

Reflect. Scout. Adopt. Sponsor.

How Gravis Robotics builds a production robotics communication stack on open-source foundations

Benjamin Rippel and Peter Siegel, Gravis Robotics

2026-11-05

Executive Summary: Building the right stack from strong foundations **1**
Reflect: Define the requirements **2**
Scout: Find the right foundations **2**
Adopt: Put the architecture to work **3**
Sponsor: Close the gaps upstream **3**
Outlook: Build the ecosystem you depend on **4**

Executive Summary: Building the right stack from strong foundations

Gravis Robotics is replacing ROS 2 (from here on referred to as ROS for brevity) with iceoryx2 and FlatBuffers, step by step and without taking its product stack offline. Instead of building a middleware in-house, we chose an open-source foundation and, together with ekxide IO GmbH, closed the two gaps that blocked us upstream: first-class FlatBuffers support in iceoryx2, and a ROS gateway that lets both middlewares run side by side during the migration.

Because both are open source and maintained by the iceoryx2 authors, we carry no fork, and other teams can follow the same low-risk path. This whitepaper explains why we moved beyond ROS, how we chose the stack, how we are putting it to work, and why we invested upstream instead of building around the gaps.

©2026 Copyright Gravis Robotics. All rights reserved.

Reflect: Define the requirements

Gravis Robotics has its foundations in academia. As an ETH spin-off, we always had strong ties to the ROS world, because it is the de facto standard in robotics research. Many of our engineers are graduates of ETH or other top-ranking universities and have extensive experience with it from their studies, private projects, competitions and work at other robotics companies. For many of us, ROS became a beloved ecosystem due to its accessibility. Plenty of datasets, state-of-the-art algorithms, drivers and tools have been created by the community and shared publicly for everyone to build on. This has been and still is the foundation of many successful robotics companies around the globe.

While moving from prototype to production, we, like many teams, experienced the friction ROS creates. In our deployments, DDS, the default ROS middleware, adds significant overhead and is hard to configure. Limited introspection into DDS from within ROS makes it error-prone in production environments. The lack of forward/backward compatibility for messages makes it easy to break interfaces and invalidate recorded rosbags, causing high maintenance effort.

The provided build tooling is easy to use and well-suited for PoC-style development, but limits design decisions down the line, when more refined workflows matter. This is also where the huge transitive dependency tree that ROS pulls in starts to hurt. When bringing the robotics software stack to the target hardware, possibly involving cross-compilation and other more involved undertakings, these seemingly small issues add up and block progress. Looking ahead, the lack of a path to safety certification with ROS will become one of the biggest blockers for companies that continue to use it.

For us, this friction has not yet turned into real pain. Gravis Robotics delivers functional products based on ROS that excite our customers and investors, but the effort to do so increases with every iteration. To secure our continued success, we decided to move beyond ROS.

This is a high-risk decision for any company, and with ROS built into the heart of our product, it is a risky undertaking for us as well, but it is also a necessity. To mitigate the risk, we decided that a transition must be incremental, replacing the functionality provided by the ROS ecosystem in a step-by-step fashion. This covers not just the middleware but also parameter handling, build tools or build tool abstractions, execution and orchestration, record and replay functionality and all the bits and pieces we wrote over time.

We decided that writing our own middleware was out of scope. The complexity is enormous, requiring experience and time. Since we already have to handle complex robotics functionality, we concluded that existing solutions were likely suited to our use case and would let us outsource development and maintenance of the middleware itself.

Given our successful past with the ROS ecosystem, we wanted a “ROS-like” solution, i.e. an ecosystem with an active community, providing a scalable, stable and mature middleware applicable to all our relevant platforms and programming languages, but without pulling in a large dependency tree or forcing design decisions on us that we don’t want.

We also want direct control over the middleware we use, without multiple layers of abstraction obfuscating configuration and functionality. While we see applications for cross-host communication, our focus is on IPC, providing low latency communication for control loops and other critical subsystems.

Scout: Find the right foundations

Many developers at Gravis Robotics support open-source software in their private and professional lives. The company’s strategy is to fix and implement things upstream, to limit our maintenance effort and give back to the many contributors who provide great tools free of charge. We wanted to carry this mindset over to our choice of middleware and its ecosystem.

After extensive research, we decided that two technologies together form the base for our future communication platform. iceoryx2 serves as the lean, stable and fast open-source IPC framework, while FlatBuffers are the IDL for schema evolution and high-performance serialization.

©2026 Copyright Gravis Robotics. All rights reserved.

The choice of iceoryx2 became clear after research, PoCs and meetings with ekxide. Their experienced developers, with a strong track record in middleware and deployment, helped us map our vision onto the current state and roadmap of iceoryx2. The strong performance in combination with available customer support made it an easy choice.

iceoryx2 also leaves room for the high-level design decisions we care about, and supports all of our required platforms without extra build-tool abstraction layers or a large dependency footprint. It supports easy-to-understand communication patterns similar to ROS and is evolving quickly, with relevant features such as the blackboard pattern added in recent months. Cross-host communication via tunnels and a planned path toward safety certification make it future-proof for our applications.

Since iceoryx2 has no native IDL, we extended it with FlatBuffers. They offer great language support and performance, the long-desired schema evolution and efficient serialization with direct access to data, without a deserialization step. They are also natively supported by the MCAP container file format, which we use heavily. The only downside was the lack of first-class FlatBuffers support in iceoryx2, which made usage tedious, though still very performant.

Adopt: Put the architecture to work

Our first migration step is the lidar pipeline, from the sensor driver to the consumers of its point clouds. It touches enough of the stack to show us what running iceoryx2 in production takes, while staying contained enough to limit the risk.

A newly created iceoryx2 to ROS gateway is what lets us do this without a flag day. The pipeline itself runs on iceoryx2 and FlatBuffers, while consumers that have not migrated yet keep their ROS subscriptions and receive the data through the gateway. New consumers subscribe on iceoryx2 directly.

Our existing tools for recording, replay and simulation keep working the same way, so the stack stays operational throughout, and each consumer can move to iceoryx2 on its own schedule. Every step can be validated against the existing ROS pipeline before anything is switched over.

Sponsor: Close the gaps upstream

Investing in iceoryx2 features was mostly a strategic decision. It allowed us to save a lot of time since ekxide handled implementation and maintenance from the start, letting us as a customer focus on preparing the migration beyond ROS. Because development happened upstream, we do not carry a fork that our own engineers would have to maintain.

Together with ekxide, we defined requirements for a generic solution that others can adopt as well. A generic solution makes it easier for tool suppliers to support iceoryx2 and FlatBuffers in their products. Open-sourcing the work also lets other companies evaluate the same path.

ekxide played a crucial role in coming to the solution. For both the FlatBuffers integration and the ROS gateway, we iterated on the requirements together, starting from our use case and working toward a design general enough for other users. Their middleware experience served as a cross-check for our own assumptions, and their support model fits our use case well.

Our first integration of FlatBuffers with iceoryx2 was cumbersome, so we decided to sponsor a first-class integration in upstream iceoryx2. Our PoC relied on workarounds that would not generalize to other use cases, which would have limited adoption by other users and companies. We showed our PoC to ekxide and talked about our options. Making it an open-source, first-class part of iceoryx2 was the best solution for both sides.

The path to sponsoring the ROS gateway was a similar experience. One of the biggest pain points was the migration strategy from ROS to iceoryx2, while limiting the risk during the transition. The software stack has to stay operational at all times during this transition.

©2026 Copyright Gravis Robotics. All rights reserved.

We decided to use a stepwise approach and envisioned a gateway between the two middlewares to bridge any information flow required to make both sides work. After several technical discussions with ekxide, we agreed on this approach and on how the gateway should work. We extended our initial concept to use reflection, which removes much of the hand-written conversion code between ROS and iceoryx2 messages. Because the gateway is open source, it enables a safe transition not only for Gravis Robotics but for any adopter.

Outlook: Build the ecosystem you depend on

Moving from ROS to iceoryx2 and FlatBuffers provides many benefits for Gravis Robotics. We get a future-proof middleware without accumulated technical debt, with support from its authors during the migration and beyond. This reduces the risk that comes with replacing a middleware.

If other companies adopt the approach, we can openly discuss middleware usage, architecture decisions and tools for all users because the middleware is not our USP, allowing everyone to collaborate and profit.

This effort also creates value for others. All users can now evaluate iceoryx2 and FlatBuffers inside an existing ROS stack without taking it offline, and the tooling is free of charge. Anyone can follow the same low-risk approach to move their product away from ROS and join the ecosystem. This also allows for adopting the open-source mindset and benefiting from the strong robotics community in Zurich, across Europe and worldwide.

With this, Gravis Robotics has laid the foundation for an easier transition from ROS to iceoryx2. ekxide can continue support and feature development, which is the basis for a slim, production-ready robotics middleware. Adopters, contributors and feature sponsors can help to strengthen its position as a production-ready alternative to ROS that keeps cost, risk and effort low.

©2026 Copyright Gravis Robotics. All rights reserved.