにスコープを当てて言及する。

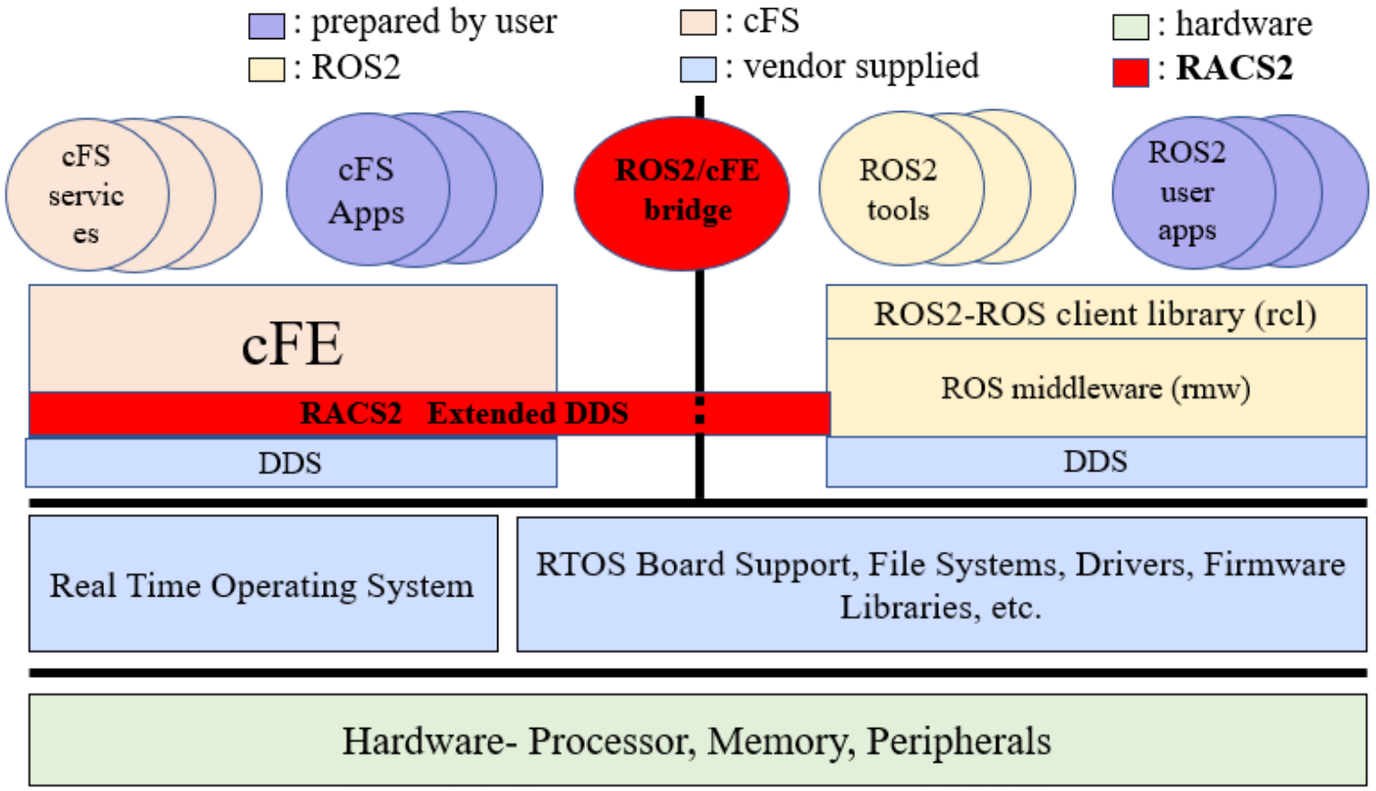


Figure1-1. The conceptual diagram of RACS2

□ 1.3 DOCUMENT CONVENTIONS

The convention used in this document is as follows:
"Shall" -- Used to indicate a requirement which must be implemented
"Should" -- Used to indicate a goal which must be addressed by the design
“To be defined (TBD)” – Undetermined features, parameters, etc. It is attempted to resolve and remove as necessary.

このドキュメントで使用されている規則は以下のとおり:
「〜すること」 -- 実装する必要がある必須の要件を示すために使用される（”Shall”要求）
「〜することが望ましい」 -- あればベターな目標を示すために使用される（”Should”要求）
「TBD」-- 未確定の機能、パラメータなど。必要に応じて解決する。

□ 1.4 REFERENCES

・https://github.com/jaxa/racs2_bridge
・Hiroki Kato, and Tatsuhiko Saito, "RACS2: the ROS2 and cFS System, launched" Flight Software Workshop 2023.

■ 2.0 PREMISE

In this section, we refer to the use cases of RACS2 Bridge. We first define the actors and then describe the use cases for each target user.

本項では、RACS2 Bridgeの前提条件について説明する。まず、ユースケースのアクターを含む用語(terms)を定義し、その後Use Case、および、Assumptionについて説明する。

□ 2.1 TERM DEFINITION

The following table defines terms describing RACS2 Bridge.

RACS2 Bridgeを説明する用語を以下の表にまとめる：

RACS2 Bridge	Target for development. Its design is described in Section 3.1, and equirements specification is described in Section 3.2.	開発対象。 その設計についてはセクション3.1で、要求仕様についてはセクション3.2で説明する。
1.1 Native cFS users	The main target users of RACS2. Therefore, they can be an actor in use cases. They are a group of people, who are used to flight software practice including cFS. Typically, they own and maintain cFS assets at different layers. This could include application software, development of on-board computers for spacecraft, or communication protocol level components such as TTE or SpaceWire. They have at least heard of how popular ROS2 is among roboticists, but may or may not be familiar with ROS2.	RACS2 の主な対象ユーザー。 cFS を含むフライト ソフトウェアの実践に慣れている人々のグループ。通常、彼らはさまざまなレイヤーで cFS 資産を所有し、維持している。これには、アプリケーションソフトウェアをはじめ、宇宙船のオンボード コンピューターの開発、または TTE や SpaceWireなどの通信プロトコルレベルのコンポーネントが含まれる可能性がある。彼らは、ROS2 がロボット工学者の間でどれほど人気があるかを知っているが、ROS2 に慣れているかどうかはどちらのケースもある。

1.2 Native ROS2 users	The main target users of RACS2. Therefore, they can be an actor in use cases. They are a group of people, who are ground roboticists. They are newly willing to develop applications for space robotic missions. They are used to ROS2 system, including conventions, supporting libraries, debugging tools, and supportive ecosystem.	RACS2 の主な対象ユーザー。 彼らは、地上ロボット工学者である人々のグループ。彼らは、宇宙ロボット ミッション用のアプリケーションを新たに開発する意欲を持っています。彼らは、規約、サポート ライブラリ、デバッグ ツール、およびサポート エコシステムを含む ROS2 システムに慣れている。
1.3 RACS2 users	They may be an actor in use cases. They may include both Native cFS users and Native ROS2 users.	Native cFS usersとNative ROS2 users両方を含む
1.4 RACS2 Developers	They may be an actor in use cases. It is us.	私たちのこと。
2.1 cFS	cFS that is connected to ROS2 using RACS2	RACS2 を使用して ROS2 に接続されている cFS
2.2 ROS2	ROS2 that is connected to cFS using RACS2	RACS2 を使用して cFS に接続されている ROS2
2.3 cFS user software	It includes cFS software modules developed by users, such as applications, communication protocol, hardware interface	アプリケーション、通信プロトコル、ハードウェア インターフェイスなど、ユーザーが開発した cFS ソフトウェア モジュールが含まれる。
2.4 ROS2 user software	It includes ROS2 software modules developed by users, such as applications, hardware interface. For communication protocol, UDP or TCP are used by default, and many users stay them.	アプリケーション、ハードウェア インターフェイスなど、ユーザーが開発した ROS2 ソフトウェア モジュールを含む。通信プロトコルは、UDP または TCP がデフォルトで使用されており、多くのユーザーはそれを使い続けている。
3.1 Communication software layer	As described. E.g., DDS、ZENOH	DDS、ZENOH等
3.2 Communication protocol layer	As described. E.g., TCP, UDP, SpaceWire communication standards	TCP, UDP, SpaceWire通信規格等
3.3 Communication hardware layer	As described, E.g., TTE hardware standards, SpaceWire hardware Standards	TTEハードウェア規格, SpaceWireハードウェア規格, 等

□ 2.2 USE CASES

In this section, we describe the use cases of RACS2 Bridge.

本項では、RACS2 のユースケースについて説明する。

(1) RACS2 users have cFS and ROS2 coexist in the software that they are developing and connect them via the RACS2 Bridge. This applies to both systems with multiple hosts and systems with a single host.

RACS2 userは、cFSとROS2を開発対象のソフトウェアに共存させ、RACS2 Bridgeにより接続する。なお、複数ホストのシステム、および、単ホストのシステム両方を想定する。

(2) RACS2 users freely select the communication software layer, communication protocol layer, and communication hardware layer when deploying flight software even when using RACS2 Bridge. In other words, RACS2 Bridge does not restrict the RACS2 user's choices in the communication software layer, communication protocol, and communication hardware layer when deploying flight software. (This is a non-functional requirement of RACS2 Bridge.)

RACS2 userは、RACS2 Bridge使用時にもフライトソフトウェアデプロイ時の通信ソフトウェア層、通信プロトコル層、通信ハードウェア層を自由に選択する。言い換えると、RACS2 Bridge はフライトソフトウェアデプロイ時の通信ソフトウェア層、通信プロトコル、および、通信ハードウェア層におけるRACS2ユーザの選択肢を狭めない。(これはRACS2 Bridgeの非機能要求となる。)

(3) The RACS2 users explicitly and directly set the ROS2-cFS connection method and conversion definitions of the RACS2 Bridge when designing and manufacturing the user application. The definitions to be converted include, for example, messages on the cFS side and topics on the ROS2 side.

RACS2 userは、RACS2 BridgeのROS2-cFS間の接続方法や変換の定義をRACS2 userがコントロールできる形でユーザーアプリケーション設計・製造時に設定する。変換される定義には、例えばcFS側のmessageやROS2側のtopic等が含まれる。

(4) RACS2 users exchange messages between ROS2 and cFS by properly configuring the RACS2 Bridge without modifying the cFS/ROS2 applications, when using the RACS2 Bridge.

RACS2 userは、RACS2 Bridgeを使用するときには、cFS/ROS2各々のアプリケーションの改変をすることなく、RACS2 Bridgeの設定を適切に記述してROS2-cFS間のメッセージのやり取りを行う。

(5) Native cFS users use ROS2 via RACS2 without modifying the cFS user software (flight software), which is a reusable design asset, except for modifications required for newly developed application software.

Native cFS userは、再利用可能な設計資産であるcFS user software（フライトソフトウェア）を改変することなく、RACS2経由でROS2を使用する。新しく開発するアプリケーションソフトウェアに対して必要な改変はその限りではない。

(6) Native cFS users connect to ROS2 using RACS2 without changing the communication protocol layer in cFS or ROS2 as much as possible. If a user wants to develop using a communication protocol layer that is not originally supported by RACS2 (although RACS2 developers are working to improve it), the user must modify the communication protocol layer of RACS2.

Native cFS userは、可能な限りcFSやROS2における通信プロトコル層の入れ替えを行わない形でRACS2を用いてROS2に接続する。（RACS2 developerはその充実を図るものの、）RACS2でもともとサポートされていない通信プロトコル層を使用して開発したいときには、ユーザ側がそのRACS2の通信プロトコル層の改修を行う。

(7) Native ROS2 users easily access and evaluate cFS using RACS2, at least for trial purposes, rather than flight software. Using RACS2, evaluation can be easily performed (for example, on a Linux PC) within the framework of ROS2/cFS coexistence without waiting for decisions on the design of the communication layer, etc. It is expected that the flight software development after evaluation including cFS, ROS2, and RACS2 will be implemented with a communication layer, etc. in detail.

Native ROS2 userは、少なくともフライトソフトウェアではなく試用時にはRACS2を用いてcFSへのアクセスを容易に行って評価する。RACS2を使ってROS2・cFS共存の枠組みで通信層等の設計の決定を待つまでもなく簡単に（例えばLinux PC上で）評価ができる。評価後の最終版としてはcFS, ROS2,とRACS2を含めて通信層等を作り込んだ形で実装する使用の流れを想定している。

□ 2.3 HYPOTHESES

In this section, we describe hypotheses of using RACS2 Bridge.

本項では、RACS2 Bridge を使用する際の重要な前提条件について記述する。

(1) The software system of the ROS2/cFS coexistence framework using RACS2 is expected to ultimately become flight software.

RACS2を使ったROS2・cFS共存の枠組みのソフトウェアシステムは、最終的にはフライトソフトウェアとなることを想定する。

(2) RACS2 users are expected to use the CCSDS Space Packet Protocol on the cFS side. Rationale: The CCSDS Space Packet Protocol is defined as a cFS feature and is part of the cFS functionality. On the other hand, while integrating CCSDS with ROS 2 is feasible, it requires careful handling of message types, topic naming, and compatibility between ROS 2 and DDS with significant amount of development and testing.

RACS2 userは、CCSDS Space Packet ProtocolをcFS側で使用することが想定される。（根拠：CCSDS Space Packet Protocol はcFSの機能として定義されていて、cFSの機能の一部である。一方で、CCSDSとROS 2の統合は技術的には実現可能と考えられるが、メッセージタイプ、トピックの命名、ROS 2とDDS間の互換性を慎重に処理する必要があり、かなりの量の開発とテストが必要となる。）

(3) RACS2 users are expected to use ROS2's convenient tools and libraries on the ROS2 side.

RACS2 userは、ROS2の便利ツールやライブラリをROS2側で使用することが想定される。

(4) Native cFS users are expected to use existing design assets on the cFS side.

Native cFS userは、既有の設計資産をcFS側で使用することが想定される。

(5) Native cFS users are expected to use existing design assets on the cFS side.

Native cFS userは、既有の設計資産をcFS側で使用することが想定される。

(6) RACS2 Bridge user is assumed to have an inter-host topology as shown in the figure below.

(A) Shared communication topology: The communication protocol layer and communication hardware layer are shared by both cFS and ROS2.
(B) Single point bridge topology: cFS and ROS2 each have their own communication protocol layer and communication hardware layer, and one of the hosts with a RACS2 bridge node accesses both the cFS and ROS2 communication buses.
(C) Single host bridge topology: cFS and ROS2 each have their own communication protocol layer and communication hardware layer, and one of the hosts bridges cFS and ROS2.

(A), (B), and (C) achieve bridging with one node, but (A'), (B'), and (C') have bridges on both the cFS and ROS2 sides. The former is more advantageous in terms of communication overhead and failure points, while the latter has more flexibility in implementation.

RACS2 Bridge userは以下の図にあるようなホスト間トポロジーを有することを想定する。

(A) Shared communication topology: 通信プロトコル層、および、通信ハードウェア層をcFS、ROS2両者で共有する。

(B) Single point bridge topology: cFS, ROS2それぞれに通信プロトコル層、および、通信ハードウェア層を有し、RACS2 bridge nodeを有するホストの１つがcFSとROS2の通信バス両方にアクセスする。

(C) Single host bridge topology: cFS, ROS2それぞれに通信プロトコル層、および、通信ハードウェア層を有し、そのうちの１つのホストの中でcFS, ROS2のブリッジを行う。

(A)(B)(C)はnode1個でブリッジを達成しているが、(A') (B') (C')はcFS, ROS2側双方でブリッジを有す。前者の方が通信オーバーヘッド、および、故障点の観点から有利であり、後者は実装の柔軟性を有する。

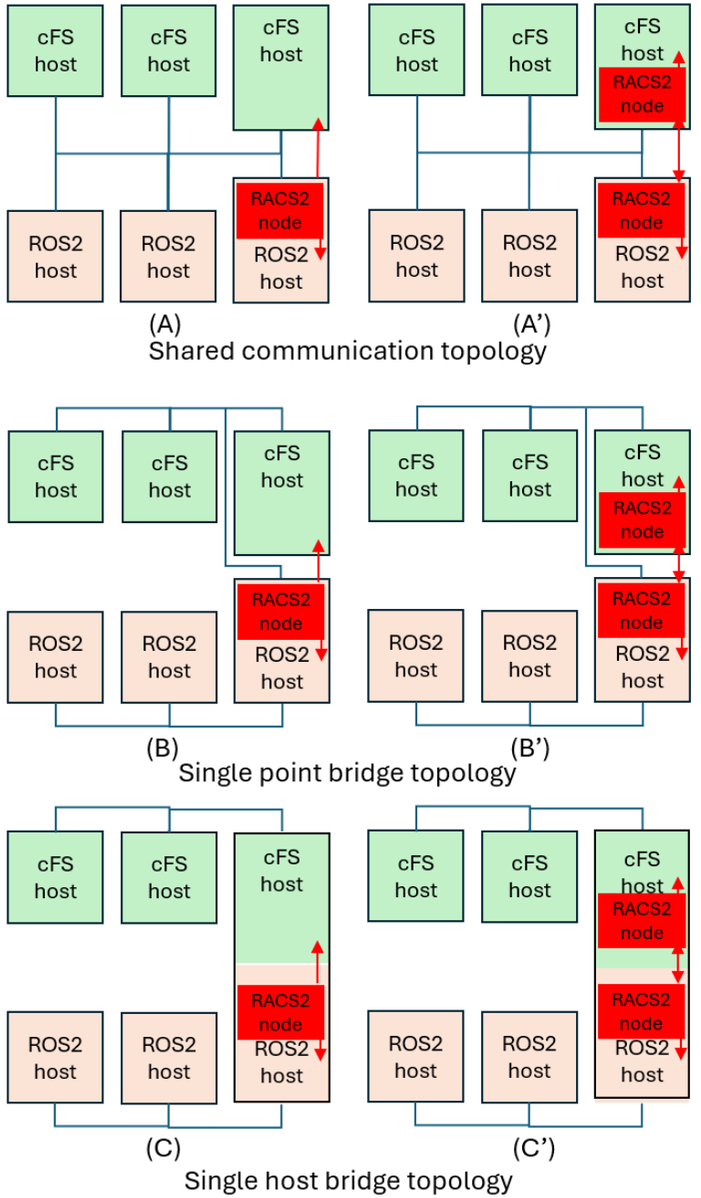


Figure 2-1 Possible host topology

■ 3.0 REQUIREMENTS SPECIFICATION DESCRIPTION

□ 3.1 DESIGN

RACS2 Bridge consists of Bridge Node and Bridge Definer. The system configuration is defined as shown in Figure 3.1. Bridge Node exists between cFS and ROS 2, and mediates messages commonly used by cFS and ROS 2, and exchanges messages with the other sides. Bridge Definer defines the mediation behavior between those messages.

RACS2 BridgeはBridger NodeとBridge Definerから構成される。システム構成をFigure3.1のように定義する。Bridger NodeがcFSとROS 2の間に存在し、cFSとROS 2各々で共通的に使用されるメッセージを仲介し、その他方へやりとりを行う。また、Bridge Definerがそのメッセージ間の仲介の挙動を定義する。

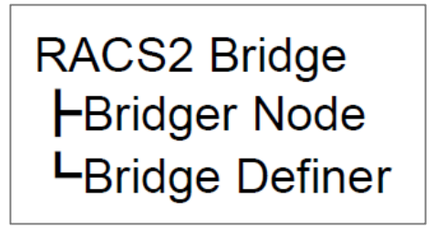


Figure 3.1 RACS2 Bridge system composition

□ 3.2 REQUIREMENTS SPECIFICATION

The top-level requirements for the RACS2 Bridge are described in Section 3.2.1. The requirements for its subsystems are shown in Section 3.2.2 onwards.

最上位のRACS2 Bridgeの要求仕様を3.2.1項に示す。以降、そのサブシステムの要求仕様を3.2.2項以降に示す。

・3.2.1 RACS2 Bridge (top-level)

(1) RACS2 Bridge shall distribute contents of cFS messages to the ROS 2 system as ROS 2 topics via Bridge Node as defined in Bridge Definer.
Rationale: From Use Case (1). This is one of the top-level requirements that can be delivered based on the system's objectives.

RACS2 Bridgeは、cFSのmessageをBridger Node経由でBridge Definerに定義した通りにROS 2のtopicとしてROS 2システムに分配できること。
根拠：Use Case (1)より。システムの目的より展開できる最上位要求の１つ。

(2) RACS2 Bridge shall distribute contents of ROS 2 topics to the cFS system as cFS messages via Bridge Node as defined in Bridge Definer. Rationale: From Use Case (1). This is one of the top-level requirements that can be delivered from the system objectives.

RACS2 Bridgeは、ROS 2のtopicをBridger Node経由でBridge Definerに定義した通りにcFSのmessageとしてcFSシステムに分配すること。
根拠：Use Case (1)より。システムの目的より展開できる最上位要求の１つ。

(3) RACS2 Bridge should distribute contents of ROS 2 messages to the cFS system as cFS messages via Bridger Node as defined in Bridge Definer.

Rationale: From Use Case (1). This is one of the top requirements that should be requested by ROS 2 users, which can be deployed based on the purpose of the system.

RACS2 Bridgeは、ROS 2のmessageをBridger Node経由でBridge Definerに定義した通りにcFSのmessageとしてcFSシステムに分配することが望ましい。

根拠：Use Case (1)より。システムの目的より展開できるROS 2ユーザから要望が出るはずの最上位要求の１つ

(4) RACS2 Bridge shall have an abstracted communication software layer and communication protocol layer. Rationale: From Use Case (2). It allows the communication software layer, communication protocol layer, and communication hardware layer on the user side (cFS, cFS user software, ROS 2, and ROS 2 user software) to be used without modification. Socket communication schemes such as UDP, which are essential for flight software, are not required for RACS2.

RACS2 Bridgeは、抽象化された通信ソフトウェア層、通信プロトコル層を有すること。
根拠：Use Case (2)より。このことにより、ユーザ側（cFS、cFS user software、ROS 2、および、ROS 2 user software）における通信ソフトウェア層、通信プロトコル層、および、通信ハードウェア層を改修なく使用可能となる。UDPのようなソケット通信がフライトソフトウェアで必須になるスキームをRACS2では使用を必須としない。

・3.2.2 Bridger Node

(1) Bridger Node should be able to communicate using shared memory when sharing data between cFS and RACS2. Rationale: From Use Case (2). Differences in the communication layers of cFS and RACS2 can be an obstacle during design. Therefore, by absorbing these differences using shared memory, RACS2 users will have more freedom in designing the communication hardware layer.

Bridger Nodeは、cFSとRACS2間でデータを共有する際には、shared memoryでも対応可能とすることが望ましい。
根拠：Use Case (2)より。cFS およびRACS2、各々の通信層の差異が設計時の障害になることが考えられる。そのため、Shared Memory を用いてその差異を吸収することにより、RACS2 userの通信ハードウェア層設計の自由度が上がると考えられる。

(2) Bridger Node should be one or less communication nodes to bridge between cFS and ROS2.

Rationale: Design options include having a bridge node on the ROS 2 side, on the cFS side, or on both, but it is desirable to have no more than one in terms of the communication topology in order to reduce communication overhead and points of failure during use.

Bridger Nodeは間に仲介する通信ノード数を１つ以下有することが望ましい。

根拠：設計の選択肢として、Bridger Node をROS 2側に持つ、cFS側に持つ、あるいはその双方で持つなどの選択肢が考えられるが、使用時の通信オーバーヘッド、故障点を減らす観点で通信トポロジー的に１つ以下とすることが望ましい。

・3.2.3 Bridge Definer

(1) Bridge Definer shall have a bridge mapping table that must be controllable within the RACS2 Bridge provided by the RACS2 developer.

Rationale: In other words, the requirement is that the mapping table must be generated in a way that can be controlled by the RACS2 user and that does not require modification of the user application.

Bridge Definerは、ブリッジのマッピングテーブルを有すること。また、そのブリッジのマッピングテーブルはRACS2 developerが提供するRACS2 Bridgeの中で制御できること。

根拠：つまり、RACS2ユーザがコントロールできる形、また、ユーザアプリの改変をしない形でマッピングテーブルを生成できることが要求となる。

(2) Bridge Definer should have a dynamic bridge mapping table that is controllable within the RACS2 Bridge provided by the RACS2 developer.

Rationale: A request from ROS2 users.

Bridge Definerは、動的なブリッジのマッピングテーブルを有することが望ましい。また、そのブリッジのマッピングテーブルはRACS2 developerが提供するRACS2 Bridgeの中で制御できることが望ましい。

根拠：ROS2の主要な機能