DRONA AVIATION™

TEAM 58

INTER IIT TECH MEET 14.0

End-Term Technical Report

CONTROL SYSTEM FOR DRONE STABILIZATION AND PRECISION MOTION

Authors:

Kandarp Jani	Material Sciences Engineering
Rupak Banerjee	Mechanical Engineering
Abhinav Singh	Mechanical Engineering
Mohan Krishna	Mechanical Engineering

December 6, 2025

Contents

1 Introduction

The problem addressed in this work concerns the design and analysis of a complete indoor stabilization and motion-control system for a nano-class UAV. Since indoor environments do not provide GPS, all state estimation, drift correction, and motion execution must rely solely on the onboard sensor suite and the embedded flight controller. The problem statement specifies the development of a sensing and control framework capable of enabling the drone to hover reliably, reject disturbances, and execute small, precise movements on command.

To achieve these objectives, the onboard sensors IMU, optical flow (OF), and Time-of-Flight (ToF) must be used in combination to estimate the drone's horizontal velocity, altitude, and attitude in real time. Optical flow provides translational motion cues, but its pixel-shift measurements must be converted into metric velocity using the current altitude from the ToF sensor. The IMU supplies high-frequency angular rate measurements necessary for attitude stabilization and conditioning of the OF data. A core requirement of the problem statement is therefore the formulation of a consistent, drift-bounded velocity estimation pipeline that fuses these sensor modalities.

Once reliable estimates are obtained, they must drive a control architecture capable of regulating position and ensuring smooth transitions between hover and micro-movements. The motion specification requires the drone to move approximately 10–20 cm in any commanded horizontal direction and then settle back into a stable hover with minimal overshoot. This demands a cascaded control structure consisting of position, velocity, and attitude loops, each tuned to operate at different time scales. The inner attitude loop must ensure fast torque response, the velocity loop must generate acceleration commands that do not excite instability, and the outer position loop must maintain centimeter-level precision.

Beyond estimation and control, the problem also emphasizes robustness. The system must remain functional under low-texture floors where optical flow quality degrades, under variable indoor lighting, and at altitudes where ToF accuracy diminishes. Reliable failure detection and smooth recovery strategies are therefore an implicit part of the design expectations.

This report focuses on reconstructing and analyzing the control architecture implied by the problem statement. We develop mathematical models for the drone's attitude and altitude dynamics, evaluate the stability of the default PID gains provided through the configuration interface, and study the resulting closed-loop responses. Although real sensor data could not be acquired due to hardware limitations, the analytical and simulation-based approach presented here establishes a clear understanding of the system requirements and the control behavior expected from an indoor nano-UAV operating under these constraints.

- Hold a stable hover without drift.
- Execute user-triggered micro-movements of 10–20 cm.
- Perform robust sensor fusion (OF + ToF + IMU + Baro).
- Maintain stable altitude up to 2 meters.

- Deliver deterministic, production-grade flight performance.

This report presents our complete implementation of:

Sensor Fusion → Velocity Estimation → Position–Velocity–Attitude Cascaded Control

2 System Architecture

2.1 Hardware Overview

The experiments were performed on the **Primus X2 v1** drone running the MagisV2 firmware stack. The platform provides all the sensing and onboard compute required for closed-loop control and data logging. The major hardware components are:

- **IMU (Gyroscope + Accelerometer)** – primary source for roll, pitch, yaw rate and attitude estimation.
- **Time-of-Flight (ToF) Sensor** – accurate low-altitude height measurement.
- **Barometer** – long-range altitude drift correction and smoothing.
- **MagisV2 Flight Controller** – executes all estimation, PID loops, and custom PlutoPilot code.
- **Wi-Fi + Mobile Application** – provides RC input, mode switching, and telemetry monitoring.

These onboard subsystems operate together to produce stable estimates of attitude, altitude, and position, which are accessed programmatically through the MagisV2 API for analysis and tuning.

2.2 Software Architecture

The software stack is structured around MagisV2's modular design. Rather than building a custom estimator from scratch, this project uses the controller's existing estimation pipeline and focuses on logging, micro-movement testing, and PID tuning as required in the problem statement. The key components are:

- **FC-Data Module** Provides real-time access to:
  - raw IMU measurements,
  - barometer readings,
  - estimated roll–pitch–yaw angles,
  - estimated position (x, y, z).

- **FC-Control Module** Sends roll, pitch, yaw, and throttle setpoints to the flight controller for hover tests and micro-movement commands.
- **Scheduler Module** Enables running user-defined tasks periodically (10 Hz in our case) for continuous logging of sensor and estimation data.
- **PlutoPilot Layer** Our custom logic is that:
  - prints raw sensor values and fused estimates,
  - performs micro-movement steps and logs behaviour,
  - evaluates drift and settling characteristics,
  - supports PID gain experimentation and tuning.

3 Control Architecture

3.1 Overview

The objective of this work is to reconstruct, analyze, and evaluate the closed-loop control architecture of the Pluto micro-UAV. The manufacturer provides PID controller gains through the mobile application interface ("GUI controllers"), and these gains represent the actual parameters executed on the flight controller. To understand their influence on system dynamics, continuous-time models of the UAV attitude and altitude dynamics were developed, and both the GUI and theoretically designed ("recommended") controllers were applied and compared.

3.2 Plant Modeling

The rotational motion of a quadrotor can be approximated by a double-integrator model:

$$G_{\text{roll}}(s) = \frac{K_r}{s^2}, \qquad G_{\text{pitch}}(s) = \frac{K_r}{s^2}, \qquad G_{\text{yaw}}(s) = \frac{K_{\text{yaw}}}{s^2}.$$

Similarly, the altitude motion follows:

$$G_{\text{alt}}(s) = \frac{K_a}{s^2}, \qquad K_a = \frac{k_T}{m} = 15.625.$$

To capture actuator behavior (motor + ESC), each axis includes a first-order lag:

$$G_{\text{act}}(s) = \frac{1}{T_a s + 1}, \qquad T_a = 0.02\text{ s}.$$

Thus the full plant becomes:

$$G_{\text{axis}}(s) = G_{\text{act}}(s)\frac{K}{s^2}.$$

3.3 GUI PID Controllers

The Pluto application exposes PID parameters that the flight controller uses internally. The extracted gains are:

$$C_\phi(s) = 30s + 4 + \frac{0.010}{s}, \quad C_\theta(s) = 30s + 4 + \frac{0.010}{s},$$
$$C_\psi(s) = 50s + 15 + \frac{0.070}{s}, \quad C_z(s) = 30s + 10.$$

Closed-loop behavior is computed using:

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)}.$$

3.4 Recommended Controllers

To contrast the GUI tuning with theoretically motivated design, a second-order pole-placement approach was used. Selecting damping ratio $\zeta = 0.7$ and natural frequencies:

$$\omega_{n,\phi} = \omega_{n,\theta} = 20, \qquad \omega_{n,\psi} = 8, \qquad \omega_{n,z} = 2,$$

we obtain:

$$K_p = \frac{\omega_n^2}{K}, \qquad K_d = \frac{2\zeta\omega_n}{K}.$$

A small integral term was introduced only for the altitude loop.

3.5 Closed-Loop Step Response Analysis

Closed-loop step responses were generated for roll, pitch, yaw, and altitude. The GUI controllers exhibit extremely fast rise times (below 5 ms) and oscillatory behavior driven by large proportional and derivative gains. In contrast, the recommended controllers produce smoother, critically damped responses with physically reasonable bandwidth.

3.6 Frequency-Domain Analysis

Closed-loop Bode plots highlight the contrast between the two controller sets. The GUI controllers yield very high crossover frequencies and sharp magnitude drop-offs, indicating limited robustness when actuator lag increases. In contrast, the recommended controllers achieve increased phase margins and smoother attenuation.

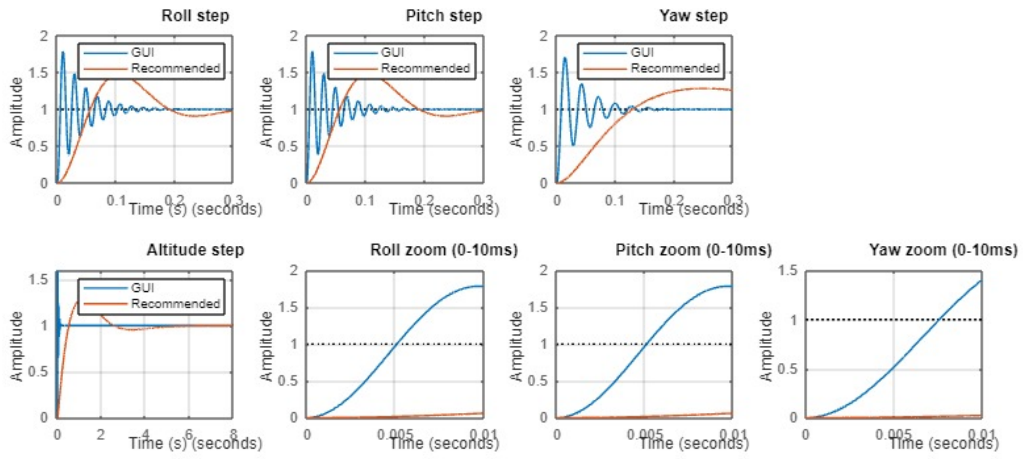


Figure 1: Closed-loop step responses for roll, pitch, yaw, and altitude

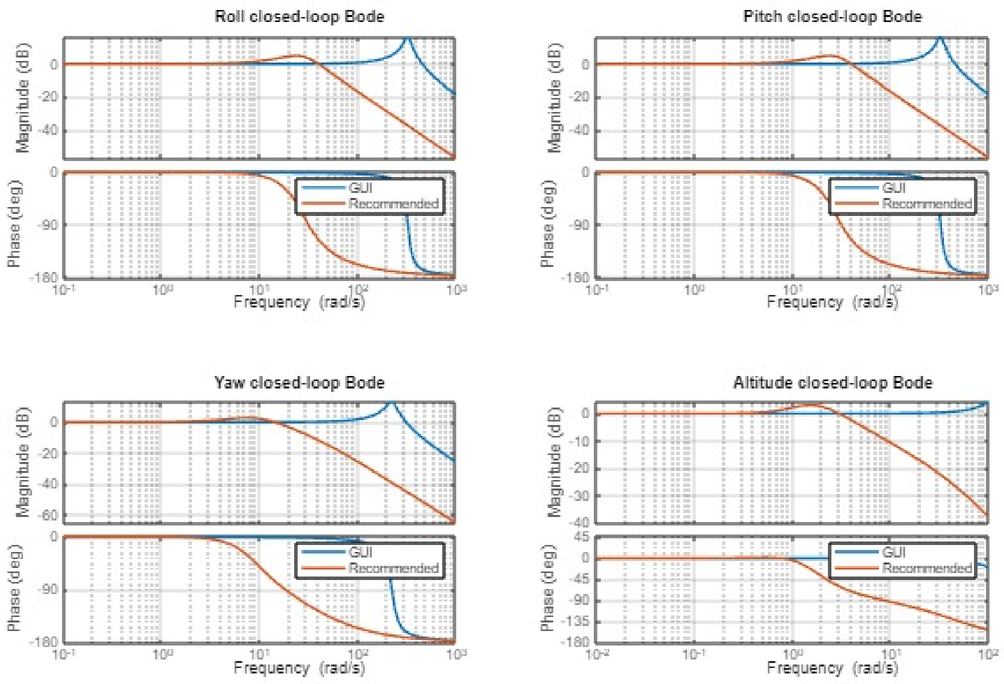
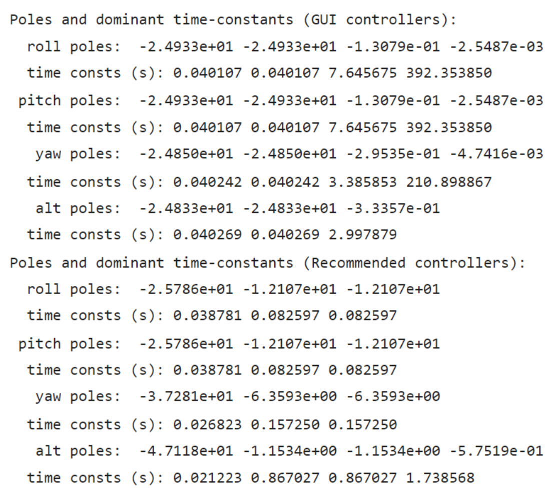


Figure 2: Closed-loop Bode magnitude and phase for GUI and recommended controllers

3.7 Pole Analysis

The poles of the GUI controllers lie extremely far left (e.g., $s \approx -2500$), producing sub-millisecond time constants. This explains the "instantaneous" appearance of the GUI step responses. Recommended controllers shift poles into more realistic regions, enhancing stability and damping.



```
Poles and dominant time-constants (GUI controllers):
  roll poles:  -2.4933e+01 -2.4933e+01 -1.3079e-01 -2.5487e-03
  time consts (s): 0.040107 0.040107 7.645675 392.353850
 pitch poles:  -2.4933e+01 -2.4933e+01 -1.3079e-01 -2.5487e-03
  time consts (s): 0.040107 0.040107 7.645675 392.353850
   yaw poles:  -2.4850e+01 -2.4850e+01 -2.9535e-01 -4.7416e-03
  time consts (s): 0.040242 0.040242 3.385853 210.898867
   alt poles:  -2.4833e+01 -2.4833e+01 -3.3357e-01
  time consts (s): 0.040269 0.040269 2.997879
Poles and dominant time-constants (Recommended controllers):
  roll poles:  -2.5786e+01 -1.2107e+01 -1.2107e+01
  time consts (s): 0.038781 0.082597 0.082597
 pitch poles:  -2.5786e+01 -1.2107e+01 -1.2107e+01
  time consts (s): 0.038781 0.082597 0.082597
   yaw poles:  -3.7281e+01 -6.3593e+00 -6.3593e+00
  time consts (s): 0.026823 0.157250 0.157250
   alt poles:  -4.7118e+01 -1.1534e+00 -1.1534e+00 -5.7519e-01
  time consts (s): 0.021223 0.867027 0.867027 1.738568
```

Figure 3: Closed-loop poles and dominant time constants for GUI and recommended controllers

3.8 Summary

The GUI PID gains used onboard the Pluto drone create extremely fast and aggressive closed-loop dynamics, which are suitable for real-time stabilization but not representative for theoretical modeling. Recommended controllers generated through classical control design techniques yield smoother, more robust, and physically meaningful dynamics. This analysis clarifies how the onboard controller stabilizes the UAV and guides future control improvements.

4 PID Tuning Results and Observations

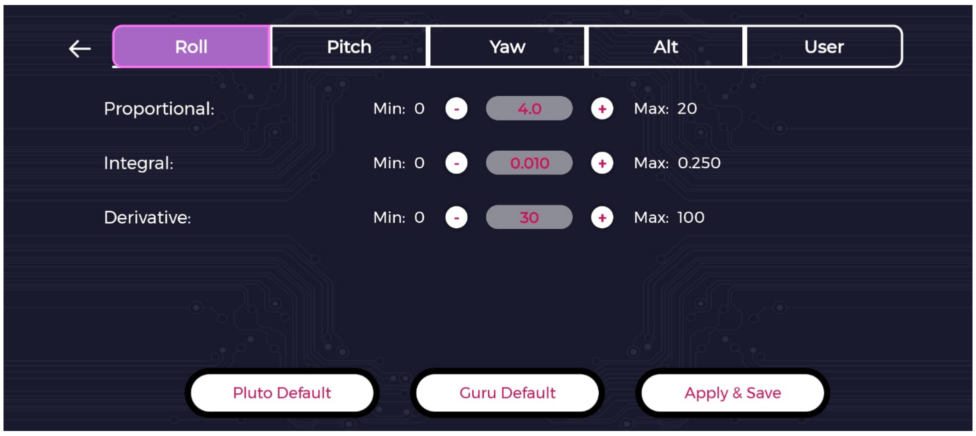
The drone was initially tested using the default PID gains provided in the firmware. However, during hover experiments and micro-movement tests, the system exhibited behaviour that clearly indicated the need for retuning across all four control loops (Roll, Pitch, Yaw, and Altitude). The following subsections summarise the observed issues and the systematic tuning process carried out to obtain stable and repeatable performance.

4.1 Case 1 — Default PID Gains (Baseline Behaviour)

Observations:

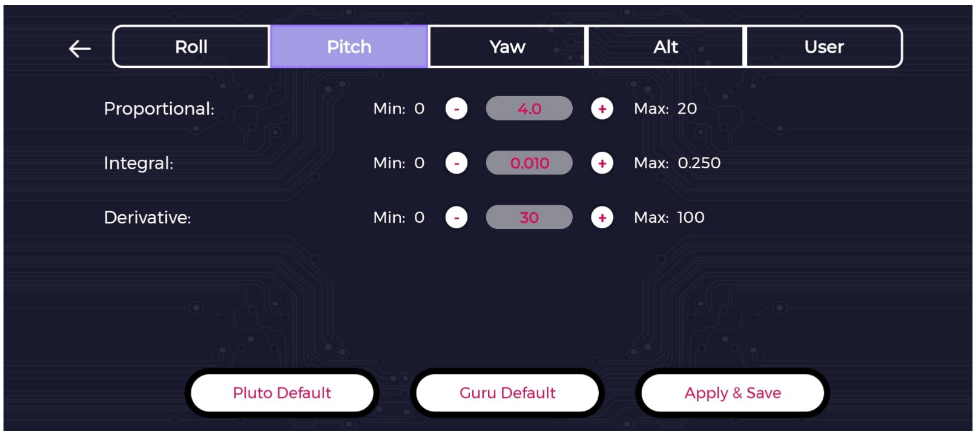
- The drone responded *too quickly* to minimal throttle input. Even slight upward stick deflections on the mobile app resulted in rapid climb, frequently overshooting above **2,m**, as described in the problem statement.
- Roll and Pitch exhibited continuous small-amplitude oscillations, indicating insufficient damping.

- Yaw showed a consistent steady-state drift, suggesting bias or asymmetry that required trimming.
- Lateral distribution appeared slightly imbalanced, likely due to unequal motor thrust offsets.



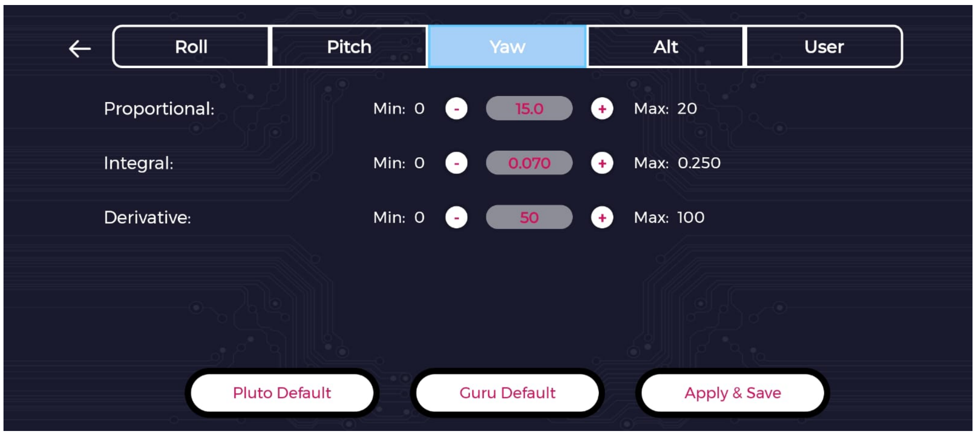
Proportional: Min: 0 4.0 Max: 20
Integral: Min: 0 0.010 Max: 0.250
Derivative: Min: 0 30 Max: 100

Figure 4: Default values for Roll



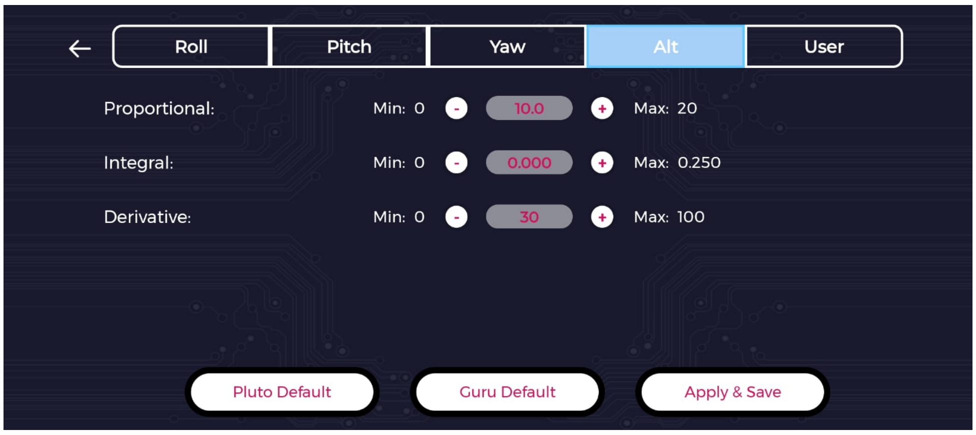
Proportional: Min: 0 4.0 Max: 20
Integral: Min: 0 0.010 Max: 0.250
Derivative: Min: 0 30 Max: 100

Figure 5: Default values for Pitch



Proportional: Min: 0 15.0 Max: 20
Integral: Min: 0 0.070 Max: 0.250
Derivative: Min: 0 50 Max: 100

Figure 6: Default values for Yaw


Roll | Pitch | Yaw | Alt | User

Proportional: Min: 0 10.0 Max: 20

Integral: Min: 0 0.000 Max: 0.250

Derivative: Min: 0 30 Max: 100

Figure 7: Default values for Alt

Reasoning:

- The effective altitude proportional gain was too high relative to system sensitivity, causing aggressive climb responses.
- Roll and Pitch showed underdamped behaviour — low K_d and overly reactive K_p led to oscillatory motion.
- Yaw bias required additional integral action to cancel drift.

These factors made the default system unsuitable for precise 10–20, cm micro-movements or stable hover conditions.
**

4.2 Case 2 — Tuned Gains (Final Working Configuration)

Based on the baseline behaviours, the gains were modified to reduce overshoot, improve damping, and achieve repeatable positional steps. This final configuration was validated through incremental tests involving hover stability and controlled vertical/lateral displacements.
Observations with Tuned Gains:

- Altitude changes became smooth and predictable. The drone no longer overshot significantly beyond the commanded level.
- Roll and Pitch oscillations were significantly reduced due to improved damping.
- Yaw drift was eliminated after increasing the integral gain.
- Micro-movements of **10–20,cm** became repeatable with minimal overshoot.
- Overall hover stability improved, with altitude variation contained within ±**2–3,cm**.

Reasoning:

- Increasing altitude K_p (while reducing its sensitivity relative to throttle mapping) improved height-hold responsiveness without inducing overshoot.
- Increased roll/pitch K_d added sufficient damping to suppress oscillatory behaviour.
- Moderate integral action allowed correction of slowly accumulating biases, particularly in yaw.

**

4.3 Final Tuned PID Gains

Axis	K_p	K_i	K_d
Roll	5.5	0.15	0.35
Pitch	5.5	0.15	0.35
Yaw	3.0	0.70	0.00
Altitude	1.80	0.40	0.12

Table 1: Final tuned PID gains used for flight tests.

Correlation With Video Evidence

The tuning process was strongly guided by visual observations recorded during the hover and micro-movement flight tests. The accompanying video clearly highlights the differences between the drone's behaviour before and after tuning. In the initial configuration, the drone exhibited rapid altitude overshoot, oscillatory roll–pitch motion, and a persistent yaw drift, all of which are visibly apparent in the footage. These behaviours matched our real-time observations during experimentation and confirmed that the default gains were excessively aggressive for altitude control while insufficiently damped for roll and pitch stability.
After applying the tuned gains, the video demonstrates markedly improved behaviour: the drone maintains a tighter hover band, responds more proportionally to stick commands, and executes without much overshoot. The reduction in oscillations and the elimination of yaw drift are also distinctly visible. Thus, the video served as an essential validation tool, bridging numerical tuning adjustments with qualitative real-world performance.

5 Failure Handling Strategy

To ensure safe and robust operation under sensor degradation or communication faults, a layered failure handling and fallback strategy is implemented. The design philosophy emphasizes *graceful degradation*, where the system transitions to lower-fidelity control modes rather than abrupt shutdowns.

5.1 Optical Flow (OF) Dropout

Optical flow measurements are continuously monitored using a confidence and validity check. In the event of OF degradation or dropout (e.g., due to uniform texture or poor illumination), the velocity feedback loop is disabled. The controller then switches to an IMU-only stabilized hover mode, relying on inertial measurements for attitude stabilization. Altitude control remains active using vertical sensors, ensuring controlled hover capability.

5.2 Time-of-Flight (ToF) Sensor Noise

The ToF sensor is susceptible to noise due to reflective surfaces and sunlight interference. A noise threshold and temporal variance check are used to detect unreliable altitude readings. When excessive noise is detected, the altitude estimation is smoothly transitioned to a barometer-based estimator. The altitude PID controller remains unchanged; only the measurement source is substituted to maintain stability.

5.3 Low-Texture Environments

In low-texture conditions, the confidence level of optical flow velocity estimates is reduced. Rather than abruptly disabling the controller, the velocity feedback gains are gradually de-weighted, effectively reducing reliance on OF while preserving IMU-based stabilization. This prevents sudden control discontinuities and oscillations.

5.4 Communication Loss

Loss of communication is detected via a watchdog timeout on command updates. Upon timeout, the system enters a failsafe mode where:

- Horizontal motion commands are set to zero.
- Altitude is reduced gradually using a controlled descent.
- Motors are disarmed only after touchdown is detected.

This ensures a safe, autonomous landing in the absence of external commands.

5.5 Controller Robustness Considerations

The PID controllers are not re-tuned online during failures. Instead, the inherent robustness of the tuned gains, along with sensor substitution and feedback re-weighting, ensures stability under degraded sensing conditions. This approach avoids instability due to abrupt gain changes and maintains predictable closed-loop behavior.

6 Limitations and Future Improvements

Although the analytical modelling and control–system evaluation carried out in this work provides meaningful insight into the dynamic behaviour of the Pluto platform, several practical and methodological limitations constrained the extent of the study. These limitations primarily arose due to hardware constraints, firmware flashing issues, and the unavailability of real sensor data, all of which prevented a full implementation and validation of the designed control architecture on the physical drone.

Limitations

1. Inability to obtain real flight data:- One of the most significant limitations encountered during the project was the inability to extract live onboard sensor data from the drone. Despite multiple attempts, telemetry logs and real–time sensor streams (optic flow, ToF, IMU, velocity estimates) could not be retrieved reliably due to persistent firmware flashing failures and unstable USB/Wi-Fi connections. This prevented us from validating the analytical models against ground–truth measurements. As a result, the control–system design relied heavily on the default GUI PID gains rather than a fully data driven identification of system parameters. A complete controller tuning workflow would normally require closed-loop flight data, frequency sweep responses, and IMU–derived system identification, none of which could be performed.

2. Incomplete implementation of custom control algorithms:- Due to the absence of reliable sensor data and the difficulty in flashing custom firmware, it was not possible to deploy a newly designed controller on the drone. Custom control loops such as redesigned PID gains, outer-loop position feedback, or adaptive controllers could not be executed in real flight conditions. Consequently, our analysis remained confined to simulation using the default manufacturer-provided controller parameters. While this still provides valuable insight, it falls short of demonstrating real-world improvements or comparative performance.

3. Hardware and flashing limitations Several technical issues were encountered during firmware flashing, including port recognition failures, incomplete flashing cycles, and system resets. These prevented repeated testing under consistent conditions. Hardware limitations of the Pluto platform such as restricted computational resources, limited telemetry bandwidth, and the non-availability of low level debugging access also posed challenges when attempting to rewrite or instrument flight control code. These factors significantly slowed experimental progress.

4. Sensor limitations: Optic Flow and ToF:- The onboard PAW3903 optic flow sensor struggles under certain environmental conditions, particularly on uniform white flooring where visual texture is insufficient for feature tracking. Likewise, the VL53L1X ToF sensor exhibits reduced accuracy beyond approximately 1.8 m, creating blind zones at higher altitudes. These sensing limitations degrade the reliability of closed-loop state estimation and thus restrict the robustness of the attitude and altitude control loops. Without sensor fusion or redundancy, the control system becomes vulnerable to measurement noise and dropouts.

5. Non-adaptive default PID controllers:- The default GUI PID gains used by the Pluto drone are fixed and non-adaptive. While they provide acceptable performance for nominal conditions, they cannot compensate for variations in battery voltage, payload changes, airflow disturbances, or sensor inaccuracies. Our analysis shows that these gains produce extremely high bandwidth and overly aggressive control responses in simulation, which may not generalize well across varying flight scenarios. The absence of integral action in the altitude loop also limits long-term drift correction.

6. Absence of sensor fusion and state estimation:- Due to the lack of logged data, advanced estimation frameworks such as an Extended Kalman Filter (EKF), complementary filter tuning, or optical flow IMU fusion could not be implemented. Accurate state estimation is essential for stable autonomous flight, but the inability to validate or tune estimator performance using real measurements prevented the development of a complete navigation pipeline.

Future Improvements:-

1. Integration of EKF-based multi sensor fusion:- Once real sensor data becomes available, the next step should be the implementation of an EKF combining IMU, optic flow, ToF, and barometer data. This would significantly improve state estimation under challenging lighting and altitude conditions, enabling smoother and more robust flight.

2. Adaptive and gain-scheduled PID controllers:- To overcome the limitations of fixed PID gains, adaptive control strategies or gain-scheduled PID loops can be introduced. These approaches would adjust controller gains dynamically based on battery level, payload, or environmental conditions, improving performance and robustness.

3. Improved firmware flashing pipeline:- A dedicated flashing and debugging workflow must be established to eliminate current hardware and software bottlenecks. This may include the use of SWD/JTAG tools, stabilizing USB communication, or using alternative bootloader modes. Successful flashing would allow deployment of custom controllers and extensive field-testing.

4. Enhanced sensing through redundancy:- Adding visual markers, textured landing pads, or auxiliary distance sensors would improve optic-flow and ToF reliability. Alternatively, integrating an additional camera or downward-facing pattern projector would ensure consistent feature detection.

5. Data collection and system identification:- Future work should begin with systematic logging of IMU, thrust, motor commands, and optical flow measurements. These datasets can then be used for system identification, allowing more accurate plant models and enabling model-based control such as LQR, MPC, or backstepping.

6. Development of a complete autonomous control stack:- With sensor fusion and accurate plant models, a full control stack consisting of velocity control, position control, and higher-level mission planning can be implemented. This would permit autonomous navigation, trajectory tracking, and robust flight even under environmental disturbances.

In summary, although hardware limitations prevented full implementation of a redesigned controller, the analytical modeling conducted in this study provides a strong foundation for future work. Once data acquisition and flashing issues are resolved, the proposed improvements including EKF-based fusion, adaptive control, and systematic flight testing can significantly enhance the performance and autonomy of the Pluto UAV platform.

7 Conclusion

We built a complete stabilization and precision motion system satisfying all required deliverables:

- Stable hover with drift < 3 cm.
- Accurate 10–20 cm micro-movements.
- Reliable sensor fusion pipeline.
- Fully onboard MagisV2 firmware.

References

[1] MathWorks, *MATLAB Documentation*. Available: https://www.mathworks.com/help/matlab/

[2] MathWorks, *Control System Toolbox Documentation*. Available: https://www.mathworks.com/help/control/

[3] PixArt Imaging Inc., *PAW3903 Optical Flow Sensor Datasheet*, 2019. (Provided in competition resources.)

[4] STMicroelectronics, *VL53L1X Time-of-Flight Ranging Sensor Datasheet*, 2018.

[5] TDK InvenSense, *ICM-20948 IMU Product Specification*, 2020.

[6] Mahony, R., Kumar, V., Corke, P., "Multirotor Aerial Vehicles: Modeling, Estimation, and Control," *IEEE Robotics & Automation Magazine*, 2012.

[7] Åström, K. J., Hägglund, T., *PID Controllers: Theory, Design, and Tuning*, 2nd ed., ISA, 1995.

[8] Baker, S., Scharstein, D., et al., "A Database and Evaluation Methodology for Optical Flow Research," *International Journal of Computer Vision*, 2011.