FREUDENBERG
INNOVATING TOGETHER

SAP

**Whitepaper**

# Exploring Voice-Controlled Drone Technology: PoC Development for SAP Labs India

**Course of Study: Business Informatics**

**Specialization: Software Engineering**


| | |
|---|---|
| Author: | Sean Tyler Straub |
| Company: | Freudenberg & Co. KG |
| Department: | Corporate IT |
| Supervisor: | Kishor Ingale |
| Company: | SAP Labs India |
| Department: | MD's office |

# Contents

# Acronyms

| | |
|---|---|
| **ASTRID** | Autonomous System for Remote and Intelligent Drone operations |
| **HLS** | HTTP Live Streaming |
| **LLM** | Large Language Model |
| **PoC** | Proof of Concept |
| **RTMP** | Real-Time Messaging Protocol |

# 1 Introduction

## 1.1 Background

This document outlines the outcomes of the development of two Proof of Concepts (PoCs) as part of the Autonomous System for Remote and Intelligent Drone operations (ASTRID) project at SAP Labs India. ASTRID, an initiative spearheaded by the Innovations Team, aims to leverage Large Language Models (LLMs) to enable intuitive drone control via voice commands. The PoCs presented here tackle key challenges in realizing this ambitious vision.

The first PoC investigates the use of few-shot prompting to generate complex and accurate drone command sequences based on user input. By integrating PyGame for simulation, this PoC demonstrates the potential of refining prompts to generate precise and actionable instructions for drone movement and behavior.

The second PoC addresses the critical issue of real-time video streaming from drones. Initially, YouTube was utilized for streaming, but this solution introduced significant latency, often exceeding 10 seconds. To overcome this limitation, an NGINX Real-Time Messaging Protocol (RTMP) server was implemented, resulting in a substantial reduction of latency to approximately 1-2 seconds, thus providing a more responsive and immersive experience.

These PoCs offer valuable insights into the practical application of voice-controlled drones and set the foundation for future developments in this field, bringing us closer to integrating LLM-powered drones into real-world use cases.

## 1.2 Preread

Before embarking on these projects, I reviewed essential resources to ensure a strong technical foundation. These included the DJI user manual and the DJI SDK documentation, which provided crucial insights into drone operation and control. Additionally, I explored various blog posts and articles detailing attempts to optimize RTMP streaming for similar use cases, helping to inform the decisions made during the implementation phase.

# 2 Implementation

The ASTRID project focuses on enabling drone control through voice commands powered by Large Language Models (LLMs). As part of the initiative, two Proofs of Concept (PoCs) were developed, each addressing a unique challenge in the application of LLMs for drone operations.

## 2.1 Methodology: Drone Command Generation

### 2.1.1 Overview

The first PoC aimed to explore the potential of few-shot prompting to generate a sequence of precise drone commands based on natural language instructions. This methodology involved designing a set of adaptable prompts capable of handling varying levels of complexity in user inputs.

### 2.1.2 Prompt Engineering

Prompt engineering focused on defining clear parameters for movement and rotation. The drone commands followed predefined formats such as "Go Front", "Turn Left", "Go Right", and "Go Up", with the goal of translating user requests for movement into a chain of executable instructions. Key challenges included refining the prompts to ensure that the LLM generated commands that were both accurate and feasible, given the constraints of the drone SDK.

For instance, a simple prompt like "Move forward 50 cm and then turn left 90 degrees" would result in a sequence of commands like: `Go Front, Go Front, Go Front, Go Front, Go Front, Turn Left, Turn Left, Turn Left`. This approach allowed the LLM to handle both simple and more complex instructions, including geometric shapes and multi-step movements. For complex shapes like squares or circles, the prompt was adjusted to guide the LLM step-by-step, ensuring that it could break down larger tasks into smaller, actionable commands.

## 2.2 Methodology: Real-time Video Streaming

### 2.2.1 Overview

The second PoC focused on improving the latency of real-time video streaming from the drone. Prior to this work, the Innovations Team had relied on YouTube to stream drone footage, but this solution suffered from significant latency, often exceeding 10 seconds. To address this challenge,

an NGINX RTMP server was set up to directly receive the RTMP stream from the drone, offering a significant improvement in latency.

### 2.2.2 RTMP Server Configuration

The implementation of the RTMP server involved several key steps:

- Configuring an NGINX server with the RTMP module to accept and forward RTMP streams.
- Setting up the server on a local machine (MacBook) and testing its performance with tools like `ffmpeg` to measure latency.
- Fine-tuning parameters to minimize delay and ensure smooth streaming.

## 2.3 Implementation: Drone Command Generation

### 2.3.1 Few-shot Prompting in PyGame

The first PoC was implemented in a simulation environment using PyGame, where the LLM-generated commands could be executed in real-time. The simulation employed the same command set as the drone SDK, including basic movement and rotation instructions. The prompts were designed to ensure that the LLM could generate commands that were both feasible and precise, considering the drone's movement constraints.

The process involved sending user input to the LLM via a chatbot interface, where the LLM would respond with a series of commands. These commands were then executed in the simulation, with the drone moving or rotating based on the generated instructions. Prompt engineering was iteratively refined based on feedback from initial tests, ensuring that the LLM understood the task and produced relevant commands.

This implementation enabled testing of complex scenarios, such as multiple turns or navigating specific shapes, and helped refine the LLM's response to these instructions.

### 2.3.2 Prompt Engineering and Refinement

Initially, the LLM used for few-shot prompting was GPT-4o-mini, but it was quickly replaced by GPT-4o due to the former's inability to generate accurate drone commands. Additionally, system prompts were refined to ensure that the LLM could produce precise and actionable instructions for drone movement and behavior. The final iteration of the system prompt, as shown in the code snippet A.1, ensured the required accuracy.

## 2.4 Implementation: Real-Time Video Streaming

### 2.4.1 NGINX RTMP Server for Video Streaming

The second PoC aimed to improve the real-time video streaming experience by reducing latency, transitioning from YouTube to a local NGINX RTMP server. This solution was tested with the different encoding settings H.264 and H.265 but the results were inconclusive in determining which encoding performed better. However, a key configuration, `sync`, was crucial for ensuring stable streaming. After testing, it was found that a `sync` value of 5ms provided the best results, though further testing under varying conditions is recommended.

The setup process involved installing and configuring the NGINX RTMP module on the MacBook, adjusting settings to handle the stream efficiently. Results showed that the RTMP server reduced latency significantly, achieving a stream delay of approximately 1-2 seconds—substantially better than the previous YouTube-based solution.

Although this setup worked well in the test environment using a mobile hotspot, subsequent tests on SAP's Wi-Fi network were hindered by firewall issues. This could be resolved in a production environment by implementing a dedicated router or an alternative network configuration.

## 2.5 Challenges and Limitations

Several challenges were encountered during the implementation of both PoCs. For the few-shot prompting task, initial testing with GPT-4o-mini proved insufficient for generating accurate drone commands. Transitioning to GPT-4o was essential for achieving the desired results, as it offered improved comprehension and response accuracy. Additionally, the system prompts required several iterations to ensure the LLM generated precise, actionable instructions for drone movement.

For the RTMP streaming, the primary hurdle was the network setup. Although the mobile hotspot provided a temporary solution, a more stable network infrastructure is necessary for long-term deployment. Additionally, the tests to determine the best video settings were inconclusive, warranting further investigation to identify the optimal configuration.

Despite these challenges, both PoCs demonstrated the potential of using LLMs in drone control and highlighted the importance of real-time video streaming in applications such as surveillance and remote monitoring.

# 3 Conclusion

## 3.1 Findings

The ASTRID project successfully explored the potential of integrating Large Language Models (LLMs) for drone control, specifically through voice commands. The two Proofs of Concept (PoCs) demonstrated promising results in both drone command generation and real-time video streaming.

The first PoC showcased the feasibility of using few-shot prompting to generate precise drone commands based on natural language instructions. The prompt engineering process, combined with iterative refinements, enabled the LLM to produce feasible and actionable commands, even for complex movements. This work opens up possibilities for more intuitive human-drone interaction through voice control, allowing users to issue commands in a natural and flexible manner.

The second PoC addressed the challenge of real-time video streaming from the drone. By transitioning from YouTube to a locally hosted NGINX RTMP server, significant improvements in latency were achieved, reducing the delay to just 1-2 seconds. This is a crucial advancement for applications where real-time video feedback is essential, such as in surveillance and remote monitoring.

## 3.2 Next Steps

Despite the successes, several areas remain for further development and refinement. The next steps for the ASTRID project include:

- **Enhancing Command Generation**: While the current LLM-generated command sequences are functional, further refinement is needed to handle even more complex scenarios, such as unpredictable environmental conditions or emergency maneuvers. This can be achieved by introducing additional context into the prompts and expanding the command set.
- **Utilizing and Deploying Command Generation in Real-world Scenarios**: Testing and deploying the command generation system in real-world scenarios will be crucial to validate its performance and reliability. This involves conducting field tests where the system is used in various operational environments to ensure it can handle diverse and unpredictable conditions. This includes testing under various weather conditions, with different drone models, and in complex operational settings like surveillance and delivery. Feedback from these tests will be invaluable for refining the system and addressing any limitations or challenges that arise.

- **Integration with other Systems**: Future work should include integrating the ASTRID system with other drone control frameworks and software platforms, enabling seamless interoperability across different devices and use cases. This may involve supporting a wider range of drone models, extending the system's compatibility beyond just DJI drones.
- **Advanced Real-time Video Streaming**: Although the NGINX RTMP server reduced latency majorly, further testing is needed to determine the optimal encoding settings and server configurations for varied network conditions. Additionally, evaluating the impact of different hardware configurations, such as dedicated servers or edge devices, will help improve streaming performance in larger-scale applications.
- **Exploring Edge Computing for Low-Latency Processing**: To further reduce latency, exploring edge computing solutions for processing drone commands and video streams could be beneficial. By offloading some processing to local devices, it may be possible to achieve even lower latency and improve the overall responsiveness of the system.
- **Transcoding from RTMP to HLS**: Another potential enhancement is the ability to transcode RTMP streams to HTTP Live Streaming (HLS) using the NGINX server. This would enable more flexible and adaptive streaming options, as HLS is widely supported across various devices and platforms. Implementing this feature would involve configuring the NGINX server to transcode the incoming RTMP stream into HLS segments, allowing for adaptive bitrate streaming and improved compatibility with different viewing devices.

In conclusion, the ASTRID project lays a strong foundation for voice-controlled drone operations and real-time video streaming, opening the door to more intuitive and efficient drone applications. As the system continues to evolve, its potential for use in industries like surveillance, remote inspection, and emergency response holds great promise. With further development and real-world testing, ASTRID could lead to a new generation of autonomous drones that respond seamlessly to voice commands, with minimal latency and enhanced operational capabilities.

# A Python Code Snippet

Source Code A.1: Prompt Engineering Drone Command Generation

```
response = client.chat.completions.create(
    model="gpt-4o",
    temperature=0,
    messages=[
        {
            "role": "system",
            "content": (
                "You are a drone controller in a simulation. You are only allowed to respond using specific commands in the following format: "
                "Go Front, Go Back, Go Left, Go Right, Go Up, Go Down, Turn Right, Turn Left, Land, Return to Home."
                "\n\n"
                "General Instructions:\n"
                "1. Movement commands (Go Front, Go Back, Go Left, Go Right, Go Up, Go Down) move exactly 10 cm per command.\n"
                "2. Rotation commands (Turn Right, Turn Left) rotate the drone by exactly 30 degrees per command.\n"
                "3. 'Land' is used to bring the drone back to the ground (z-coordinate = 0).\n"
                "4. 'Return to Home' commands the drone to return to its starting point.\n"
                "5. Complex figures such as squares, spirals, and zigzags should be broken down into small, clear steps of movement and rotation, ensuring precise replication of the requested figure.\n"
                "6. For every figure, change the amount of Go Fronts to fit the size requested.\n"
                "7. Combine all commands in a single response separated by a comma.\n"
                "8. Distances and rotations must be rounded to the nearest multiple of 10 cm or 30 degrees.\n"
                "9. Do not put any other characters than the commands and the commas. Do not end the list of commands with a period.\n"
                "\n\n"
```

```
22                  "Square Instructions:\n"
23                  "1. For a square with side length n cm, move
                        forward n/10 times (rounded to the nearest
                        multiple of 10 cm) at each side.\n"
24                  "2. After each movement, turn right 3 times (to
                        make a 90-degree turn).\n"
25                  "3. Repeat for all 4 sides of the square.\n"
26                  "Example: 'Go Front (repeat n/10 times), Turn Right
                        , Turn Right, Turn Right, Go Front (repeat n/10
                        times), Turn Right, Turn Right, Turn Right, Go
                        Front (repeat n/10 times), Turn Right, Turn
                        Right, Turn Right, Go Front (repeat n/10 times)
                        '\n"
27                  "\n\n"
28                  "Circle Instructions:\n"
29                  "1. For a circle with radius n cm, calculate the
                        circumference using the formula C = 2 * pi * n,
                        where n is the radius.\n"
30                  "2. The drone can only move forward in increments
                        of 10 cm, so round the forward movement to the
                        nearest 10 cm to fit the circumference.\n"
31                  "3. Divide the circle into 12 equal segments of 30
                        degrees each (to complete 360 degrees). For each
                         segment, move forward by a distance
                        proportional to the radius.\n"
32                  "4. The distance moved forward per segment can be
                        calculated by dividing the circle's
                        circumference by 12 (the number of 30-degree
                        segments). Since the drone moves in 10 cm
                        increments, round the distance to the nearest 10
                         cm.\n"
33                  "   - For example, if the circumference is 628 cm,
                        the drone moves approximately 52 cm per segment
                        (rounded to the nearest 10 cm, which is 50 cm).
                        This would mean the drone moves forward 5 times
                        (10 cm each) in each 30-degree turn.\n"
34                  "5. Repeat the movement and rotation steps 12 times
                         to approximate the circle.\n"
35                  "6. Example: For a circle with radius 100 cm (
                        circumference = 628 cm), the drone would move
                        forward 5 times (10 cm per step) for each of the
                         12 segments, turning right 30 degrees after
                        each movement. The sequence of commands would
                        look like this: 'Go Front, Go Front, Go Front,
```

```
                        Go Front, Go Front, Turn Right, Go Front, Go
                        Front, Go Front, Go Front, Go Front, Turn Right,
                         ...' (repeat for 12 steps).\n"
                ),
            },
            {"role": "user", "content": chatbox_text},
        ],
    )
```