Drone Simulation with 1-DOF Gimbal and MAVLink Control

Robotics Simulation Engineer Assignment

Kankati Mokshagna Anurag

December 13, 2025

Contents

1 Introduction

This document describes the complete implementation of a drone simulation using **Gazebo**, **ArduPilot SITL**, and **MAVLink**, with a focus on attaching and controlling a **1-DOF gimbal-mounted camera**. The assignment was divided into three major parts:

- Part 1: Drone simulation setup in Gazebo using ArduPilot SITL
- Part 2: Integration of a 1-DOF gimbal with camera
- Part 3: Gimbal control using MAVLink commands

2 Approach and Architecture

2.1 Overall Approach

The overall approach followed a modular simulation design:

- Use ArduPilot SITL as the flight controller
- Use Gazebo as the physics and visualization engine
- Define the drone and gimbal using SDF models
- Control the gimbal using MAVLink commands via a Python script

Each part was validated independently before integration to ensure stability and correctness.

2.2 System Architecture

The simulation architecture consists of four main components:

- Gazebo (Physics + Rendering)
- ArduPilot SITL (Flight Control Logic)
- MAVLink Communication Layer
- Python-based MAVLink Control Interface

3 SITL ↔ Gazebo Data Flow Diagram

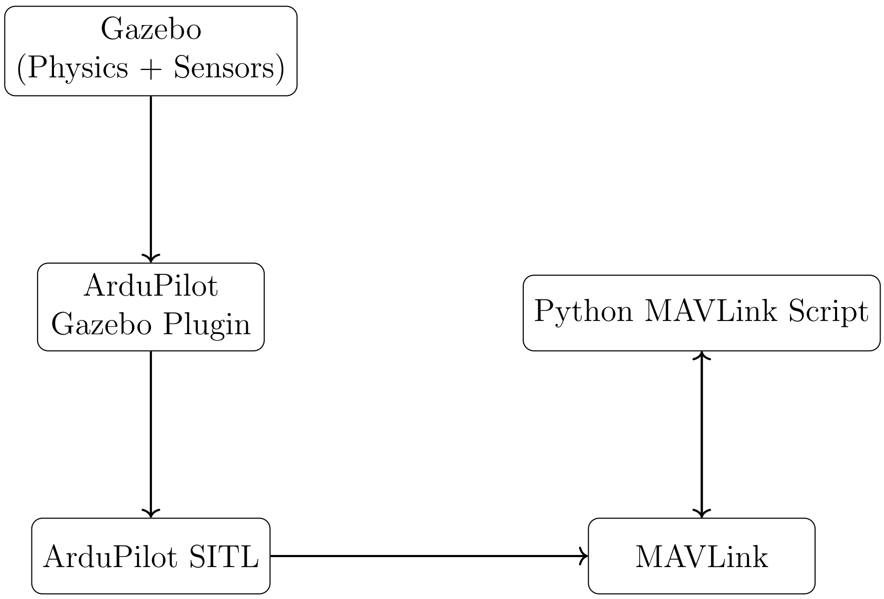

Figure 1: Data flow between Gazebo, ArduPilot SITL, and MAVLink

4 Part 1: Drone Simulation in Gazebo

4.1 Steps Performed

1. Installed Gazebo and ArduPilot SITL
2. Created a Crazyflie SDF model with:
   - Rigid body
   - Four propellers
   - IMU, GPS, and Barometer sensors
3. Integrated `libArduPilotPlugin.so` into the SDF
4. Verified motor joint connections with ArduPilot control channels

4.2 Outcome

The drone successfully spawned in Gazebo, with all sensors detected by ArduPilot SITL and motor joints responding to control inputs.

5 Part 2: 1-DOF Gimbal and Camera Integration

5.1 Steps Performed

1. Imported the `gimbal_small_1d` model
2. Attached the gimbal base to the drone body using a fixed joint

3. Created a revolute joint for 1-DOF tilt motion
4. Mounted a camera sensor on the gimbal tilt link
5. Enabled camera visualization inside Gazebo

5.2 Design Considerations

- Very low mass and inertia were used to avoid destabilizing the drone
- Gravity was disabled for the gimbal links
- Joint damping was added to prevent oscillations

5.3 Outcome

The camera feed was visible inside the Gazebo client window, and the gimbal motion was clearly observable.

6 Part 3: Gimbal Control via MAVLink

6.1 MAVLink Commands Used

- `MAV_CMD_DO_MOUNT_CONFIGURE`
- `MAV_CMD_DO_MOUNT_CONTROL`

6.2 Steps Performed

1. Established MAVLink connection using pymavlink
2. Configured the gimbal mount in MAVLink
3. Sent pitch angle commands in incremental steps
4. Verified physical gimbal movement in Gazebo

6.3 Execution Model

The control script is event-based:

- Python script runs
- Sends MAVLink commands
- Gimbal moves
- Script exits

Continuous execution is not required unless implementing real-time tracking.

7 Issues Faced and Solutions

7.1 Drone Instability

Issue: Drone moved uncontrollably due to gimbal mass. **Solution:** Reduced gimbal mass and inertia and disabled gravity.

7.2 Missing Visuals

Issue: Propellers or gimbal meshes were not visible. **Solution:** Corrected mesh paths and ensured proper link naming.

7.3 MAVLink Errors

Issue: Incorrect MAVLink command arguments. **Solution:** Fixed parameter ordering in `command_long_send`.

8 Final Result

- Stable drone simulation in Gazebo
- Functional 1-DOF gimbal with camera
- Successful MAVLink-based gimbal control
- Camera feed visible during gimbal motion

All assignment objectives were successfully completed.