

ASTRO

Autonomous Satellite Test & Robotics Operations

Cannon Whitney

Dylan Long

Caleb Jackson

2026-02-14

1. Project Overview

The STAR laboratory at UF relies on ad hoc UDP client-server architectures to connect simulations with flight software. This limits:

- **Interoperability** – Projects cannot easily work together
- **Modularity** – Logging, calculations, and networking are bundled into monolithic executables
- **Scalability** – Simulations cannot serve multiple systems simultaneously

1. **Interoperate** – Enable seamless communication between simulation desktops and Jetson edge hardware through ROS 2 nodes
2. **Modularize** – Package existing lab software (orbital mechanics, GNC, logging) into discrete ROS 2 components
3. **Extend** – Build a framework that connects to *any* simulation, not just Basilisk, expanding the lab's testing and operational capabilities

2. Project Design

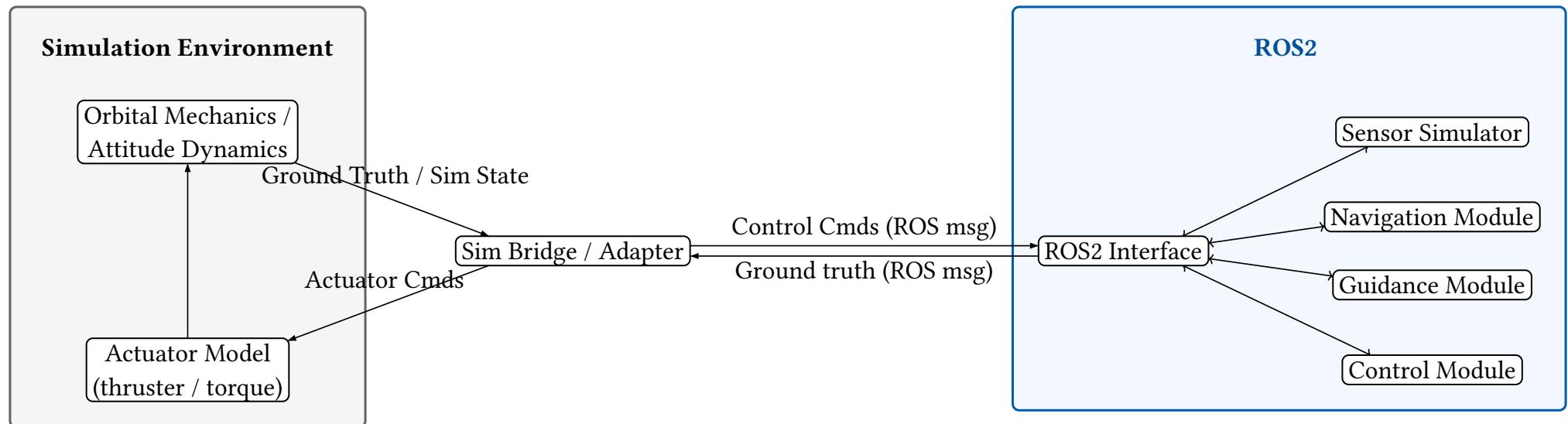
2.1 Software Architecture – External Sim

2. Project Design

When the simulation environment cannot be wrapped as a ROS 2 node, a bridge adapter translates between raw simulation data and ROS messages.

2.1 Software Architecture – External Sim

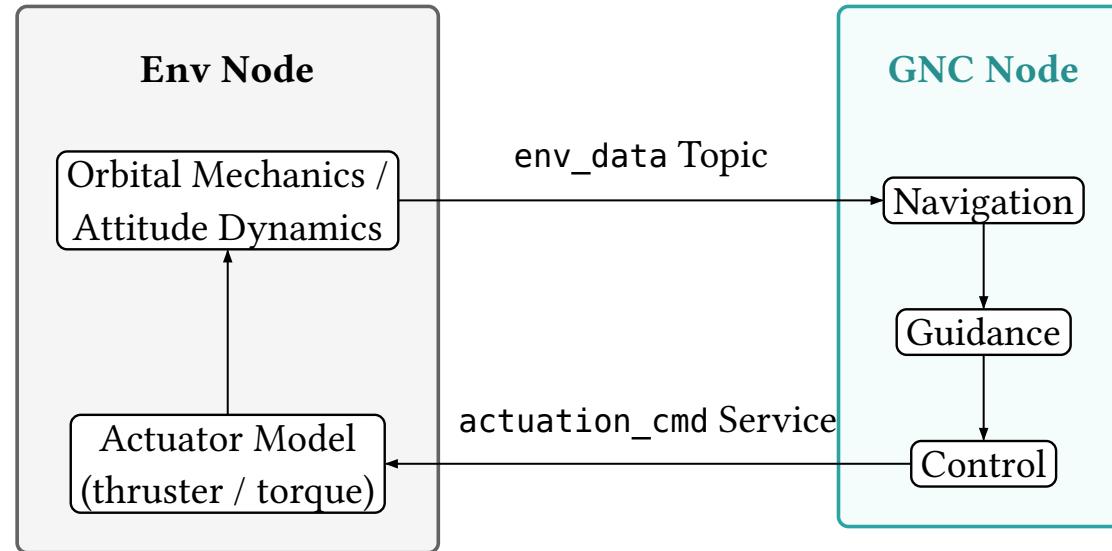
2. Project Design



When the simulation *can* be wrapped in ROS 2, all components live inside the ROS 2 graph as native nodes.

2.2 Software Architecture – Internal ROS 2

2. Project Design



2.3 ROS 2 Component Mapping

2. Project Design

Component	Role
Env Node	Executes simulation software (orbital mechanics, actuator models)
GNC Node	Navigation, Guidance, and Control – ported from DLQR on NJON
env_data Topic	High-frequency telemetry / state from Env to GNC
actuation_cmd Service	Control signals sent from GNC to the environment

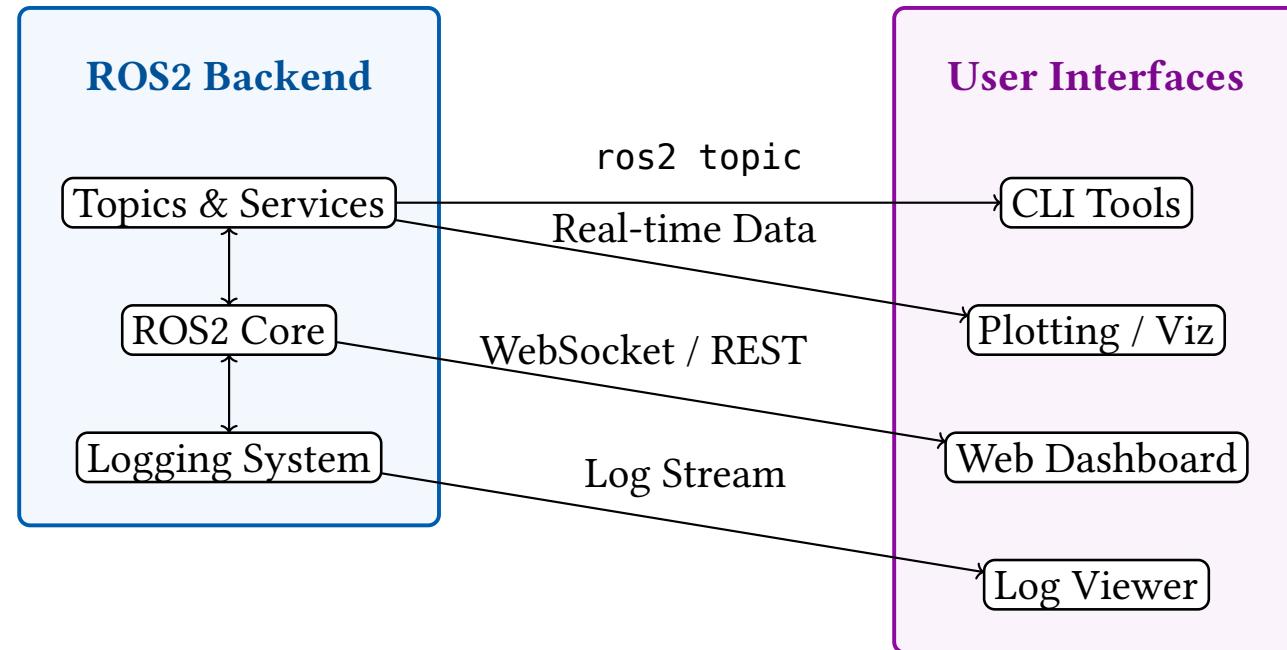
2.4 User Interface Design

2. Project Design

ROS 2 exposes system data through topics, services, and logging – accessible via CLI, dashboards, and plotting tools.

2.4 User Interface Design

2. Project Design



3. Project Progress

3.1 Milestones

3. Project Progress

#	Milestone	Status
1	Complete integration with DLQR	In Progress
2	Complete integration with QP_MPC	Planned
3	Add tasking layer	Planned
4	Integrate UI	Planned

3.2 Current Sprint

3. Project Progress

- Laying out the ROS 2 package structure
- Mapping existing hardware topology (Desktop + NJON) to ROS 2 nodes
- Wrapping DLQR functionality into Env and GNC nodes
- Removing ad hoc networking; replacing with ROS 2 topics and services

4. Individual Responsibilities

4.1 Dylan – SCRUM Master

4. Individual Responsibilities

- Build the ROS 2 component skeleton (package layout, launch files)
- Populate Env Node from `udp_hw_discrete_txrx` reference code
 - ▶ Remove networking layer
 - ▶ Replace logging with ROS 2 logging

4.2 Caleb – Backend Developer

4. Individual Responsibilities

- Populate Control Node from `udp_roundtrip_discrete` reference code
 - ▶ Remove networking layer
 - ▶ Replace logging with ROS 2 logging
- Verify outputs against the original standalone executable

4.3 Cannon – Project Manager

4. Individual Responsibilities

- Coordinate with STAR Lab
- Coordinate architecture decisions and team schedule
- Oversee integration testing between nodes