

Supplementary Materials: A Three-Layer Multi-Agent Framework for PHM-Enabling Autonomous Condition Monitoring of Power ICT Infrastructure in Underground Facilities

— Supplementary Material S1: KEP-ROS Implementation Details —

Jaekyung Lee^{1,2}, Byung Sung Ko², Jiwon Lee², Jaeheon Park², Taewon Kim², Seoktae Kim² and Wonhee Kim^{1,*}

¹School of Energy Systems Engineering, Chung-Ang University, Seoul, Republic of Korea

²KEPCO Research Institute, Daejeon, Republic of Korea

*Corresponding Author: Wonhee Kim. Email: whkim79@cau.ac.kr

ABSTRACT: This document serves as Supplementary Material S1 to the main manuscript entitled "A Three-Layer Multi-Agent Framework for PHM-Enabling Autonomous Condition Monitoring of Power ICT Infrastructure in Underground Facilities." It details the implementation of KEP-ROS, an open multi-robot operating framework, covering its system architecture, technology stack, operational flow, database schema, heterogeneous robot control interfaces, multimodal sensor configuration, and operational strategy.

KEYWORDS: Prognostics and health management; condition monitoring; multi-agent AI system; edge computing; robotic framework

1 System Architecture and Technology Stack

Table S1 presents the technology stack of KEP-ROS, organized by layer.

Table S1: KEP-ROS technology stack configuration

Layer	Technology	Role
Backend API	FastAPI + Pydantic	REST/WebSocket endpoints, type-safe schemas
ORM/DB	SQLAlchemy + SQLite + Alembic	Persistence of missions, schedules, and inspection results
Distributed Tasks	Celery + Redis	Execution of mission schedules, real-time status monitoring
Robot Control	Spot SDK (gRPC)	Autonomous navigation, manipulator, and PTZ control
ROS Integration	ROS Bridge (WebSocket)	cmd_vel commands and topic subscription for ROS2-based generic robots
Video Streaming	aiortc (WebRTC) + go2rtc	Low-latency PTZ video transmission
Frontend	React 19 + Vite + Tailwind CSS	Dashboard, schedule calendar, and 3D visualization
3D Visualization	Three.js + URDF Loader	Real-time 3D rendering of robot poses and paths
Deployment	Docker Compose	Container orchestration

Note: REST, Representational State Transfer; URDF, Unified Robot Description Format.

In the backend layer, the integration of FastAPI and Pydantic ensures type safety for REST and WebSocket endpoints. In the distributed task layer, the combination of Celery and Redis processes data-collection and analysis tasks concurrently across multiple inspection points, avoiding single-process bottlenecks. This architecture establishes a fault-tolerant system in which the failure of an individual task does not disrupt the overall mission flow. The robot control layer directly controls the quadrupedal robot through the Spot SDK (gRPC) and manages Robot Operating System 2 (ROS2)-based generic robots through

the ROS Bridge within the same application programming interface (API) layer. The frontend integrates real-time 3D visualization of the robot’s pose and path with a mission-scheduling calendar in a single dashboard. Container orchestration through Docker Compose enables component-level updates without service disruption during module integration.

2 Overall Operational Flow

Fig. S1 illustrates the overall operational flow of KEP-ROS as an eight-stage circular structure.

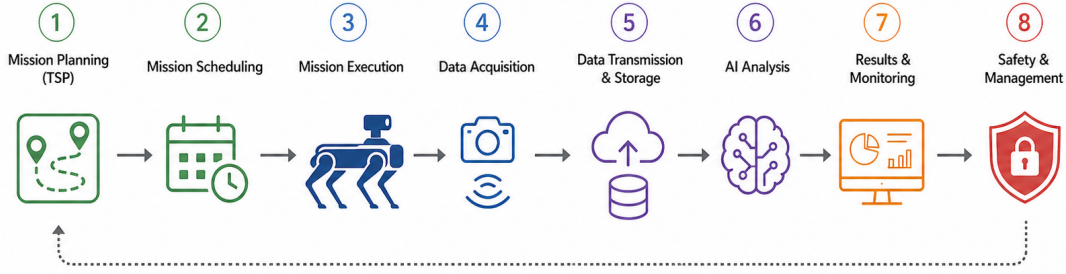
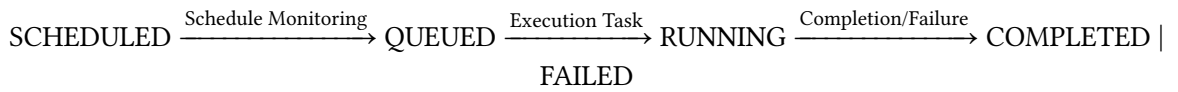


Figure S1: Overall operational flow of KEP-ROS (eight-stage circular structure)

- ① **Mission Planning:** The optimal inspection route is generated by solving the Traveling Salesman Problem (TSP) over the predefined inspection points on the map, taking the robot’s battery runtime into account. If the trajectory intersects a predefined obstacle boundary during navigation—detected using a line segment–polygon intersection test—the route is automatically replaced with an alternative. To minimize computational overhead, previously computed routes are reused for repeated executions of the same mission.
- ② **Mission Scheduling:** When a planned mission is registered on the calendar, a schedule monitoring task automatically places it into the execution queue at the scheduled time. To prevent duplicate concurrent execution of the same mission, its status is immediately updated to QUEUED. The mission lifecycle is managed by an explicit state machine as follows:



- ③ **Mission Execution:** The mission execution task transmits the autonomous navigation command sequence to the robot, initializes the mission context in the database, and records the baseline status of the robot at the starting location.
- ④ **Data Acquisition:** The status monitoring task receives and parses the navigation state and sensor data from the robot, automatically detecting arrival and departure events at each inspection point. Once arrival is confirmed, the corresponding timestamp is recorded, and multimodal data collection, including RGB and thermal images, is initiated.
- ⑤ **Data Transmission and Storage:** To guarantee data consistency across the distributed system, the acquired data are transmitted to the server in real time and persisted in the relational database.
- ⑥ **AI Analysis:** The collected images are automatically fed into the AI diagnostic pipeline. Note that diagnostic execution is performed asynchronously, continuing after the robot completes its physical patrol. The diagnostic status follows a separate queue structure (QUEUED → RUNNING → COMPLETED / FAILED) independent of the mission state machine.

- ⑦ Results and Monitoring: Diagnostic results and robot statuses are visualized in real time on the dashboard, and measurements exceeding safety thresholds automatically trigger operator alerts. The accumulated inspection history is persisted in the database to generate historical trend charts for time-series analysis of equipment degradation.
- ⑧ Safety and Management: Emergency stop (E-Stop) systems, battery levels, and communication statuses are continuously monitored; if an anomaly is detected, control is immediately transferred to manual remote operation. Note that the system periodically updates an E-Stop keep-alive signal. A fail-safe protocol is implemented to halt the robot automatically if the signal is not updated within the predefined timeout window.

Upon completion of a single inspection cycle, the subsequent cycle automatically commences according to the schedule defined in ②, forming a continuous, fully automated inspection workflow.

Fig. S2 shows the three-step asynchronous pipeline for mission automation.

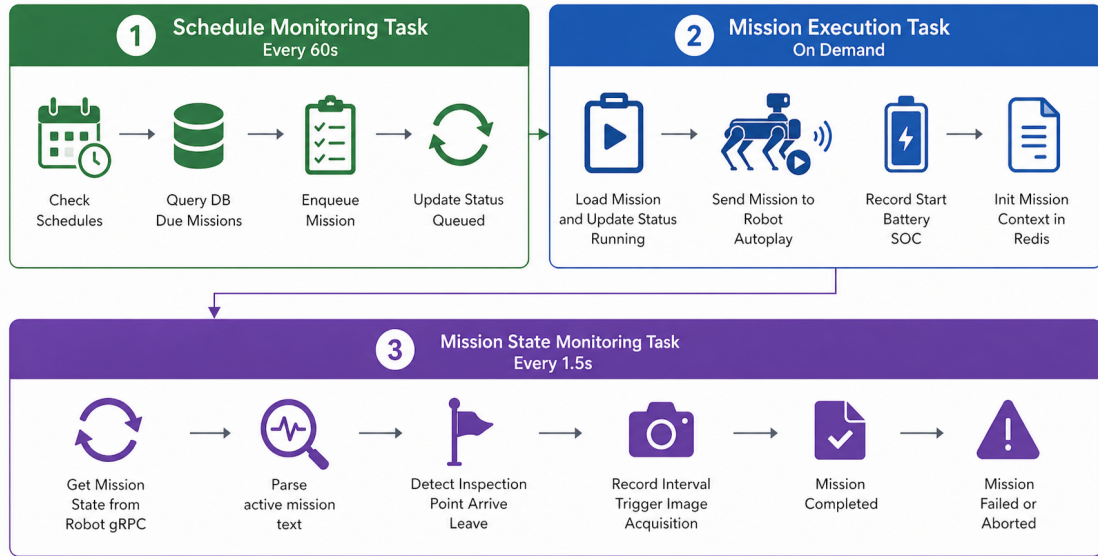


Figure S2: KEP-ROS mission automation pipeline

3 Database Schema and Heterogeneous Robot Interface

All operational logs in KEP-ROS are recorded in a relational database. Table S2 presents the core database tables.

Table S2: Core database schema of KEP-ROS

Table	Main Content	Remarks
robot	Installation location, robot type (Spot / ROS / Etc), connection address, camera module name, manipulator installation status	Robot registration
mission	Deployment site, mission name, waypoint coordinates list, path connection information, obstacle coordinates, TSP optimal path	Path/Obstacles
schedule	Assigned robot, mission to execute, scheduled time, daily repetition interval, schedule activation status	Calendar schedule
mission_result	Scheduled/actual start and end times, remaining battery capacity (%), operational status	Execution history
inspection_point	Inspection point name, equipment classification hierarchy, diagnostic model, physical gauge range, threshold limits for normal conditions	Inspection criteria
inspection_data	Inspection point name, acquisition timestamp, source image file path, output result storage path, analysis status	AI diagnostic queue
inspection_result	Reading completion timestamp, applied model, numerical reading value, state classification value, source/result paths	Reading results
event_log	Target robot, classification (general / alarm / error), key phrase, details, occurrence timestamp	Audit log

For the integration of heterogeneous robots, KEP-ROS incorporates a dual-protocol abstraction layer. For the quadrupedal robot, the Boston Dynamics Spot client is abstracted as a single managed object. Over a dozen functions—including status query, power control, autonomous navigation, manipulator control, pan-tilt-zoom (PTZ) camera control, and emergency stop—are managed through a unified application programming interface (API). For ROS2-based generic robots, KEP-ROS subscribes to and publishes standard topics—such as velocity commands (`cmd_vel`), pose estimation, occupancy grid maps, and battery status—through the ROS Bridge, enabling concurrent operation of heterogeneous platforms within the same dashboard. This abstraction layer decouples the application logic from the underlying protocol heterogeneity, thereby enabling transparent multi-robot control.

4 Multimodal Sensor and Data Acquisition Structure

The inspection targets in the underground infrastructure of the KEPCO Power ICT Center span diverse modalities—equipment appearance, gauge readings, and environmental conditions—that require a multimodal sensor suite for simultaneous capture.

In this study, sensor modules were deployed to acquire visual, thermal, and environmental data. The operational role and data structure of each sensor are defined as follows. First, the RGB camera is used to capture visual information required for inspection items such as equipment appearance, panel displays, and status LEDs. Second, the thermal imaging camera is deployed to detect anomalous heating patterns inside the equipment by analyzing temperature distributions. This enables proactive detection of latent anomalies that manual visual inspection cannot identify. Third, the environmental data module collects temperature, humidity, volatile organic compound (VOC) concentration, nitrogen oxide (NOx) concentration, and noise level to monitor atmospheric conditions in the underground infrastructure. The acquired environmental data are used to automatically assess whether inspection tasks can be performed safely, according to the criteria in Table S3. Upon entering a designated zone, the robot first acquires environmental data. If environmental parameters exceed the safety thresholds, the inspection mission for the target zone is automatically restricted or deferred. Note that environmental conditions are continuously monitored during the patrol; if anomalous fluctuations are detected, inspection schedules and routing priorities are dynamically adjusted.

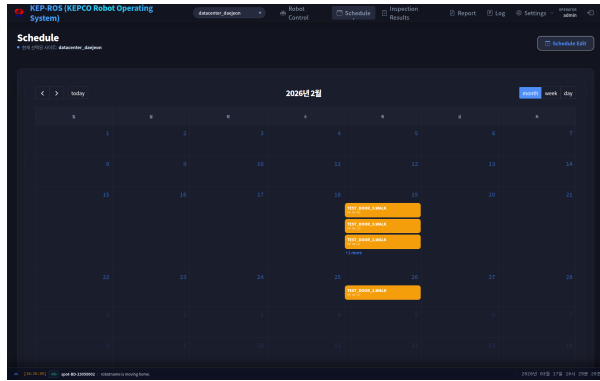
Table S3: Environmental monitoring criteria linked with Internet of Things (IoT) sensors

Classification	Measurement Item	Reference Value	Decision Criteria	Action
Environmental Safety	Temperature	0 to 40°C	Normal range	Execute inspection
	Humidity	0 to 80%RH	Normal range	Execute inspection
Environmental Hazard	VOC (Gas)	Index ≤ 100	Exceeds 100	Alarm and additional inspection
	NOx (Hazardous Gas)	Index ≤ 1	Exceeds 1	Alarm and additional inspection
Environmental Anomaly	Noise	Below reference	Rapid increase	Additional inspection required

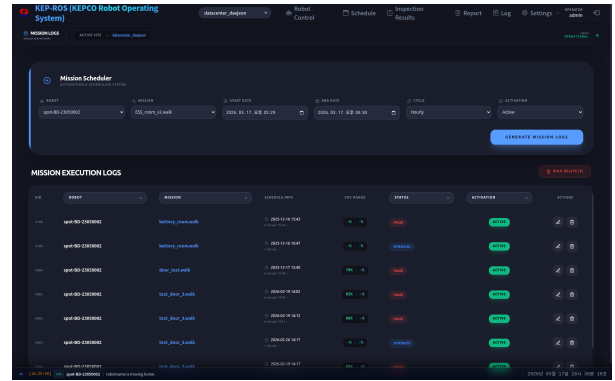
In the current framework, environmental data serve as a safety pre-screening filter for the patrol zone rather than as direct diagnostic variables. This multimodal dataset, however, could be leveraged in future work to enhance the accuracy of anomaly detection by fusing environmental variables with vision-based diagnostic models. Prior to transmission, the data acquired from each sensor undergo time synchronization and format normalization. Visual, thermal, and environmental data are processed independently but linked through a common synchronized timestamp to enable relational queries. Because real-world operational environments exhibit unpredictable lighting fluctuations, the reliability of the acquired image data was preserved by optimizing camera angles, supplementing lighting conditions, and stabilizing sensor mount positions. Obstacles and communication latency were addressed separately through the real-time path replanning and asynchronous data transmission described in Sections 5 and 1, respectively.

5 Operational Strategy

The underground infrastructure of the KEPCO Power ICT Center consists of multiple equipment zones with distinct functions, such as machine rooms, electrical rooms, UPS rooms, and ESS rooms. In this study, a zone-based autonomous patrol scheme was implemented to segment the inspection area based on equipment layouts and inspection targets. Each zone is defined beforehand in the database as a set of distinct inspection targets and mission units. By leveraging map-based pose estimation and path planning within KEP-ROS, the robot navigates reliably through narrow passages and high-density equipment layouts. The standard patrol sequence is structured as follows: "Enter zone → Acquire environmental data and verify safety → Navigate to inspection point → Stop → Acquire data → Move to next point." Note that the mission is temporarily suspended for safety if an abnormal gas concentration or a thermal anomaly is detected. Real-time obstacle avoidance is achieved by detecting static and dynamic obstacles during patrol execution and triggering local path replanning. Particularly in confined spaces, safety is ensured by dynamically adjusting the navigation velocity and, if necessary, halting the robot to replan the trajectory. To maximize inspection efficiency, a schedule-based patrol execution framework (Fig. S3) was adopted. Patrol tasks are executed automatically according to a predefined priority sequence. If a delay or sensor failure occurs at a specific inspection point, the subsequent sequence is dynamically rescheduled to prevent the entire patrol mission from halting.



(a) KEP-ROS inspection schedule



(b) KEP-ROS inspection results

Figure S3: Schedule-based robot operation