

Problem Statement

Control system for Drone Stabilization and Precision Motion Using Optical Flow & ToF Sensors

Drona Aviation – Inter IIT Tech Meet 14.0

Problem Statement Description

Modern nano-drones operate with extremely limited sensing and computational budgets. Indoors, without GPS or external tracking, stability depends entirely on how well the control system interprets on-board sensors and closes the loop in real time. Even with good IMU data, horizontal drift and altitude variations appear within seconds unless corrected by a dedicated feedback controller.

Modern flight systems solve this by fusing **optical-flow-based motion**, **precision altitude from Time-of-Flight (ToF) sensors**, and **IMU orientation data** into a stable, real-time estimation and control pipeline.

In this challenge, you will build exactly that pipeline inside the **MagisV2 flight firmware** used by the Pluto Drone platform. Your job is to convert raw sensor streams into consistent velocity and position estimates and use them to create a **production-grade closed-loop stabilization system** capable of centimeter-level accuracy.

This is not a waypoint navigation or autonomy challenge.
This is a **real flight-control engineering challenge**.

You must engineer a system that reads data continuously, filters noise, rejects poor signal conditions, generates predictable control actions, and results in a drone that can stay fixed in space — and move precisely when commanded.

Your Mission

Build a complete, on-board stabilization and precision-motion system that can:

- Take off safely
- Hold position with minimal drift
- Maintain altitude up to 5 meters
- Move in exact centimeter-level increments
- Resist and recover from external disturbances
- Demonstrate stable, repeatable, production-grade behavior

Your implementation must run entirely within MagisV2 using only the provided APIs, sensors, and microcontroller compute.

What You Must Build

1. Real-Time Velocity Estimation

You must compute actual horizontal velocities from sensor data by:

- Transforming optical-flow pixel shifts using real-time altitude
- Adjusting for yaw variations
- Handling low-texture or low-light conditions
- Maintaining high update rate (native sensor frequency)

These velocity estimates serve as the backbone of your control algorithm.

2. Closed-Loop Control System

The drone must autonomously:

- Take off
- Stabilize
- Maintain a fixed X–Y position
- Hold altitude accurately
- Recover from disturbances (pushes/pulls)
- Minimize drift
- Respond to commands without oscillation

A cascaded control architecture (Position → Velocity → Attitude) is expected, though teams may innovate beyond PID if properly justified.

3. Sensor Fusion Under Real Conditions

You must combine **Optical Flow + ToF + IMU** using deterministic, explainable logic such as:

- Complementary filtering
- Kalman-style filters
- Adaptive gain fusion
- Any well-motivated alternative

Fusion must remain stable under:

- Low/uneven textures
- Lighting variation

- Optical flow dropouts or spikes
- ToF noise at high altitude
- Slight yaw rotations
- Micro-vibrations

Failure cases must degrade gracefully — not diverge.

4. Altitude Stability up to 5 Meters

Your altitude controller must demonstrate:

- Stable hover \geq 10 cm for minimum one minute
- Low oscillations
- Noise filtering
- Consistent behavior across heights

A special evaluation includes a **high-altitude consistency test up** where stability, drift, and control authority are examined.

No surface, lighting, or texture biases are allowed.

5. Precision Motion (Centimeter Commands)

Your system must support precise movement:

- “Move forward 20 cm”
- “Move left 35 cm”
- “Go up 15 cm and hold”

Requirements:

- Controlled acceleration and deceleration
- No overshoot
- Repeatable, smooth motion
- Immediate return to stable hover

This tests your estimation accuracy and controller tuning quality.

6. Production-Grade Engineering Expectations

Your design should reflect:

- Deterministic behavior
- Predictable responses
- Low oscillations
- Modular architecture
- Clearly tuned control gains
- Reliability across multiple flights

We are not looking for a hacky drone demo.

We are looking for a system a real drone engineering team could ship.

Deliverables

1. Firmware Implementation

A complete, modular codebase running inside MagisV2 that includes:

- Sensor reading & filtering
 - Velocity and short-term position estimation
 - PID-based control modules
 - Hover logic
 - Commanded movement logic
 - Tunable flight configuration parameters
-

2. Documentation

A clear technical document covering:

- Architecture overview
 - Controller design & reasoning
 - Sensor fusion strategy
 - PID tuning approach
 - Stability tests & performance graphs
 - Limitations and learning insights
-

3. Demonstration Video

Showcase:

- Takeoff → hover sequence
- Stable hover for at least 1 minute
- Horizontal drift under 5 cm
- Altitude stability ≥ 10 cm
- Disturbance recovery (pull/push ± 5 cm in any direction)
- Precise centimeter movements in all directions

Dashboard overlays (velocity, ToF, OF) are encouraged.

Evaluation Criteria (100 Points)

1. Stability & Precision — 40 pts

- Hover stability
- Drift minimization
- Smooth corrections
- Accurate altitude hold

2. Control System Design — 25 pts

- Correct controller architecture
- Structured PID tuning
- Responsiveness without oscillation

3. Sensor Fusion Quality — 20 pts

- Clean velocity estimation
- Noise handling
- Robust behavior under difficult surfaces/lighting

4. Engineering Innovation — 10 pts

- Advanced filtering Techniques
- Visualization tools
- Predictive/adaptive strategies

5. Documentation & Presentation — 5 pts

- Clarity, structure, and completeness