Turkish Journal of Science & Technology
20(1), 281-297, 2025

Research Paper
https://doi.org/10.55525/tjst.1623326

# Real-Time Drone Command Processing: A Large Language Model Approach for IoD Systems

**Anıl SEZGİN[1*], Aytuğ BOYACI[2]**
[1]Research and Development, Siemens A.S, Istanbul, Turkey
[2] Department of Computer Engineering, National Defence University, Air Force Academy, Istanbul, Turkey
[*1] anil.sezgin@siemens.com, [2] aytug.boyaci@msu.edu.tr

**(Geliş/Received: 20/01/2025; Kabul/Accepted: 10/03/2025)**

**Abstract:** One of the most critical steps toward autonomous capabilities, where natural language instructions can be successfully converted into executable API calls, is integrating Large Language Models (LLMs) into the ecosystem of the Internet of Drones (IoD). This study introduces an end-to-end LLM-based framework for enhancing real-time drone operation and problem handling in intent recognition, parameter extraction, and ambiguity resolution. It has resorted to a spectrum of methodologies in the form of Retrieval-Augmented Generation (RAG) and customized fine-tuning specific to each domain, towards accurate command interpretation and successful API generation. This paper evaluates the performance of the model on a carefully designed dataset containing 1,500 commands for the different scenarios of IoD with an average BLEU score of 89.6 and a cosine similarity of 0.94. Optimization techniques, such as parallel processing and better query handling, reduced the latency of this system by 15%, with an average query processing time of 0.9 seconds. This work gives considerable importance to the scalability and flexibility of the system, which is quite crucial for applications in domains like disaster response, precision agriculture, and surveillance. Proposed LLM-based framework thus tries to bridge gap between human intent and drone execution for intuitive, reliable, and efficient IoD deployments.

**Key words:** Large language models, internet of drones, natural language processing, real-time command processing.

## Gerçek Zamanlı Drone Komut İşleme: IoD Sistemleri için Büyük Dil Modeli Yaklaşımı

**Öz:** Doğal dil talimatlarının yürütülebilir API çağrılarına başarıyla dönüştürülebildiği otonom yeteneklere doğru atılan en kritik adımlardan biri, Büyük Dil Modellerinin (LLM) İnsansız Hava Araçları İnterneti (IoD) ekosistemine entegrasyonudur. Bu çalışma, amaç tanıma, parametre çıkarımı ve belirsizlik çözümleme alanlarında gerçek zamanlı drone operasyonlarını ve sorun çözümünü geliştirmek için uçtan uca bir büyük dil modeli tabanlı çerçeve sunmaktadır. Çalışmada, her alana özgü doğru komut yorumlama ve başarılı API oluşturma için Retrieval-Augmented Generation (RAG) yönteminden yararlanılmıştır. Çalışma, IoD'nin farklı senaryolarını kapsayan ve 1.500 komuttan oluşan bir veri kümesi üzerinde modeli değerlendirmiştir. Elde edilen sonuçlara göre, modelin ortalama BLEU skoru 89,6 ve kosinüs benzerliği 0,94 olarak ölçülmüştür. Paralel işlem ve daha iyi sorgu işleme gibi optimizasyon teknikleriyle sistemin gecikme süresi %15 azaltılmış ve ortalama sorgu işleme süresi 0,9 saniye olmuştur. Bu çalışma, özellikle afet müdahalesi, hassas tarım ve gözetim gibi alanlarda uygulamalar için oldukça kritik olan sistemin ölçeklenebilirliğine ve esnekliğine büyük önem vermektedir. Önerilen LLM tabanlı çerçeve, böylece insan amacı ile drone uygulamaları arasında sezgisel, güvenilir ve verimli IoD dağıtımları için bir köprü kurmayı amaçlamaktadır.

**Anahtar kelimeler:** Büyük dil modelleri, insansız hava araçları interneti, doğal dil işleme, gerçek zamanlı komut işleme.

## 1. Introduction

The Internet of Drones is one of the advanced technologies today, where drones no longer work in isolation but, rather, in coordination with each other under a single network controlled by centralized systems. Among the many functions such systems make possible are coordinated logistics in the delivery of products, vast monitoring in security uses, and dynamic environmental sensing to gather critical data for better decision-making. IoD systems have revolutionized domains such as disaster management, where drones can be rapidly deployed to aid search and rescue operations; logistics, through route optimization of delivery networks in both urban and rural environments; and precision agriculture, with detailed crop health assessment and efficient application of pesticides.

At the core of this new technology lies the challenge of effective communication between humans and machines. Most often, users give commands in natural language, but it is crucial that such a command is translated into a precise, executable API request so that drones can execute the intended actions. For instance, even the straightforward command “Send Drone A to scan Sector 7” involves finding the right drone, assessing its current status and location, building the appropriate API request, and verifying that the operation satisfies a range of

* Corresponding author: anil.sezgin@siemens.com. ORCID Number of authors: [1] 0000-0002-5754-1380, [2] 0000-0003-1016-3439

constraints, including battery levels, no-fly zones, and task priorities. This task is significantly harder in dynamic environments where the drone networks keep altering, the operational parameters get updated quite often, and user intents change greatly due to the variation of context and applications.

IoD systems, on the other hand, have to cater to a wide range of users, from professional experts to laypersons not necessarily versed in specialized vocabulary. This kind of heterogeneity demands a solution capable of correctly interpreting vague, incomplete, or contextually complex queries while maintaining high levels of accuracy and efficiency. Such is the challenge requiring activation of modern security measures and effective detection systems, discussed in [1], pointing out a number of vulnerabilities in IoD networks while suggesting new methods for data and operational integrity. The increasing sophistication of drone operations and requirements for real-time responsiveness only add another layer of complexity to an already difficult challenge of translating human intent into machine-executable commands. Improvement in the object detection process, as explained in [2], is the key to allowing drones to recognize and react to dynamic objects in their surrounding to accomplish their tasks successfully in different setups.

The focus of this research, by its very nature, integrates large language models in overcoming such challenges—LLMs. These models, through the strengths of advanced generative language capabilities coupled with specialized fine-tuning for specific tasks, are able to analyze user inputs and generate accurate responses that are tailored to specific domains. Methods like Retrieval-Augmented Generation (RAG) and supervised fine-tuning make LLMs very adaptable to perform specialized tasks, like translating natural language instructions into API requests for IoD systems. Besides adaptability in performing different functions, LLMs have proven very efficient in ambiguity resolution and consistency within context; these are factors that further improve performance on incomplete queries or in multi-turn dialog. If fine-tuned carefully, even for the most complex operational settings, LLMs could still come up with quite coherent and stable responses.

The application of Artificial Intelligence (AI) technologies and LLMs in IoD systems is one giant leap toward human-machine interaction and operational efficiency. Similarly, some of the critical IoD challenges, such as resource constraints and scalability, can be overcome by these IoD systems through advanced AI-driven methodologies such as trajectory optimization, intelligent caching, and adaptive edge computing [3]. Simultaneously, LLMs improve the processing of natural languages so that human commands can be correctly translated into executable API requests [4]. The integration follows frameworks investigated in [5], underlining the need for scalable and secure API-driven architectures to enable seamless interactions between humans and machines. It would also enable real-time collaboration in multi-drone environments, ensuring adaptability to dynamic mission requirements and laying the foundations for resilient IoD operations. Exploiting these synergies between AI and LLM offers new solutions for applications in disaster response, smart city, and agriculture [6].

The contribution of LLMs goes beyond command processing to contextual understanding, scalability, and real-time responsiveness of the IoD functionality. In this regard, LLMs have integrated domain-specific retrieval mechanisms with scalable architectures to optimize API generation for mission-specific demands [7]. It will ensure that drones perform complicated tasks with precision, bridging the gap between human intent and machine execution. Furthermore, LLM-based frameworks allow the IoD system to adapt easily in diverse, fast-changing operational contexts, which improves the reliability of the system with less development effort [8]. With advances in artificial intelligence and large language model technologies, simpler, intuitive, and intelligent IoT systems would emerge, enabling larger degrees of innovations on different applications and fields while ensuring that, in themselves, they have roles of promoters for autonomous devices. The capacities of large language models go beyond task completion; they also bring better contextual understanding to the Internet of Things, offer crucial scalability, and make feasible instantaneous responsiveness. Large language models, using retrieval mechanisms focused on given domains combined with flexible architectural designs, enhance the creation of mission-specific APIs. Complex tasks are thus performed with precision by drones, and the gap between the human intent and machine execution is effectively bridged. This will enable the development of LLM-based frameworks to continue, allowing IoD systems to be adaptive under different and dynamic operational contexts with less development effort but increased reliability. With the continuing evolution of AI and LLM technologies, IoD systems are going to be even more accessible, intuitive, and efficient, driving innovation in a lot of fields and further solidifying their position as key enablers of autonomous systems. Another core aspect is that scalability is going to form another cornerstone in our approach: IoD systems frequently demand the simultaneous operation of several drones with very different roles, constraints, and real-time demands. The new framework, based on LLM, is going to efficiently scale up with the incorporation of knowledge-retrieval mechanisms peculiar to a specific domain that adjusts operating parameters for each drone; it ensures API requests are valid and optimized against requirements peculiar to any given context in operation. Hence, this allows perfect integration into each one of the most variable and dynamic situations.

Another core operation of IoD is real-time execution. Any delay in processing and execution of commands could have a tremendous impact on mission outcome, more so in time-critical scenarios involving disaster response or security surveillance. Our system performs advanced indexing techniques along with low-latency retrieval algorithms to reduce delays and ensures that API requests are created and executed within operationally acceptable time frames. This is further complemented by error detection and correction mechanisms, which proactively identify and resolve inconsistencies in generated commands to prevent operational failures.

The potential applications of an IoD command system based on LLM go far beyond the simple execution of tasks. Offering seamless interaction between humans and machines, this technology opens possibilities for much more intuitive and accessible drone operations. As IoD systems will continue to evolve, so must the frameworks to provide support for operation. LLM integration has provided a considerable advance in this aspect by filling the gap between human intent and machine execution in scalable, efficient, and user-friendly solutions for the problems posed by natural language-to-API request conversion. By addressing these issues directly, our research enhances the functionalities of IoD systems and forms a basis for future developments of autonomous systems and intelligent control frameworks.

To better outline the following paper, its structure will be as follows: Section 2 discusses related work on IoT systems, RAG models, and approaches to command generation. Section 3 elaborates on the system architecture with a focus on the retriever and generator components. Section 4 discusses the creation of the dataset and model development. Section 5 evaluates the system using real-world scenarios and benchmarks. Finally, Section 6 concludes with key findings and future directions for advancing IoD command systems.

## 2. Related Work

### 2.1. Introduction to AI and large language models

The development of large language models has been a turning point for artificial intelligence, enabling new developments in reasoning, communication, and automation. The core of all this lies in the development of diverse datasets and integration methods that expand these models' abilities. Many research efforts have shed light on some important aspects of LLMs, hence setting the stage for their application in many different domains.

High-quality, diverse datasets are indispensable for large language models. Study [8] classifies datasets into pre-training sources such as Common Crawl, C4, and GitHub repositories, and fine-tuning datasets like Alpaca and InstructGPT, highlighting the importance of dealing with challenges in data duplication, biases, and quality filtering. It calls for open-source initiatives to improve non-English corpora for democratized LLM development and better global utility. The versatility of LLMs is seen even in robotics, as discussed in Study [9], with regard to how models like GPT-4 and LLaMA would impact areas of communication, perception, planning, and control. It highlights zero-shot planning, adaptive control, and robotics-specific prompt engineering, together with challenges in consistencies and computational demands. In a comparable manner, foundational models, including large language models and vision-language models (VLMs), enhance the functionalities of autonomous systems, as indicated in Study [10], by augmenting the comprehension of semantic contexts and facilitating task decomposition, thereby fostering improved robotic autonomy in previously untrained environments. The research highlights progress in semantic scene representation alongside natural language processing, effectively connecting perception with action. Beyond robotics, LLMs revolutionize software development, Study [11] shows how generative AI automates tasks of code generation, debugging, and testing in order to make workflows faster and increase productivity. This paper further stresses balancing automation with human oversight to address ethical and cybersecurity issues, hence showing the transformative role that AI plays in modern software engineering.

### 2.2. Security and anomaly detection

With the Internet of Things, this increased dependency on interconnected systems and networks has raised the bar for strong security frameworks. Hence, artificial intelligence has come to be among those most significant enablers for addressing the increasingly complex challenges of security in anomaly detection and mechanisms of trust. The modern security environment is now being fundamentally transformed through the use of advanced machine-learning techniques and leading-edge technologies.

Few-shot learning has been promising in the direction of addressing zero-day vulnerabilities by enhancing real-time anomaly detection. Study [12] proposes an unsupervised few-shot learning framework utilizing FastText embeddings and Approximate Nearest Neighbor (ANN) search for API injection attack detection. The methodology proposed has a high accuracy and is flexible, with crucial innovations such as incremental learning

methods and tokenizers tailored for specific APIs, which makes it practical for real-world applications despite the issues regarding dataset representativeness and protocol diversity. The security vulnerabilities associated with Unmanned Aerial Vehicles (UAVs) are thoroughly taxonomized in [13], analyzing susceptibilities present in the hardware, software, and communication layers. The paper discusses AI-driven techniques like federated learning and reinforcement learning, in the context of adaptive security and decentralized processing, which implies the necessity of standardized and scalable benchmarks. Similarly, autonomous vehicles are attacked by adversarial attacks, as addressed in Study [14], which analyzes attack scenarios using ISO 21448. This article synthesizes recent advances in adversarial defenses and robust sensor technologies, thereby offering a roadmap for simulation-based evaluations and enhanced safety. A novel approach to increasing AI trust is introduced by integrating blockchain into LLMs in Study [15]. The BC4LLM framework uses the decentralized and immutable characteristics of blockchain to secure the training and output of LLMs for applications in highly sensitive domains, such as fintech and healthcare. It also addresses the challenges of scalability and energy efficiency. All these studies together present how AI is shaping the future of security across different domains with the help of complementary technologies such as blockchain.

### 2.3 UAV-enabled wireless networkds and IoT integration

Integration of UAV-enabled wireless networks (UWNs) with the Internet of Things (IoT) systems is one of the modern communication and intelligent infrastructure paradigms. Using artificial intelligence, these networks can respond to key issues of resource management, scalability, and fault tolerance and thus can offer autonomous and adaptive capabilities. In many fields, like smart cities, agriculture, disaster management, etc., UAVs have been indispensable nodes in transmitting real-time data to enhance network efficiency. This integration of artificial intelligence-based systems and new approaches enabled such advancement in these interconnected systems, as could be inferred from the following studies.

The transformative potential of AI in UAV networks and IoT ecosystems is substantiated through the advances in optimization, resource management, and data generation. Study [3] comprehensively reviews the AI applications in UAV networks focusing on trajectory optimization, radio resource management, and edge computing. Techniques such as deep reinforcement learning (DRL) and multi-agent learning have been instrumental in improving resource optimization and adaptability. Practical implementations like intelligent caching and UAV placement address the issues of scalability and energy efficiency, though there is always the computational constraint. The IoIT paradigm further pushes these innovations as Study [16] discusses, for instance, TinyML and 5G in applications of smart cities, healthcare, and military. IoIT, while at the nascent stage, even with interoperability and resource-constrained limitations, shows a future direction with promises of federated learning and adoption of 6G. Enhancing IoT networks further, Study [17] presents the LLM-ENFT framework, where large language models are integrated into edge networks to perform proactive fault diagnosis and decision-making. Simulation results have shown an improvement in throughput and resilience, showing that LLMs will be the cores for autonomous fault management, though challenges remain in computational overhead. In response to the pressing demand for scalable training datasets, Study [18] introduces the ASDA framework, which utilizes procedural generative methods alongside domain randomization (DR) to augment the diversity of datasets pertinent to machine learning in drone-related applications. When combined with the AirSim engine, ASDA markedly enhances both data diversity and efficiency, addressing deficiencies in the availability of real-world data while promoting advancements in aerial autonomy. Together, these studies highlight AI's pivotal role in optimizing UAV networks, IoT systems, and edge computing while addressing scalability and adaptability challenges.

### 2.4 LLMs for command interpretation and human-robot interaction

Large Language Models, in their development, have revolutionized the way humans can interact with machines by enabling natural language to become a practical interface for understanding and executing commands. This has fostered smooth human-machine collaboration, more so in the complex and dynamic contexts, due to the effectiveness of bridging linguistic input and autonomous responses.

The integration of NLP and LLMs in UAV systems and robotics has greatly enhanced command mechanisms, human-robot interaction (HRI), and industrial automation. Study [4] shows the application of a fine-tuned BERT model for UAV navigation, mapping natural language commands to North-East-Down (NED) coordinates with 89% accuracy. This framework, using Raspberry Pi Zero W and MAVLink protocols, achieves resource-efficient, real-time navigation and shows the potential for integration of multimodal input. Beyond UAVs, Study [6] looks at the applications of LLMs in HRI, with a strong emphasis on planning, contextual reasoning, and ethical decision-

making. The paper classifies the techniques into domains such as education and healthcare, which meet the scalability and safety needs for adaptive and robust robotic systems. Similarly, Study [7] presents LLM-driven advances in humanoid robotics, where these models are used as trajectory planners and reinforcement learning designers to achieve stable locomotion, validated through simulations in Unity. Though the challenges in multi-task integration and computational demands remain, the research shows the transformative potential of LLMs in humanoid robotics. In industrial settings, Study [19] introduces a hierarchical framework leveraging LLMs to translate human commands into precise robotic actions for manufacturing. This line of approach has an 81.88% success rate in tasks such as tool path design and decision-making, thus saving production while addressing scalability challenges in complex 3D tasks. Taken together, these studies show the critical role that LLMs play in advancing UAV navigation, HRI, humanoid robotics, and manufacturing automation, with future directions for improving adaptability, efficiency, and multi-modality.

### 2.5 Multi-agent systems and collaboration

Integration of Large Language Models in multi-agent systems has revolutionized collaborative workflows in autonomous environments and opened new directions for efficient coordination and decision-making. LLMs help drive these systems with human-like reasoning and interaction, leading to tremendous progress in several domains such as robotics, logistics, and aerial intelligence. This section will cover innovative frameworks and applications where LLMs and AI foster collaboration among agents and outline some of the challenges and future directions that are molding this field.

The integration of large language models and advanced computational techniques has enhanced MAS, UAV coordination, vehicle-drone logistics, and aerial intelligence. Study [20] presents an LLM-based MAS framework with a profile, perception, and mutual interaction module for structured management of complex workflows in industrial and societal simulations. The main contributions of this study are the taxonomy of MAS methodologies and innovative multimodal data integration to improve coordination and efficiency. In UAV coordination, Study [21] presents a linguistics-driven system using SDF and artificial potential field algorithms for real-time swarm geometry adjustments and safe flocking behaviors, with applications ranging from drone shows to VR simulations. Future efforts will focus on improving scalability and aesthetic interactions. The study [22] investigates vehicle-drone cooperative systems for logistics by classifying operational models into single-vehicle single-drone (SVSD) and multiple-vehicle multiple-drone (MVMD) among others, and proposing mathematical formulations such as TSP-D. Focusing on urban and disaster scenarios, the study underlines hybrid models and heterogeneous drones for better synchronization and cost efficiency. Advancing aerial intelligence, Study [23] discusses CNN-based methods for aerial human action recognition and object detection, which overcome challenges such as occlusion and limitations of datasets. The study highlights the potential in multi-modal learning and scalability by combining CNNs with transformers for spatiotemporal analysis, with enhancements in applications in surveillance and urban planning. Taken together, these developments show that LLMs are at the cusp of a transformative role across domains, enabling better adaptability, scalability, and efficiency in collaborative and real-time operations.

### 2.6 Code generation and API recommendations

The rapid evolution of software ecosystems has created an urgent need for tools able to support code generation and improve API suggestions. Large Language Models have been powerful enablers in this domain, exploiting their understanding of semantics and reasoning capabilities in dealing with ever-increasing complexities of development workflows. The present chapter discusses major LLM-powered code generation and API optimization platforms that could be game-changers across a very broad spectrum of settings.

Improvements in large language models revolutionized graph representation, API usage, conversational agents, software automation, and legacy code modernization. Study [24] introduces the L-MTAR framework, which combines semantic and positional encoding using motifs with a linearizing Transformer architecture, achieving state-of-the-art results in API recommendation tasks where HR@10 exceeds 0.93 on datasets such as ProgrammableWeb, hence showing promising results when dealing with diversity and complexity of APIs. Apart from this, Study [25] proposes the CAPIR framework that solves the problems related to low-resource libraries by decomposing tasks into smaller parts, retrieving and ranking APIs, and directly incorporating APIs into prompts. Showing huge gains in Recall@5 and pass@100, CAPIR is an example of domain-specific adaptability of LLMs. In the healthcare domain, Study [26] explores integrating REST APIs with conversational agents using LLMs for empathetic and personalized health counseling. With a modular design, seamless API integration, and URI-based resource optimization, this framework has great potential in digital healthcare while addressing issues of scalability

and data security. In software automation: Study [27] introduces LLM_GP, an evolutionary algorithm for mutation, crossover, and selection, using LLMs in code generation. LLM_GP, through tokenized code sequences with prompt engineering, is outperforming traditional approaches to symbolic regression, a strong example of the synergy between LLMs and evolutionary algorithms. Study [28] is working on the modernization of legacy code with an LLM-driven type-migration pipeline for the resolution of ambiguities in type conversion, to adapt idiomatic Rust structures. This will significantly lower the effort of developers while preserving maintainability and security in the modernized system. Together, these architectures show the LLM's powerful ability to accelerate automation, scalability, and adaptability within a wide range of software and application domains.

### 2.7 Environmental monitoring and applications

Integration of such advanced technologies as GIS, large language models (LLMs), and AI-driven tools is revolutionizing workflows in environmental engineering and industrial workflows. Study [29] demonstrates a system that integrates GIS with weather data and GPT-based analysis for the identification of fertile land and crop recommendation with an accuracy of 80% using some techniques like Panchromatic Sharpening and Brovey Transformation. While it highlights its potential to enhance agricultural productivity and sustainability, the study recognizes the limitations in the scope of the datasets and plans to include more expansive datasets and economic factors. In engineering, Study [30] investigates the use of LLMs like GPT-4 in automating multibody system dynamics modeling. Using Python's Exudyn library, the study shows that LLMs can simplify workflows in simulation for simple scenarios. However, challenges in managing complex rigid-body dynamics and a need for multi-modal inputs and domain-specific training data remain. Expanding into industrial applications, Study [31] evaluates the TAPE framework for testing REST API tools like EvoMaster and StarCoder. While these tools do not yet surpass manual test quality, they significantly improve efficiency and usability, providing practical solutions for managing API complexities. Collectively, these advancements underscore the transformative role of LLMs and AI-driven technologies in fostering automation, efficiency, and adaptability across diverse applications while highlighting avenues for future improvements in scalability, accuracy, and dataset integration.

### 2.8 Emerging frameworks

The development and use of large language models, or LLMs for short, together with associated technologies, have brought profound influence on a wide circle of fields, among them dealing with difficult computational tasks, improving automation, and shrinking perception-action gaps. More recently, it has been explained in the literature how such applications of LLMs revolutionize discrete controller synthesis, server management, robotic assembly, and autonomous operation of systems. A theme common to many of these studies is combining semantic reasoning with feedback mechanisms, enabling iteration to improve efficiency and flexibility.

Recent advances in large language models and foundation models are overcoming the key barriers of computational efficiency, automation, and robotic autonomy in these areas. Study [32] demonstrates that through a structured policy design combined with semantic reasoning in prompt engineering, the computational cost of discrete controller synthesis can be drastically reduced by significantly lowering the need for state and transition analysis. Herein, Research [33] advises using GPT-4-based AI agents for such complicated multitasking in categories that demand a high success rate for the simplification of workflows to reduce the chance of human error in the server management domain. In the field of construction robotics, Research [34] introduces a hierarchical framework that integrates high-level planning with low-level policy development to improve the accuracy and modularity of robot control scripts, consequently enhancing flexibility in intricate assembly operations. Taken together, these studies show the huge impact of LLMs and foundation models on decision process optimization, improvement of automation, and robotic adaptability, while some of the most important challenges remain in scalability and precision.

## 3. System Architecture

The overall architecture of the system is designed to translate natural language instructions into executable API calls within the IoD ecosystem. This architecture leverages LLMs to understand human intentions and then translates them into suitable IoD functionalities, hence offering a seamless and efficient interface for interaction between humans and drones. The current section provides an end-to-end design consideration in line with its various components, methodologies, and technical aspects in terms of natural language processing and application programming interface integration. LLM processing module is the engine that enables understanding and

processing of user inputs. The module consolidates all the text normalization activities, tokenization, intent recognition, and parameter extraction into a single straightforward workflow. Especially tailored and created for IoD datasets, it ensures correct transformation of the instructions into well-structured API calls to be executed in the IoD operating environment. The LLM is next to be processed, where the natural language input will undergo processing so that it can infer the intent of the user. Extracts actionable parameters—drone IDs, geographical locations, and task-specific goals—via the use of NER, dependency parsing, and semantic analysis. For instance, an instruction such as “Monitor the western sector at sunrise” is converted into a formal request: POST /monitor {“location”: “western sector”, “time”: “sunrise”}.

It also supports real-time ambiguity resolution to enable user interaction. The LLM performs. For instance, if a command lacks essential parameters, such as in the sentence “Send drones to the field” it uses contextual awareness to infer the missing parameters from previous messages or to form clarification questions, like “Which field do you mean?”. This iterative process makes sure that commands are well understood before they are translated into API requests. More of these pre-processing tasks are done in the workflow of the LLM. Typo correction and colloquial handling are a part of these. Thus, it is able to understand instructions like “pls snd drone 2 south” as “Please send a drone to the South” making it truly usable by a wide variety of users. It even remembers follow-up commands, like “Use the same settings as before” with contextual memory—meaning, it recalls and reapplies without having to restate the relevant setting.

In that sense, API Gateway serves as the important middleman in this whole IoD ecosystem, assuring that processed requests are smoothly translated into executable commands. That is what it guarantees for the continuity of operations under standards by IoD, and hence ensures efficiency and reliability between the LLM-driven processing units and drone operation systems. Validation is definitely one of the very basic but most important elements a gateway uses to check API requests for completeness and correctness, and to be compliant with the operation protocols defined in IoD. This way, it does ensure that any such request that is malformed or incomplete is caught well in advance to prevent errors that may arise during execution. For instance, it would ask for more inputs or corrections if any of the key parameters were missing, such as geolocation or drone identifiers.

Dynamic Mapping then enables the API Gateway to route requests efficiently to appropriate endpoints in the IoD ecosystem. Restful APIs or gRPC, whichever are the protocols used in these communications, this mapping mechanism ensures that each request is properly dispatched to its target. Deployment requests are thus broadcast to the fleet management module, while commands from the analytics subsystem concerning data gathering are propagated via a broadcast mechanism. Adaptive routing is bolstered by the fact that the mapping system is context-aware and hence autonomously changes itself as the conditions in which the system is operated change. Most optimization work focuses on reducing payload overhead to increase the efficiency of request execution. The overall system efficiency is improved because the request payload is compressed, unnecessary data transmission is reduced, and API calls are optimized. What's more, this feature assumes greater importance in the case of scenarios with high demand, such as simultaneous drone operations, since it avoids network congestion and assures a quick response.

First of all, validation is the most critical aspect in which a gateway checks API requests for their completeness, correctness, and compliance with operation protocols defined in IoD. In this manner, it makes sure that any such request which is found malformed or incomplete is flagged well in advance before any error in execution could occur. For example, if some necessary parameters are missing, such as geolocation or identifiers of drones, it will ask for more inputs or correction. After that, the mechanism allows for routing requests to endpoints in the IoD ecosystem using Dynamic Mapping, ensuring the process continues uninterrupted. Whatever the communication protocol may be, be it RESTful APIs or gRPC, a mechanism of mapping ensures that the requests are mapped accordingly to their respective destinations. The deployment requests, therefore, reach the fleet management module, whereas the commands aimed at gathering data go to the analytics system. Adaptive routing of the same nature would only be possible using a context-aware mapping system capable of reconfiguring itself in dynamically changing operational conditions.

The primary optimization here is fundamentally to reduce the payload overhead and increase the efficiency of request execution. The gateway can significantly raise the overall performance of the system, mainly by compressing the payload of requests, cutting down the transmission of unrequired data, and optimizing the API calls. This becomes an important aspect in high-demand scenarios with several drones working concurrently, as this helps to lighten the network burden, ensuring quick responses.

Error Handling is another crucial part of the system; it warns users about possible issues that might be generated during operations. For instance, when the user enters the wrong command, near real-time feedback identifies the mistake and corrects it, as is evident from the message, “Invalid location specified. Please provide a valid geolocation”. This functionality not only minimizes the possibility of the system going wrong but also

enables users to rectify their mistakes efficiently, thus promoting an improved interaction experience. There is an adaptive learning process in the system that has a tendency to improve the error handling capacity with time on the basis of learning from the user-specific error trends. This process runs as a two-stage process of error detection and refinement of the adaptive response. Under the error detection stage, errors like invalid commands, incomplete parameters, or wrong identifiers are detected by carrying out rule-based verification processes. When errors are brought to the surface, the system logs such occurrences together with the involved user activity and corrections in a feedback database. The system mines the logs over time to identify recurring patterns of error for a given user or user groups. Machine learning techniques, including clustering techniques or pattern-discovery techniques, are used to identify common issues. For instance, if the user keeps inputting incorrect drone sector IDs (e.g., "Sector X" instead of correct "Zone 5"), the system associates the occurrences and learns to make personalized suggestions like, "Did you mean Zone 5 from past tasks?" or auto-suggest the most likely correct ID from historical data. This adaptive functionality improves efficiency as well as user experience. By minimizing the occurrence of redundant manual corrections, the system lowers command execution latency and enables interaction to be smoother. Adaptive processing is also employed to handle scenarios with sophisticated commands where partial or ambiguous input (e.g., "Send drones to the location") is obtained. In these instances, the system generates contextual memory by drawing on previous interactions and responds with intelligent follow-up questions such as, "What location do you want to survey?".

The Feedback Loop will be a two-way mechanism: on one side, it will provide actionable information and updates to the user so that effective monitoring and control of the activities by the drones are possible; on the other side, it will collect user feedback that can be used in fine-tuning system performance over time. For example, this can be achieved by tuning the performance of the LLM processing module such that better intent recognition and parameter extraction capabilities will be achieved based on trends of user edits.

This is further amplified by advanced analysis methodologies coupled with adaptive learning systems. To a certain degree, the study of user interactions and performance metrics will enable the system to predict and deliver proactive suggestions on potential problems that may arise. In the illustrated example above, repeated delays in one region could enable the system to suggest changes to operating parameters even before a user issues a command.

Furthermore, the Feedback Loop brings in scalability, as the same performance is provided in the face of changing user scenarios. Whether one is controlling a small fleet or managing a large operation, the module will scale dynamically in its communications approaches to meet the needs of the user. In high-pressure situations involving a lot of drones, it will prioritize key updates and push non-critical information into the background in an attempt to reduce the possibility of overwhelming the user with the EMCS.

One of the architecture elements of IoD is the Feedback Loop, which provides for real-time updating, offers complete error management, and is adaptive. The raw data is transformed into actionable intelligence, filling the gap in understanding and building a user-centric framework for interaction. To that end, it assures IoD to be intuitive, responsive, and effective in all feasible operational environments underpinned by the three principles of transparency, efficacy, and continuing improvement.

### 3.1 Technical Workflow

This architecture shall be cautious from the point of view of data flow so that fluidity and heterogeneity in user directives are handled and, most importantly, such natural language inputs can be concretized. While this architecture design follows a linear progression in itself, it does suit all those complex variations of command patterns for robustness and scalability.

The entire process initiates with the submission of input, wherein, through an interface-directly integrated with the LLM Processing Module-users are able to feed instructions. This may be one of those interaction points which will collect inputs ranging from simple directives to complicated queries such as, "Deploy two drones" or "Survey the eastern field at sunrise and report anomalies". In this context, an interface does very minimal pre-processing of inputs before passing it to the LLM module for deeper analysis.

At the very next step of processing, several levels of analysis are done at a large language model or LLM. The LLM recognizes the intent of the user, extracts the relevant parameters, and contextualizes the command further. State-of-the-art techniques, namely semantic parsing, Named Entity Recognition, and dependency parsing, are utilized while dealing with information that may be either explicit or implicit. Considering a very simple example, parsing of the instruction "Inspect the crop fields tomorrow" identifies "Inspect" as the action verb, "crop fields" as the place identified, and "tomorrow" as the temporal reference and forms an API call like POST /inspect { "location": "crop fields", "time": "2024-01-05"} in structured form. This confirms that the instruction was

understood correctly, even in the case of ambiguity or partial lack of information from the user's original utterance, with iterative clarification in case of need. The structured request is passed further to API Execution through the API Gateway. In this stage, the request gets checked against IoD standards for completeness and correctness. On successful validation, API Gateway forwards the request to a concrete IoD endpoint capable of realizing an action corresponding to one of the three categories: drone deployment, data acquisition, or system calibration. Such a monitoring request is then forwarded to the analytics subsystem, while a deploy command is forwarded to the fleet management module. Dynamic routing depends on changes in drone availability, network conditions, and operational priorities.

During the System Action phase, the core execution of the command is done by the core IoD system. The drones realize programmed actions in the mission, for example, flying to a coordinate, taking photos, or gathering environmental data. Real-time orchestration of multiple drones, using edge computing nodes for localized processing and cloud systems to extend visibility of the mission, will be needed. Mechanisms of feedback are enabled while at this stage, which allows the drones to make out real-time conditions, including obstacle avoidance or recasting of tasks due to unplanned environmental changes.

By any measure, the last phase of the interaction cycle is the User Feedback stage. In such a regard, when raw telemetry data and operational contexts are interpreted into intelligible insights for the users, the Feedback Loop provides actionable updates. For instance, it notifies users: “Mission completed successfully”, “Drone 3 encountered an obstacle and rerouted”, and “Data analysis in progress”. The error alarms, such as “Invalid location specified”, nag the user to make necessary improvements for easier interaction flow. The workflow is visualized in Figure 1.

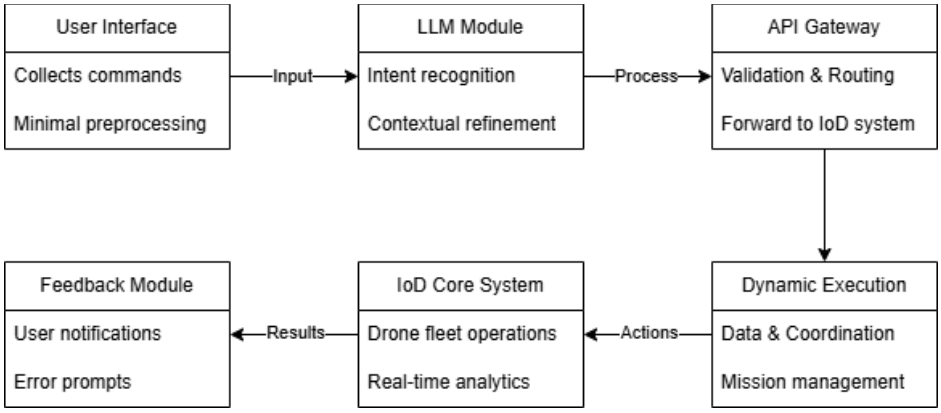


**Figure 1.** System architecture workflow.

This pseudo code, which can be seen in Table 1, illustrates the core workflow of an LLM-based IoD command processing system that translates natural language commands into executable API requests, ensuring real-time drone operation through systematic input parsing, validation, execution, feedback delivery, and performance optimization.

## 4. Dataset and Model Development

The creation of an effective system in the conversion of natural language to API calls within the Internet of Drones ecosystem requires well-structured datasets and powerful training methodologies. The subsequent section describes the creation and extension of the IoD application-specific datasets and the training, fine-tuning, and validation methodologies followed for LLM.

The first step towards building a powerful model is the collection of a large dataset. For IoD applications, datasets have to contain various sets of natural language directives, operating scenarios, and their corresponding API mappings. The creation process of the dataset consists of the following big steps: data collection, preprocessing, and augmentation.

**Table 1.** Pseudo code for LLM-Based IoD command processing system.

**Pseudo Code Workflow**

```
# Initialize the system
Initialize LLM_Model (fine-tuned for IoD commands)
Initialize API_Gateway
Initialize Feedback_Loop
Initialize Command_Log

# Main processing loop
While system is active:
   Receive user_input_command
   Log user_input_command to Command_Log

   # Step 1: Input Parsing
   parsed_command = LLM_Model.parse(user_input_command)
   If parsed_command is incomplete:
      clarification = LLM_Model.request_clarification(parsed_command)
      Receive user_clarification
      parsed_command = LLM_Model.update_command(parsed_command, user_clarification)

   # Step 2: API Request Generation
   api_request = LLM_Model.generate_API(parsed_command)
   Validate api_request with API_Gateway
   If api_request is invalid:
      Feedback_Loop.notify_error("Invalid API Request. Please check command.")
      Continue

   # Step 3: API Execution
   response = API_Gateway.execute(api_request)
   If response is error:
      Feedback_Loop.notify_error("Execution Failed. Check system logs.")
      Continue

   # Step 4: Feedback Delivery
   Feedback_Loop.send_update(response)

   # Step 5: Performance Monitoring
   Record processing_time, latency
   If processing_time > threshold:
      Optimize system components (parallel processing, query handling)

   # Loop End
End While
```

### 4.1. Data collection

Initially, the system was bootstrapped using simulated data, which included designing representative scenarios for IoD typical operations—like disaster response, agricultural monitoring, and urban surveillance. The commands were developed manually by experts in the domain and extended by paraphrasing techniques in order to simulate the inputs of different users. For instance, a base instruction like "Survey the northern field" was expanded to variations like "Inspect the northern area", "Deploy drones for surveillance in the north", and "Monitor the field in the north". These artificial conversations made up the core training data for the initial models.

As the system evolved, real-time logging of user interaction became very important in collecting data from active operational contexts, which consists of collecting inputs made by the user, system reactions, and feedback obtained in practical situations. For instance, a logged command like "Deploy two drones to monitor crop health in sector 5" can be traced back to its corresponding API response for model improvement. These logs contain a record of the actual user behavior, colloquial phrasings, unclear instructions, and corrections. For example, a user might enter "Check the area" in a first iteration and then clarify to "Survey the southern zone for anomalies". Iterative logs capture such subtlety present in real-world interactions and are used in model updating. In order to increase dataset diversity further, additional techniques were used. Paraphrasing of commands was done to represent regional and contextual differences. A command "Deploy drones to survey the field" was paraphrased as "Send UAVs for field observation" or "Survey the crop area". Edge cases involving partial directives were added to validate the model's capacity to request clarifications and disambiguate in real-life situations.

Furthermore, the system records exceptional cases and infrequent occurrences, including incomplete directives or instructions involving multiple steps. For example, a directive such as “Inspect the field” may elicit a subsequent clarification from the system: “Which field would you like to inspect?”. The user's reply, “Sector 4” is subsequently integrated into the log, thereby enhancing the dataset with instances of contextual resolution.

Over time, this two-fold approach—simulated and real data combined—ensures the dataset matures to capture both expected and new trends in user engagement. Further, the collected data is enriched with the inclusion of region-specific linguistic patterns like using “patrol” instead of “survey” making it more inclusive and adaptable across different operational environments.

### 4.2. Augmentation

Augmentation techniques—such as paraphrasing, synonym replacement, and sentence reordering—are applied to increase the size and diversity of the dataset. These make sure that the model generalizes well to varied phrasings like “Send two drones to the northern field at dawn” and “At sunrise, deploy two UAVs for northern field surveillance”. Paraphrasing tools and LLMs are used to generate sentence variants that preserve the original intent but offer linguistic variety. For instance, the phrase “Monitor the southern area” may be rephrased as “Inspect the southern region”, “Survey the south”, or “Deploy drones to observe the southern sector”.

Synonym replacement: It replaces the key terms with synonymous words or phrases. For example, “deploy” can be substituted with “send” or “dispatch”, and “inspect” can be substituted with “examine” or “observe”. In this way, the model understands the commands with different vocabularies but retains semantic correctness. Sentence reordering Similarly, sentence reordering creates structural variations. A fundamental instruction such as “At dawn, deploy drones to the eastern field” can be restructured to “Deploy drones to the eastern field at dawn” or alternatively “Drones should be deployed at dawn for the eastern field”.

Augmentation also contains edge cases such as ambiguous or incomplete commands, including the partial directive “Survey the field” expanding to “Survey the western field”, “Survey the crop fields”, or “Survey the field at 8 AM”. Such edge cases will eventually make the model strong enough to deal with real-world scenarios in which users give very ambiguous or fragmented input. Also, in order to make the model more robust, adversarial examples are included, such as intentionally misspelled or grammatically incorrect commands. The command “Sennd drones 2 west” is mapped to a corrected interpretation, “Send two drones to the west”.

Sophisticated augmentation techniques involve the integration of contextual knowledge to generate domain-specific information. In the agricultural monitoring domain, commands such as “Analyze crop health in sector 4” can be grounded into other alternatives like “Check vegetation health in zone 4”, “Monitor plant conditions in section 4”, or “Survey crops for anomalies in region 4”. In the disaster response domain, commands like “Assess flood damage in the northern area” can be further detailed with, for instance, “Evaluate water levels in the north” or “Inspect flood impact in northern regions.”

With such augmentation strategies, the dataset will not only increase in size but also in representativeness and robustness, meaning the model can generalize well over a large space of user inputs, operational contexts, and unexpected linguistic variations. The diverse and enriched dataset is more able to ensure the model can give accurate API outputs even in complex or unexpected scenarios.

### 4.3. Model development and fine-tuning

Only after curating such a dataset does developing and fine-tuning an LLM to understand natural language instructions and convert them into API calls come into question. This approach thus fits into model selection, pretraining, fine-tuning, and subsequent evaluation. Because it has been confirmed to work well, especially in cases requiring precise intent identification, fine-tuned versions of BERT are selected to perform IoD. BERT models are pretrained on large corpora and thus capable of capturing the linguistic nuances and semantic relationships important in mapping user intents into structured API requests. In this respect, fine-tuning BERT on IoD-specific datasets optimizes it for certain domain-specific tasks associated with API formatting, such as parameter extraction and intent classification; hence, guaranteeing superior performance in converting natural language commands into actionable output.

It is fine-tuned on the curated IoD-specific dataset by adapting the pre-trained BERT model to the domain-specific tasks. In this step, the model is optimized with task-specific objectives: intent recognition, parameter extraction, and API formatting. The RAG operation is performed over these labeled command-API pairs. This would further solidify the model's capability of extracting relevant data from a structured knowledge base during training and generating outputs that closely match the operational API requirements. The outputs of the model are

then fine-tuned with reinforcement learning using human feedback to achieve high accuracy for diverse input scenarios.

Model performance is assessed on the basis of a set of established metrics, which are tailored to natural language processing in order to guarantee linguistic accuracy as well as functional accuracy by ensuring functional reliability. In the methodology of evaluation, the techniques and their applications include:

Some of the metrics used in measuring the model's ability to correctly classify user intent are accuracy and the F1 score. Taking a command, for example, like “Deploy drones to inspect the northern region”, it should be able to recognize “deploy” as the intent. Accuracy measures the number of correct intentions identified divided by the total number of inputs; the F1 score balances precision and recall in showing a holistic view of performance.

Parameter extraction: Extracting geolocations, timestamps, and action objects from user commands. Precision measures the ratio of correctly identified parameters to all extracted parameters, while recall measures the ratio of correctly identified parameters to all relevant parameters in the input. For example, given the command “Monitor the eastern zone at 6 AM”, it should be able to correctly tag “eastern zone” as the location and “6 AM” as the time. High F1 score here ensures complete parameter extraction.

BLEU is short for Bilingual Evaluation Understudy and a metric to calculate the similarity of generated API requests and reference requests. In our example, it computes how well the model generated POST /monitor request with parameters {“location”: “eastern zone”, “time”: “6 AM”} matches a predefined reference API call. It then uses BLEU for the computation, combined with cosine similarity, to compute the semantic closeness of the generated and reference outputs by comparing vector representations of the commands, ensuring that the underlying intent and semantics are correctly aligned even when syntax differs. Integration of BLEU in assessing syntactic precision with cosine similarity in measuring semantic accuracy forms a complete evaluation framework for assessing API generation quality. Realistic scenarios are simulated in order to exercise the entire system workflow, which includes input commands, execution of the API, and feedback presentation. These are simulated in an attempt to duplicate operational conditions for the purpose of ensuring that the system can, with skill, handle complex and dynamic user inputs. For instance, the system could interpret a directive such as “Survey the southern area and send images by noon”, producing a formatted API request exemplified by POST /survey {“area”: “southern”, “deadline”: “noon”}. Upon execution, the feedback mechanism assesses the results, verifying that the requested images have been accurately captured and transmitted.

All of these scenarios are used by simulations to ensure robustness, from multi-step tasks and ambiguous inputs to high-demand operations. For example, the multi-step instruction “Deploy three drones to inspect the northern field, then report findings to the base” is translated into a series of sequential API requests. The system ensures that each step gets performed in the correct sequence, while the feedback mechanism keeps updating in real-time, such as “Drone 1 deployed successfully” or “Inspection data transmitted to the base”.

Edge cases are also an essential aspect of such simulations. Similarly, ambiguous instructions, such as “Inspect the area”, make the system request more context: “Which area would you like to inspect?”. Through more user-provided context, the system demonstrates dynamic adaptation—how it handles partial inputs. High-load scenarios like managing a number of simultaneous operations in several regions are also tested to validate the scalability and latency of the system. Commands such as “Deploy drones to all sectors for fire monitoring” are processed to have adequate load balancing and an effective execution of tasks.

These simulations include metrics such as success rates, latency assessments, and user feedback evaluations. In a simulated scenario of disaster response, for example, it measures the system's ability to deploy drones within critical time limits and provide actionable updates by comparing the results against set benchmarks. High success rates on these assessments make a strong case for the system's reliability in the face of actual conditions.

Through holistic simulations, this system integrates diverse and challenging situations, giving support to wide examination of its operational processes and therefore emphasizing benefits while highlighting areas for improvement. This iterative process ensures the IoD platform will retain resilience, flexibility, and efficiency in meeting user needs across a large number of applications.

## 5. Evaluation and Results

The paper evaluates the proposed system by assessing the effectiveness of Llama-3.2-1B-Instruct in producing accurate API requests based on natural language inputs. The evaluation uses Retrieval-Augmented Generation (RAG) methodology to extend the model with an ability to extract and apply relevant information from a structured knowledge base. The evaluation framework consists of metrics such as BLEU score and cosine similarity in order to measure syntactic and semantic accuracy, respectively. The next few sections describe the evaluation methodology, the experimental results, and a detailed analysis of the system's features.

To ensure a solid evaluation, the system was tested with a curated dataset containing 1500 commands representative of diverse IoD scenarios, from simple commands like “Deploy three drones” to complex, multi-step instructions like “Inspect the southern sector and relay images back to the base”. Each command was accompanied by its corresponding ground truth API request for the comparison process. Below are some illustrative pairs (Figures 2–5).

Illustration 1:

- Command: “Deploy three drones to survey the eastern field.”
- API Request:


```
{
    "action": "deploy",
    "number_of_drones": 3,
    "location": {
        "region": "eastern_field",
        "coordinates": {
            "latitude": 39.837128,
            "longitude": 26.241863
        }
    },
    "task": {
        "type": "survey",
        "parameters": {
            "altitude": "300m",
            "coverage": "full"
        }
    },
    "priority": 2,
    "timestamp": "2025-01-10T10:00:00Z"
}
```


**Figure 2.** API request generated for deploying three drones to survey the eastern field.

Illustration 2:

- Command: “Monitor crop health in sector 7 and send a report by 5 PM.”
- API Request:


```
{
    "action": "monitor",
    "task": {
        "type": "crop_health",
        "parameters": {
            "sensors": ["thermal", "visual"],
            "data_format": "geojson"
        }
    },
    "location": {
        "sector": 7,
        "coordinates": {
            "latitude": 39.837128,
            "longitude": 26.241863
        }
    },
    "deadline": "17:00",
    "notifications": {
        "email": "anil.sezgin@hypersense.dev",
        "sms": "+123456789"
    },
    "priority": 2
}
```


**Figure 3.** API request for monitoring crop health in sector 7 and sending a report by 5 PM

Illustration 3:

- Command: “Inspect the southern sector for anomalies and relay images back to the base.”
- API Request:


```
{
    "action": "inspect",
    "location": {
        "sector": "southern",
        "coordinates": {
            "latitude": 39.837128,
            "longitude": 26.241863
        }
    },
    "task": {
        "type": "anomaly_detection",
        "parameters": {
            "camera": {
                "type": "infrared",
                "resolution": "4K"
            },
            "detection": {
                "objects": [
                    {
                        "type": "person",
                        "threshold": 0.85
                    },
                    {
                        "type": "vehicle",
                        "threshold": 0.9
                    }
                ]
            }
        }
    },
    "priority": 2,
    "timestamp": "2025-01-10T14:30:00Z"
}
```


**Figure 4.** API request for inspecting the southern sector and relaying images to the base.

Illustration 4:

- Command: “Return all drones to the base immediately.”
- API Request:


```
{
    "action": "return",
    "target": "all_drones",
    "location": {
        "type": "base",
        "coordinates": {
            "latitude": 39.837128,
            "longitude": 26.241863
        },
        "details": {
            "name": "central_operations_base",
            "capacity": "6_drones"
        }
    },
    "priority": 1,
    "status": {
        "current": "in_progress",
        "estimated_completion_time": "2025-01-10T14:30:00Z"
    },
    "alerts": [
        {
            "type": "notification",
            "method": "email",
            "address": "anil.sezgin@hypersense.dev",
            "message": "All drones are returning to base."
        },
        {
            "type": "log",
            "level": "info",
            "timestamp": "2025-01-10T14:00:00Z"
        }
    ]
}
```


**Figure 5.** API request for the immediate return of all drones to the base.

Metrics such as the BLEU score and cosine similarity give a holistic framework for evaluating how effective the system is at generating accurate API requests. The BLEU score measures n-gram precision by calculating the amount of overlap between the generated API requests and their reference requests. This ensures the syntactic correspondence of the output to measure how faithful the model is to the expected phrasing. For example, for an instruction such as “Survey the northern region”, if the model generates POST /survey {“region”: “north”}, then a high BLEU score reflects that the generated request is in exact agreement with the expected format.

On the other side, cosine similarity captures semantic similarity by transforming API requests to vector representations and then calculating the cosine of the angle between them. Such an approach catches much better those deeper relationships between input and output, especially where syntax might differ, but the essential meaning remains the same. For instance, the instructions “Inspect the western field at sunrise” and “Survey the west sector early in the morning” may lead to API requests that are structurally distinct; however, they exhibit a cosine similarity score of 0.97, which signifies an almost complete semantic correspondence.

By combining BLEU and cosine similarity, the evaluation model ensures a comprehensive assessment of both syntactic accuracy and semantic correctness. Theoretically, instructions with a BLEU score above 90 and a cosine similarity of more than 0.95 stably reflect good model reliability. For multi-step commands like “Deploy drones to sector 4, inspect for anomalies, and report findings”, discrepancies in timestamp formatting or parameter extraction might reduce the BLEU score to 85, but a cosine similarity of 0.92 would confirm semantic alignment. This dual-metric approach ensures the system's robustness across a variety of IoD command complexities, enabling precise and meaningful API generation.

The testing scenarios were designed to determine how well the model would perform across a variety of complexities and uncertainties. Simple directives were single-intent instructions with very clear parameters—for example, “Deploy two drones to monitor sector 5”. These directives supported basic tests of intent recognition and parameter extraction. In contrast, complex directives examined the model's ability to handle multi-step instructions that require sequential API generation. For instance, a command like “Inspect the southern sector, relay findings to the base, and prepare a status report” demanded the accurate execution of several interdependent tasks. Ambiguous commands presented additional challenges, requiring the model to interpret context or seek clarification. Instructions such as “Survey the area” or “Send drones” were designed to specifically test the system's ability to obtain more information from the users and to adapt its responses in answer to these refinements. The test of those various scenarios sufficiently proved the model's robustness and adaptability in realistic IoD contexts.

## 6. Conclusion and Future Directions

The development of advanced IoD systems shows the transformative power of large language models in creating a bridge between human intent and machine execution through generation. It achieves a high level of accuracy and adaptation by bringing to bear on a comprehensive dataset for training and a two-metric assessment framework, consisting of BLEU scores and cosine similarity. This allows the human operators to engage drones uninterrupted, bringing in accurate and contextual API generations across vastly complex and diverse operational environments. All the same, despite the high-level achievement attained, there are still quite a number of aspects primed for improvement and further exploration. This research has several important achievements, and perhaps the most prominent among them is the combination of RAG methodologies with the Llama model. This would not only improve the ability of the model to handle complex, multi-step commands but also improve its capacity to retrieve and incorporate relevant contextual information dynamically. The ability of the system to understand ambiguous or incomplete user inputs and produce actionable, precise API calls has made it a robust solution for real-world IoD applications ranging from disaster response to agricultural monitoring. Challenges in high-demand contexts, such as delays, and sporadic errors in command classification, demonstrate that there is always room for improvement. Future research should focus on improving these limitations to ensure both scalability and reliability in large-scale applications. Another important direction is expanding the dataset with more diverse domain-specific scenarios and edge cases. While the existing dataset forms a good base, region-specific language variations, multilingual support, and more real-time data from field operations can only make the system more robust. Similarly, with multimodal data—like images and sensor inputs—the model can learn to generate richer and more contextually relevant API calls, enabling multimodal IoD systems that integrate vision, language, and real-time analytics.

From a technological standpoint, the inclusion of edge computing may be critical in reducing latency and assuring real-time responsiveness. The closer commands are executed to where an operation is being performed, the more edge nodes can offload the central servers from unnecessary computational tasks—especially in applications demanding real-time response, such as wildfire monitoring or disaster relief. Combination of this

approach with lightweight model architecture developments could allow effective deployment even in resource-constrained environments.

The adaptive learning methodologies will be definitely under consideration in integrating with the future versions of this system so that the model itself will keep evolving with new data and interactions from users. For example, RLHF and self-supervised learning can help the model keep pace with the changing operational demands; in reinforcement learning loops, the model could improve the understanding of ambiguous instructions by giving more weight to the cases where successful clarification has happened.

Another hopeful direction is the ethical and secure deployment of such systems. The most central challenges that would need focused attention are data-privacy protection, misuse of information, and dealing with potential biases in command interpretation. Major efforts in security protocols, including encrypting data transmissions and secure API access, could mitigate risks and foster trust among users. Equally importantly, the mechanisms for auditing—that is, those systems that monitor and explain model decisions—may most importantly bring transparency into the system, ensuring it remains accountable and interpretable.

Collaboration with experts in these diverse domains can significantly accelerate the deployment and enhancement of this technology. Alliances with response teams dealing with emergencies, agricultural organizations, or experts in urban development could expose specific problems within an industry and give way to the development of solutions tailor-made for that particular case. Working with disaster response teams could, for example, highlight very concrete operational needs—such as independent control of drone swarms in chaotic environments—that can guide successive feature enhancements.

The real potential of this system lies beyond the traditional applications of IoD. Its use in autonomous shipping, precision forestry, and smart city infrastructure will unlock new opportunities and further drive innovation. These industries are in a good position to harvest the ability of the system in understanding complicated commands and performing precise actions to fully realize the benefits that come with efficiency and automation—hence, wider adoption of IoD platforms. This research has laid a strong foundation that large language models can be integrated into IoD systems with great potential for improving operational efficiency, flexibility, and user experience. Addressing the current limitations and exploring new directions, the system will evolve to become a core technology for IoD platforms and beyond, revolutionizing the way humans interact with autonomous systems.

**Acknowledgement**

We would like to thank Siemens Turkey for their support in the completion of the study. This study was supported by TUBITAK 1512 Grant, project no: 2220423.

## References

[1] Sezgin A, Boyacı A. Securing the Skies: Exploring Privacy and Security Challenges in Internet of Drones. In: 2023 10th International Conference on Recent Advances in Air and Space Technologies (RAST); 2023; Istanbul, Turkiye, pp. 1-6.

[2] Sezgin A, Boyacı A. Advancements in Object Detection for Unmanned Aerial Vehicles: Applications, Challenges, and Future Perspectives. In: 12th International Symposium on Digital Forensics and Security (ISDFS); 2024; San Antonio, TX, USA, pp. 1-6.

[3] Zhou L, Yin H, Zhao H, Wei J, Hu D, Leung VCM. A Comprehensive Survey of Artificial Intelligence Applications in UAV-Enabled Wireless Networks. Digit Commun Netw 2024; 1-26.

[4] Savenko I. Command interpretation for UAV using language models. In: 2024 IEEE 7th International Conference on Actual Problems of Unmanned Aerial Vehicles Development (APUAVD); 2024; Kyiv, Ukraine, pp. 228-231.

[5] Sezgin A, Boyacı A. Rising Threats: Privacy and Security Considerations in the IoD Landscape. Journal of Aeronautics and Space Technologies (JAST) 2024; 17: 219-235.

[6] Zhang C, Chen J, Li J, Peng Y, Mao Z. Large language models for human–robot interaction: A review. Biomim Intell Robot 2023; 3(4): 1-15.

[7] Sun S, Li C, Zhao Z, Huang H, Xu W. Leveraging large language models for comprehensive locomotion control in humanoid robots design. Biomim Intell Robot 2024; 4(4): 1-16.

[8] Du F, Ma X, Yang J, Liu Y, Luo C, Wang X, Jiang H, Jing X. A Survey of LLM Datasets: From Autoregressive Model to AI Chatbot. J Comput Sci Technol 2024; 39: 542-566.

[9] Kim Y, Kim D, Choi J, Park J, Oh N, Park D. A survey on integration of large language models with intelligent robots. Intell Serv Robot 2024; 17: 1091-1107.

[10] Jang D, Cho D, Lee W, Ryu S, Jeong B, Hong M, Jung M, Kim M, Lee M, Lee S, Choi H. Unlocking Robotic Autonomy: A Survey on the Applications of Foundation Models. Int J Control Autom Syst 2024; 22: 2341-2384.

[11] Sauvola J, Tarkoma S, Klemettinen M, Riekki J, Doermann D. Future of software development with generative AI. Autom Softw Eng 2024; 31: 1-8.

[12] Aharon U, Dubin R, Dvir A, Hajaj C. A classification-by-retrieval framework for few-shot anomaly detection to detect API injection. Comput Secur 2025; 150: 1-13.
[13] Tlili F, Ayed S, Fourati LC. Advancing UAV security with artificial intelligence: A comprehensive survey of techniques and future directions. Internet Things 2024; 27: 1-27.
[14] Ibrahum ADM, Hussain M, Hong J. Deep learning adversarial attacks and defenses in autonomous vehicles: a systematic literature review from a safety perspective. Artif Intell Rev 2024; 58: 1-53.
[15] Luo H, Luo J, Vasilakos AV. BC4LLM: A perspective of trusted artificial intelligence when blockchain meets large language models. Neurocomputing 2024; 599: 1-20.
[16] Oliveira F, Costa DG, Assis F, Silva I. Internet of Intelligent Things: A convergence of embedded systems, edge computing and machine learning. Internet Things 2024; 26: 1-20.
[17] Fang H, Zhang D, Tan C, Yu P, Wang Y, Li W. Large Language Model Enhanced Autonomous Agents for Proactive Fault-Tolerant Edge Networks. In: IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2024; Vancouver, BC, Canada: IEEE. pp. 1-2.
[18] Sabet M, Palanisamy P, Mishra S. Scalable modular synthetic data generation for advancing aerial autonomy. Robot Auton Syst 2023; 166: 1-17.
[19] Fan H, Liu X, Fuh JYH, Lu WF, Li B. Embodied intelligence in manufacturing: leveraging large language models for autonomous industrial robotics. J Intell Manuf 2024; pp. 1-17.
[20] Li X, Wang S, Zeng S, Wu Y, Yang Y. A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. Vicinagearth 2024; 1: 1-43.
[21] Lykov A, Karaf S, Martynov M, Serpiva V, Fedoseev A, Konenkov M, Tsetserukou D. FlockGPT: Guiding UAV Flocking with Linguistic Orchestration. In: 2024 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct); 2024; Bellevue, WA, USA: IEEE. pp. 1-4.
[22] Zhou J, Yi J, Yang Z, Pu H, Li X, Luo J, Gao L. A survey on vehicle–drone cooperative delivery operations optimization: Models, methods, and future research directions. Swarm Evol Comput 2025; 92: 1-29.
[23] Maheriya K, Rahevar M, Mewada H, Parmar M, Patel A. Insights into aerial intelligence: assessing CNN-based algorithms for human action recognition and object detection in diverse environments. Multimed Tools Appl 2024; 1-43.
[24] Zheng X, Wang G, Xu G, Yang J, Han B, Yu J. A LLM-driven and motif-informed linearizing graph transformer for Web API recommendation. Appl Soft Comput 2025; 169: 1-12.
[25] Ma Z, An S, Xie B, Lin Z. Compositional API Recommendation for Library-Oriented Code Generation. In: 2024 IEEE/ACM 32nd International Conference on Program Comprehension (ICPC); 2024; Lisbon, Portugal: IEEE. pp. 1-12.
[26] Park S, Kim A, Lee S, Kamyod C, Kim CG. Design of REST API Client for Conversational Agent using Large Language Model with Open API System. In: 2024 IEEE/ACIS 22nd International Conference on Software Engineering Research, Management and Applications (SERA); 2024; Honolulu, HI, USA: IEEE. pp. 1-4.
[27] Hemberg E, Moskal S, O'Reilly U. Evolving code with a large language model. Genet Program Evolvable Mach 2024; 25: 1-36.
[28] Hong J, Ryu S. Type-migrating C-to-Rust translation using a large language model. Empir Softw Eng 2024; 30: 1-38.
[29] Balasundaram A, Aziz ABA, Gupta A, Shaik A, Kavitha MS. A fusion approach using GIS, green area detection, weather API and GPT for satellite image based fertile land discovery and crop suitability. Sci Rep 2024; 14: 1-16.
[30] Gerstmayr J, Manzl P, Pieber M. Multibody Models Generated from Natural Language. Multibody Syst Dyn 2024; 62: 249-271.
[31] Poth A, Rrjolli O, Arcuri A. Technology adoption performance evaluation applied to testing industrial REST APIs. Autom Softw Eng 2024; 32: 1-34.
[32] Ishimizu Y, Li J, Yamauchi T, Chen S, Cai J, Hirano T, Tei K. Towards Efficient Discrete Controller Synthesis: Semantics-Aware Stepwise Policy Design via LLM. In: 2024 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia); 2024; Danang, Vietnam: IEEE. pp. 1-4.
[33] Cao C, Wang F, Lindley L, Wang Z. Managing Linux servers with LLM-based AI agents: An empirical evaluation with GPT4. Mach Learn Appl 2024; 17: 1-13.
[34] Luo H, Wu J, Liu J, Antwi-afari MF. Large language model-based code generation for the control of construction assembly robots: A hierarchical generation approach. Dev Built Environ 2024; 19: 1-18.