Pololu-rs: A Rust-Based Framework for Reproducible Multi-Robot Experiments

Jiaming Li, Charlotte Stentzler, Johannes Roser, Wolfgang Hönig

I. Introduction

Robotics research is often hindered by the lack of low-cost, open-source frameworks; existing systems are typically custom-built, expensive, or proprietary. This not only raises the barrier to entry but also hinders the reproducibility of experimental results.

To address these limitations, we present **pololu-rs**, which consists of two parts: i) an open firmware written in Rust for commercially off-the-shelf (COTS) mobile robots made by Pololu; ii) a low-cost, open-hardware add-on board that supports onboard data logging and low-latency wireless communication. Together, these contributions form a reproducible, extensible, and easy-to-deploy research framework.

The Crazyswarm project [1] has demonstrated the advantages of such an open COTS-based framework for real-world multi-robot research on small quadcopters and has seen widespread adoption within the research community. More recently, several agile hardware platforms have further highlighted the value of compact, reproducible experimental testbeds. For example, the Sanity quadrotor platform [2] enables high-speed coordinated flight of up to 20 micro-UAVs in motion-capture environments, the Cambridge RoboMaster platform [3] provides an integrated hardware and software stack for agile multi-robot research on holonomic ground vehicles, and the Mini Wheelbot [4] serves as a dynamic balancing testbed for learning-based control and benchmarking.

Our framework employs differential-drive and tracked ground robots, enabling controlled 2D experimental setups with fast and repeatable motions using the compact Pololu platforms, which can reach speeds of up to 4 m/s.

In addition, because the framework is built on the same radio and communication protocol as Crazyswarm, existing users can adapt quickly, and the shared architecture facilitates future fusion of the two frameworks.

II. Technical Details

A. Hardware Platform

The framework is built upon a COTS open-hardware mobile robot platform developed by **Pololu Corporation**, based on an RP2040 microcontroller. We support two robot types: the differential drive Pololu 3pi+ 2040 and the tracked Pololu Zumo 2040. Both are available in several variants (e.g. Hyper and Turtle) that differ in maximal velocity; see Fig. 1.

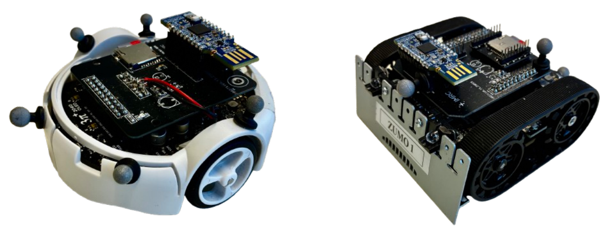

Fig. 1. The two supported robots, equipped with the custom extension deck. Left: Pololu 3pi+ 2040. Right: Pololu Zumo 2040.

The robots are available for around 200 USD. The underlying platform provides two DC motors with quadrature encoders, an onboard inertial measurement unit (IMU) and line sensors.

In addition, we develop modular open-hardware communication and storage decks that can be easily mounted on top of the robot. Each deck integrates an nRF52840 radio dongle and a microSD card interface and costs less than 15 USD.

B. Rust Firmware

While the manufacturer provides a C/C++ SDK for building firmware and a MicroPython-based firmware for high-level scripting, our framework introduces a custom **Rust** firmware. Rust was selected for its compile-time memory safety guarantees, modern language features such as operator overloading, and high performance.

Using an asynchronous architecture based on the **Embassy** [5] framework, high-frequency tasks (motor control, data logging, radio communication) run concurrently in an event-driven design without the overhead of a traditional real-time operating system.

For trajectory following we employ a geometric controller [6], followed by a control allocation step formulated as a constrained quadratic program. The embedded **quadratic programming (QP) solver**, based on Alternating Direction Method of Multipliers (ADMM) [7] solves convex QPs with linear constraints on platforms with tight computational budgets (e.g., STM32, RP2040 etc.), directly in the control loop explicitly enforcing actuator and kinematic constraints. This allows the robot to prioritize steering accuracy over velocity near physical limits and ensures safe, kinodynamically

All authors are with Technical University of Berlin, Berlin, Germany. Contact: charlotte.stentzler@campus.tu-berlin.de.

Video: https://youtu.be/4XJbtHaBDLs

Docs: https://imrclab.github.io/pololu3pi2040-rs/

feasible motions during aggressive maneuvers, providing a distinct advantage over standard PID-based approaches that may command infeasible actions leading to actuator saturation.

C. ROS 2 Embedding

The platform is designed to integrate with the Robot Operating System 2 (ROS 2) [8] through a lightweight bridge architecture. Rather than running a resource-intensive ROS node directly on the microcontroller, we implement a specialized protocol that offloads high-level processing to a host workstation. This design offers two primary benefits: i) **motion capture integration**: We receive state estimates from standard motion capture systems via ROS 2 topics, allowing for precise closed-loop trajectory tracking; and ii) **modular design**: The separation of low-level execution (firmware) and high-level planning (ROS 2) allows researchers to swap trajectory planners or control algorithms in Python or C++ on the host side without reflashing the robot.

D. Communication

Robot-host communication uses a wireless link based on an nRF52840 dongle and Crazyradio, following the Crazy RealTime Protocol (CRTP) [9]. The overall architecture consists of a bidirectional wireless connection between the Crazyradio and the nRF52840 dongle, enabling low-latency data exchange between the robot platform and a host computer.

Figure 2 is depicting the communication flow: On the downlink, the host transmits position ground truth (acquired from the motion capture system) and control commands via the radio interface. The dongle subsequently forwards them to the robot's RP2040 microcontroller through a UART connection. In parallel, the robot streams its internal state and identification information back to the dongle via UART, which relays the data wirelessly to the host. This communication pipeline enables synchronized state and command exchange with a ROS 2-based control node running on the host computer, supporting closed-loop control and monitoring across different robot configurations.

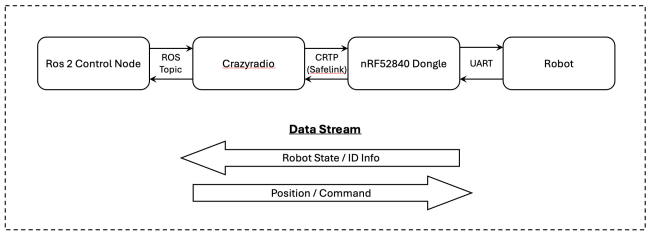

Fig. 2. System overview of the communication architecture between the host computer and the robot, showing the data flow among different levels.

III. Usage

To demonstrate the applicability of the framework in real-world robotics research, we note that recent work in kinodynamic motion planning has successfully employed it in real experiments. Figure 3 shows the car-with-trailer ground robot experiment from db-LaCAM [10], in which four Pololu robots are following precomputed trajectories including a swap maneuver.

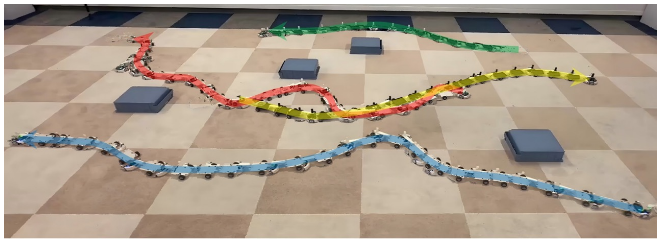

Fig. 3. Representative experimental result on the proposed platform showing a four-robot swap experiment.

In pc-dbCBS [11], Pololu-based ground robots are used in physical experiments, including a setup in which multiple differential-drive robots are coupled via rigid rods.

Similarly, db-ECBS [12] reports real-world experiments with heterogeneous ground–aerial teams using Pololu 3pi+ 2040 robots and Crazyflie drones, validating multi-robot coordination within a shared communication architecture for ground and aerial platforms.

These examples highlight the robustness of the hardware, the real-time feasibility of the firmware, and the scalability of the framework to multi-robot experiments. While our experiments validate operation with up to six robots, the communication and control architecture is designed to scale to larger teams, such as ten robots, without sacrificing low-latency performance.

IV. Conclusion

We introduce a modular, low-cost, open-source framework for differential-drive and tracked multi-robot experiments. The framework combines COTS hardware, a Rust-based real-time firmware, and a ROS 2 integration that together support reproducible, real-world multi-robot experiments. The successful usage of the framework presented in published research demonstrates its real-world applicability.

The development of the framework is ongoing and future work will focus on three directions: (1) extending the firmware to support 2.5D experiments, such as navigation over ramps and jumps; (2) enabling on-board state estimation to reduce the reliance on external motion capture systems and significantly lower upfront costs; and (3) seamless integration with Crazyswarm to enable mixed ground–aerial robot cooperation experiments.

References

[1] J. A. Preiss, W. Hönig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304.

[2] H. Woo, K. R. I. Sanim, K. Okumura, G. Yang, A. Shankar, and A. Prorok, *Sanity: An agile brushless quadrotor for multi-agent experiments*, https://openreview.net/pdf/8e2e978377d2793388c80d477162585c60668728.pdf, Accessed: 2025-01-XX, 2025.

[3] J. Blumenkamp, A. Shankar, M. Bettini, J. Bird, and A. Prorok, “The cambridge robomaster: An agile multi-robot research platform,” in *International Symposium on Distributed Autonomous Robotic Systems*, Springer, 2024, pp. 439–456.

[4] H. Hose, J. Weisgerber, and S. Trimpe, “The mini wheelbot: A testbed for learning-based balancing, flips, and articulated driving,” *arXiv preprint arXiv:2502.04582*, 2025.

[5] Embassy Developers, *Embassy: Embedded async executor for rust*, `https://github.com/embassy-rs/embassy`, Accessed: 2026-01-11, 2025.

[6] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, “A stable tracking control method for an autonomous mobile robot,” in *Proceedings., IEEE International Conference on Robotics and Automation*, 1990, 384–389 vol.1.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[8] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, “Robot operating system 2: Design, architecture, and uses in the wild,” *Science Robotics*, vol. 7, no. 66, eabm6074, 2022.

[9] Bitcraze AB, *CRTP — crazy realtime protocol*, `https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/crtp/`, Accessed: 2026-01-08, 2025.

[10] A. Moldagalieva, K. Okumura, A. Prorok, and W. Hönig, “db-LaCAM: Fast and scalable multi-robot kinodynamic motion planning with discontinuity-bounded search and lightweight mapf,” *arXiv preprint arXiv:2512.06796*, 2025.

[11] K. Wahba and W. Hönig, “pc-dbCBS: Kinodynamic motion planning of physically-coupled robot teams,” *arXiv preprint arXiv:2505.10355*, 2025.

[12] A. Moldagalieva, J. Ortiz-Haro, and W. Hönig, “db-ECBS: Interaction-aware multirobot kinodynamic motion planning,” *IEEE Transactions on Robotics*, vol. 42, pp. 244–260, 2026.