Fleet-Level Robot Orchestration Dashboard for Performance Monitoring and Remote Debugging

Software Engineering & Human–Robot Interaction

08.06.2025

Robot fleet on a construction site (AI-generated)

Background

When construction and logistics sites scale up their use of mobile robots, project managers face a new challenge: keeping dozens—or hundreds—of autonomous robots healthy, productive, and safe at once. While individual robots ship with their own GUIs, site-wide orchestration still depends on ad-hoc terminal commands, siloed dashboards, or even walk-around inspections.

A web-based orchestration interface that provides a single, real-time overview of every robot's status, tasks, and KPIs would:

- cut mean-time-to-repair by surfacing low-level diagnostics early
- boost throughput by re-allocating idle robots automatically, and
- simplify certification and -shift-handover reports with high-level analytics.

Using modern front-end stacks (TypeScript + React + Tailwind) together with ROS 2 (Jazzy) back-end services, this thesis will design, implement, and evaluate such a fleet-management dashboard.

SUPERVISOR
Prof. Dr.-Ing. Matthias Althoff

ADVISOR
M.Sc. Julian Hoffmann

TYPE
Bachelor's/Master's thesis

LANGUAGE
English

CONTACT
julian.a.hoffmann@tum.de

REQUIREMENTS
Programming: TypeScript / JavaScript, React (or Vue/Svelte), Node.js, Rust or Go for high-throughput telemetry services
Robotics Frameworks: ROS 2 (Jazzy), rosbridge / WebSockets, basic URDF
Data & Visualization: REST / gRPC, WebSockets, Plotly / ECharts, Prometheus
DevOps & Tooling: Docker, Git, CI (GitHub Actions)

Description

This project investigates interaction paradigms for **fleet-level robot orchestration** and delivers a browser dashboard that enables operators to:

1. **Monitor ↔ Overview**
    a. Live list of all registered robots (ID, type, last heartbeat, firmware).
    b. Health KPIs: battery, temperature, network latency, CPU/GPU load.
    c. Task throughput metrics: completed / failed / pending counts, Gantt-style timelines.
2. **Inspect ↔ Deep Dive**
    a. Drill-down panel per robot: ROS topic bandwidth, diagnostic_msgs, error logs.
    b. Real-time joint-state or wheel-odometry plots for quick motion verification.
    c. Snapshot + download of the last 60 s rosbag for offline replay.
3. **Control ↔ Command**
    a. Send high-level actions: “pause”, “resume”, “clear cost-map”, “return to dock”.
    b. Trigger parameter overrides (e.g., max velocity) with safety confirmation dialogs.
    c. Upload and schedule task lists (YAML or JSON) with version-controlled history.
4. **Debug ↔ Assist**
    a. Automatic anomaly detection (e.g., motor current spikes) with notification system.
    b. One-click link to remote SSH / web-terminal into the affected robot.
    c. Visual diff of configuration files between healthy and faulty units.
5. **Report ↔ Analytics**
    a. Daily & weekly performance reports: MTBF, mission success rate, utilization heat-maps.
    b. Export to CSV / PDF for regulatory compliance and stakeholder briefings.

Tasks

1. **Literature & Technology Survey**
    a. Fleet-management in AMR warehouses (e.g., Amazon Robo-Runner).
    b. Open-source monitoring stacks (Prometheus, Loki, Grafana) and ROS diagnostics.
    c. HCI design guidelines for operations dashboards.
2. **System Architecture**
    a. Define telemetry schema (JSON & ROS 2 msg) and data pipeline (FastDDS → WebSocket → React context).
    b. Choose persistence layer (e.g., InfluxDB for time-series, Postgres for metadata).
    c. Security model: authentication, TLS, and role-based command permissions.
3. **Interface Implementation**
    a. Fleet Overview Page with sortable / filterable table and status badges.
    b. Robot Detail Page featuring live plots (Plotly), YAML parameter editor, and log tailing.
    c. Task Board (Kanban style) showing queued, running, and done missions.
    d. Command Palette (⌘ + K) for quick actions—power users can type “robot/alpha restart-nav”.
    e. Responsive design for tablets carried by field technicians
4. **Documentation & Dissemination**
    a. Developer guide (setup, extension points, API docs via Swagger / AsyncAPI).
    b. Operator manual with troubleshooting playbooks.
5. **Thesis report and defense.**

Deliverables

1. **Web-based Orchestration Dashboard**.
2. **ROS 2 Integration Package**.
3. **Evaluation Report** with quantitative & qualitative study results.
4. **Thesis Document**

Further Readings

- Chen, Shengkang, et al. "Multi-modal user interface for multi-robot control in underground environments." *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.