AeroCommand GCS – Advanced Drone Dashboard

A Python-based Ground Control Station (GCS) designed to interface with a physics-based Quadcopter Simulator. This dashboard provides real-time telemetry monitoring, autonomous waypoint navigation, live visualization, and flight data logging using UDP communication.

Project Features

- **Real-Time Telemetry**
    - **Live monitoring of:**
        - **Battery percentage**
        - **Flight mode**
        - **Position (X, Y, Z)**
        - **Yaw**
- **Visual Instrumentation**
    - **Live Altitude Graph**
      A moving green line graph tracking the drone's height in real time.
    - **Compass**
      A visual indicator showing the drone's heading (North, South, East, West).
- **Flight Controls**
    - **TAKEOFF**
    - **LAND**
    - **RTL (Return to Launch)**
- **Autonomous Mission Planning**
    - **Upload a list of 3D waypoints in JSON format for automated flight paths.**
- **Data Logging**
    - **Automatically records all flight data to a CSV file:**
    - **flight_log_YYYY-MM-DD_HH-MM-SS.csv**
    - **Useful for post-flight analysis.**

Prerequisites

Ensure Python is installed along with the required scientific libraries.

Required Libraries

- **numpy**

- **matplotlib**
- **scipy**
- **tkinter (included with standard Python installations)**

Install Dependencies

pip install -r requirements.txt

1.. How to Run the Project

This system requires **two separate terminal windows** running at the same time: one for the drone simulator and one for your dashboard.

Step 1: Start the Simulator

Open your first terminal window and run the physics engine. This acts as the "drone".

Bash

python udp_quad.py

Wait until you see the message: [UDP] Listening for incoming GCS commands...

Step 2: Start the GCS Dashboard

Open a second terminal window and run your control station.

Bash

python main.py

The dashboard window will appear, and you should see the telemetry update immediately.

2.. Controls & Explanation

Quick Actions

- **TAKEOFF (2m):** Commands the drone to arm motors and ascend to a safe altitude of 2 meters.
- **LAND:** Commands the drone to slowly descend to the ground (Altitude 0) and disarm.
- **RTL (Return to Launch):** Commands the drone to return to home. It will descend to **Z=0**, but may deviate between **-1 to 1** on the X and Y axes due to simulated wind or drift.

Mission Planning

This section allows you to program an autonomous flight path.

- **Waypoints JSON:** Enter a list of [x, y, z] coordinates in the text box.
    - *Example:* [[0, 0, 5], [2, 2, 2], [-2, 2, 1]]

    - *Meaning:* Fly to 5m high -> Fly to x=2, y=2 -> Fly back to x=-2.

- **UPLOAD & FLY:** Sends the waypoint list to the drone and switches the mode to **GUIDED**. The drone will visit each point in order.

System

- **EMERGENCY REBOOT:** Instantly resets the simulator. The drone teleports back to [0, 0, 0], and the battery is refilled to 100%. Use this if the drone crashes.

Data Logging

Every time you run the GCS, a new CSV file is created in the project folder (e.g., flight_log_2026-02-02_12-30-00.csv).

Columns Recorded:

- Timestamp
- Flight Mode
- Battery Level
- Position (X, Y, Z)
- Yaw Orientation

🔧 **Technical Details**

- **Communication:** UDP Sockets
- **RX Port (Telemetry):** 9001
- **TX Port (Commands):** 9000
- **GUI Framework:** Tkinter (with ttk styling)