From Data to Decisions: Optimizing Robotic Fleets with Telematics and Artificial Intelligence

Bachelorarbeit

University of Applied Sciences Koblenz
Faculty of Engineering
Bachelor's Program in Mechatronics

by
Ivana Navarrete-Santeliz
Matriculation Number: 546322

University Supervisor:
Prof. Dr.-Ing. Matthias Flach

Submission Date: 29 December 2025

Declaration of Independent Work

Koblenz, 29 December 2025	Ivana, Navarrete-Santeliz
Place, Date	First Name, Last Name

Selbstständigkeitserklärung

Koblenz, 29 Dezember 2025	Ivana, Navarrete-Santeliz
Ort, Datum	Vorname, Nachname

Table of Contents

List of Abbreviations

AMR	Autonomous Mobile Robot
AI	Artificial Intelligence
CAN	Controller Area Network
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
FMS	Fleet Management System
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
IoT	Internet of Things
ML	Machine Learning
OBD-II	On-Board Diagnostics II
PID	Parameter ID
PM	Predictive Maintenance
RPM	Revolutions Per Minute
SoC	State of Charge
TCU	Telematics Control Unit

List of Figures

List of Tables

1 Introduction

Ever since the introduction of Artificial Intelligence (AI), the number and speed of discoveries and inventions in the scientific world have substantially increased, enabling researchers to make breakthroughs that were previously considered near-impossible [1]. However, there is one factor that has remained constant throughout, and that is data.

Data refers to information, mostly facts or figures, that is collected and organized to be analyzed, considered, and ultimately used to inform or guide decisions [2]. For this reason, data plays a key role across all developments, from standard databanks to powering AI chatbots, and as this work will explore, in telematics and smart cities.
The larger the amount of data, the more informed and qualitative systems can be built. Large datasets empower researchers and engineers to create advanced architectures capable of learning, predicting, and communicating. However, the ability to collect data is not quite enough. Knowing how to interpret, optimize and effectively use the data proves even more important. This challenge of meaningful data utilization becomes especially significant in complex, dynamic environments such as modern transportation and robotic systems.

Telematics is one of the best examples of the power of data. It not only enables data collection from diverse vehicles but also gives organizations the power to use this information to make smarter decisions, optimize routes and improve vehicle functionalities. Unfortunately, up to the writing of this work, its full potential has not been fully explored. Current telematics focus mostly on basic functionalities, such as simple tracking, navigation, or trip logging, missing opportunities for deeper intelligence and predictive power.

Ever since the introduction of the term in 1978, telematics has referred to the integration of telecommunications and information technology to handle information [3]. Today, however, modern telematics capabilities extend far beyond this foundational purpose. By supporting real-time data acquisition, cloud-based analytics, and most recently, AI-driven decision-making, telematics systems have evolved to become central components of self-driving vehicles and smart infrastructure [4]. Combined with machine learning models, telematics has the potential to become a powerful tool capable of not only monitoring, but also predicting failures, forecasting energy usage, identifying optimal routes, and maximizing fleet efficiency.

Even with these advancements, some significant limitations remain. Current telematics systems typically function as descriptive or diagnostic tools. They report on what is happening or has already happened but rarely influence decisions autonomously. Consequently, crucial tasks like routing, maintenance scheduling, and energy management are often managed separately or based on rigid rules, such as running maintenance every fixed number of hours or charging below a certain threshold.
Integrating AI into telematics-based maintenance systems can significantly improve fleet performance. For example, Massaro et al. [5] developed an intelligent electronic control unit capable of collecting vehicle data and applying machine learning models to support predictive maintenance, driver behavior analysis, and fleet-level decision-making, showing that AI-enhanced telematics can meaningfully reduce downtime and optimize fleet operations. However, even these enhanced systems focus mainly on traditional vehicle fleets, leaving a gap in the integration of predictive intelligence into autonomous robotic fleets, where unique challenges, such as energy constraints, mechanical stress, and continuous operation, demand far more sophisticated decision-making capabilities.

This gap creates a significant research opportunity. Autonomous delivery robots and mobile robotic fleets are rapidly gaining relevance across various sectors, including logistics, retail, campus mobility, and smart cities. These systems require intelligent, adaptive decision-making, especially when operating continuously, since they must balance energy consumption, health monitoring, environmental variability, and dynamic workloads. Successfully integrating AI with telematics would enable robotic fleets to become self-optimizing and self-maintaining, thereby minimizing downtime, maximizing efficiency, and directly contributing to a smarter, more sustainable urban ecosystem.

The primary objective of this thesis is to investigate how telematics and artificial intelligence can be integrated to create a predictive, autonomous optimization framework for robotic fleets. Through the development of a simulated fleet environment, this work compares a traditional rule-based (baseline) telematics model with an AI-enhanced predictive model that incorporates energy forecasting, health estimation, and intelligent task scheduling. By contrasting these approaches, the thesis aims to demonstrate how AI-driven telematics can significantly enhance operational efficiency, reduce unplanned failures, and support broader smart-city initiatives.

2 Theoretical Background and Literature Review

2.1 Telematics: Foundations, Data, and Architecture

Introduced in 1978 from the French word *télématique*, telematics was originally defined as the combination of telecommunications (data transmission) and informatics (computerized processing) [3]. Since, telematics has evolved far beyond simple data transmission. It now serves as a sophisticated link between physical hardware and digital insights. By combining sensing technology with high-speed processing, telematics allows for the continuous tracking of mobile assets. Over the past decades, this concept has evolved into a technological foundation where vehicles and autonomous systems are no longer isolated but are instead constantly monitored and optimized to improve performance.

2.1.1 Evolution and Purpose of Telematics

Early telematics systems were fairly limited, relying mostly on basic GPS tracking and periodic diagnostic messages, enabling simple route visibility and security monitoring. Later on, in the late 90's, the introduction of onboard diagnostics (OBD-II) built the foundation for sending real-time vehicle data to remote operators over cellular networks, turning simple tracking into a much more powerful tool [6].

As communication networks advanced and embedded sensors became more capable, telematics evolved into a far more intelligent, connected system. Through the integration of high-bandwidth connectivity (4G/5G), multi-sensor data acquisition, and cloud-based analytical pipelines, telematics has expanded into a far more comprehensive data ecosystem.
As can be seen in Figure 2-1, modern telematics platforms allow the continuous observation of both the mechanical and operational states of vehicles and robots. This expansion enables real-time decision support and lays the groundwork for intelligent, automated fleet management [7].

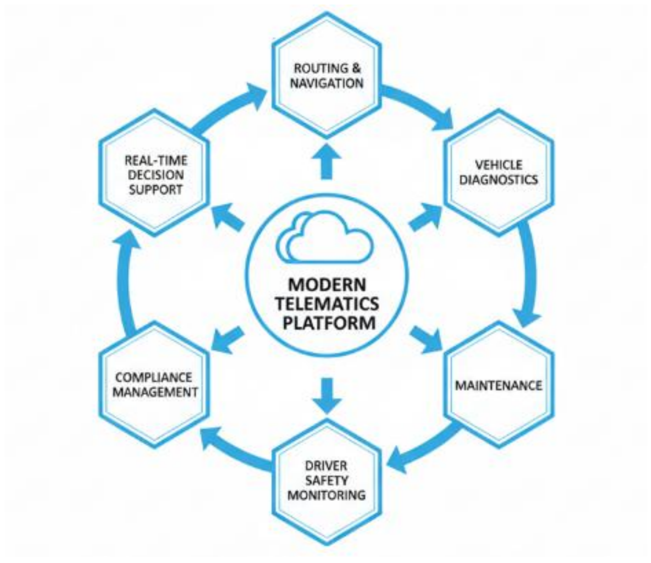

Figure 2-1 Core Capabilities and Interfaces of Modern Telematics Systems
Source: Author's own creation

Through this evolution, telematics has shifted from being a simple static reporting tool to becoming a dynamic digital infrastructure. By bridging the gap between raw data and action, modern telematics now provides the essential framework for truly intelligent and autonomous decision-making.

2.1.2 Communication Networks: CAN, OBD-II, FMS, and Gateways

Modern telematics systems rely on a layered communication architecture that connects onboard sensors, electronic control units (ECUs), diagnostic ports, and telematics gateways to remote cloud platforms, as illustrated in Figure 2-2. This architecture determines how data is acquired, transmitted, processed, and made available for different applications such as routing or diagnostics. Thus, making it fundamental for all telematics functionalities.

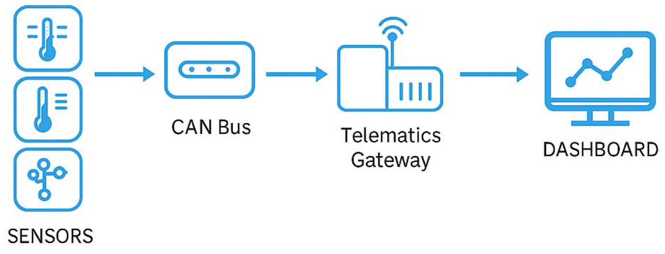

Figure 2-2 Generic Vehicle Telematics Communication Chain
Source: Author's own creation

2.1.2.1 Controller Area Network (CAN)

Introduced by Bosch in 1991 as a way to reduce wiring complexity and enable reliable communication between ECUs, Controller Area Networks (CANs) remain at the core of modern in-vehicle communication. CAN allows sensors, actuators, and controllers to communicate with each other by using a priority-based arbitration method, that ensures reliable communication even in noisy environments [8].

CAN is capable of supporting a wide variety of essential operational data, including:

- Motor performance: torque, RPM
- Vehicle status: brake signals, temperature values
- Energy metrics: battery voltage and current
- Fault indicators: diagnostic and error codes

Thanks to its reliability, fast response time, and ability to handle errors, CAN continues to be used as the main communication method in vehicles, agricultural machinery, industrial robots, and autonomous delivery devices.

2.1.2.2 OBD-II Diagnostic Interface

Introduced in the mid-1990s, On-Board Diagnostics II (OBD-II) plays a central role in modern telematics. Serving as an industry benchmark to access vehicle diagnostic information, OBD-II has the important goal of ensuring emissions regulations are consistently monitored. Through its standardized 16-pin diagnostic connector, displayed in Figure 2-3, OBD-II enables real-time access to engine and subsystem data [8, 9].

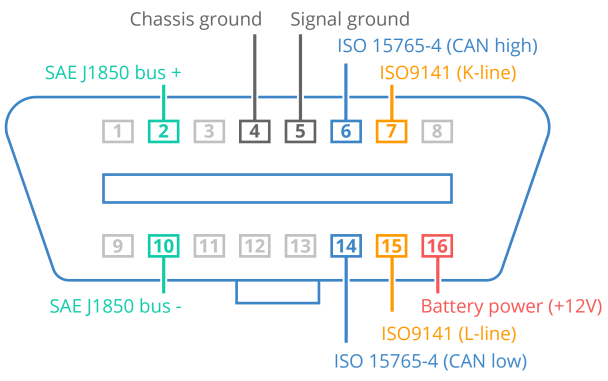

Figure 2-3 OBD-II Connector Pinout
Source: Image from [10]

Using Parameter IDs (PIDs), defined commands to request specific sensor readings, and Diagnostic Trouble Codes (DTCs) for systematic error sorting and identification
OBD-II provides access to several operational and diagnostic parameters, including:

- Engine performance (RPM, throttle position)
- Thermal status (coolant, air temperature)
- Vehicle speed
- System status (fuel pressure, run time).

"OBD-II's ability to provide standardized, real-time data access is why it remains central to traditional telematics. While autonomous robots do not use this specific interface, they rely on its fundamental logic: a reliable diagnostic gateway that provides a continuous window into the system's internal health.

2.1.2.3 FMS Gateway Integration

To overcome the complexity of proprietary CAN signals across vehicles, six major truck manufacturers created the Fleet Management System (FMS) Standard in 2002 [8]. Rather than requiring telematics devices to decode dozens of unique brand-specific signal structures, the FMS Standard processes and converts the vehicle's internal data into a consistent set of parameters, regardless of the vehicle's brand, model, or production year.

This data dictionary covers several parameters from vehicle and engine speed to fuel consumption, trip distance, and engine hours [11]. By unifying these metrics, the FMS Standard enables more efficient telematics tracking in mixed fleets, where consistent data is essential for route planning, performance monitoring, and intelligent transportation systems.

As shown in Figure 2-4, the FMS Gateway, alongside the OBD-II and CANs, collects vehicle information and forwards the standardized signals to an external telematics device, such as the Lantronix FOX4. These telematics devices then combine the FMS data with GNSS coordinates and mobile network connectivity, which then get transmitted to cloud-based fleet platforms to perform advanced functionalities such as routing, diagnostics, and predictive maintenance.

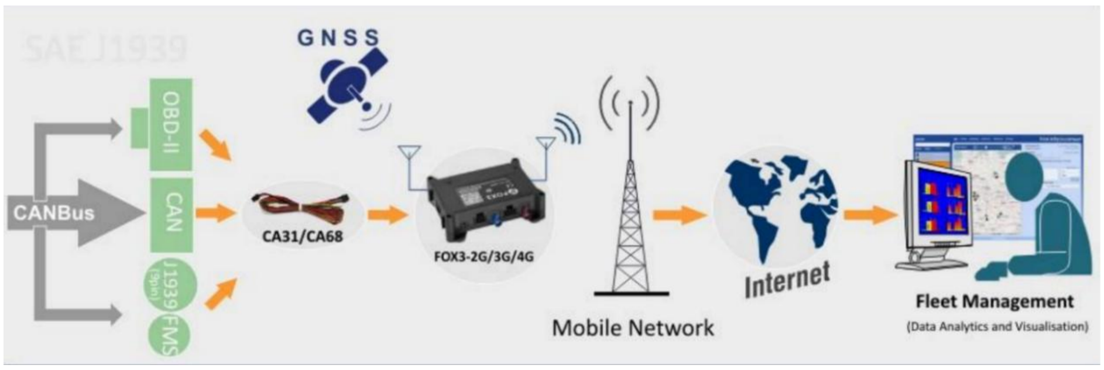

Figure 2-4 End-to-End Telematics Data Flow Architecture
Source: Image from [11]

Although autonomous delivery robots do not directly implement the FMS Standard, robotic fleets also require a unified method for accessing essential telemetry regardless of hardware differences. In this sense, the FMS Gateway serves as a conceptual design pattern for driverless, more compact vehicles by establishing the unified data foundation required for intelligent automation and closed-loop fleet management.

2.1.2.4 Telematics Control Units (TCUs)

As shown in Figure 2-4 with the FOX series example, Telematics Control Units (TCUs) serve as the primary communication center between a vehicle's internal systems and external telematics platforms. By integrating microcontrollers, wireless communication modules, GNSS receivers, transceivers, and power supplies, the TCU enables a bidirectional data exchange between the vehicle and the cloud. This capability is essential for continuous telemetry reporting, remote configuration, firmware updates, and event-triggered communication, making the TCU a primary driver of connected mobility solutions [12].

Unlike interface-specific standards, such as the OBD-II and the FMS, the TCU concept can be directly applied to autonomous robotic systems. These platforms require continuous access to telemetry, as well as the ability to receive remote configuration and software updates. As a result, TCUs provide the data foundation needed for AI-assisted optimization, predictive maintenance, and achieving closed-loop fleet management.

2.1.3 Types of Telematics Data

Telematics systems collect an extensive range of data. To achieve this, they utilize vehicle networks, diagnostic standards, GNSS receivers, and embedded sensors within the telematics device itself. These data streams form the required foundation to support monitoring, diagnostics, energy management, and higher-level analytics.

Table 2-A provides an overview of the primary data types relevant to both vehicle and robotic fleets, as well as a brief description and some example parameters.

Data Type	**Description**	**Example Parameters**
Operational & Performance Data	Real-time metrics reflecting engine/motor performance and supporting system functionality.	RPM, torque, engine load, battery voltage, coolant temperature.
Energy & Consumption Metrics	Indicators related to fuel or battery consumption, energy efficiency, and overall long-term performance.	Fuel level, fuel used, engine hours, state of charge, oil pressure.
Location & Movement Data	Geospatial and motion-related information used for tracking and route optimization.	GPS coordinates, speed, geofencing events, distance traveled.
Diagnostic & Fault Data	Standardized error messages and warnings used for maintenance and anomaly detection.	Diagnostic Trouble Codes, error or n/a messages.
Environmental & Motion Data	Data describing ambient or physical operating conditions that affect performance and safety.	Accelerometer readings, ambient temperature, yaw rate sensors.
System Health & Connectivity Data	Status information confirming the operation and communication stability of the telematics device itself.	Signal strength, GNSS quality, device uptime, firmware version, connection state.

Table 2-A Overview of Telematics Data Types
Source: Adapted from [13]; author's own evaluation

These data categories provide a better understanding of how vehicles and robotic fleets operate. By integrating these sources, it is possible to build robust databases that support predictive maintenance, route optimization, and AI-assisted fleet management, topics explored in detail in the following sections.

2.2 Artificial Intelligence and Machine Learning in Fleet Management

2.2.1 Artificial intelligence in Telematics

Modern telematics excel at data collection. The challenge arises, however, on knowing how to interpret it in a meaningful and actionable way. Artificial Intelligence (AI) and Machine Learning (ML) have the potential to provide the necessary analytical foundation to transform that raw telemetry into meaningful insights for prediction, optimization, and autonomous decision-making across vehicle and robotic fleets.

AI is usually defined as the field that allows systems to perform tasks usually associated with human intelligence, such as pattern-recognition, reasoning, and decision-making, by using computational methods and data-driven approaches [14]. Within this field, Machine Learning focuses on the use of algorithms to learn from data and thus improve its performance over time [15]. These abilities to quickly recognize operational patterns and anticipate future behavior, make AI and ML perfectly suited for the dynamic environments found in modern mobility systems.

The integration of AI into telematics can significantly enhance traditional monitoring by shifting systems from a reactive to a predictive and adaptive state. Rather than relying exclusively on descriptive indicators, such as current speed, battery levels, or static fault codes, AI-driven platforms leverage data to forecast battery degradation and estimate energy consumption for upcoming tasks. Furthermore, these systems can detect early signs of component failure to prevent breakdowns and calculate optimal routes by analyzing complex environmental and operational conditions in real time.

Beyond improving efficiency and reducing downtime, AI provides the necessary framework for scaling fleet operations. As the size of a fleet grows, manual human supervision is impractical due to the massive volume and frequency of data generated. With AI and ML models, these processes can be automated and optimized, to process information, prioritize critical events, and even propose or execute optimal actions. This shift enables the system to move beyond static, reactive decisions into a proactive regime, facilitating more sophisticated approaches to maintenance, tasking, and energy management.

2.2.2 AI for Predictive Maintenance

Predictive Maintenance (PM) has quickly become one of the most impactful applications of AI for fleet management and telematics. Traditional maintenance strategies usually follow either corrective (fixing after failure) or preventive (following fixed time-based schedules) approaches. While simple to implement, these methods often result in unnecessary maintenance, excessive idle time, or unexpected hardware/software failures. Artificial Intelligence introduces an innovative approach by using both real-time and historical data to forecast durability, predict failures, and optimize maintenance timing, consequently shifting the focus from repairing broken assets to maintaining healthy ones.

Machine learning models are ideal for predictive maintenance. They can identify complex patterns in sensor data that tend to be invisible or hard-to-spot for human operators or simple rule-based systems. By using technical features like changes in temperature, vibration patterns, voltage irregularities, engine load, or brake pressure, ML models are capable of recognizing upcoming failures. When integrated within telematics platforms, operators can be alerted before a failure occurs, transforming maintenance from a disruptive variable into a predictable, scheduled event.

A study conducted in 2020 demonstrated this capability by developing a smart electronic control board integrated with machine learning algorithms for bus fleets. Their system continuously collected vehicle data, processed onboard diagnostics, and applied AI to detect abnormal behaviors and predict risks of failure. The study concluded that predictive models could reliably anticipate malfunctions such as battery faults, overheating, and electrical issues, leading to fewer unplanned service interruptions and more efficient maintenance. Clearly showing how AI transforms simple diagnostic data into actionable predictions that enhance fleet reliability [5].

AI-driven PM is capable, not only of early failure detection, but also of contributing to operational cost reduction and road-safety improvement. Modern telematics gather endless amounts of data from CAN, OBD-II, GNSS, and external sensors [8], making them ideal environments for implementing predictive algorithms. With the additional support of TCUs streaming real-time diagnostics to the cloud, fleet operators can implement predictive models at scale without requiring powerful computers on every vehicle.

In the world of autonomous robotic fleets, predictive maintenance proves even more critical. Unlike standard vehicles, mobile robots run on smaller batteries, have lighter mechanical structures, and continuously operate. Meaning, their performance can degrade much faster due to diverse factors such as motor strain, temperature variations, uneven loads, or weather exposure. Hence, reliable predictive models are essential to prevent mid-operation breakdowns, optimize charging schedules, and ensure continuous task availability. By integrating predictive diagnostics into robot telematics, fleets can dynamically plan maintenance windows, which minimizes interruptions in smart city environments where continuous delivery or mobility services are expected.

AI-driven predictive maintenance directly supports the shift toward closed-loop fleet management, where robots not only report their status but also autonomously schedule charging, request maintenance, or adjust tasks based on their predicted health. This work aims to adopt such principles in a simulation, where predictive elements actively inform the task scheduling, routing, and energy management strategies.

2.2.3 AI for Energy Forecasting

Energy consumption is one of the most critical limitations in autonomous robotic fleets. Unlike regular vehicles, Autonomous Mobile Robots (AMRs) often operate with smaller battery capacities, limited power reserves, and more frequent charging cycles. Factors such as cargo weight, route elevation, motor efficiency, among others, also play a key role in AMRs' maximum travelable distance. This makes accurate energy forecasting essential to ensure robots can complete their assigned tasks, while preventing them from running out of power mid-route and maintaining predictable operations within smart cities.

Machine learning models have been shown to improve the prediction of fuel efficiency by integrating vehicle information, such as tire efficiency and emissions data, revealing non-linear correlations that would have otherwise proven challenging to discover through manual analysis [16]. Similarly, AI forecasting applied to electric and hybrid vehicles demonstrated that learning algorithms can predict energy demand much more accurately than classical methods, since they are capable of recognizing patterns in speed changes, idle periods, and temperature [17]. These proven findings can be seamlessly adapted to autonomous robots, since they rely on a lot of the same data streams to monitor their energy consumption.

For robotic fleets in real world applications, energy forecasting directly influences operational decision-making. Robots must constantly decide whether they can accept a delivery, if they need to reroute for different reasons, or if they must return to a charging station before battery levels become critical. Telemetry-powered AI supports these decisions by predicting whether the remaining charge is sufficient for any given task. Studies on autonomous logistics systems highlight how AI-driven route optimization and dynamic scheduling significantly reduce idle times and enhance the overall efficiency of delivery systems [18]. By building these predictive energy models into fleet management, charging queues can be optimized, and idle time minimized, which are necessary for the continuous service expected in smart cities.

2.2.4 AI for Route Optimization

Efficient routing is a crucial performance factor for both traditional vehicle fleets and autonomous robotic systems. Fleet operators must minimize travel time, reduce energy consumption, avoid congestion, and ensure timely task completion. For AMRs, routing becomes even more challenging due to their limited ranges, sensitivity to environmental conditions, and the need to dynamically coordinate with other robots in the same environment.

Classical routing methods in transportation typically use deterministic algorithms such as Dijkstra, A*, or variations of the Traveling Salesman Problem (TSP) [19] for static maps, however these methods alone do not account for energy constraints, deadlines, or system-level fleet coordination. Machine learning, in contrast, offers a necessary upgrade by incorporating live data, such as robot load, battery level, temperature, and historical patterns. This allows the system to generate optimal routes that evolve alongside changing circumstances, rather than sticking to a rigid, pre-calculated path.

By integrating telemetry, environmental conditions, and predictive models, AI-powered route optimization significantly improves routing decisions. These systems can identify emerging traffic patterns and predict delays, while computing routes to minimize energy consumption or avoid risks. Large-scale implementations, like the UPS ORION navigation system, which successfully reduced daily travel by up to 12 miles per driver [20], demonstrate the impact and benefits of dynamic, AI-powered routing.

To function effectively in a smart city environment, AI-driven route optimization must combine several key capabilities:

- Dynamic rerouting: automatically adjusting paths in response to obstacles, temperature changes, battery state-of-charge, or unexpected delays.
- Energy-aware navigation: predicting whether a given route can be completed with the available battery levels and minimizing high-consumption maneuvers, like steep inclines, when energy is low.
- Task-aware scheduling: selecting routes that align with delivery priorities, distance constraints, and required completion windows.
- Multi-robot coordination: managing traffic in shared spaces to prevent congestion, collisions, and deadlocks.

The impact and value of these capabilities can be observed in some recent studies. Mohsen [18] highlights that, in order to overcome urban congestion and reduce carbon footprints, AI-driven dynamic rerouting is essential. Similarly, Rinaldi et al. [21] demonstrate that energy-aware navigation and task-aware scheduling are no longer optional features, but critical constraints to prevent mission failure in battery-limited fleets operating under strict time windows.

In the context of this thesis, AI-based route optimization serves as the primary mechanism for developing a closed-loop fleet management model. By combining telematics with AI optimization, the goal is to enable robotic fleets to make intelligent decisions that maximize operational efficiency while remaining robust in dynamic environments.

2.2.5 Limitations of Current Systems

Although the integration of telematics and AI across mobility and logistics sectors has been on the rise, most current fleet management systems remain limited in their predictive and autonomous capabilities. Traditional telematics platforms function almost exclusively as descriptive tools. While modern TCUs are capable of collecting vast amounts of quality data [12], cloud backends and dashboards usually provide only retrospective analysis, in other words, they only report on what has already happened. Granting this type of tracking can be valuable, it is insufficient for autonomous robotic fleets, which need to continuously adapt to their environment rather than just react to errors after they occur. Without dedicated predictive layers, current telematics systems cannot support advanced functions, such as fully optimized routing, predictive charging, or early component wear detection, resulting in failures that could have otherwise been avoided [5, 17].

Another main concern in current telematic systems is the lack of a unified decision model, a problem stemming largely from data fragmentation. Vehicle data, maintenance records, energy metrics, environmental conditions, and other dynamic information are often stored and processed in isolation. Due to this lack of integration, systems are forced to rely on fixed thresholds and static logic. For example, an alert may only trigger when temperature exceeds a specific limit, or maintenance modules track errors but do not influence the actual workload distribution. Relying on these isolated metrics makes it nearly impossible to make accurate predictions, resulting in suboptimal planning and failures that could have otherwise been avoided [5]. In order to maximize fleet efficiency, integrated models capable of coordinating all these interdependent decisions are essential [18].

Finally, scalability remains a significant challenge. As fleet sizes increase, conventional monitoring methods that rely purely on human intervention become unsustainable. Current platforms still require manual management for basic functions like interpreting alerts, assigning tasks, and planning interventions. While AI has the potential to automate these workflows, AI-enhanced decision mechanisms remain underutilized, especially in smaller or mixed-vendor fleets where varying data standards (e.g., OBDII vs. FMS) and proprietary protocols act as barriers to widespread deployment [8]. These limitations are summarized in Table 2-B and clearly point to the need for a new framework that integrates telematics with AI-driven predictions and autonomous, real-time decision-making.

Current Limitation	**Operational Consequence**	**Autonomous Requirement**
Descriptive and retrospective data analysis: reports primarily on what has already happened.	Unable to explain root causes or predict failures before they occur.	Predictive data analysis: anticipates errors and continuously adapts to environmental changes.

Fragmented and threshold-based decision logic: logic operates in isolation using static limits.	Suboptimal planning ignores dynamic and interdependent variables like environmental conditions or component wear.	Unified and dynamic decision logic: a central model coordinating codependent decisions in real-time.
Human-Dependence: relies on manual intervention to interpret alerts and assign interventions.	Unsustainable for large-scale operations.	Automation: Closed-loop strategies where AI autonomously manages workflows.

Table 2-B Summary of Current Telematics Limitations and Autonomous Requirements
Source: Adapted from [5, 8, 17, 18]; author's own evaluation

2.3 Robotic Fleets and Autonomous Mobile Robots (AMRs)

Autonomous Mobile Robots (AMRs) are becoming a standard in urban logistics. Designed to navigate environments without human supervision, these systems rely heavily on telemetry, perception, and intelligent decision-making to operate reliably. As cities continue to digitize transportation systems and expand their logistics capabilities, robotic fleets represent one of the fastest-growing applications of autonomous technology in smart cities.

The rapid expansion of commercial delivery services demonstrates the importance of intelligent fleet management. Starship Technologies, which surpassed 8 million autonomous deliveries globally in 2024 [22], proves that robotic fleets are no longer just experimental prototypes. They are fully deployed in real-world environments, requiring telematics-driven coordination at a massive scale.

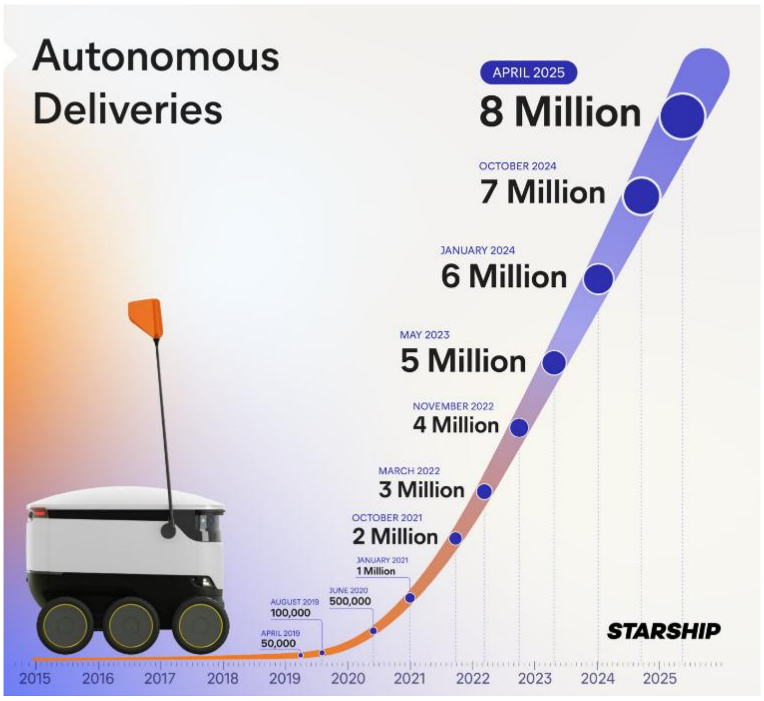

Figure 2-5 Autonomous Delivery Growth by Starship Technologies
Source: Image from [22]

Operating large robotic fleets introduces challenges that far exceed those found in traditional vehicle telematics. Unlike conventional automobiles, which benefit from high energy reserves and human oversight, delivery robots operate on strict energy budgets that limit both how much they can carry and how far they can drive. Simultaneously, they must be capable of localizing themselves accurately in dynamic environments, avoiding pedestrians, and maintaining thermal and mechanical stability.

Telematics helps solve these challenges by aggregating data from onboard sensors, such as motor currents, IMUs, and GNSS, to monitor system health alongside the vehicle's operating environment. Because robotic fleets have such fast dynamics, this telemetry needs to be high-resolution and low latency [17]. However, relying on descriptive data alone is insufficient, since static, rule-based management strategies cannot handle the variability of real-world operations. This is why, integrating predictive models for activities, like energy forecasting and adaptive task scheduling becomes essential to minimize deadlocks and significantly improve mission success rates [19].

As smart cities evolve into connected mobility ecosystems, AMRs are expected to integrate with external systems, like traffic signals, IoT infrastructure, charging networks, and real-time municipal logistics data. Such environments demand adaptive, predictive, and highly coordinated fleet management systems capable of operating with minimal human intervention. This establishes why robotic fleets require a fundamentally different telematics architecture than traditional vehicles, one that combines real-time sensing with predictive analytics, and intelligent decision-making.

2.4 Smart Cities and Connected Autonomous Mobility

Smart cities are all about integration. By combining digital infrastructure, intelligent transportation systems, and connected services, they improve the efficiency, sustainability, and safety of urban environments. The digitalization of logistics and mobility networks in these cities makes autonomous systems such as delivery robots, AMRs, and connected vehicles a standard in daily logistic operations. As such, their successful deployment depends on the seamless interaction between robotic fleets, sensor-rich infrastructures, and cloud-based coordination systems, which are made possible thanks to telematics and data-driven decision mechanisms.

In the context of smart urban logistics, telematics plays a central role by enabling real-time monitoring, communication, and coordination of autonomous vehicles. Effective smart city frameworks rely on the data sharing capabilities between AI, IoT devices, and autonomous vehicles to optimize delivery and traffic flows, as well as to increase operational resilience [18]. By facilitating this exchange, telematics allows mobility systems to react dynamically to changing road conditions, traffic density, and infrastructure status.

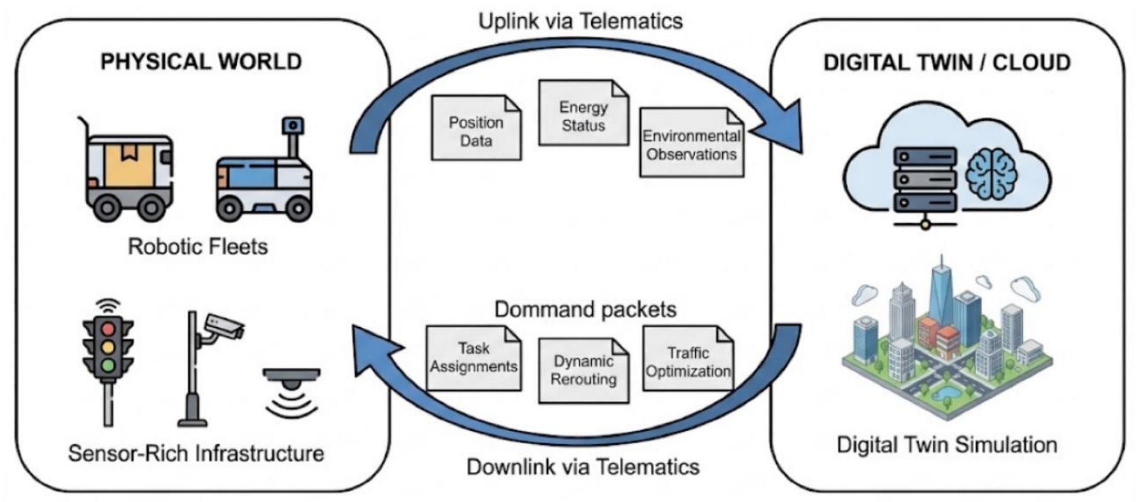

Figure 2-6 Data Driven Feedback Loop in Smart Cities
Source: author’s own creation

To operate efficiently within a smart-city ecosystem, robotic fleets must transition away from isolated operation and remain continuously connected to a network of key infrastructure elements. This integration involves coordinating with smart delivery hubs to manage parcel storage and transfer, alongside charging stations that demand precise scheduling to prevent congestion and minimize downtime. Furthermore, these fleets rely on environmental sensing networks for real-time data on variables such as temperature, humidity, and surface conditions, as well as IoT-enabled infrastructure to avoid conflicts with vehicles and pedestrians.

Despite significant progress, current smart-city mobility systems still lack unified models that combine telematics, AI-driven forecasting, and autonomous fleet coordination into a single, closed-loop framework. Existing deployments tend to separate routing logic, energy estimation, and task allocation rather than integrating them into unified systems. This fragmentation limits the true operational potential of robotic fleets. Addressing this gap requires models capable of continuously analyzing telemetry, diagnosing risks, forecasting energy needs, and dynamically adjusting robot behavior.

2.5 Research Gap

While there has been significant progress in telematics, predictive maintenance, and autonomous navigation, these fields are still treated as separate disciplines. Current telematics systems are generally limited to reporting what has happened, while AI applications usually focus on optimizing one specific task at a time. Similarly, research on Autonomous Mobile Robots (AMRs) typically focuses on how a single robot navigates, rather than how a fleet coordinates in real-time.

However, in a smart city, these variables are deeply interconnected. A fleet cannot function efficiently unless it can simultaneously analyze telemetry, forecast energy needs, and manage component health under real-world constraints. Current research has yet to fully establish a framework that unifies these data streams with predictive models to optimize routing, scheduling, and energy management as a single process. This thesis aims to bridge that gap by proposing an AI-enhanced telematics framework, validated within a simulated environment, that integrates these distinct tasks into a cohesive decision-making model.

3 System Design and Simulation Framework

3.1 System Overview

In order to evaluate how telematics and artificial intelligence can effectively optimize the operational performance of autonomous robotic fleets, this paper proposes the simulation of a closed-loop fleet management system.

In the simulation, robots continuously transmit sensor and operational data to a central decision engine, which then calculates real-time commands for task assignments, routing, charging, and maintenance scheduling. This continuous feedback cycle effectively creates a digital twin (a virtual replica that mirrors the live state of all robots) of the robotic fleet, with the aim of accurately mirroring how modern smart-city mobility infrastructures operate. This continuous feedback loop can be observed in Figure 3-1.

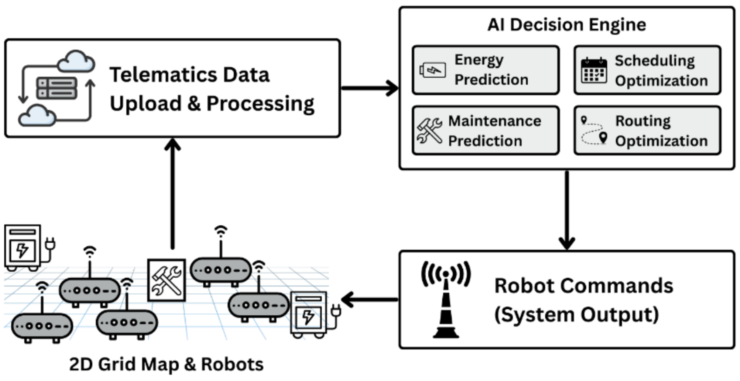

Figure 3-1 Closed-Loop Telematics Architecture for Autonomous Robotic Fleets
Source: author's own creation

The process functions as follows:

1. Telemetry Uplink: First, the robots transmit their operational data, including current position, battery state of charge (SoC), internal temperature, health status, and task progress.
2. Digital Twin & Telematics Processing: This incoming data is then validated and combined. Creating a unified, real-time state representation of the entire fleet within the simulation.

3. AI-Enhanced Decision Engine: Based on the grouped state, the system runs its predictive models. These models estimate future energy consumption, evaluate scheduling risks, and anticipate maintenance needs to compute the optimal assignment and charging decisions.
4. Command Downlink: Once the decisions are made, updated commands (such as which robot should take a new job, when to head to a charger, or when to enter maintenance) are transmitted back to the specific robot.
5. Actuation & Behavior: Finally, the robots execute these commands in the simulation environment. Their actions generate new telemetry, which triggers the cycle again, effectively closing the feedback loop.

This architecture is designed to be consistent with connected mobility systems in smart cities, where traffic analytics, vehicle telemetry, and cloud-based optimization interact continuously. While this simulation operates on a simplified 2D grid, its structure was designed to conceptually align with real-world fleet telematics platforms used in industry.

3.2 Simulation Environment

The simulation environment is designed as a simplified representation of a city area, designed to enable controlled experimentation on routing, energy consumption, scheduling, and maintenance decisions, attempting to imitate the spatial and operational constraints typically encountered by delivery robots in real cities.

3.2.1 Grid Layout and Navigation Model

The system operates in a 20×20 grid where every reachable cell represents a valid location a robot can occupy. To simulate real-world constraints, static obstacles can be placed arbitrarily to represent inaccessible regions such as buildings, construction sites, or restricted-access zones, which need to be taken into consideration by the system when planning routes. A visual example of this layout is shown in Figure 3-2.

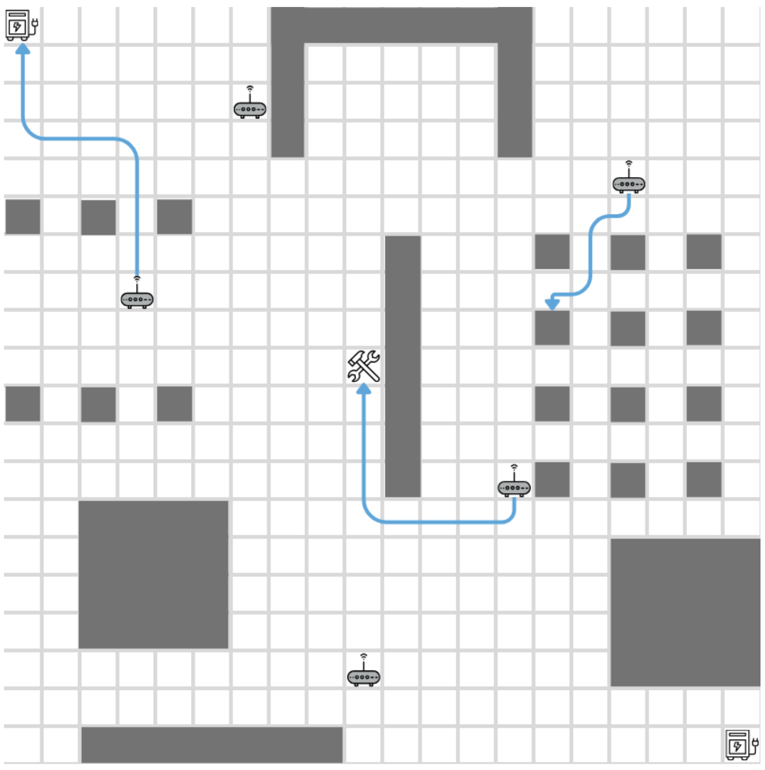

Figure 3-2 Simulation Environment 20×20 Urban Grid Example
Source: author's own creation

To handle navigation, the system interprets this grid as a graph, where every reachable cell is a node connected to its orthogonal neighbors by an edge. This structure allows the robots to utilize Dijkstra's algorithm to navigate efficiently from its current position to any target, whether that is a pickup, drop-off, charging station, or the maintenance depot [23, 24].

This grid-based approach and the robots' orthogonal movement through it, allow the use of the Manhattan metric [25] to calculate the distance between any two cells at any given time:

$$d_{Manhattan}(a,b) = |a_x - b_x| + |a_y - b_y| \tag{3.1}$$

The use of this function allows the system to estimate travel costs, predict battery usage, and prioritize robots based on their proximity to a task.

3.2.2 Charging Stations and Maintenance Depot

To ensure full coverage across the environment, two charging stations are positioned at opposite corners of the grid (coordinates (0,0) and (19,19)). Robots can route themselves to either of these stations whenever their SoC drops below the threshold determined by the active decision policy (baseline or AI-enhanced).

For maintenance operations, a depot is located at the center of the grid (10,10). Robots are routed here when their health degrades beyond an acceptable limit or when maintenance is proactively recommended by the AI decision module.

The placement of these facilities was strategically kept simple and symmetric, in order to keep the focus of the simulation on the decision-making logic.

3.2.3 Task Modeling and Job Generation

To capture the behavior of a broad range of urban delivery scenarios, without being forced to mimic the specific constraints of a single service model (such as food or parcel delivery), delivery tasks are modeled as pickup–drop-off pairs located anywhere on the accessible grid. Each delivery task is fully defined by a pair of independently chosen locations (pickup and drop-off, both selected from valid, obstacle-free cells) and several critical operational attributes, such as a specific payload weight, a priority level, and a completion deadline. This structure captures the essential characteristics of urban logistics: the robot must reach the pickup, collect the payload, and deliver it, all while managing battery constraints and health degradation.

To simulate dynamic demand, new jobs are set to arrive stochastically. At every simulation step, a new job has a 5% probability of being created. For example, a standard run consists of 2,000 simulation steps, which would result in approximately 100 new tasks on average. By using this probabilistic approach, realistic fluctuations are created without losing control over the test conditions.

3.3 Robot Model

Each robot in the simulation is designed to represent an autonomous delivery unit equipped with sensing, mobility, and telematics capabilities. With this model, all key operational factors relevant to routing, energy consumption, scheduling, and predictive maintenance are captured.

The robots operate on the grid defined in Section 3.2 and move one cell per simulation step along their planned routes. Their overall behavior is determined by both internal states, such as battery level, temperature, and mechanical health, as well as external demands, like job assignments, charging or maintenance needs. This behavior is summarized in the state machine diagram shown in Figure 3-3.

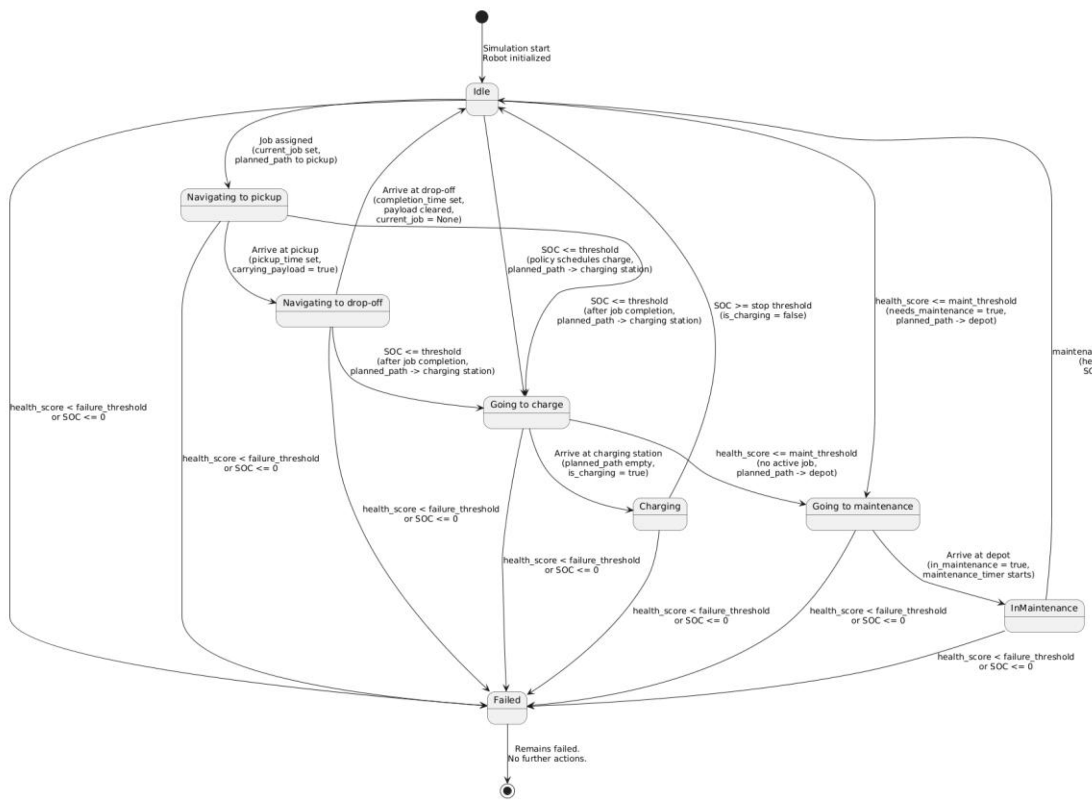

Figure 3-3 State Diagram: Robot Class
Source: author's own creation

At every step of the simulation, the decision-making system uses the following variables to define the dynamic state of each robot:

- Position (x, y): current grid location.
- State of Charge: battery level as a fraction $[0, 1]$.

- Payload Weight: weight of the carried package in kg.
- Internal Temperature T: approximate internal temperature in degrees Celsius.
- Health Score h: hardware integrity in [0,1], where 1.0 is fully operational.
- Planned Route: sequence of grid cells calculated using shortest-path routing
- Job Assignment Variables: current job and its related status flags (pickup/drop-off status, timestamps).
- Flags: *is_charging, is_idle, needs_maintenance, in_maintenance,* and *failed.*

These internal variables ultimately determine how the robot uses energy, degrades over time, and how the simulation policies assign work or trigger maintenance actions.

3.3.1 Movement and Energy Consumption Model

Robots move through the grid at a fixed speed of one cell per step along the shortest paths calculated by Dijkstra's algorithm. They only become stationary when waiting for charging, undergoing maintenance, or pausing due to policy updates.

This movement constitutes the primary source of energy drain. Elements such as distance travelled and payload weight, help define the change in SoC per step, as can be seen in Formula 3.2., where α_d is the base energy cost per cell and $\alpha_w \cdot w$ the additional cost per payload weight.

$$\Delta SOC = \frac{\alpha_d + \alpha_w \cdot w}{Battery\ capacity} \tag{3.2}$$

Once SoC drops below a policy-defined threshold (fixed for baseline, dynamic for AI mode), robots stop movement after job-completion and route themselves to the nearest charging station. During charging, SoC increases at a constant rate until a predefined stopping threshold is reached.

3.3.2 Health Degradation and Maintenance Strategy

Baseline mode follows a "problem-solving" logic. A robot will continue operating until there is a system or mechanical failure. Only then will the robot go offline and stop operating until the end of the run.

On the other hand, the AI mode uses a health score system, to help determine whether a robot needs to go to the maintenance depot. Factors such as baseline wear (H_{base}), distance moved ($H_{dist} \cdot d$), and penalties for low State of Charge (H_{lowSOC}) and high internal temperature ($H_{highTemp}$) change a robot's health score per step (Δh). This degradation is modeled as:

$$\Delta h = H_{base} + H_{dist} \cdot d + H_{lowSOC} + H_{highTemp} \tag{3.3}$$

The accumulated degradation gives robots a score between 0 and 1. Thus, allowing the AI model to determine whether a robot can continue operating or if it should be directed to the maintenance depot. This reflects the difference between reactive and predictive maintenance strategies.

3.3.3 Telemetry Noise and Failure Conditions

To ensure the decision engine operates realistically, all telemetry readings (SoC, temperature, health estimates) include Gaussian noise to simulate sensor inaccuracies and measurement uncertainty. For any given measurement x, the reported value $\tilde{x}$ is not a simple deterministic reading but is drawn from a normal distribution centered around the true value, with a standard deviation σ_x specific to the variable x. This relationship is modelled as:

$$\tilde{x} \sim \mathcal{N}(x, \sigma_x^2) \tag{3.4}$$

Where σ_x^2 is the variable-specific noise standard deviation and the level of this uncertainty is predefined in the simulation parameters.

Since sensor data in real-world conditions can be unreliable and noisy, robots must run safety checks. If this check determines the robot can no longer operate safely or reliably, it immediately switches to a Failed state. Failure can occur if:

- Critical Health: The health score falls below a critical threshold ($h < 0.1$).
- Power Loss: The State of Charge reaches zero ($SOC \leq 0$).

This failure is permanent, representing a critical operational challenge that reduces total fleet capacity due to the sudden, unexpected loss of assets.

3.4 Main Simulation Loop

The simulation relies on a main loop to advance the system state in discrete time steps. Each iteration represents a single time unit, during which the system executes a defined logic sequence to manage the fleet. Figure 3-4 illustrates this sequential process.

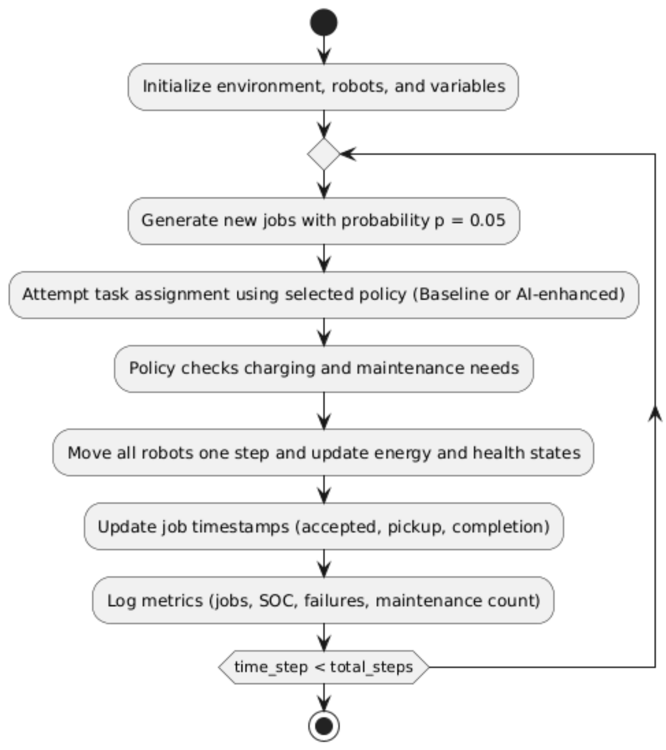
Initialize environment, robots, and variables
Generate new jobs with probability p = 0.05
Attempt task assignment using selected policy (Baseline or AI-enhanced)
Policy checks charging and maintenance needs
Move all robots one step and update energy and health states
Update job timestamps (accepted, pickup, completion)
Log metrics (jobs, SOC, failures, maintenance count)
time_step < total_steps

Figure 3-4 Activity Diagram: Main Simulation Loop
Source: author's own creation

The loop consists of the following key phases:

1. Job Generation: New delivery tasks are created probabilistically (currently set at $p = 0.05$ per step) to simulate dynamic demand. Each generated job is assigned specific attributes:
    a. pickup and drop-off coordinates,
    b. payload weight,
    c. a priority level,
    d. and a hard deadline.
2. Task Assignment: This phase varies depending on the selected control strategy.
    a. In Baseline mode, the system assigns jobs using the "nearest-robot" logic.

   b. In AI-enhanced mode, the system utilizes predictive energy models, dynamic thresholds, and cost-based optimization to make assignment decisions.

3. Charging & Maintenance Scheduling: The policies evaluate every robot's State of Charge (SoC) and accumulated health degradation. Based on these values, the system decides whether a robot must break from its routine to route to a charger or the maintenance depot.
4. Robot Movement & Energy Consumption: Robots physically advance along their planned paths on the grid. As they move, the system calculates energy depletion based on the current payload and distance traveled, while concurrently increasing the health degradation factor based on operational strain.
5. Job Lifecycle & Metric Logging: Finally, the system updates critical timestamps (acceptance, pickup, completion) when robots reach relevant coordinates. It also records global performance metrics, such as the number of active jobs, average fleet SoC, robot failures, and charging queue sizes, to allow for detailed post-simulation analysis.

A high-level view of this logic, can be found in Pseudocode 3-1 in Appendix A.

3.5 Job Lifecycle

With the objective of mimicking real-world last-mile delivery scenarios, every delivery order that enters the simulation goes through a structured process, moving through different stages like dispatching, pick up, transport, and final delivery.

As outlined in Figure 3-5, at each simulation step, the system has a 5% probability of generating a new job. Once created, the job remains active until it is either assigned to a robot, successfully delivered, or the simulation run ends without it being completed. This model allows both the baseline and AI systems to make routing and scheduling decisions, while simulation real-world conditions.

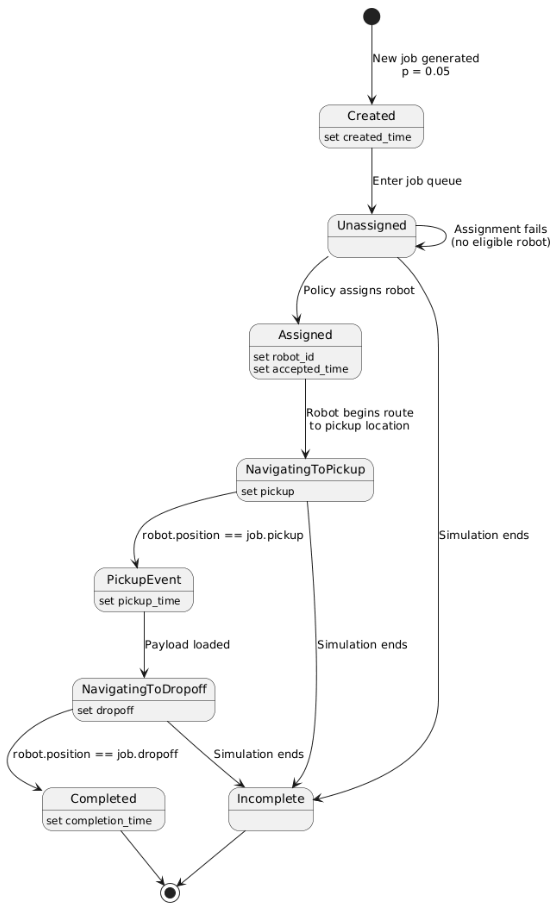

Figure 3-5 State Diagram: Jobs Lifecycle
Source: author's own creation

3.5.1 Job Creation and Assignment Phase

New jobs are created probabilistically ($p = 0.05$ per step) to simulate a constant but unpredictable arrival of delivery requests. For each new job, the system assigns:

- pickup (x_p, y_p) and drop-off (x_d, y_d) locations
- a payload weight w randomly chosen from 1 to 8 kg
- a deadline offset randomly chosen between 30 and 200 simulation steps
- a priority integer between 1 and the maximum number of robots
- a creation time t

These attributes allow the decision-making policies to evaluate the urgency, feasibility, and energy requirements for the task.

Once a job has been created, it can be assigned based on the active policy. The baseline assigns the job to the nearest available robot with a State of Charge above a fixed threshold. While the AI policy evaluates predicted energy requirements, deadline flexibility, and priority to select the best candidate.

If a robot is selected, the job records the assignment timestamp t_{accept}, and the robot receives a planned route to the pickup and drop-off location. If, however, no robot qualifies, the assignment attempt is logged but no state change occurs, and the job remains unassigned until a suitable robot becomes available.

3.5.2 Pickup and Delivery Phases

A job moves into the Pickup Phase when the robot's position is the same as the job pickup location. At this moment, the robot records its pickup time, loads the payload (increasing energy usage proportional to the payload weight), and is immediately routed along the second portion of its planned path. The job is then successfully Delivered when the robot's position equals the job drop-off location. At this final event, the robot unloads the payload, clears its current job attribute, and the completion time is logged for metrics. Any job that does not reach the drop-off location by the end of the simulation is considered incomplete.

3.5.3 Completion Metrics

To properly analyze and compare the performance of the baseline and AI models, the simulation records several key Completion Metrics. Beyond tracking the creation, assignment, pickup, and completion times, the system also records the delay relative to the deadline and the assignment success status. These records are essential to assess total jobs completed, timeliness, fleet capacity utilization, and overall assignment efficiency.

4 Experimental Setup & Evaluation

4.1 Experimental Setup

To properly evaluate the impact of telematics-supported decision-making on autonomous fleet performance, the simulation was executed in repeated experiments, comparing two different control strategies:

1. a Baseline policy with simple, rule-based dispatching and maintenance behavior
2. an AI-inspired predictive policy, that incorporates estimated energy requirements and health aware thresholds when assigning jobs and scheduling charging and maintenance.

All experiments use the same environment, robot model, and job generation process described in Chapter 3.

4.1.1 Common Simulation Parameters

To represent a simplified urban area, the experiments use a fixed 20 × 20 grid environment containing static obstacles, two charging stations located at opposite corners, and one centrally placed maintenance depot. Five identical robots operate in this environment and share the same internal dynamics for movement, energy consumption, temperature, and health degradation.

Table 4-A summarizes the main simulation parameters and constraints applied across all policy comparisons.

Parameter	**Value/Range**
Number of Robots	5
Grid Size	20 × 20 cells
Single Run Length	2,000 steps
Job Creation Probability	0.05
Job Deadline	30 to 200 steps
Job Priority	1 (low) to 5 (high)
Payload Weight	1 kg to 8 kg
Battery Capacity	1.0 kWh
Charging Rate	0.02 kWh/step

Maintenance Duration	80 time steps

Table 4-A Simulation Main Configuration Parameters
Source: author's own creation

Alongside these parameters, telemetry noise plays a crucial part in the simulation. In order to achieve a realistic representation of imperfect sensor measurements, telemetry noise is applied to the internal robot variables. The noise is implemented by applying Gaussian distribution with zero mean and a variable-specific standard deviation (e.g. $\sigma_{SOC} \approx 0.01, \sigma_{temp} \approx 0.5\ °C$). These values were chosen to be small enough not to dominate the system dynamics, yet substantial enough to accurately reflect the uncertainty present in real-world telematics data.

A single simulation run consists of 2,000 time steps, where up to roughly 100 delivery jobs are generated, assigned, executed, charged, and possibly delayed or dropped depending on robot availability, energy levels, and failures.

4.1.2 Policies Comparison

Both policies rely on the same low-level robot model and only differ in how they make decisions based on the available telemetry, as observed in Table 4-B.

Feature	**Baseline Policy**	**AI-Inspired Policy**
Job Assignment	Based on the nearest available robot with an SoC above a fixed threshold (e.g., 20%).	Estimates energy required for the entire job (pickup to drop-off) and assigns only if the robot has enough energy plus a safety margin.
Allocation Logic	Nearest available robot.	Minimizes a cost function that combines distance to pickup, remaining slack to deadline, and job priority.
Charging Decisions	Triggered only when a robot's SoC falls below the fixed threshold (e.g., 20%) and it has no active job.	Uses a dynamic SoC threshold that increases as the robot's health declines, routing older/less healthy robots to charge earlier.

Maintenance Strategy	Reactive: Only triggered when health drops below a relatively low limit. Robots are driven close to failure before withdrawal.	Proactive: Triggered once health passes a higher threshold (and no job is active), routing the robot to maintenance before failure occurs.
Core Philosophy	Simple, rule-based controller. Decisions made on immediate, static criteria.	Telematics-supported controller. Decisions utilize predictive information (energy and degradation estimates).

Table 4-B Baseline vs AI-Inspired Policy Comparison
Source: author's own creation

These two strategies represent a clear comparison between a simple rule-based fleet controller and an advanced telematics-supported controller that utilizes predictive information about energy and degradation.

4.1.3 Multi-Run Evaluation Procedure

To obtain statistically meaningful results and reduce the influence of randomness in job arrivals, the experiments are carried out as multiple independent runs:

- Number of Runs: 100 independent iterations.
- Modes: Each run is executed twice, once under the Baseline policy and once under the AI-inspired policy.
- Fair Comparison: To ensure a direct and fair comparison of operational efficiency, both policies observe the exact same job sequence (locations, weights, deadlines) for any given run index.

The simulation records per-step, per-job, and per-assignment data, which can then be collected to compute run-level metrics such as the average number of completed jobs, failure counts, and deadline adherence. The following sections define these metrics formally and present the comparative results.

4.2 Metrics Definitions

To allow an objective comparison between the Baseline and AI-inspired control strategies, a set of quantitative performance metrics has been defined. These metrics are derived directly from the simulation logs and exported datasets, capturing both operational efficiency and system robustness. For reliable reporting, all metrics are first calculated per simulation run and then aggregated across the complete set of runs.

4.2.1 Job Completion Metrics

The Job Completion Metrics quantify the fleet's ability to fulfill the generated delivery demand.

4.2.1.1 Job Completion Count

The Job Completion Count displays the total number of delivery jobs successfully completed within a run, where $1(\cdot)$ is an indicator function equal to 1 if the job is completed, and 0 otherwise.

$$N_{completed} = \sum_{j=1}^{N_{jobs}} 1(j\ completed) \tag{4.1}$$

4.2.1.2 Job Completion Rate

The Job Completion Rate is a fraction of generated jobs that are successfully completed. This reflects the overall effectiveness of the fleet in fulfilling demand.

$$Completion\ Rate = \frac{N_{completed}}{N_{generated}} \tag{4.2}$$

4.2.2 Timeliness Metrics

Timeliness Metrics evaluate how well the policy manages urgency and adheres to deadlines.

4.2.2.1 On-Time Delivery Rate

The On-Time Delivery Rate measures the ratio of completed jobs that meet their assigned deadline. This evaluates how well the policy manages time-critical tasks and deadline pressure.

$$OnTime\ Rate = \frac{N_{completed\ on\ time}}{N_{completed}} \tag{4.3}$$

A job is considered on time if:

$$completion\_time_j \leq deadline_j \tag{4.4}$$

4.2.2.2 Average Delivery Delay

For jobs completed after their deadline, the Average Delivery Delay is defined as:

$$Average\ Delay = \frac{1}{N_{late}} \sum_{j \in late\ jobs} (completion\_time_j - deadline_j) \tag{4.5}$$

This helps capture how severely deadlines are missed when delays occur.

4.2.3 Fleet Energy Metrics

Fleet Energy Metrics monitor energy consumption and the efficiency of the charging process.

4.2.3.1 Average State of Charge

The Average State of Charge is the mean battery state of charge across all robots, averaged over all simulation steps.

$$\overline{SOC}(t) = \frac{1}{N_{robots}} \sum_{r=1}^{N_{robots}} SOC_r\ (t) \tag{4.6}$$

4.2.3.2 Charging Utilization

Charging Utilization calculates the average number of robots actively charging per time step. This metric is meant to reflect how efficiently energy replenishment is scheduled and whether robots are being excessively withdrawn from service for charging.

$$Charging\ Utilization = \frac{1}{T}\sum_{t=1}^{T} N_{charging}\,(t) \tag{4.7}$$

4.2.4 Health and Maintenance Metrics

The Health and Maintenance Metrics help quantify the hardware wear and the effectiveness of the maintenance strategy.

4.2.4.1 Average Robot Health

The Average Roboth Health calculates the mean health score across all robots, averaged over all simulation steps. Lower values indicate higher cumulative wear and degradation.

$$\overline{H}(t) = \frac{1}{N_{robots}} \sum_{r=1}^{N_{robots}} H_r\,(t) \tag{4.8}$$

4.2.4.2 Maintenance Frequency

Maintenance Frequency is defined as the number of robots that enter the maintenance state at least once during a simulation run, where 1(·) is an indicator function equal to 1 if robot r enters maintenance at least once during the run, and 0 otherwise.

$$N_{maintenance} = \sum_{r=1}^{N_{robots}} 1\,(r\ enters\ maintenance) \tag{4.9}$$

4.2.4.3 Failure Count

Failure Count shows the number of robots that experience a breakdown, whether due to critical health or battery depletion, during a run.

$$N_{failures} = \sum_{r=1}^{N_{robots}} 1\,(H_r \leq H_{min} \vee SOC_r = 0) \tag{4.10}$$

4.2.5 Assignment Efficiency Metrics

Assignment Efficiency Metrics evaluate the reliability of the task-allocation logic and the system's ability to successfully connect robots with pending jobs.

4.2.5.1 Assignment Success Rate

The Assignment Success Rate quantifies the effectiveness of the scheduling and resource allocation logic, measured as the proportion of all assignment attempts that result in a successful robot-job allocation:

$$Assignment\ Success\ Rate = \frac{N_{successful\ assignments}}{N_{assignment\ attempts}} \tag{4.11}$$

4.2.6 Aggregation Across Runs

All defined metrics are computed individually for each simulation run. The results presented in the next section are obtained by aggregating these individual metrics across the 100 independent runs using mean values and standard deviation. This multi-run evaluation methodology is crucial as it minimizes the influence of random fluctuations in job generation sequences and initial system conditions.

4.3 Results and Discussion

This section presents and interprets the quantitative results obtained from the multi-run simulation experiments described in the previous sections. The goal is to compare the performance of the Baseline (rule-based) and AI-inspired (telematics-supported) control strategies across key dimensions, including task fulfillment, energy management, system reliability, and maintenance behavior. Unless stated otherwise, all reported values represent averages across the 100 simulation runs.

4.3.1 Overall Job Performance

Table 4-C summarizes the total job generation and completion results for both control strategies across all simulation runs.

	Jobs generated		**Jobs completed**		**Completion rate**	
Mode	**Mean**	**Standard Deviation**	**Mean**	**Standard Deviation**	**Mean**	**Standard Deviation**
Baseline	100.58	9.04	59.96	20.67	0.60	0.20
AI	100.71	11.05	98.52	10.84	0.98	0.02

Table 4-C Overall Job Completion Performance per Control Strategy
Source: author's own creation

As can be observed, the AI-inspired strategy consistently completes a significantly higher number of delivery jobs than the Baseline approach. This improvement can be primarily attributed to two factors:

1. energy-aware task assignments, which prevents robots with insufficient battery reserves from accepting infeasible jobs, and
2. dynamic charging decisions, which reduce the number of robots becoming unavailable due to mid-operation battery depletion.

In contrast, the Baseline strategy usually assigns jobs based only on how close they are. This often results in deliveries being abandoned or delayed, due to insufficient battery, as energy constraints are not factored into the initial decision.

Looking at the boxplots in Figure 4-1, which map the completed jobs per run, the difference is made clearer. The Baseline strategy's broad box and long lines indicate that its performance is unstable and highly unpredictable. The AI-inspired strategy, however, shows a narrower box, demonstrating it maintains a more steady, reliable output. Ultimately, the concentrated distribution of the AI-inspired strategy's output demonstrates a higher level of predictability and operational resilience, establishing it as the more robust choice for large-scale fleet management.

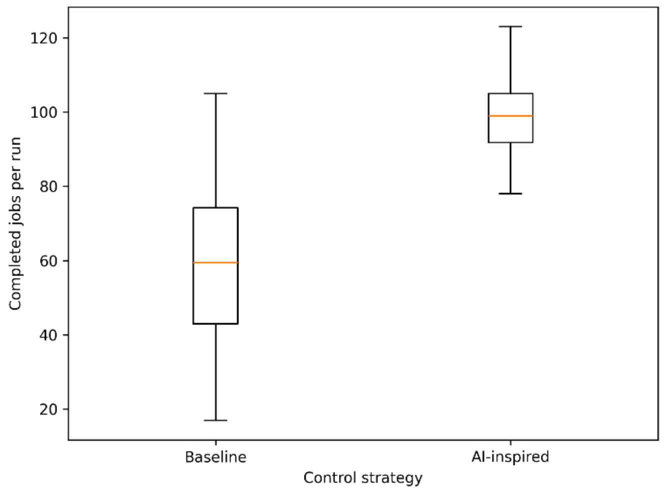

Figure 4-1 Distribution of Completed Jobs per Simulation Run (Baseline vs. AI-Inspired Control)
Source: author's own creation

4.3.2 Fleet Operational States and Temporal Effects

Figure 4-2 illustrates the temporal evolution of fleet status for both the Baseline and AI-inspired control strategies. These data points represent the average count of robots occupying specific operational states (charging, undergoing maintenance, or failed) at each discrete simulation step. By analyzing these fluctuations, it is possible to observe how different approaches manage resource availability and respond to the cumulative stresses of continuous operation.

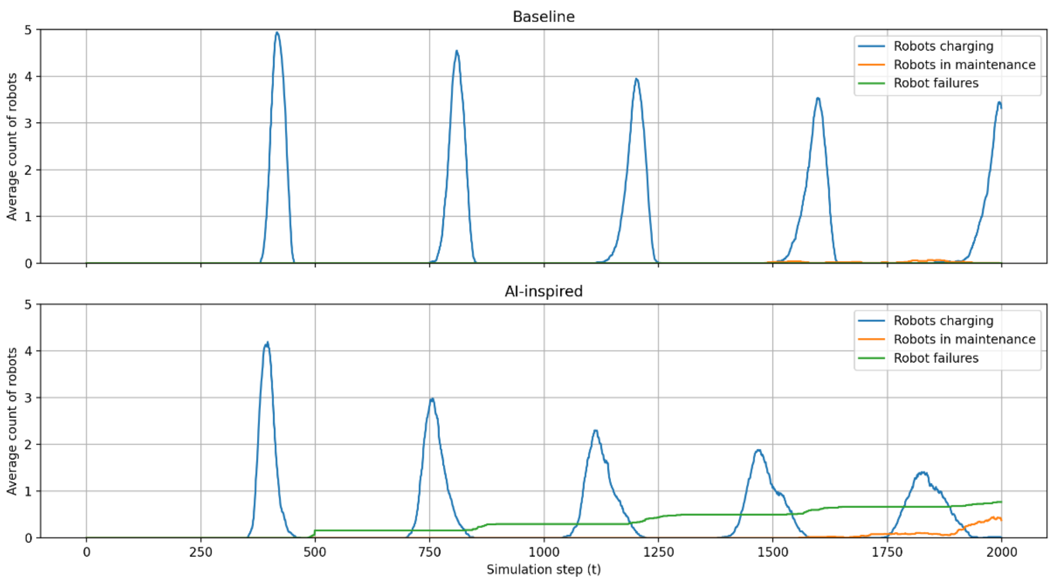

Figure 4-2 Fleet Operational States Over Time (Averaged Across Runs)
Source: author's own creation

The two strategies function relatively differently in practice. The Baseline strategy exhibits pronounced, periodic peaks in charging activity. This happens due to the nature of the reactive policy that only triggers charging once a fixed threshold is reached, thus making multiple robots frequently hit their battery limits simultaneously. This synchronization causes robots to become unavailable in groups, significantly reducing fleet flexibility during high-workload periods.

On the other hand, the AI-inspired strategy creates smoother, more evenly distributed charging patterns. Instead of waiting for a battery to hit a specific limit, it uses flexible thresholds based on both battery levels and the robot's overall health. This approach prevents robots from all needing to charge at the same time, keeping more of them active and ready to work. As a result, the fleet is more available and consistently finishes more tasks.

An interesting observation in Figure 4-2 is that the AI-inspired strategy actually shows a slightly higher average of robot failures. However, this does not actually mean the performance is inferior. Instead, it is a result of the fleet being much more active. Due to the higher volume of completed deliveries in AI-controlled robots, they undergo more operational stress and travel further than those in the Baseline group. The higher failure rate is essentially a trade-off for a much more productive fleet that is being used to its full potential, which further emphasizes the importance of predictive maintenance, since it allows the system to detect this accelerated degradation early, long before a total breakdown can occur.

These results demonstrate how intelligent, closed-loop decision-making shifts fleet behavior from reactive, synchronized patterns toward a more proactive, performance-driven model. While the higher activity levels of the AI-inspired strategy lead to more frequent hardware wear, the fleet accomplishes a much higher total volume of completed tasks. This trade-off highlights the necessity of the proposed framework; by using predictive telematics, the system can push the fleet to its full potential while still catching early signs of degradation before they lead to total breakdowns.

4.3.3 Timeliness and Deadline Compliance

Figure 4-3 presents a comparison of the on-time delivery rates achieved by both the Baseline and AI-inspired control strategies across all simulation runs. Each data point represents the percentage of completed jobs in a single run that were successfully delivered on or before their assigned deadline.

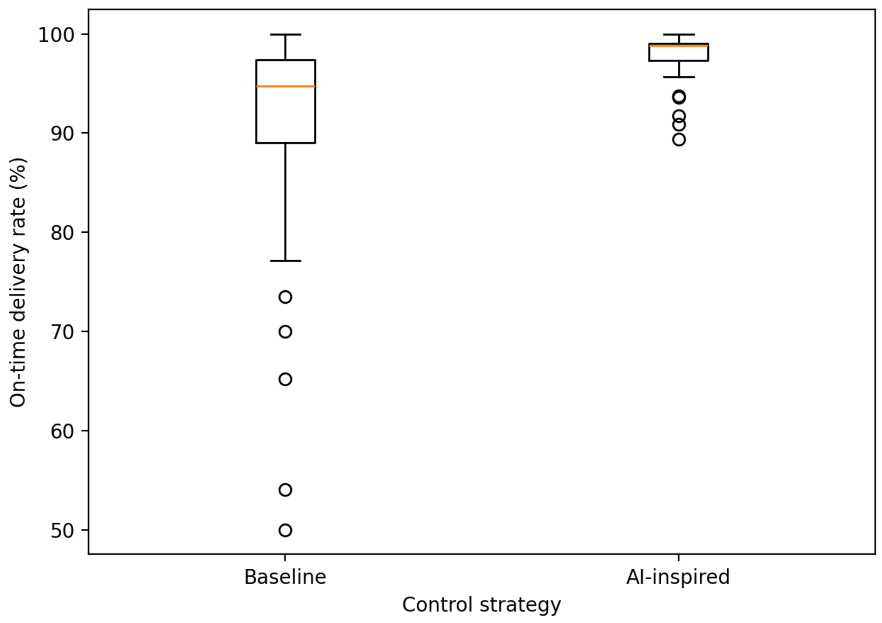

Figure 4-3 On-Time Delivery Rate per Run as Percentage
Source: author's own creation

The AI-inspired strategy consistently outperforms the baseline, with a median on-time rate close to 99% and a very narrow interquartile range. This suggests that the system is not only better at meeting deadlines but is also consistent, showing very little variability regardless of the specific job distribution in a given run.

In contrast, the Baseline strategy shows a lower median on-time rate and significantly more data dispersion, including several low-performing outliers. This variability is a direct result of the strategy's lack of temporal awareness. Since it only considers proximity during task assignment, it fails to adapt when several urgent jobs appear simultaneously.

This performance gap highlights the value of the AI policy's cost function, which incorporates both deadline slack and job priority. By taking these timing constraints into account, the system can prioritize urgent tasks and avoid late deliveries, even if that means skipping over the closest robot in favor of one better positioned to finish on time. These results prove that even relatively simple, deadline-aware logic can significantly improve reliability compared to traditional distance-based rules.

4.3.4 Energy Behavior and Charging Utilization

Figure 4-4 illustrates the average fleet State of Charge (SoC) over time, aggregated across all simulation runs for both control strategies.

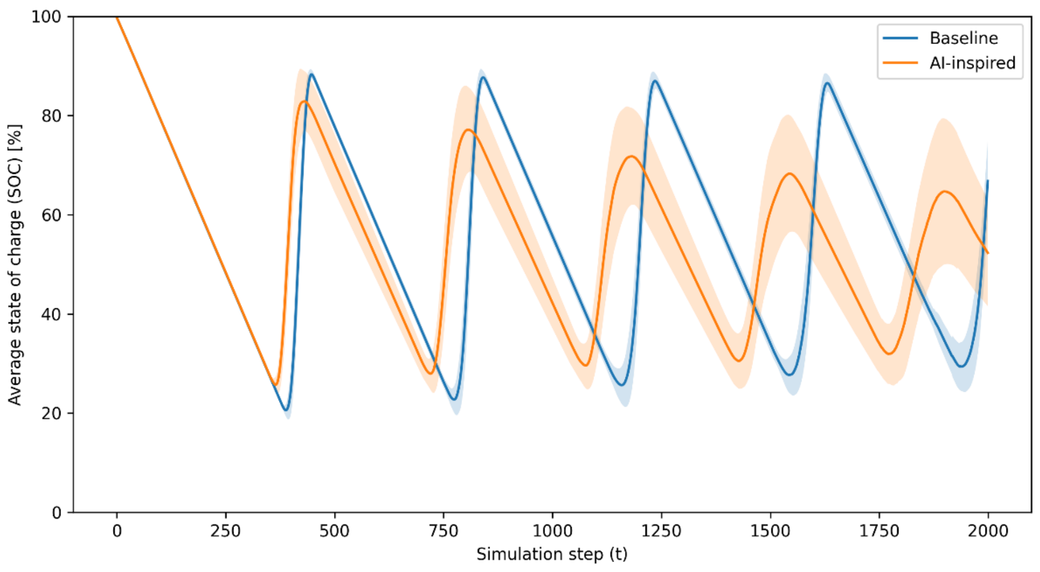

Figure 4-4 Average State of Charge as Percentage
Source: author's own creation

While both approaches successfully maintain the SoC within safe operating limits, there are some distinct differences in energy management. The Baseline strategy exhibits more pronounced oscillations, with robots going through deep battery drains followed by sudden, synchronized charging phases. This sawtooth behavior is a direct result of its reactive logic. Due to delayed charging until a fixed low threshold is reached, multiple robots often hit this limit simultaneously, which can lead to temporary congestion at charging stations and sudden drops in fleet availability.

In contrast, the AI-inspired strategy demonstrates smoother SoC trajectories with reduced amplitude. Interestingly, the average SoC under the AI strategy is slightly lower than in the Baseline case. This does not indicate inferior management, but rather higher efficiency. Instead of focusing on higher energy levels, the AI policy makes the fleet more efficient. By forecasting demand and scheduling charging proactively, the system allows robots to continue running efficiently just above critical thresholds, eliminating the downtime associated with energy stockpiling. Energy is therefore converted more consistently into productive delivery work.

Altogether, these results lead to a major shift in how the fleet operates. Intelligent policies move the system away from just reacting to low batteries and toward a model where energy is managed based on the actual workload. By balancing safety buffers with the needs of the job, the AI-inspired approach hits a higher level of efficiency without ever putting the robots at risk of running out of power.

4.3.5 Health Degradation and Maintenance Events

To better understand the shift from reactive to preventive maintenance, it is necessary to analyze each policy's degradation patterns and service triggers found in Figures 4-5, 4-6, and 4-7.

Figure 4-5 provides a comparison of the number of maintenance entries per run between the two policies. The AI-inspired strategy triggers a higher number of maintenance events than the Baseline approach. Although it may seem as though the AI-inspired robots have poorer health, this metric actually reflects the proactive maintenance philosophy, where robots are serviced before critical degradation occurs. In contrast, the Baseline strategy records very few maintenance visits, mainly because maintenance is triggered too late, only after severe degradation has already progressed.

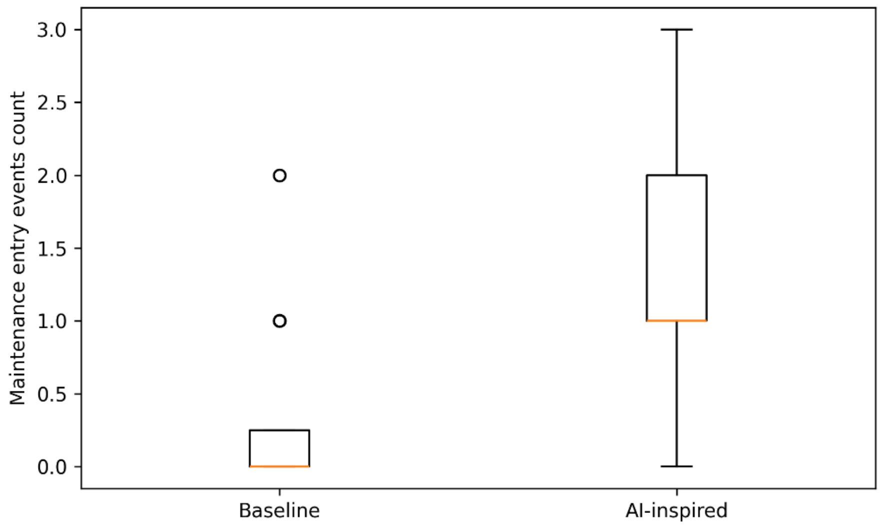

Figure 4-5 Maintenance Entries per Run
Source: author's own creation

The impact of this degradation is clearly illustrated in Figure 4-6, which shows the robot health score at the moment maintenance is initiated. Under the AI-inspired strategy, robots head to maintenance while their health is still relatively high. This creates a safety margin, ensuring that repairs happen while the robot is still functioning well. The Baseline strategy, however, waits until health levels are much lower, which significantly increases the risk of a robot breaking down on its way to the depot.

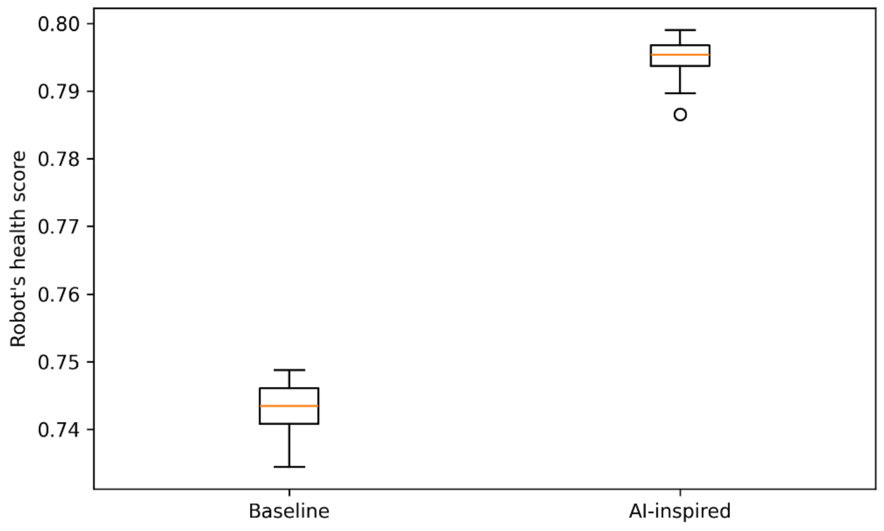

Figure 4-6 Health Score at Maintenance Entry
Source: author’s own creation

To better understand how these strategies play out over the course of a full simulation, Figure 4-7 illustrates when maintenance normally happens. The AI-inspired strategy shows a higher concentration of maintenance events toward the end of the simulation. Rather than a sign of delayed repairs, this pattern is a direct result of the fleet being much more active. Because the AI-controlled robots finish significantly more deliveries, they accumulate wear more rapidly and therefore require maintenance more frequently as the simulation progresses. On the other hand, the Baseline strategy shows fewer and more scattered maintenance events. This does not necessarily indicate that robots are healthier, but rather that they often break down completely before even meeting maintenance conditions.

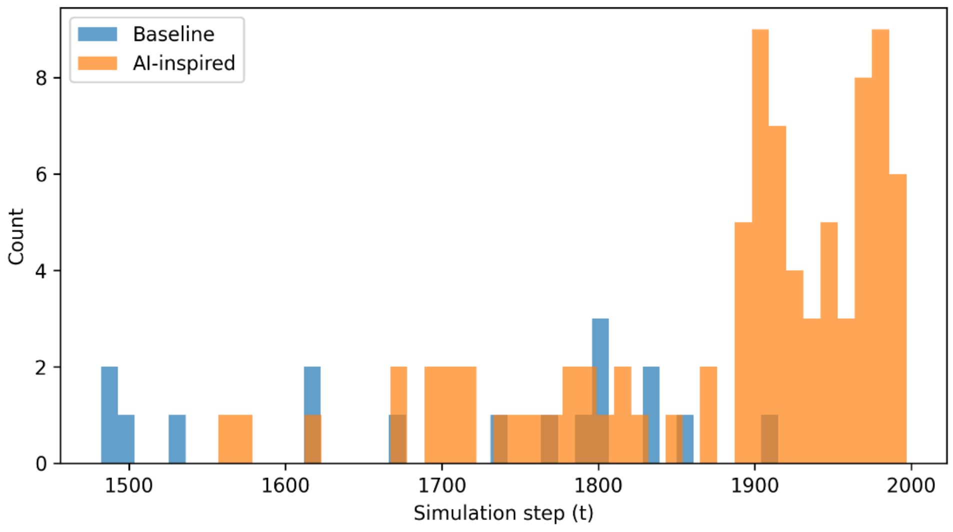

Figure 4-7 Temporal Distribution of Maintenance Events
Source: author’s own creation

These results highlight a fundamental trade-off between reactive and preventive maintenance strategies. While the AI-inspired approach pauses robots more often for quick check-ups, it pays off by nearly eliminating unexpected breakdowns and making the whole fleet more reliable. By incorporating health feedback into the operational decision-making, the AI-inspired strategy demonstrates how predictive maintenance can actually drive higher, long-term performance in autonomous fleets.

4.3.6 System-Level Performance Trade-offs and Assignment Efficiency

The multi-factor assignment approach of the AI-inspired strategy, summarized in Pseudocode A-3 in the Appenidx, is the leading contributor to the stronger overall performance observed across the experiments. Instead of the Baseline’s narrow focus on nearness, the AI policy evaluates the full context of every job. It does not focus exclusively on distance, but rather factors in battery life, estimated energy use, time limits, and the robot's current condition before making a move.

This intelligent assignment logic also plays a key role in stabilizing fleet availability. As discussed in Chapter 4.3.2, the Baseline strategy suffers from "synchronized charging," where multiple robots hit their low-battery thresholds simultaneously, creating bottlenecks. To avoid this, the AI strategy distributes charging events more proactively over time. Even though this leads to a lower average battery level overall, it prevents those sudden capacity drops and ensures that enough robots remain operational during peak demand.

Timeliness also benefits directly from this improved efficiency. By taking into consideration how much time is left in a job, the AI policy prevents the scheduling conflicts that often trouble the distance-based Baseline policy. This proves that even adding a basic awareness of time to the assignment logic has a measurable impact on service reliability.

However, this higher performance comes with a physical cost. As detailed in chapter 4.3.5, the AI strategy triggers maintenance interventions earlier and more frequently. Although this can translate into an increase in downtime, it is a strategic trade-off. By accepting a busier workload and scheduled service, the system avoids the catastrophic failures that occur when maintenance is ignored, as seen in the Baseline results.

Ultimately, these results point to a clear strategic choice: the AI strategy accepts more wear and tear and higher energy use in exchange for better reliability and higher delivery volumes. In this context, assignment efficiency is about more than just shortening travel distances; it acts as a coordination layer that aligns each robot's individual actions with the bigger goals of the fleet. These findings prove that even simple, context-aware rules can lead to massive gains when they are part of a closed-loop system.

4.3.7 Discussion Summary

This comparative analysis demonstrates that the AI-inspired control strategy consistently outperforms the Baseline approach across multiple key metrics. By incorporating energy awareness, deadline sensitivity, and health feedback into task assignment and operational decisions, the AI-inspired strategy achieves higher job completion rates, improved on-time delivery, and more stable fleet availability.

Another significant insight is how the system handles operational stress. While the AI-inspired policy keeps the fleet much busier, which naturally speeds up wear and tear, it manages that risk through proactive maintenance. This prevents the total breakdowns and synchronized charging bottlenecks seen in the Baseline runs, proving that it is possible to push a fleet to work harder without sacrificing overall reliability.

Ultimately, these results highlight the importance of closed-loop, context-aware decision-making in autonomous fleet management. The data shows that even straightforward, predictive rules can offer massive benefits over old-fashioned, distance-based logic. These conclusions establish the groundwork for the final analysis in Chapter 5, including design advice and directions for future work.

5 Conclusions & Future Work

5.1 Interpretation of Results

The findings in Chapter 4 demonstrate that the AI-inspired control strategy does more than just tweak performance; it fundamentally changes how the fleet behaves. By teaching the system to account for time and robot health in every decision, the operation shifts from a short-sighted, reactive mode to one that is proactive and focused on getting the most out of every robot.

Across all evaluated dimensions (operational states, energy behavior, timeliness, maintenance activity, and overall task throughput) the AI-inspired strategy consistently shifts the fleet away from reactive, synchronized behavior toward a proactive and utilization-driven operation. This change is most noticeable in how the system smooths out charging demand, the redistribution of task assignments under deadline pressure, and the emergence of preventive maintenance patterns.

One of the most important insights from the results is that improved performance does not stem from keeping robots in an artificially “safe” or conservative operating state. Instead, the AI-inspired strategy deliberately operates the fleet closer to its functional limits, as evidenced by lower average state-of-charge levels and increased health degradation rates. However, this increased stress is continuously monitored and incorporated into the decision logic, allowing the system to intervene through charging or maintenance before failures occur.

The Baseline strategy’s reactive logic, on the other hand, leads to periodic congestion and inefficiencies. Due to its reliance on fixed thresholds, it forces multiple robots to become unavailable simultaneously, thus reducing fleet flexibility when it is needed most. Similarly, the absence of deadline awareness results in delayed task execution under high demand, even when sufficient fleet capacity exists. Over time, these inefficiencies compound, leading to high performance variability and susceptibility to cascading failures.

The observed increase in maintenance activity under the AI-inspired policy should therefore not be interpreted as a degradation in system quality, but rather as a direct indicator of higher effective utilization. Robots accumulate wear more rapidly because they complete more tasks, travel longer distances, and spend less time idle. By favoring early maintenance interventions, the AI-inspired strategy effectively mitigates the likelihood of unexpected system breakdowns, ensuring that the fleet remains productive even under the strain of a high-volume workload.

Ultimately, these results indicate that true system-level reliability does not come from focusing on energy or wear in isolation, but from a closed-loop approach that balances these competing needs. The findings emphasize the central thesis' argument: for an autonomous fleet to be effective, its decision-making must simultaneously account for timing, energy levels, and hardware health. Strategies that ignore these interdependencies may appear sufficient in low-stress environments, but lack the robustness required for high-intensity, large-scale operations required in modern cities.

5.2 Trade-offs Between Efficiency, Health, and Reliability

The comparison between the Baseline and AI-inspired strategies shows that running an autonomous fleet is essentially a balancing act. Performance is not just about one specific goal; it is governed by the constant trade-off between keeping operations efficient, maintaining hardware health, and ensuring the system stays reliable over time. Ultimately, managing a fleet effectively means weighing these competing needs against one another to find the right balance for the mission.

The Baseline strategy prioritizes short-term asset preservation by minimizing immediate wear, charging activity, and delaying maintenance. Because robots are only charged or serviced once they hit predefined thresholds, the fleet may appear healthier on paper, maintaining higher average battery levels and requiring fewer service stops. However, this conservative approach creates significant system-level inefficiencies. Charging demand becomes synchronized, leading to robots becoming unavailable in clusters, while task assignments suffer from a lack of temporal awareness. As a result, system performance deteriorates under high workload or deadline pressure, proving that lower mechanical stress does not automatically translate to operational success.

In contrast, the AI-inspired strategy explicitly accepts higher operational stress in exchange for improved fleet performance. By utilizing robots more aggressively, the system achieves higher productivity and smoother energy profiles, even when this results in lower average battery levels and faster health degradation. While these specific metrics might seem like a disadvantage, they are strategic trade-offs required to maintain a more stable and predictable operation.

A central insight from the results is that reliability is not equivalent to minimizing degradation. While the Baseline strategy records fewer maintenance events, it suffers from higher failure risks and performance variability. When degradation is left unchecked, the likelihood of sudden robot failures increases, removing assets from service unexpectedly and disrupting the entire workflow. The AI-inspired strategy mitigates this risk by converting uncontrolled failures into managed downtime, by monitoring health indicators and intervening before critical thresholds are reached.

This trade-off is particularly evident in energy management. While the AI-inspired policy maintains a lower average SoC, it avoids deep discharge events and distributes charging demand more evenly over time. This contrasts with the Baseline approach, where delayed charging leads to sharp SoC drops followed by concentrated recharging periods. The AI strategy therefore prioritizes availability and flexibility over simply maintaining high battery levels, demonstrating that a high SoC does not necessarily imply better operational performance.

These observations highlight a fundamental principle of autonomous fleet control: maximizing system-level reliability often requires operating individual assets closer to their limits. The AI-inspired strategy does not eliminate wear or degradation; instead, it leverages predictive awareness to ensure that increased utilization is managed safely, preventing high-intensity work from turning into uncontrolled failure. The results suggest that reactive policies are insufficient for scaling autonomous operations in complex environments.

5.3 Implications Real-World Autonomous Fleet Deployment

The findings of this study extend beyond the simulation environment, offering practical implications for the design and deployment of autonomous robotic fleets in the real world. As these systems transition from controlled simulations to complex, time-critical applications such as last-mile delivery or urban service robotics, the limitations of purely reactive control strategies become increasingly problematic.

One of the most significant conclusions is the necessity of health-aware and time-aware decision-making. The results clearly demonstrate that strategies relying exclusively on static thresholds for charging or maintenance are unable to effectively adapt to the fluctuating workloads and deadline pressure of real-world operations. In real deployments, where traffic conditions, order surges, and environmental factors vary constantly, such rigid policies are likely to lead to synchronized downtime and unpredictable performance drops. To remain robust, the control layer must be dynamic.

The success of the AI-inspired strategy suggests that operational intelligence does not require prohibitive computational complexity. The system achieved significant performance gains using relatively simple predictive logic, like estimating battery usage and deadline slack, rather than computationally expensive global optimization planners. This is highly relevant for scalability, since it implies that fleet operators can achieve smart behavior without requiring massive centralized computing power, making the approach viable for large-scale deployments.

Another important implication is the traditional engineering view of maintenance. Often times, fleet management treats maintenance as an undesirable interruption to be minimized. However, the results of this study show that more frequent, preventive interventions are actually beneficial for system reliability by reducing the incidence of unexpected failures. In practice, this suggests operators should move away from fixed service intervals and instead trigger maintenance based on real-time telemetry, accepting short, planned interruptions to prevent costly, unplanned failures.

Energy management also emerges as a critical system-level consideration. While keeping individual robots at a high State of Charge feels intuitive, the system-level analysis shows that availability depends more on how charging demand is distributed over time. Real-world charging algorithms should prioritize avoiding congestion and deep discharges rather than simply trying to keep every battery full.

Finally, the observed trade-offs show how important it is to align a control strategy with the actual goals of the operation. Fleets optimized for high productivity and strict deadlines might have to accept higher and accelerated wear and tear on its robots. On the other hand, for operations with lower urgency or limited maintenance capacity, a more conservative approach might be preferred, even if it leads to reduced performance. The framework and results in this thesis provide a structured basis for evaluating these trade-offs before a fleet is deployed.

The framework presented in this thesis provides a structured methodology for evaluating these trade-offs before hardware is ever deployed. Ultimately, reliable autonomous operations are not defined by the complete elimination of degradation, but by the ability to manage it predictably. Health-aware, adaptive control offers the most practical path toward achieving this stability in the real world.

5.4 Limitations of the Study

While the results presented in this thesis provide meaningful insights into autonomous fleet control, it is important to acknowledge some specific constraints and assumptions that shape these findings. These limitations primarily stem from the necessary abstractions required to make the simulation experimentally feasible.

First, the study is based entirely on a simulated environment. Although the simulation captures key aspects of fleet operation such as task assignment, energy consumption, health degradation, charging behavior, and maintenance events, it cannot fully replicate the complexity of real-world conditions. Factors such as sensor noise, communication delays, hardware faults, environmental variability, and unexpected human interactions are not explicitly modeled. As a result, absolute performance values should not be interpreted as directly transferable to physical deployments.

Second, the fleet size and operational scale are limited. The experiments consider a fixed number of robots operating within a predefined spatial and temporal environment. While this allows for controlled comparisons between strategies, larger fleets or more heterogeneous fleets might exhibit additional complex behaviors not captured in this study, such as network congestion or coordination overhead. The dynamics of a more diverse fleet population remain an area for further investigation.

Third, the AI-inspired strategy implemented in this work relies on smart, hand-coded rules rather than machine learning or complex optimization. Although this was a deliberate design choice to ensure the system's logic remains transparent and easy to interpret, it does limit the adaptability of the strategy. Since the system does not learn from its mistakes or use historical data to improve its own models over time, these results are likely just the tip of the iceberg. The improvements presented represent a starting point for what more advanced, learning-based systems could eventually achieve.

Another limitation lies in the modeling of health degradation and maintenance. In this simulation, health evolves according to simplified degradation rules and threshold-based maintenance triggers. In reality, mechanical wear is influenced by a vast range of interacting variables, including load conditions, terrain, component variability, and manufacturing tolerances. While the simplified model is sufficient for comparative analysis, it does not capture the full non-linear unpredictability of physical degradation processes.

It is also important to note that the workload was the same for every simulation run. Although this supports fair comparison between the strategies, it does not account for extreme or adversarial scenarios such as sudden demand spikes, prolonged idle periods, or chain reaction of mechanical failures. A different set of challenges might change how much of an edge the AI strategy really has.

Finally, the evaluation metrics focus primarily on operational performance indicators rather than economic ones. Factors such as dynamic energy pricing, maintenance labor, and asset replacement costs were not incorporated into the model.

Despite these limitations, the study successfully demonstrates the qualitative advantages of health-aware and time-aware control strategies under controlled conditions. The identified trends and trade-offs remain valuable as conceptual guidance for system designers, providing a robust foundation for future validation in real-world environments.

5.5 Future Work and Research Directions

The framework and results presented in this thesis establish a strong foundation for further research into autonomous fleet management using telematics-driven decision-making. The limitations identified in the previous section naturally point toward several exciting avenues for future research.

One of the most logical extensions of this work is the integration of learning-based models into the closed-loop control framework. While the current AI-inspired strategy relies on hand-coded rules and threshold-based logic, future systems could incorporate machine learning techniques to predict energy consumption, battery degradation, and maintenance needs with much higher accuracy. By using historical telemetry data to build models that continuously adapt to the fleet's health, the system could move from simply reacting to degradation to anticipating it based on long-term trends. This evolution could be taken even further through reinforcement learning, allowing the system to move away from pre-written rules entirely and instead learn its own policies. This would enable robots to make much smarter decisions, balancing immediate productivity with long-term health preservation, even as workloads and demands fluctuate.

The simulation environment itself could also be expanded. Future work could incorporate heterogeneous robot fleets, where robots differ in battery capacity, payload limits, speed, or degradation characteristics, to better reflect real-world logistics. Additionally, introducing dynamic obstacles, variable travel speeds, or temporary connectivity losses would provide a much clearer picture of how robust these strategies are when faced with unexpected disruptions.

Another significant opportunity exists in integrating the fleet controller with broader smart city infrastructure. While the current model treats the operational space as an isolated environment, future iterations could incorporate external data streams, such as real-time traffic density, pedestrian flow, or charger availability. By integrating city-level information into the decision loop, fleet behavior could adapt dynamically to broader urban conditions, further reinforcing the closed-loop concept.

From an operational perspective, engineering efficiency is mostly linked to cost and sustainability. Future versions of this framework should incorporate economic factors, such as dynamic energy pricing and labor costs associated with maintenance. Furthermore, aligning the system with green engineering standards by factoring in the carbon footprint of energy use and the lifecycle impact of battery replacements would ensure that the optimization targets both financial and environmental sustainability.

Finally, the ultimate validation of this framework lies in bridging the gap between simulation and reality. Transitioning the proposed logic to physical robotic platforms or commercial fleet data would offer critical insights into sensor reliability, hardware constraints, and system scalability. Real-world deployment remains the final step in confirming that health-aware, adaptive control can provide the necessary stability for autonomous systems in urban environments.

In summary, this thesis demonstrates that even straightforward health-aware and time-aware mechanisms can substantially improve system-level performance. Future research can build upon this modular foundation to develop fully adaptive, learning-enabled management systems capable of navigating the complexities of modern autonomous operations.

Appendix

Appendix A: Pseudocode

```
Initialize environment E
Initialize robot fleet R
Initialize empty job list J
time = 0

for each simulation step t in [0, T]:

    # Job Generation
    if random() < job_arrival_probability:
        job = generate_job(t)
        add job to J

    # Task Assignment
    for each job j in J:
        if j is unassigned:
            policy.assign_job(R, j, t)

    # Charging and Maintenance
    policy.manage_charging(R)
    policy.manage_maintenance(R)

    # Robot Actions
    for each robot r in R:
        r.step(E)

    update_pickups_and_completions(R, J, t)

    record_metrics(R, J, t)
```

```
end for
```

Pseudocode A-1 Main Simulation Loop

```

for each simulation step t:

    # Job creation
    if random() < job_arrival_probability:
        job = create_new_job(t)
        add job to job_list

    # Assignment attempt
    for each job in job_list:
        if job is unassigned:
            result = policy.assign_job(robots, job, t)
            record result

    # Pickup and delivery
    for each job in job_list:
        if job is assigned to robot r:

            if job.pickup_time is None and r.position == job.pickup:
                job.pickup_time = t
                r.load_payload(job.weight)

            if job.pickup_time is not None and r.position == job.dropoff:
                job.completion_time = t
                r.unload_payload()
                mark job as completed
end for
```

Pseudocode A-2 Job Lifecycle Processing

```
for each unassigned job j:
    for each available robot r:
        estimate energy_required(r, j)
        compute deadline_slack(j, r)
        if SOC_r < energy_required + safety_margin:
            skip robot
        compute cost(r, j) =
            w1 * distance_to_pickup +
            w2 * deadline_slack +
            w3 * health_penalty
    assign job j to robot with minimum cost
end for
```

Pseudocode A-3 AI-Inspired Assignment Decision

Bibliography

[1] O. Fink, T. Hartung, S. Y. Lee, and A. Maynard, “AI for Scientific Discovery,” in Top 10 Emerging Technologies of 2024, World Economic Forum, June 2024, https://www.weforum.org/publications/top-10-emerging-technologies-2024/in-full/1-ai-for-scientific-discovery/
[Last accessed on November 24, 2025].

[2] Cambridge University Press, “Data,” Cambridge Dictionary, https://dictionary.cambridge.org/dictionary/english/data
[Last accessed on November 24, 2025].

[3] A. Goel, “Telematics,” in Fleet Telematics (Operations Research/Computer Science Interfaces, vol. 40), Springer, Boston, MA, 2008, pp. 7-30. DOI: 10.1007/978-0-387-75105-4_2.

[4] Cohen, A., and Shaheen, S., “Planning for Shared Mobility,” Berkeley, CA: University of California Berkeley, Transportation Sustainability Research Center, 2018, https://escholarship.org/content/qt0dk3h89p/qt0dk3h89p.pdf
[Last accessed on November 24, 2025].

[5] Massaro, A., Selicato, S., and Galiano, A., “Predictive Maintenance of Bus Fleet by Intelligent Smart Electronic Board Implementing Artificial Intelligence,” IoT, vol. 1, no. 2, pp. 180–197, 2020, https://doi.org/10.3390/iot1020012
[Last accessed on November 24, 2025].

[6] Radius Telematics, “A Brief History of Telematics”, https://www.radius.com/en-gb/telematics/explained/history/
[Last accessed on November 24, 2025].

[7] Ortiz, F. M., Sammarco, M., Costa, L. H. M. K., & Detyniecki, M, “Vehicle Telematics Via Exteroceptive Sensors: A Survey,” arXiv pre-print arXiv:2008.12632, Aug 2020, https://arxiv.org/abs/2008.12632
[Last accessed on November 24, 2025].

[8] Young, R., Fallon, S., Jacob, P., and O'Dwyer, D, "Vehicle Telematics and Its Role as a Key Enabler in the Development of Smart Cities," IEEE Sensors Journal, vol. 20, no. 19, pp. 11713–11724, Oct 2020. DOI: 10.1109/JSEN.2020.2997129.

[9] Singh, S. K., and Singh, A. K, "A Study: OBD III Standard and Its Predecessors OBD II and OBD I," Mody University, India, https://www.academia.edu/download/88791685/Review_Paper_I_accepted_at_IPEM.pdf.
[Last accessed on November 24, 2025].

[10] CS Selectronics, "OBD2 Explained – A Simple Intro", 2025, https://www.csselectronics.com/pages/obd2-explained-simple-intro,
[Last accessed on November 24, 2025].

[11] Lantronix. "How to collect CAN FMS/J1939/OBD-II data with FOX3-2G/3G/4G Series," October 2019, https://cdn.lantronix.com/wp-content/uploads/pdf/AppNote_CAN_FMS_CAN_OBDII_Howto.pdf.
[Last accessed on November 24, 2025].

[12] Panasonic Industry, "What is a Telematics Control Unit (TCU)? – Equipment configuration and key points for achieving higher performance," Technical Information, December 2022, https://industrial.panasonic.com/ww/ds/ss/technical/b19.
[Last accessed on November 24, 2025].

[13] Lantronix, "PFAL Command Reference," April 2020, https://cdn.lantronix.com/wp-content/uploads/pdf/AVL-PFAL-CR.pdf
[Last accessed on November 24, 2025].

[14] Russell, S. J., and Norvig, P., "Artificial Intelligence: A Modern Approach", 3rd ed., Pearson Education, Inc., 2010, ISBN: 978-0-13-604259-4.

[15] Bishop, C. M., "Pattern Recognition and Machine Learning", Springer Science+Business Media, 2006, ISBN: 978-0-387-31073-2.

[16] Yoo, S., Shin, J.-W. and Choi, S.-H., "Machine learning vehicle fuel efficiency prediction," Scientific Reports, vol. 15, no. 14815, April 2025,

https://doi.org/10.1038/s41598-025-96999-0
[Last accessed on November 24, 2025].

[17] M. Wilbur, A. Sivagnanam, A. Ayman, S. Samaranyake, A. Dubey, and A. Laszka, "Artificial Intelligence for Smart Transportation," arXiv:2308.07457, Aug 2023, https://arxiv.org/abs/2308.07457
[Last accessed on November 24, 2025].

[18] B. M. Mohsen, "AI-Driven Optimization of Urban Logistics in Smart Cities: Integrating Autonomous Vehicles and IoT for Efficient Delivery Systems," Sustainability, vol. 16, no. 24, 11265, Dec 2024, https://www.mdpi.com/2071-1050/16/24/11265
[Last accessed on November 24, 2025].

[19] S. M. LaValle, "Planning Algorithms", Cambridge University Press, 2006, https://lavalle.pl/planning/
[Last accessed on November 24, 2025].

[20] Delta-Q Technologies, "Autonomous Vehicle Telematics Maximize Productivity," Aug 2022, https://delta-q.com/industry-news/autonomous-vehicle-telematics-maximize-productivity/
[Last accessed on November 24, 2025].

[21] M. Rinaldi, S. Primatesta, M. Bugaj, J. Rostáš, and G. Guglieri, "Urban Air Logistics with Unmanned Aerial Vehicles (UAVs): Double-Chromosome Genetic Task Scheduling with Safe Route Planning," Smart Cities, vol. 7, pp. 2842–2860, Oct 2024, https://doi.org/10.3390/smartcities7050110
[Last accessed on November 24, 2025].

[22] Starship Technologies, "Starship Technologies Surpasses 8 Million Deliveries," Apr 2025, https://www.starship.xyz/press/starship-technologies-surpasses-8-million-deliveries/
[Last accessed on November 24, 2025].

[23] D. Wei, "Lecture 10: Dijkstra's Shortest Path Algorithm," Dept. of Computer Science and Engineering, Hong Kong University of Science and Technology

(HKUST), https://www.cse.ust.hk/~dekai/271/notes/L10/L10.pdf
[Last accessed on December 04, 2025].

[24] NetworkX Developers, “shortest_path,” NetworkX Reference Documentation, https://networkx.org/documentation/stable/reference/algorithms/shortest_paths.html
[Last accessed on December 04, 2025].

[25] C. Jotin Khisty and B. Kent Lall, “Transportation Engineering: An Introduction”, 4th ed., Elsevier, 2016, ISBN: 978-0128038185