Unmanned Aerial Vehicle Autopilot Firmware Design and Implementation

1st Luca Ciancanelli
Electrical Engineering Department
Colorado School of Mines
Golden, United States
leciancanelli@mines.edu

***Abstract*—This paper presents the design and partial implementation of a custom autopilot firmware system for a fixed-wing unmanned aerial vehicle (UAV). The objective of the project was to develop a modular firmware architecture capable of autonomous waypoint navigation, with core components including real-time data acquisition, digital signal filtering, state estimation, and control system design. Sensor drivers were developed for five onboard sensors and integrated into a real-time data logging and processing pipeline. A complementary and IIR filtering framework was implemented to estimate UAV orientation and altitude, enabling the construction of virtual flight instruments for feedback and characterization. A nonlinear three-degree-of-freedom (3-DoF) longitudinal model of the aircraft was created in MATLAB/Simulink, linearized around a trimmed flight condition, and used for the synthesis of an altitude controller based on linear quadratic regulation. While full autonomous navigation was not achieved during the project period, foundational components of the autopilot were validated through simulation and flight data analysis, establishing a basis for future development of complete autonomous flight capabilities.**

I. Introduction

The development of autonomous unmanned aerial vehicles (UAVs) has become increasingly prominent in the last few decades as the number of use cases for UAVs has grown. With the growing number of use cases for UAVs, development of specialized hardware and software has become more common to address the specific needs of UAV applications. This paper documents the process of developing firmware for a UAV autopilot tailored to fixed-wing UAVs over the course of the Spring 2025 academic semester. The initial research objective was to develop an autopilot that could navigate a small fixed-wing UAV through several waypoints and conduct a live demonstration of the autopilot. The development of this autopilot would include developing sensor drivers, digital signal processing algorithms for state estimation, flight control logic, and flight data recording.

While the full scope of the autopilot was not achieved within the duration of the semester due to unanticipated challenges, foundational elements for an autopilot were successfully developed. Future development aims to take the work completed during this research period and complete the development of an autopilot that can achieve the initial goals set at the start of this project.

II. Hardware Overview

A. Fixed-Wing UAV

The fixed-wing UAV platform used for gathering flight data and testing autopilot functions was a custom developed UAV. The UAV was a single engine, top mounted wing, conventional tail airframe that was specifically designed for UAV autopilot development (Fig. 1). The design of the airframe aimed to provide stable flight characteristics that would be relatively simple to design control algorithms around.

Fig. 1. Fixed-Wing UAV Platform Used for Development (Final Configuration)

Using this UAV for autopilot development ended up causing delays due to the experimental nature of the UAV and need for modifications to address design issues. For future development, it would be more ideal to use an existing UAV platform for the development of an autopilot from scratch to reduce or eliminate the need to work on aspects of the UAV that were unrelated to the autopilot.

B. Flight Controller

The flight controller used for the development of the autopilot was the LEC Innovations Limited Flight Control Manager V1. This flight controller utilizes an STM32F405 microcontroller programmed using embedded C for communicating with sensors, running filtering and control algorithms, and outputting control signals to the actuators and motor (Fig. 2). This flight controller was designed in a similar style to a development circuit board for ease of firmware development and troubleshooting.

Over the course of developing the autopilot, several design issues were discovered with the flight controller. However, the flight controller was able to meet all minimum requirements for running the autopilot firmware and thus was a reasonable choice for a flight controller platform. Future iterations of the flight controller will address the issues discovered during development and will serve as a better platform for future work on the autopilot.

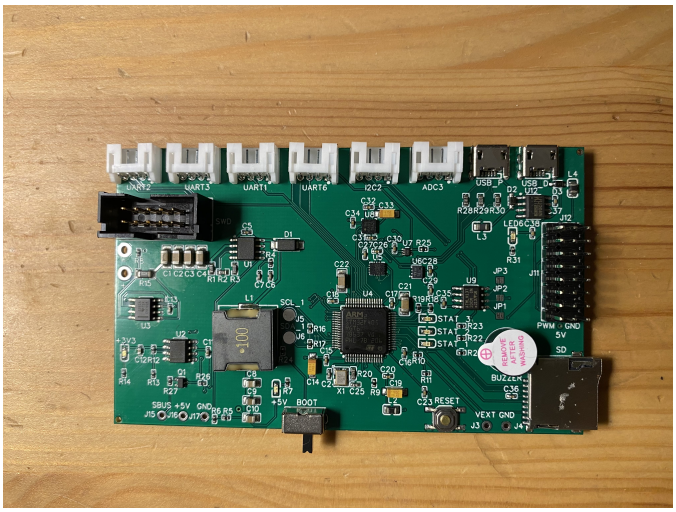

Fig. 2. LEC Innovations Limited Flight Control Manager

III. Development Overview

The development of the UAV autopilot system was structured as a sequence of seven primary tasks to be completed over the course of the semester. The first task involved the implementation of firmware drivers to enable communication between the flight control microcontroller and onboard/external sensors, including the accelerometer, gyroscope, magnetometer, barometer, and GPS receiver. The second task focused on the design of signal processing algorithms to transform raw sensor measurements into filtered signals suitable for use in feedback control.

The third task consisted of conducting flight tests to acquire real-world sensor data for validating the filtering algorithms and enabling preliminary modeling for control systems. In the fourth task, a state-space model of the UAV's longitudinal dynamics was formulated to support control system design. Subsequent tasks included the implementation of low-level controllers for attitude, airspeed, and altitude regulation; the development of autonomous flight modes for level flight and waypoint navigation; and the final demonstration of autonomous waypoint traversal in flight.

At the conclusion of the project period, the first three tasks and a portion of the fourth were successfully completed. These achievements established a functional sensor-processing pipeline and laid the groundwork for future control and navigation development.

IV. Sensor Driver Development

Each sensor driver was developed to manage two primary functions: device configuration and data acquisition. Configuration tasks included setting communication interfaces and initializing the respective sensor's control registers. Data aquisition tasks included defining interrupt and timer based scripts for retrieving sensor data as it was made available.

A. Sensor Configuration

There was a total of five independent sensor ICs that drivers were developed for: the inertial measurement unit (IMU), the magnetometer, the barometer, the thermometer, and the GPS receiver. Each sensor except for the GPS receiver had a set of configuration registers that controlled important aspects of sensor data parameters such as the measurement range, on-board filtering, output data rate (ODR), etc. The device configuration functions in the drivers would configure the communication lines (I2C, UART, or SPI) used to communicate with each IC and set values of each configuration register according to desired parameters for each sensor.

B. Data Acquisition and Interrupt Handling

Sensor data was acquired using either hardware interrupts or a timer-based polling approach. For the IMU, magnetometer, barometer, and GPS modules, dedicated interrupt lines were configured to signal data readiness. The thermometer, lacking an interrupt interface, relied on periodic polling via a microcontroller timer. Within the associated interrupt service routines (ISRs), sensor registers were read, and newly acquired data was stored in dedicated memory locations on the microcontroller.

C. Flight Data Recording Framework

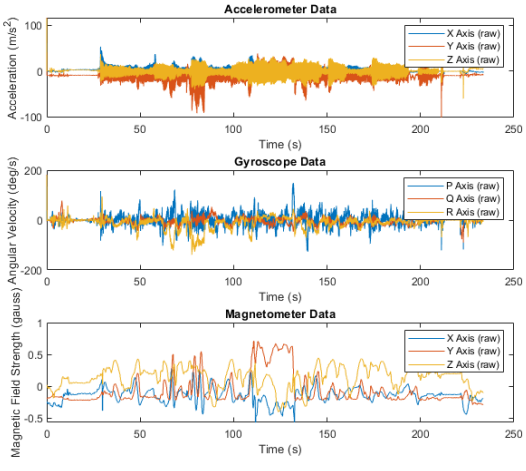

Fig. 3. Raw Accelerometer, Gyroscope, and Magnetometer Data (Flight 4)

To facilitate offline filter development and diagnostic analysis, a custom flight data recording driver was implemented. This module periodically stored sensor data to a microSD card

using a double-buffering approach. While one buffer was actively filled with incoming data, the other was asynchronously written to non-volatile memory via SPI communication. This design allowed for uninterrupted data logging at high acquisition rates while preserving data integrity and minimizing write delays.

Binary flight data files generated during test flights were parsed using a MATLAB script, enabling structured import and visualization of raw sensor outputs. These datasets served as the foundation for the development and validation of subsequent filtering algorithms (see Fig. 3).

V. Sensor Filtering and Estimation Architecture

To facilitate robust state estimation for state space control algorithms and flight data analysis, a modular filtering architecture was developed to process raw sensor data from the onboard IMU, magnetometer, barometer, thermometer, and GPS. This architecture was first conceptualized through a block diagram outlining the flow of raw sensor data through filtering elements to instruments used for autonomous navigation (See Fig. 4). Filter development was primarily done using MATLAB for rapid prototyping, visualization, and parameter tuning. Filters were tested on flight data to validate proper state estimation.

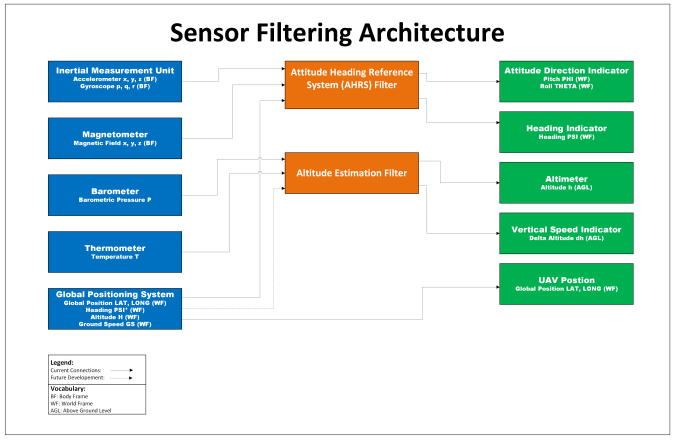

Fig. 4. Sensor Filtering and Estimation Architecture

A. Attitude and Heading Reference System

The attitude and heading reference system (AHRS) provides real-time estimates of the UAV's orientation in the Earth-fixed frame, represented by pitch, roll, and yaw angles. While the AHRS was initially designed to use IMU, magnetometer, and GPS data, current hardware limitations required exclusion of magnetometer data due to sensor inaccuracy. Thus, the present implementation fuses only IMU and GPS inputs (See Fig. 5).

Analysis of test flight data revealed significant high-frequency noise in IMU measurements, primarily from motor vibration. To suppress this noise while preserving low frequency data, a first-order infinite impulse response (IIR) low-pass filter was implemented for each axis of the accelerometer and gyroscope (See Eq. 1) [1]. The filter coefficient α was empirically selected ($\alpha > 0.8$) by comparing filtered and unfiltered signals to attenuate undesired noise while maintaining responsiveness (See Fig. 6-7).

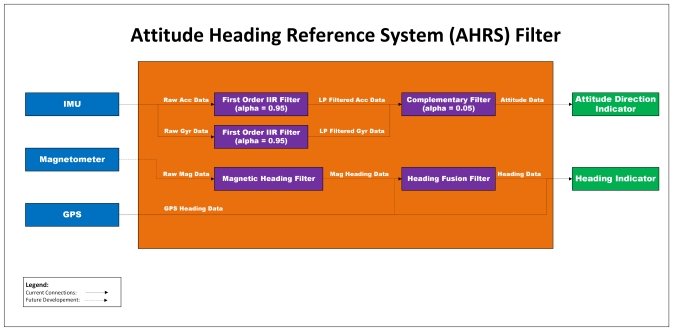

Fig. 5. Attitude and Heading Reference System Architecture

$$y[n] = (1 - \alpha) \cdot x[n] + \alpha \cdot y[n-1] \tag{1}$$

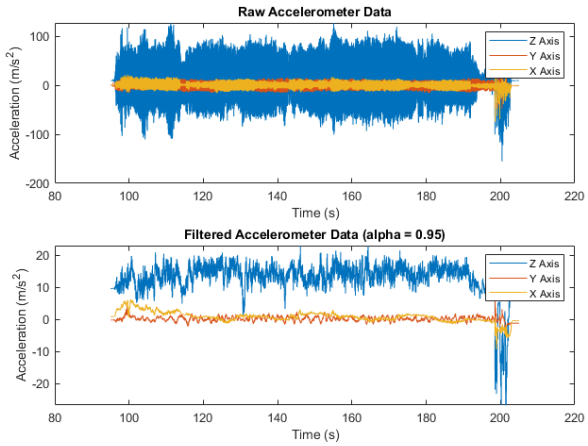

Fig. 6. Accelerometer Prefilter Plot (Flight 3)

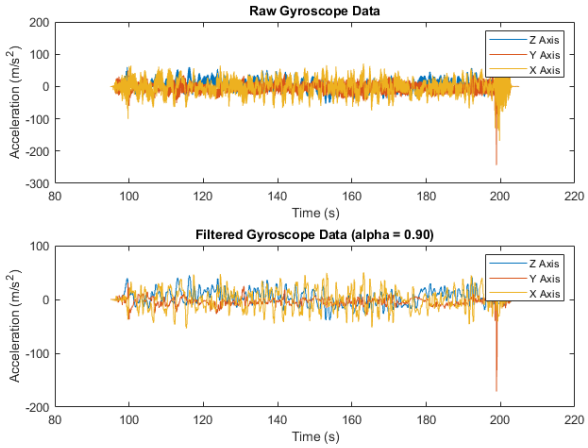

Fig. 7. Gyroscope Prefilter Plot (Flight 3)

Filtered IMU data was then passed into a complementary filter to estimate pitch and roll angles. This approach combines gyroscope-derived angle integration and accelerometer-based inclination measurements (See Eq 2-7) [2]. The fusion is controlled by a weighting factor α, which was tuned to balance the drift-prone nature of gyroscope integration with the noisier but drift-free accelerometer data (See Fig. 8). Lower α values were selected to prioritize the stability of accelerometer inputs, minimizing drift while accepting some additional noise. While the complementary filter was selected for its simplicity and computational efficiency, future iterations may implement an extended Kalman filter (EKF) for improved estimation performance under varying flight conditions.

$$\hat{\Phi}_{acc}[n] = \arctan\left(\frac{y_{acc}[n]}{x_{acc}[n]}\right) \tag{2}$$

$$\hat{\Theta}_{acc}[n] = \arcsin\left(\frac{x_{acc}[n]}{g}\right) \tag{3}$$

$$\begin{aligned}\dot{\Phi}_{gyr}[n] =& p_{gyr}[n] \\ &+ \tan\left(\hat{\Theta}[n-1]\right) \cdot \sin\left(\hat{\Phi}[n-1]\right) \cdot q_{gyr}[n] \\ &+ \cos\left(\hat{\Phi}[n-1]\right) \cdot r_{gyr}[n]\end{aligned} \tag{4}$$

$$\begin{aligned}\dot{\Theta}_{gyr}[n] =& \cos\left(\hat{\Theta}[n-1]\right) \cdot q_{gyr}[n] \\ &- \sin\left(\hat{\Phi}[n-1]\right) \cdot r_{gyr}[n]\end{aligned} \tag{5}$$

$$\hat{\Phi}[n] = \alpha \cdot \hat{\Phi}_{acc}[n] + (1-\alpha) \cdot \left(\hat{\Phi}[n-1] + T_s \cdot \dot{\Phi}_{gyr}\right) \tag{6}$$

$$\hat{\Theta}[n] = \alpha \cdot \hat{\Theta}_{acc}[n] + (1-\alpha) \cdot \left(\hat{\Theta}[n-1] + T_s \cdot \dot{\Theta}_{gyr}\right) \tag{7}$$

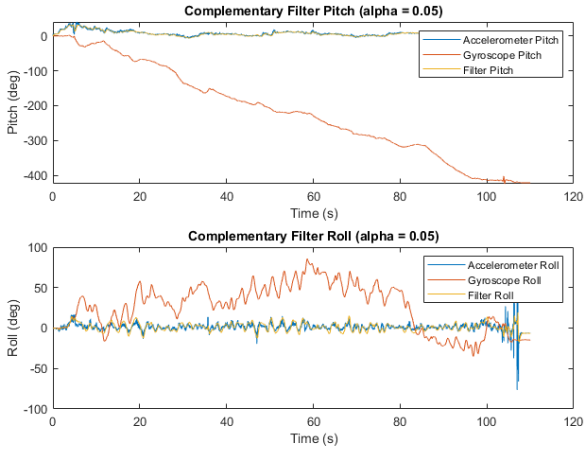

Fig. 8. Complementary Filter Pitch and Roll Plots (Flight 3)

B. Altitude Estimation Filter

The altitude estimation filter computes the UAV's altitude above ground level (AGL) using barometric pressure and temperature data (See Fig. 9). The algorithm is based on the hydrostatic equation, which relates altitude changes to variations in atmospheric pressure, assuming a standard temperature lapse rate in the troposphere (See Eq. 8) [3], [4]. Ground-level pressure and temperature are recorded at system initialization and used as reference conditions for subsequent altitude calculations. This method offers high precision and low latency, making it suitable for real-time altitude feedback within a state space control system.

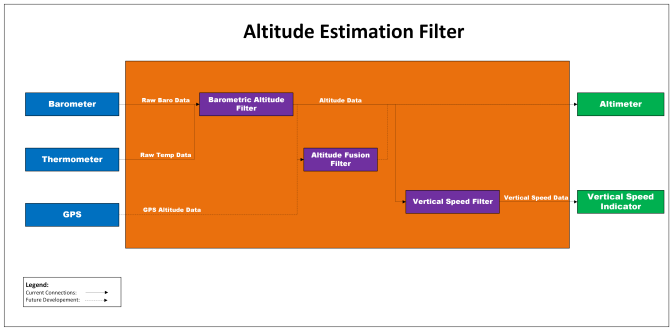

Fig. 9. Altitude Estimation Filter Architecture

$$\hat{h}[n] = \frac{T[0]}{L} \cdot \left(\left(\frac{P[n]}{P[0]}\right)^{\frac{-L \cdot R}{g}} - 1\right) \tag{8}$$

Although GPS altitude was available, it was excluded from the primary altitude estimate due to its lower update rate. Initial flight tests demonstrated that barometric altitude tracking was sufficiently accurate for autopilot operation, with observed deviations remaining within acceptable tolerances (See Fig. 10).

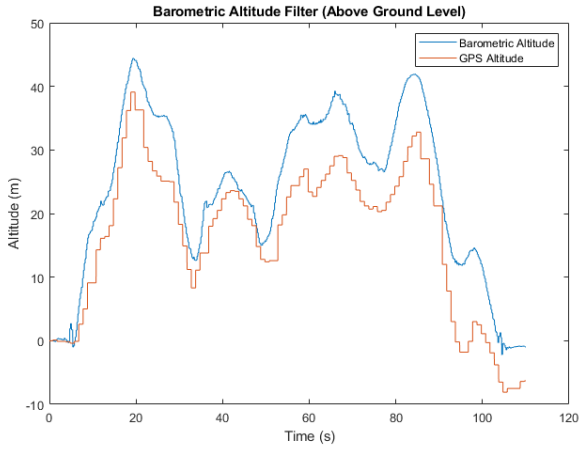

Fig. 10. Barometric/Temperature Computed Altitude and GPS Altitude Comparison (Flight 3)

To estimate vertical speed, a finite-difference derivative of the altitude signal was computed using a backward difference method (See Eq. 9) [5]. This discrete derivative was further smoothed using a first-order IIR filter to suppress high-frequency noise introduced by atmospheric fluctuations and sensor quantization (See Eq. 1 and Fig. 11).

$$d\hat{h}[n] = \frac{\hat{h}[n] - \hat{h}[n-1]}{T_s} \tag{9}$$

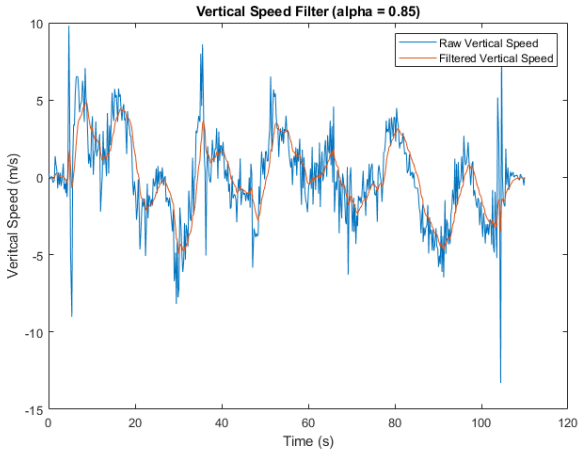

Fig. 11. Vertical Speed Filter Plot (Flight 3)

C. Flight Instrument Outputs

The final outputs of the filtering architecture were integrated into a minimal set of virtual flight instruments required for autonomous UAV navigation and system characterization. Filtered pitch, roll, and yaw estimates from the AHRS were mapped to the Attitude Direction Indicator (ADI) and Heading Indicator (HI). Altitude estimation and vertical speed from the altitude estimation filter were mapped to the Altimeter and Vertical Speed Indicator (VSI) respectively.

The instruments outputs combined with the raw GPS data form the core flight data required for autonomous flight stabilization and performance analysis. The modular structure of the filtering and instrumentation system ensures compatibility with standard state space autopilot architectures and supports future extensions for higher-order control systems.

VI. Real-World Flight Testing

To test the robustness of the flight data recorder and flight data filtering algorithms, real-world flight testing was conducted, and the captured data was analyzed. Originally flight testing was planned to be performed as a single task, but due to the iterative nature of the development of the flight data recorder and flight data filtering algorithms, flight testing was performed several times over the course of the semester. Flight testing was carried out at a designated model airfield (Arvada Associated Modelers Airpark) (See Fig. 12).

Several flights were successfully recorded by the flight data controller, giving several data sets to analyze. Running the flight data filtering algorithms on the test data proved relative success of the algorithms in terms of noise reduction and accurate state estimates when compared to flight footage. Due to the lack of a reference for the actual states of the aircraft during flight, limited analysis could be performed to determine the accuracy of the estimates empirically. For pitch, roll, and yaw, the flight footage was compared to the estimated states from the AHRS filter to determine relative accuracy (did the AHRS filter capture pitch up/down, roll left/right, and yaw left/right movements that could be observed in the flight footage). Visual observation confirmed that the AHRS filter was at least roughly tracking orientation. For altitude, the GPS altitude served as a reference for the barometric/temperature computed altitude and demonstrated successful altitude tracking (See Fig. 10). For more accurate validation of flight data filtering algorithms, a commercially available flight controller could be used to provide an accurate reference to determine filter performance.

Fig. 12. Flight Testing at Arvada Associated Modelers Airpark

One of the main constraints that limited the amount of real-world testing that could be accomplished over the course of the semester was weather. Flight testing was delayed several times due to high wind speeds, snow, and rain. Future development could be performed during the summer season to reduce weather related delays.

VII. State-Space Modeling and Control

To enable autonomous flight with closed-loop regulation, a state-space representation of the UAV's longitudinal dynamics was developed. This model captures the relationships between the control inputs (elevator deflection and throttle) and key flight states, including airspeed, angle of attack, pitch angle, pitch rate, and altitude. By linearizing the nonlinear aircraft dynamics about a trimmed flight condition, a reduced-order linear time-invariant (LTI) system was obtained. This representation enabled the systematic design of state-feedback controllers using classical and modern control techniques, with an initial focus on altitude stabilization.

A. Three-Degree-of-Freedom Longitudinal Model

Given the inherent complexity of six-degree-of-freedom aircraft dynamics, a reduced three-degree-of-freedom (3-DoF) longitudinal model was first constructed to establish a baseline for control system design. An existing nonlinear model of longitudinal dynamics was implemented in MATLAB/Simulink based on simulated UAV airframe characteristics [6]. The model included six state variables: airspeed, angle of attack, pitch angle, pitch rate, absolute altitude, and horizontal distance. As horizontal distance was not essential for vertical guidance, it was excluded from the control-relevant state vector.

The simulation environment was developed using a MATLAB function block that computes the time derivatives of the aircraft states based on current state values and control inputs. The equations of motion were parameterized using aerodynamic coefficients obtained from Open Vehicle Sketch Pad (OpenVSP) simulations over a range of flight conditions and control surface deflections. Physical airframe parameters such as mass and moments of inertia were extracted from a CAD-based model developed in Autodesk Fusion 360. The resulting nonlinear system was numerically integrated using Simulink integrator blocks to simulate state evolution over time (See Fig. 13).

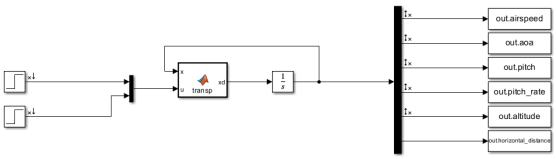

Fig. 13. Non-linear 3-DoF Simulink Model

To determine an appropriate trim condition for linearization, a steady-state flight solution was obtained using a constrained optimization procedure. MATLAB's `fminsearch` function was employed to minimize deviations from force and moment equilibrium, yielding steady-state values for both state variables and control inputs. These values were verified via time-domain simulation (See Fig. 14). Linearization about the trimmed operating point was performed using the MATLAB Model Linearizer tool, resulting in state-space matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ for use in controller design.

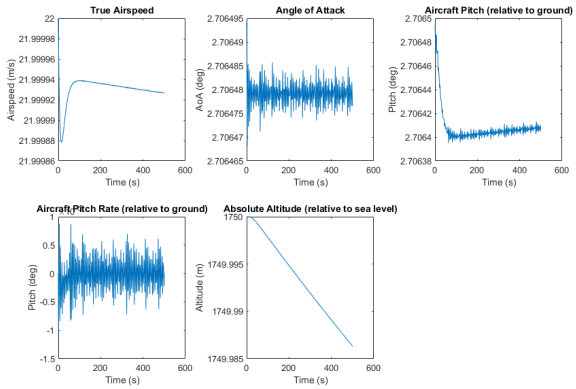

Fig. 14. Steady State Flight Simulation of Non-linear Model

B. Altitude Control System

An altitude control system was implemented in Simulink using the linearized 3-DoF model as the basis for controller synthesis (See Fig. 15).

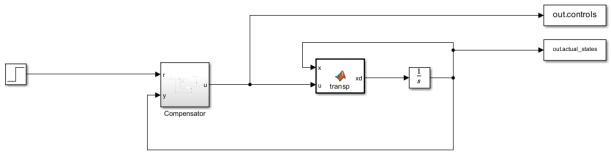

Fig. 15. 3-DoF Altitude Controller Architecture

The estimated states are fed into a compensator designed via linear quadratic regulation (LQR), which computes the optimal control inputs to minimize a weighted cost function of tracking error and control effort [7]. Reference scaling was applied to ensure correct steady-state tracking behavior under the linear model assumptions [8] (See Fig. 16).

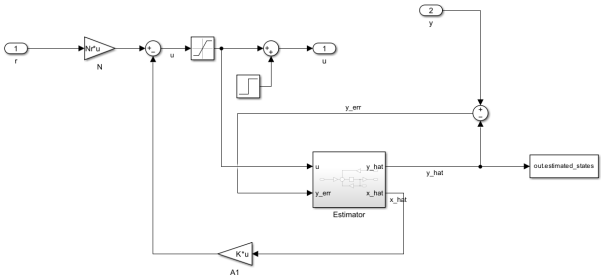

Fig. 16. 3-DoF Altitude Controller Compensator

The resulting controller successfully demonstrated regulation of altitude and convergence to reference commands within the nonlinear simulation environment, validating the efficacy of the linear control design (See Fig. 17-18).

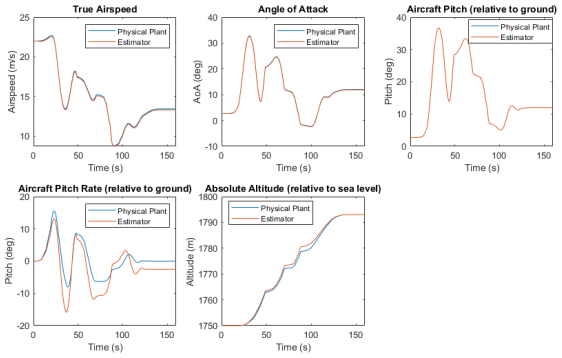

Fig. 17. Altitude Controller Simulation States (Climb 100ft)

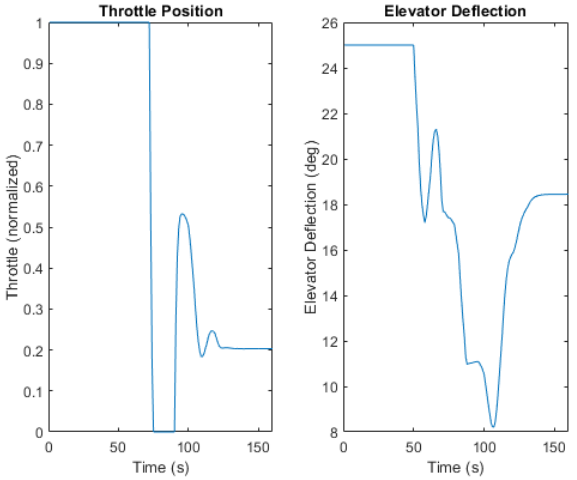

Fig. 18. Altitude Controller Simulation Control Inputs (Climb 100ft)

VIII. Conclusion

This paper presented the design and implementation of a custom autopilot firmware architecture for a fixed-wing unmanned aerial vehicle. The development process involved sensor interface design, real-time data acquisition, digital signal processing for state estimation, flight data logging, and preliminary state-space control system design. Emphasis was placed on building a modular and extensible framework suitable for fixed-wing UAV navigation and control.

Although the complete implementation of waypoint navigation and autonomous flight modes was not achieved within the project timeline, key foundational components—including sensor drivers, filtering algorithms, and a state-space-based altitude control system—were successfully developed and validated through real-world flight testing and simulation. The filtering architecture demonstrated the ability to extract usable state estimates from noisy flight data, and the preliminary controller showed effective altitude tracking within a nonlinear simulation environment.

These results establish a functional platform for continued development of autonomous flight capabilities. Future work will focus on expanding the state estimation framework to include more advanced filtering methods, completing the implementation of full flight control modes, and executing autonomous waypoint navigation. Improvements to both hardware reliability and sensor accuracy will further support the transition from simulation to in-flight control demonstrations.

References

[1] P. Salmony, "IIR Filters - Theory and Implementation (STM32) - Phil's Lab #32," YouTube, Sep. 07, 2021. https://www.youtube.com/watch?v=QRMe02kzVkA [accessed Mar. 03, 2025].

[2] P. Salmony, "Complementary Filter - Sensor Fusion #2 - Phil's Lab #34," YouTube, Sep. 30, 2021. https://www.youtube.com/watch?v=BUW2OdAtzBw [accessed Mar. 03, 2025].

[3] "ICAO standard atmosphere," Oxford Reference, 2025. https://www.oxfordreference.com/display/10.1093/oi/authority.20110803095955775 [accessed Mar. 03, 2025].

[4] "A Quick Derivation relating altitude to air pressure." Portland State Aerospace Society, Dec. 12, 2004. https://archive.psas.pdx.edu/RocketScience/PressureAltitude_Derived.pdf [accessed Mar. 03, 2025].

[5] P. N. Shah, "Newton's Backward Difference formula (Numerical Interpolation) Formula & Example-1," Atozmath.com, 2025. https://atozmath.com/example/CONM/NumeInterPola.aspx?q=B&q1=E1 [accessed Mar. 03, 2025].

[6] B. L. Stevens, E. N. Johnson, and F. L. Lewis, Aircraft control and simulation, 3rd ed. : dynamics, controls design, and autonomous systems. Malden, MA: Wiley-Blackwell, 2015.

[7] K. Johnson. "EENG417: Modern Control Design: Optimal Control," Colorado School of Mines, 2024.

[8] K. Johnson. "EENG417: Modern Control Design: Systems with Non-zero Reference Signals," Colorado School of Mines, 2024.