An Architecture for Onboard Flight Control of a Sub-Gram Piezo-Actuated Aerial Vehicle

Sriram Kodey

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Mechanical Engineering

University of Washington
2025

Committee:
Sawyer B. Fuller
Santosh Devasia
Mehran Mesbahi

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

An Architecture for Onboard Flight Control of a Sub-Gram Piezo-Actuated Aerial Vehicle

Sriram Kodey

Chair of the Supervisory Committee:

Sawyer B. Fuller
Mechanical Engineering

The development of agile, autonomous insect-scale aerial vehicles opens promising avenues for applications in environmental monitoring, search and rescue, and swarm robotics. However, the creation of such miniature platforms is hindered by stringent constraints on size, weight, and computational resources to achieve sensing, control, and actuation all within tight limits. This thesis presents a comprehensive architecture for onboard flight control of a sub-gram, piezo-actuated aerial vehicle. The design and implementation of a microcontroller-based high-voltage actuation system capable of generating precise signals for piezoelectric actuators is discussed in detail. A finite state machine (FSM) combined with a cascaded PID control architecture was developed for stabilization and maneuvering, utilizing real-time feedback from external motion capture systems operating at 240 Hz. Additionally, a lightweight and efficient logging infrastructure was developed to enable continuous recording of state estimates and control signals for post-flight analysis and debugging. Experimental results demonstrate the system's successful real-time generation of smooth, continuous sinusoidal waveforms for the piezoelectric actuators in response to pose feedback, as well as reliable capture of flight log data. Collectively, these results highlight the effectiveness of the proposed system and lay the foundation for control autonomy in insect-scale aerial robotics.

Acronyms

DAC Digital-to-Analog Converter. 3

DMA Direct Memory Access. 3

FIRs Flying Insect Robots. 3

FSM Finite State Machine. 3

LQR Linear Quadratic Regulator. 3

MPC Model Predictive Control. 3

PID Proportional-Integral-Derivative. 3

RTOS Real-Time Operating System. 3

SPI Serial Peripheral Interface. 3

Acknowledgements

I would like to sincerely thank Prof. Sawyer Fuller, Prof. Santosh Devasia, and Prof. Mehran Mesbahi for serving on my thesis committee. I am especially grateful to Prof. Fuller for his unwavering support and guidance throughout this research. His insights have been instrumental in shaping both the direction and outcomes of this work.

I am deeply appreciative of Dr. Daksh Dhingra, under whose mentorship I began my journey in the lab. I would also like to acknowledge my labmates in the Autonomous Insect Robotics Laboratory for cultivating a supportive and stimulating environment. In particular, I'm thankful to Yash Talwekar for his camaraderie, encouragement, and help throughout this project.

My sincere thanks go to my friends in the department, Amy and Thomas, for their steady support and friendship. I'm especially grateful to Chuou—for all the moments we've shared on and off the badminton court, and for her continued presence, encouragement, and joy throughout this journey. To my friends back home, thank you for your continued support in all aspects of my life.

Finally, I am profoundly grateful to my parents and sister for being my foundation. Their encouragement to aim higher, their constant positivity, and their unwavering emotional support have carried me through this journey. None of this would have been possible without them.

DEDICATION

To my father, Kodey V. Satyanaryana

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Insect-scale flapping-wing robots (FIRs) have emerged as a compelling platform for developing agile, lightweight, and energy-efficient aerial systems. Inspired by the biomechanics and control strategies of natural flyers such as bees and flies, these micro-robots—often weighing less than 150 mg—are capable of navigating confined environments and executing precise maneuvers, making them ideal candidates for a wide range of applications [1], [2], [3]. Their small size and low cost of mass production further enhance their suitability for distributed tasks across complex and inaccessible terrains [4], [5]. The potential use cases for FIRs span across several domains: search and rescue operations in collapsed or cluttered structures, weather and microclimate monitoring in forests or urban canopies, hazardous material detection in contaminated zones, and even exploration of extraterrestrial environments such as the Martian atmosphere [6], [7], [8]. Recent designs such as the Tumbler robot have demonstrated rapid post-collision recovery and gust resilience in cluttered environments [9].

However, flapping wing insect-scale robots (FIRs) remain largely constrained to laboratory environments, relying on external systems for power, sensing, and control. As a result, true autonomy for these platforms is still out of reach. However, recent advances in microscale power systems and sensor suites represent important progress toward overcoming these limitations. Notably, the liftoff of a 190 mg laser-powered aerial vehicle demonstrated the feasibility of wireless flight without tethers, marking a key milestone in power autonomy [10]. In parallel, the development of a lightweight, power-efficient avionics system has enabled onboard sensing and estimation capabilities at the sub-gram scale—advancing FIRs toward sensor autonomy [11], [12], [13].

Controlling flapping insect robots (FIRs) is inherently challenging due to the unique and extreme demands

of their dynamics and form factor. Insect flight is characterized by natural instability, requiring control mechanisms that operate at high frequencies to counteract the rapid growth of instabilities [14]. The problem is further exacerbated by the robots' low inertia and highly reactive dynamics, which demand swift and precise control inputs. FIRs typically exhibit a high torque-to-moment-of-inertia ratio—on the order of 10^3 rad/s^2—which results in extremely fast and sensitive responses to control inputs [15]. Additionally, these systems are highly nonlinear and susceptible to variability introduced by manufacturing inconsistencies and rapid mechanical degradation.

Compounding these difficulties are the extreme constraints on Size, Speed, Weight, and Power (SSWaP), which limit the onboard computation resources available for implementing sophisticated control algorithms. These constraints necessitate the development of lightweight, high-frequency control and actuation frameworks that can operate reliably under tight computational budgets and dynamic uncertainty.

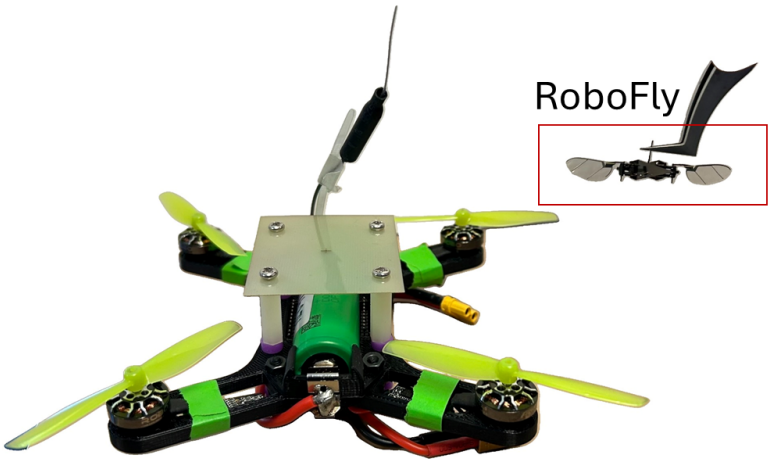

Figure 1.1: Showcasing the scale of the UW RoboFly next to a CrazyFlie (commercially available 30 g drone).

1.1 Analysis on the Effect of Scaling

Let us choose a characteristic length, the wingspan of the robot, L, to perform a scaling analysis to understand the necessity for high-speed control. We use a setup with time scales T_F and T_I, representing the forward (translational) motion time scale and inertial (oscillatory) motion time scale for pitch motions, respectively. This is the same setup used in [14], but we ignore pitch damping for simplicity, i.e, $T_P \to \infty$.

With this assumption, using the dynamics equations (Eq 3.3 & Eq 3.4) from [14], we get an equation in λ, solutions being the eigenvalues of the system. The positive real parts of the eigenvalues of the intrinsically unstable system, $Re(\lambda)$, determine the time scale of instability growth T_{INST}.

$$\lambda^3 + \frac{1}{T_F}\lambda^2 - \frac{1}{T_F T_I^2} = 0 \tag{1.1}$$

The mass of the robot, $M \propto L^3$, the moment of Inertia $I \propto L^5$, and the distance between the center of drag and center of mass (independent parameter based on the design), h, determine the Inertial Time Scale T_I.

$$T_I \equiv \sqrt{\frac{I}{Mg|h|}} \implies T_I \propto L\sqrt{\frac{1}{|h|}} \implies T_I = k_I L\sqrt{\frac{1}{|h|}}$$

The following equation (Eq 4.2 from [14]) defines T_F. Note that Mass, $M \propto L^3$, and Wing Surface Area, $S \propto L^2$, are the only factors that scale with size or wingspan, L. Thus, we can arrive at a scaling relationship between T_F and L.

$$T_F = \frac{M}{2\rho S C_D w} \implies T_F \propto L \implies T_F = k_F L$$

From (1.1), using the relationship between T_I, T_F, and L we get the following relationship between λ, and L.

$$\lambda^3 + \frac{1}{k_F L}\lambda^2 - \frac{|h|}{k_F k_I^2 L^3} = 0 \tag{1.2}$$

Assume $\lambda = a + bi$, substituting this in (1.2) and setting $k_F = k_1$ and $k_F k_I^2 = k_2$ gives us

$$\underbrace{(a^3 - 3ab^2) + \frac{1}{k_1 L}(a^2 - b^2) - \frac{|h|}{k_2 L^3}}_{\text{Real Part}} + \underbrace{i((3a^2 b - b^3) + \frac{1}{k_1 L}(2ab))}_{\text{Imaginary Part}} = 0 \tag{1.3}$$

Setting the imaginary part to $zero$, we can solve for b. The trivial solution is $zero$, since the cubic equation (1.2) should have at least one real solution. We can substitute b in the real part, which should also equal $zero$, giving us the following equation.

$$8a^3 + \frac{8a^2}{k_1 L} + \frac{2a}{k_1^2 L^2} + \frac{|h|}{k_2 L^3} = 0 \tag{1.4}$$

Substituting $a = \frac{\alpha}{L}$, yields the following equation

$$8\alpha^3 + \frac{8}{k_1}\alpha^2 + \frac{2}{k_1^2}\alpha + \frac{|h|}{k_2} = 0 \tag{1.5}$$

Since, $a = Re(\lambda)$ the following relationship can be established, where $\alpha(h)$ is the real valued solution to the cubic in (1.5)

$$\implies Re(\lambda) = \frac{\alpha(h)}{L} \tag{1.6}$$

This relationship translates to the following between T_{INST} and L, following $T_{INST} \propto \frac{1}{Re(\lambda)}$, i.e., a larger positive real part of the eigenvalue leads to faster growth of instability resulting in a smaller timescale.

$$T_{INST} \propto \frac{L}{\alpha(h)} \tag{1.7}$$

We can interpret (1.7) as the following: with the scale of wingspan or length L decreasing, the instability growth timescale decreases, i.e., instabilities grow faster. Previous work in control theory [16][17][18] has shown that a phase margin of 45° is indicative of acceptable control performance. Let's define a timescale T_{RXN} that corresponds to how fast reactive control is applied, which is directly related to the sampling time of the system, the time required for computing the feedback control input, and the application of this control input to the system. A smaller T_{RXN} corresponds to running the feedback control loop at a higher frequency. [14] demonstrates that the ratio $\frac{T_{INST}}{T_{RXN}} = 6$ provides a performance metric equivalent to a phase margin of 45°, and thus this time-scale ratio can be used as a practical criterion for our case. Therefore, as T_{INST} varies with scaling of L, T_{RXN} should also follow suit to maintain the ratio. Thus, we arrive at the following relationship.

$$T_{RXN} \propto \frac{L}{\alpha(h)} \tag{1.8}$$

Hence, as the size or wingspan of the robot decreases, T_{RXN} decreases, i.e., faster reaction is necessary for good quality control. The design parameter h also has an influence on the required reaction speed,

depending on its magnitude, $\alpha(h) \approx \sqrt[3]{|h|}$ or $\alpha(h) \approx |h|$. h can be varied accordingly by modifying the design to influence the required reaction times.

1.2 Motivation

The cost of transport for the UW RoboFly, a piezo-actuated FIR, decreases exponentially with flight velocity [Fig. 1.2], highlighting the need for advanced control mechanisms—such as receding horizon control—to enable efficient flight.

Recent work from our lab has successfully demonstrated stable hovering and trajectory tracking in a 150

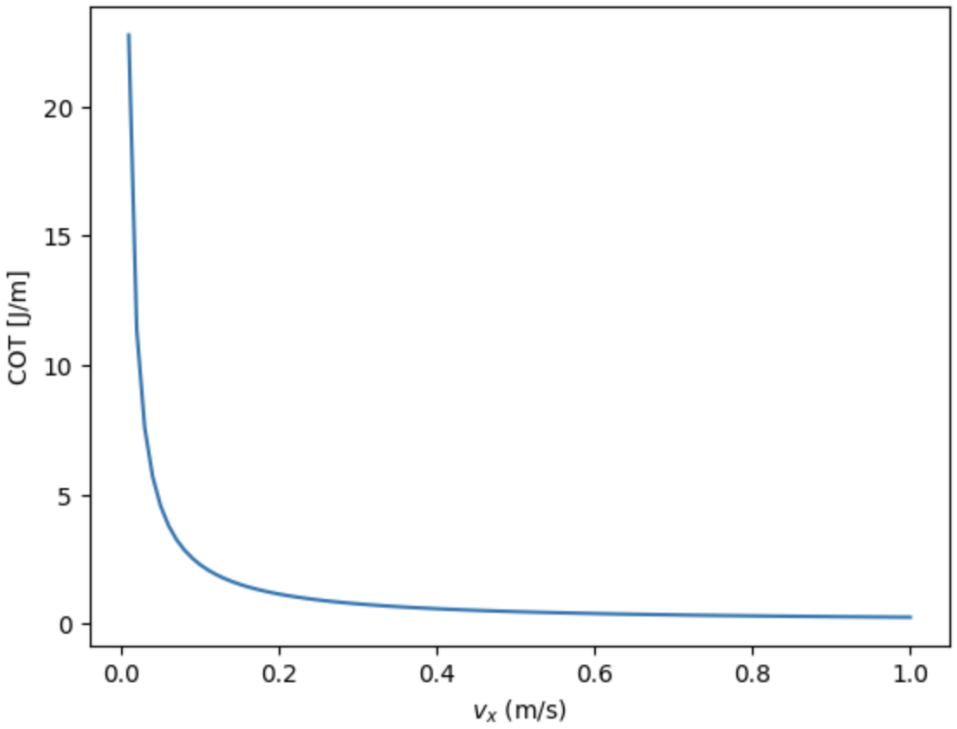

Figure 1.2: Cost of Transport vs Flight Speed characterization of the UW RoboFly.

mg flying insect robot (FIR), the UW RoboFly, using an infinite-horizon Linear Quadratic Regulator (LQR) based on a more accurate stroke-averaged force and torque model. The controller achieved RMS position errors of approximately 4 cm and tracked translational maneuvers up to 25 cm/s, marking the first demonstration of optimal control on an FIR using force-torque data validated with motion capture [15]. Additionally, data-driven MPC approaches like Deep Learned Tube-MPC show promise for optimal control under actuator constraints and trajectory tracking for complex maneuvers such as ramps and infinity loops [19],

[20]. However, both the methods were deployed on a desktop PC with SimuLink Real-Time, the controller running at 2 kHz and the signal generator at 10 kHz—a framework not suitable for onboard implementation on sub-150 mg robots. Microprocessors small enough to be carried onboard, such as the 10 mg, 120 MHz STM32F4 used in the first wireless liftoff of an FIR, the UW RoboFly in [10], are capable of floating-point math operations. Nevertheless, their performance will be limited to a fraction of desktop capabilities, just a few hundred MHz, for the foreseeable future. Recently, the TinyMPC framework demonstrated MPC running at 500 Hz on an STM32F405 using a motion capture system, showing promise for deploying optimal control on resource-constrained platforms [21]. However, to achieve control autonomy in FIRs, we must develop an embedded system-based actuation and control framework that eliminates reliance on external infrastructure.

1.3 Primary Contribution

This thesis presents the design and implementation of a lightweight control autonomy stack tailored for sub-gram piezo-actuated flapping-wing robots.[1] My primary contributions span the development of a custom microcontroller-based actuation system capable of generating high-voltage signals for piezoelectric actuators, the design of a real-time control framework integrating feedback from external or onboard sensors for flight stabilization and maneuvering, and the creation of a logging infrastructure to capture flight data for post-analysis. Together, these components form a cohesive autonomy architecture [Fig. 1.4] that enables closed-loop flight experiments and lays the groundwork for future work on control autonomy for piezo-actuated FIRs.

1.4 Thesis Outline

Chapter 2 details the design of a microcontroller-based actuation system capable of generating high-voltage output signals required for driving piezoelectric actuators on a sub-gram flying robot. This includes the hardware architecture, signal conditioning circuitry, and firmware implementation for generating voltage waveforms.

Chapter 3 presents the integrated control framework developed to stabilize and maneuver the robot using

[1] The code for the work presented in this thesis is available at the RoboFly cf-firmware repository on GitHub.

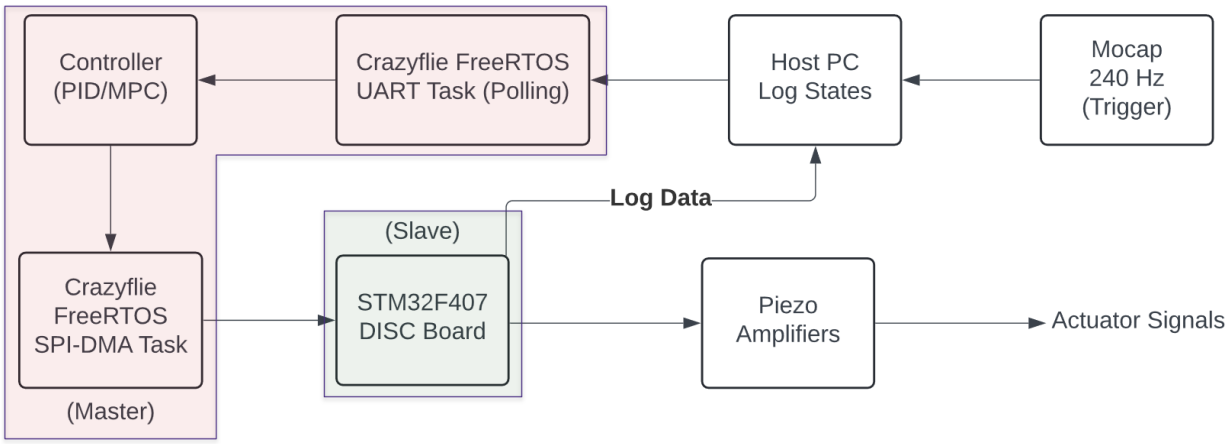

Figure 1.3: High level description of the developed system architecture.

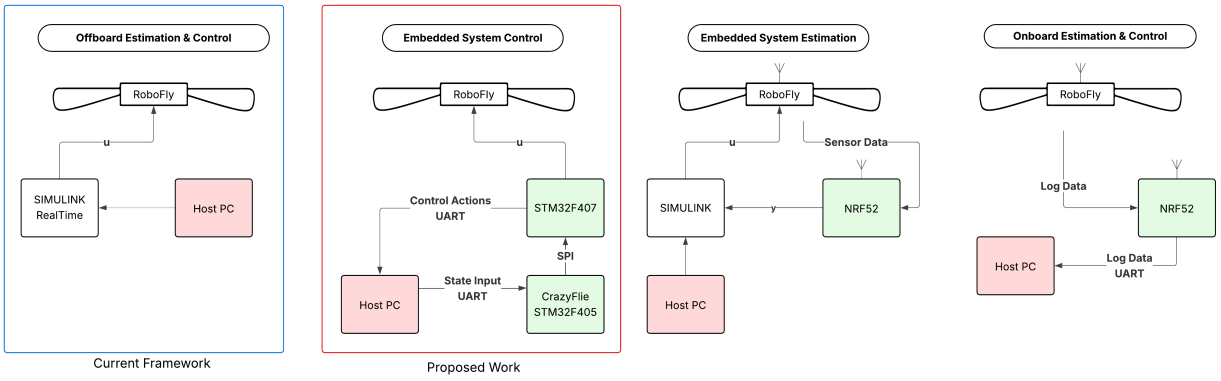

Figure 1.4: Overview of the planned autonomy development architecture, highlighting key stages and their interconnections

real-time feedback from an external motion capture system or onboard sensing. This includes task scheduling, control loop design, and communication protocols between sensing and actuation components.

Chapter 4 describes the logging infrastructure constructed to store state estimates and control commands during flight experiments continuously. It outlines the memory management strategy, data synchronization techniques, and the tools developed for post-flight analysis and debugging.

Chapter 2

Actuation

2.1 Actuator Description

Actuation in FIRs can be realized using a variety of mechanisms, each with specific advantages and limitations. The primary strategies include piezoelectric [22], [23], electromagnetic [24], dielectric elastomer actuators [25], electrostatic [26], and shape memory alloy actuators [27], as well as less common hydraulic and pneumatic systems [28]. In this work, we focus exclusively on FIRs utilizing piezoelectric actuators, as the UW RoboFly platform employs this actuation method.

The wing actuation mechanism of the insect-scale robots utilizes a bimorph cantilever structure composed of a carbon fiber substrate layered between two piezoelectric sheets. In the "simultaneous drive" configuration, the top surface of the bimorph is maintained at a constant high voltage (bias), while the bottom surface is connected to ground. An alternating driving signal is applied to the middle layer, generating an oscillating electric field across the piezoelectric material, as shown in Fig(2.1) [1]. This configuration produces small, alternating strains in the piezo layers via the reverse piezoelectric effect, which results in tip displacement of the cantilever. The tip motion is further amplified by a microfabricated transmission to achieve wing stroke amplitudes of approximately 90 degrees. Viewed from above, the motion of the wings generates downward airflow, enabling aerodynamic lift for flight [30]. In the case of the UW RoboFly, the top layer is maintained at 260V (Bias voltage) and the actuators are controlled by a voltage signal described by the equation:

$$V(t) = \left(V_0 \pm \frac{\Delta_{\text{offset}}}{2}\right) + \left(A_0 \pm \frac{\Delta_A}{2}\right) \sin(\omega t)$$

[1] *Image courtesy of Dr. Johannes James and Prof. Sawyer Fuller* [29]

The amplitude A_0, Δ_A, and Δ_{offset} are the control inputs to the system. A larger amplitude corresponds to

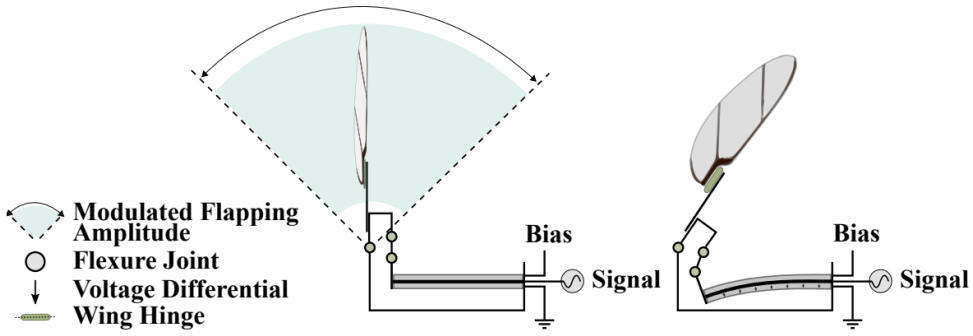

Figure 2.1: Illustration of how a single piezoelectric actuator controls wing motion in the UW RoboFly.

a larger wing stroke, which provides control over the lift produced by the wing, as shown in [Fig. 2.2a]. Δ_A is used to create a difference in the lift generated by the two wings, allowing control of body roll. Δ_{offset} is used to shift the mean of the wing stroke, enabling control of body pitch, as illustrated in [Fig. 2.2b]. Driving Mechanism: The piezo actuators are driven by Op-amps coupled with 3 high-voltage amplifiers

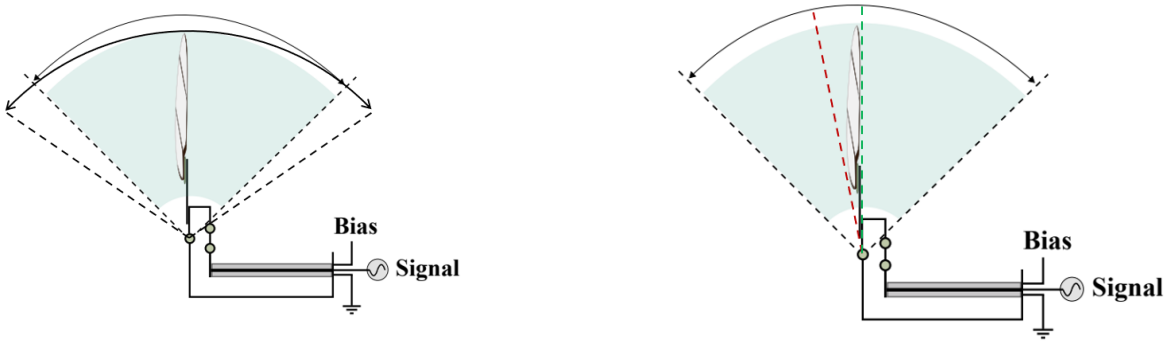

Figure 2.2: (a) Larger amplitude wing stroke produces greater lift. (b) Offset controls the mean position of the wing stroke and can generate pitch torque.

(Trek 2205). A powered switch-box provides the DC "bias" signal to both actuators, while the other two amplifiers supply separate sinusoidal drive signals to each of the two wings.

Actuation Signal Requirements

- The voltage signal must be smooth and continuous, as the actuators are brittle and susceptible to damage. Voltage surges can cause crack formation.
- The actuators must be carefully ramped up and down from 0 V to their baseline voltages to avoid

abrupt changes that could cause cracks or failure.

- The desired actuation waveform is sinusoidal, with a frequency of up to 200 Hz, typically in the range of 170–180 Hz.

- If each sine-wave cycle is discretized into 1000 time steps (each representing a specific voltage level), the actuation system must support a voltage generation rate of up to 200 kHz.

2.2 DMA-based DAC Actuation System

This system is designed for high-precision signal generation and control, integrating both real-time control signal processing and high-speed waveform output. It is built around an STM32F407 microcontroller that includes dedicated hardware peripherals for efficient data transfer and analog signal generation. The primary purpose of the system is to produce a precisely controlled analog output waveform— for driving the two piezoelectric actuators.

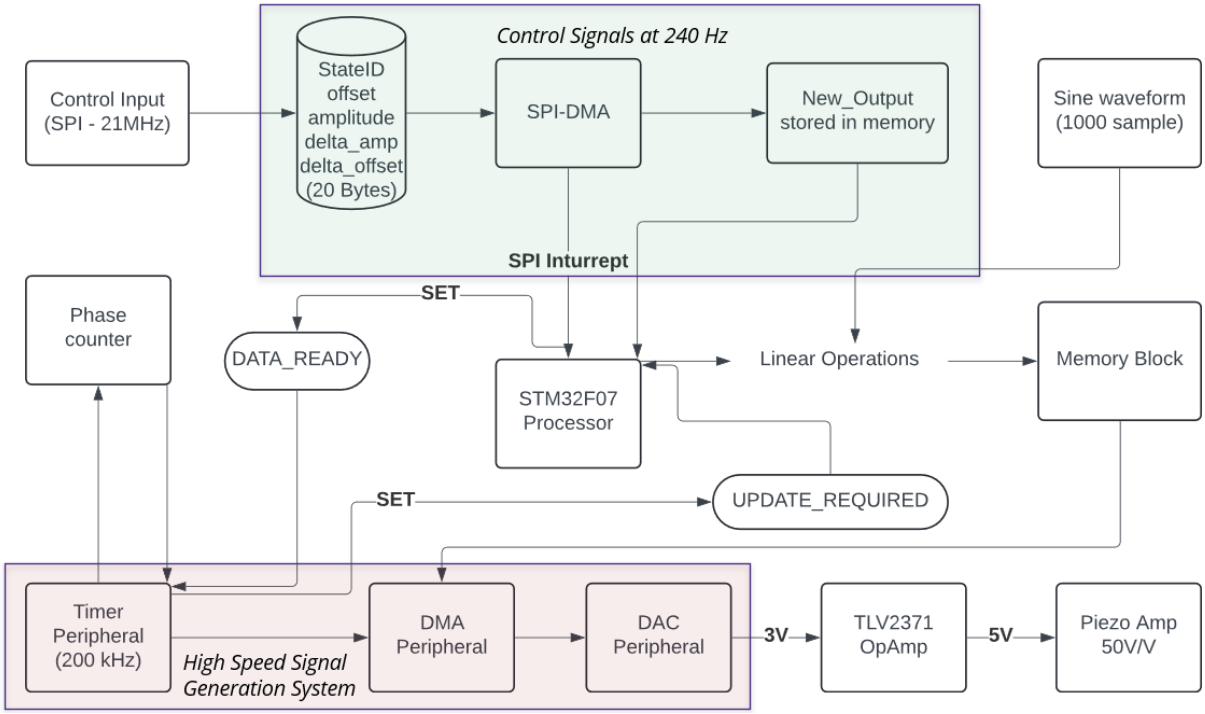

Figure 2.3: Description of the developed DMA-based DAC actuation system.

2.2.1 Control Input Management via SPI-DMA

The process begins at the control input, which communicates via an SPI interface operating at 21 MHz. Control data is received by the microcontroller in packets of 20 bytes, containing the following fields: `StateID`, `offset`, `amplitude`, Δ`amp`, and Δ`offset`. These parameters define the characteristics required for generating the output sinusoidal waveform.

The incoming control data is managed by the Serial Peripheral Interface (SPI) module using the Direct Memory Access (DMA) peripheral for efficient, non-blocking transfers. The DMA ensures that control packets are relayed seamlessly into memory without CPU interference, triggering an SPI interrupt only after a complete transaction and setting the `DATA_READY` flag. This approach ensures efficient use of precious compute resources.

2.2.2 Parameter Storage and Propagation

Once received, the new waveform parameters are stored in RAM. The STM32F407 core fetches these parameters when the `DATA_READY` flag is set, enabling immediate responsiveness to input commands. The processor performs linear operations on a lookup table using the updated parameters and sets an `UPDATE_REQUIRED` flag, indicating that the waveform should be updated in the next signal generation cycle. The aforementioned lookup table is a static memory block containing 1000 `float` (4-byte) variables representing a baseline sinusoidal waveform. This waveform is computed once during system startup, eliminating the need for recalculation during operation and reducing computational load.

2.2.3 High-Speed Signal Generation System

A hardware timer peripheral operates at up to 200 kHz (180 kHz in this work), acting as the heartbeat for high-rate signal generation. On each timer event interrupt, a phase counter advances to track the signal phase and ensure waveform updates occur without introducing discontinuities. New waveforms are written into memory only when the phase of the voltage signal is either 0 or π. When new waveform data or parameter updates are available (as indicated by the `UPDATE_REQUIRED` flag), and the current phase permits modification, the system updates another static memory block exposed to the DMA peripheral. This block contains the signal pattern derived from the most recent control parameters and is used for output generation.

DMA and DAC Peripheral

DMA is used to efficiently fetch samples from waveform memory and stream them to the DAC peripheral in real time, again minimizing CPU involvement. The DAC converts the digital data stream into an analog voltage, producing a continuous analog waveform. The same hardware timer peripheral also coordinates this transfer. Upon each update event interrupt (triggered by timer overflow), data from a specific address in the static memory block is transferred to the DAC peripheral's Data Holding Register (`DHR`). After one clock cycle, this data is transferred to the DAC peripheral's Data Output Register (`DOR`) and becomes available as voltage on the output pin after a settling time t_{settling}, which depends on the power supply voltage and the analog output load. DMA is configured to operate in circular mode, ensuring that the DAC peripheral can continuously cycle through the waveform without requiring external intervention. The DAC output is given by the expression:

$$\text{DAC}_{\text{output}} = V_{\text{REF}} \times \frac{\text{DOR}}{4096}$$

Thus, a 12-bit resolution is available for the analog channels.

2.3 High-Voltage Output

The DAC output (typically in the range of 0–3V) is buffered and conditioned by a TLV2371 operational amplifier, providing impedance matching and signal stabilization. This op-amp output is then fed to a high-gain Piezo Amplifier. Add more details here The Piezo Amplifier boosts the analog voltage by a factor of 50, raising the signal level from 5V up to 250V, suitable for driving the piezoelectric actuators.

2.4 Functional Summary

- **Responsiveness**: Real-time SPI control logic updates operational parameters at up to $240\,\text{Hz}$.
- **Efficiency**: DMA-based data handling in both input (SPI) and output (DAC) paths minimizes CPU load and maximizes throughput.
- **Precision Output**: Hardware-timed signal generation ensures precise, low-jitter analog waveform output at up to $200\,\text{kHz}$.

- **Scalability**: The system can accommodate varying output signal forms by adjusting the waveform memory contents and control parameters.

This architecture is well-suited to this application, requiring fast update rates, low-latency control, and precise, high-voltage analog signal generation.

Chapter 3

Controller Framework

The controller framework proposed in this work utilizes an RTOS-based architecture to enable integration of various functional tasks, including sensor polling, data acquisition, pose estimation, and controller execution within the robotic control system. By supporting both blocking and non-blocking execution modes, the framework allows for flexible and adaptive real-time scheduling, leading to efficient utilization of computational resources under diverse operational scenarios. Inter-task communication and data exchange are facilitated through message queues and notification mechanisms, ensuring reliable and timely information flow between tasks. The use of FreeRTOS—a widely adopted real-time operating system in embedded systems—enables deterministic task management and emulates real-time execution on resource-constrained single-core platforms such as the STM32F4 series, thereby addressing challenges commonly encountered in embedded robotic applications.

A finite state machine (FSM) is utilized within the control framework to systematically govern the system's operational modes and transitions. By encoding the controller's key behaviours—such as initialization, actuation, and shutdown—into discrete, well-defined states, the finite state machine (FSM) facilitates safe and orderly operation of the piezo-actuators. Operating independently of the controller itself, the FSM serves as an intermediary that manages transitions between system modes in response to external commands or internal events. This design not only enforces operational safety but also decouples high-level system logic from low-level control implementation, thereby enhancing both system robustness and modularity.

Key Design Requirements

- **Real-time State Feedback**: The system should integrate feedback at 240 Hz (the frequency at which the motion capture system operates) to enable fast, accurate control.
- **Actuator Ramping**: Smoothly ramps the actuators' offset and amplitude up and down to prevent crack formation in the actuators.
- **Future-Proof Structure**: The overall design and modularity make it easy to introduce additional controllers or onboard estimators when required.

Core Design Choices

- **RTOS Foundation**: Built on FreeRTOS, the framework exploits multitasking and task synchronization for responsive, high-throughput real-time operations.
- **Finite State Machine (FSM)**: The FSM approach streamlines state handling, specifically managing the controlled ramp-up and ramp-down of actuators.
- **Extensive Modularity**: Individual tasks and modules (state machine, controller, communications, filtering, pose conversions) are logically separated, allowing independent development, testing, and future extension.
- **Inter-Task Communication**: Data queues and task notifications manage the passage of information between the UART Task, State Machine, and Controller Logic—enabling both decoupling and low-latency response.

3.1 Architectural Flow

3.1.1 External Pose Data (Host PC, 240 Hz)

External pose updates stream into the system via a Universal Asynchronous Receiver Transmitter (UART) peripheral. The UART Task operates the peripheral in polling mode, executing every 4 ms. The pose data packet from the Host PC includes marker bytes at the start and end. Upon wake-up, the task scans for a single byte in timeout mode to avoid CPU starvation. Once the start marker is detected, the task resets its

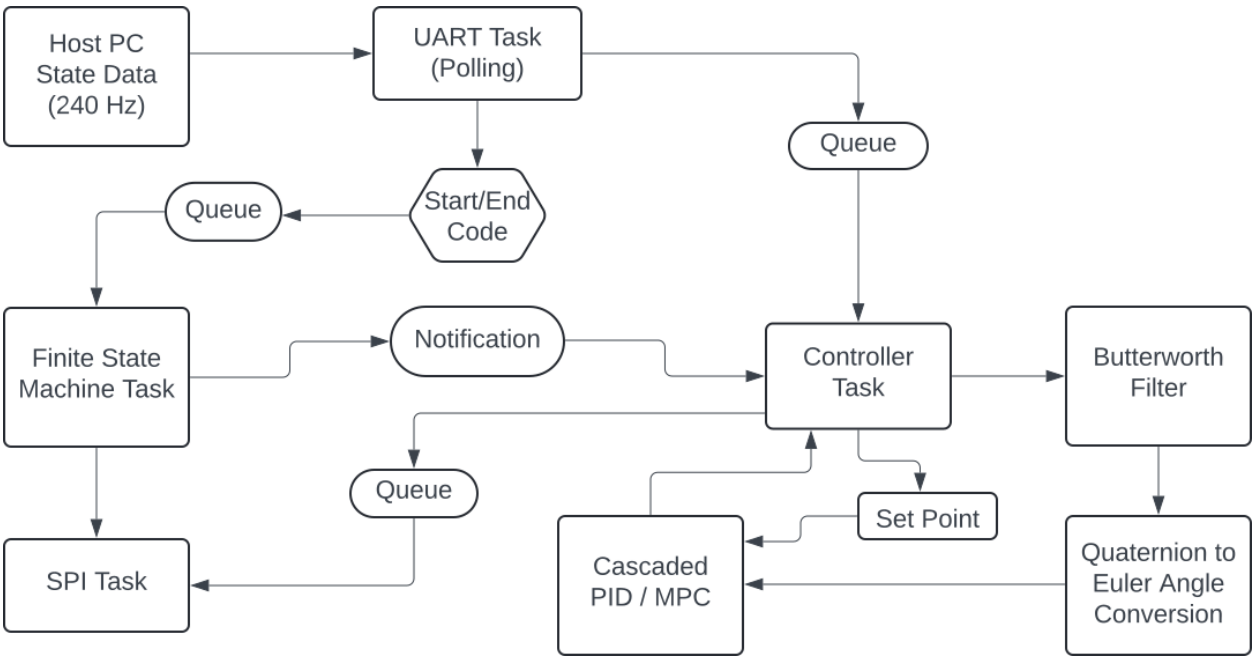

Figure 3.1: Description of the developed RTOS-based Finite State Machine and Controller framework.

clock, reads the complete packet, checks its integrity using the end marker byte, and then enqueues the pose data to the `flyControllerTask` and the start code to the `stateMachineTask`. The task then goes back to sleep, waking back up again in 4 ms, theoretically 0.16 ms before the next pose update.

3.1.2 Finite State Machine (FSM) Task

The Finite-State Machine (FSM) task is responsible for managing system behavior and ensuring the safe operation of piezo actuators through methodical state transitions. Interprets incoming start and end codes, manages actuator command profiles, including critical ramp-up and ramp-down sequences, and communicates directly with the SPI task for hardware-level actuation updates. In addition, the `stateMachineTask` (FSM task) issues explicit mode notifications determining the operational mode of the `flyControllerTask`, which serves as an intermediary between the high-level supervisory logic and the implementation of the underlying controller. For task synchronization, the FSM leverages the FreeRTOS task notification mechanism, which is highly efficient with minimal CPU overhead and ideally suited for lightweight, low-latency inter-task communication. The FSM and flyController transition through several defined states and modes:

FSM States and Transitions

- **Idle State**: In this state, the task remains dormant and immediately enters a low-power sleep. No actuator commands are issued until a start signal is received.

- **Offset Ramp**: Upon activation, the FSM begins en-queueing control outputs to the SPI task, initiating with an offset of zero and linearly increasing to the baseline offset voltage of 130 V. During this period, the controller is set to IDLE mode. While in idle, the flyControllerTask estimates the initial position of the robot by calculating a time-averaged value of the incoming pose data, providing a reliable baseline for subsequent operation. Once the offset reaches 130 V, the FSM transitions to the next state.

- **Amplitude Ramp**: Maintaining the offset at the baseline value (130 V), the FSM ramps the amplitude from 0 to the nominal value of 140 V, en-queuing the corresponding control outputs to the SPI task. Upon reaching 140 V, control transitions to the liftoff phase.

- **Liftoff**: The FSM now delegates command output duties to the `flyControllerTask`, signaling a switch to liftoff mode. The `flyControllerTask` then activates the altitude and attitude control loops, utilizing feedback from the input pose data stream to stabilize the system during liftoff. This phase lasts approximately 100 ms, after which the FSM transitions to full control mode.

- **Control On**: The FSM notifies the `flyControllerTask` to switch into the CONTROL_ON mode and then enters an idle state, awaiting new system events. During this mode, the `flyControllerTask` enables the lateral controller and sets the desired setpoint. It processes pose data as received from the UART task, computing and en-queuing actuator commands accordingly.

- **Landing**: Upon reception of an end signal from the UART interface, the FSM transitions the system into landing mode and notifies the `flyControllerTask` to do likewise. In this phase, the controller independently generates ramp-down sequences for both amplitude and offset—without waiting for additional pose data input—safely decreasing actuation levels and sending corresponding outputs to the SPI task. This ensures a controlled descent and hardware protection. After 500 ms, the FSM advances to amplitude ramp-down

- **Amplitude Ramp-Down**: Control of the output commands reverts to the FSM, with the controller placed in RESET mode to facilitate parameter re-initialization. The FSM decreases the amplitude

from its baseline value (140 V) to zero while maintaining a constant offset, sequentially issuing the requisite commands to the SPI task.

- **Offset Ramp-Down**: The FSM continues the shutdown sequence by ramping the offset voltage from 130 V to zero. Upon completion, the FSM resets its internal parameters and returns the system to idle mode, completing the operational cycle.

State	Action	**Controller Mode**
IDLE	–	**–**
OFFSET_RAMP	Offset to 130V, Turn on Bias	**–**
AMP_RAMP	Amp to 140V	**–**
LIFTOFF	–	**LIFTOFF**
CONTROL_ON	–	**CONTROL_ON**
LAND	–	**LAND**
AMP_RAMP_D	Amp to 0V	**RESET**
OFFSET_RAMP_D	Offset to 0V, Turn off Bias	**–**
RESET	–	**–**

Table 3.1: *Description of FSM states and the controller notifications they send.*

Controller Mode	Action
IDLE	Estimate the initial position
LIFTOFF	Altitude and Attitude control switched on
CONTROL_ON	Lateral control also switched on
LAND	Ramp ΔAmp and $\Delta Offset$ to 0v
RESET	Reset controller parameters

Table 3.2: *Description of the different controller modes in the framework.*

Through this structured sequencing, the FSM task and flyController task ensure that all transitions between operational phases are conducted in a safe, predictable, and coordinated manner, particularly during events such as ramp-up, liftoff, and landing. By decoupling supervisory logic from real-time control execution, the FSM architecture enhances both system modularity and operational reliability.

3.1.3 Fly Controller Task

The `flyControllerTask` operates as the computational core of the control framework. It receives mode notifications from the Finite State Machine (FSM) and acquires state or setpoint data through dedicated queues. Upon receipt of new data, the Controller Task invokes the active control algorithm—in this work, a cascaded PID (Proportional-Integral-Derivative) controller—using the incoming pose information, the current setpoint, and the initialized controller object. The resultant computed outputs are then dispatched to the SPI Task as actuator commands for the downstream signal chain. The operational flow of the Controller Task is inherently coordinated with the FSM Task, as detailed in the preceding section. The FSM dictates the control mode and hands over control duties where appropriate, ensuring seamless state transitions and system safety. A notable feature of the Controller Task design is its modularity: the controller instantiation and computation routines are abstracted such that the specific control algorithm can be substituted with minimal changes to the overall system architecture. This future-proofs the framework and facilitates the introduction of advanced or alternative control schemes for future research work. Moreover, the input queue managed by the Controller Task is accessible not only to the existing data pipeline (pose data received over UART) but also to potential onboard state estimation modules. This extensible architecture enables a transition from reliance on external motion capture to integrated, autonomous state estimation without major redesign, further enhancing system flexibility and scalability.

3.1.4 SPI Task

The SPI Task is dedicated to transmitting actuator command data to the physical hardware interface, specifically interfacing with the STM32F407 Discovery board via high-speed Serial Peripheral Interface (SPI) communication. SPI is a synchronous, full-duplex protocol in which the master device initiates data exchanges, and hardware NSS (slave-select) lines are used to activate and synchronize communication with the slave device. Unlike encapsulating SPI transfers within a function, implementing SPI communication as a separate FreeRTOS task offers significant advantages in a real-time system. Due to the FreeRTOS scheduling policy, a function-based SPI transfer cannot be guaranteed exclusive execution unless called from the highest-priority task. Even in that case, a running task may be preempted if another task of equal priority becomes schedulable. By delegating SPI transfers to a dedicated high-priority task, the system ensures that the high-speed data transmissions are performed with minimal latency and interruption, thereby maintaining

reliable and deterministic actuator updates. This design is particularly critical at the high operational speeds required in this application, and it safeguards the integrity of command delivery to the hardware.

3.2 Control System Design

3.2.1 Cascaded PID Control

The control framework implements a cascaded Proportional-Integral-Derivative (PID) architecture in which altitude and lateral motion are regulated through distinct yet interconnected control loops. Pose estimates provided by the motion capture system—position and orientation—are first processed through a Butterworth filter and passed to the individual controllers.

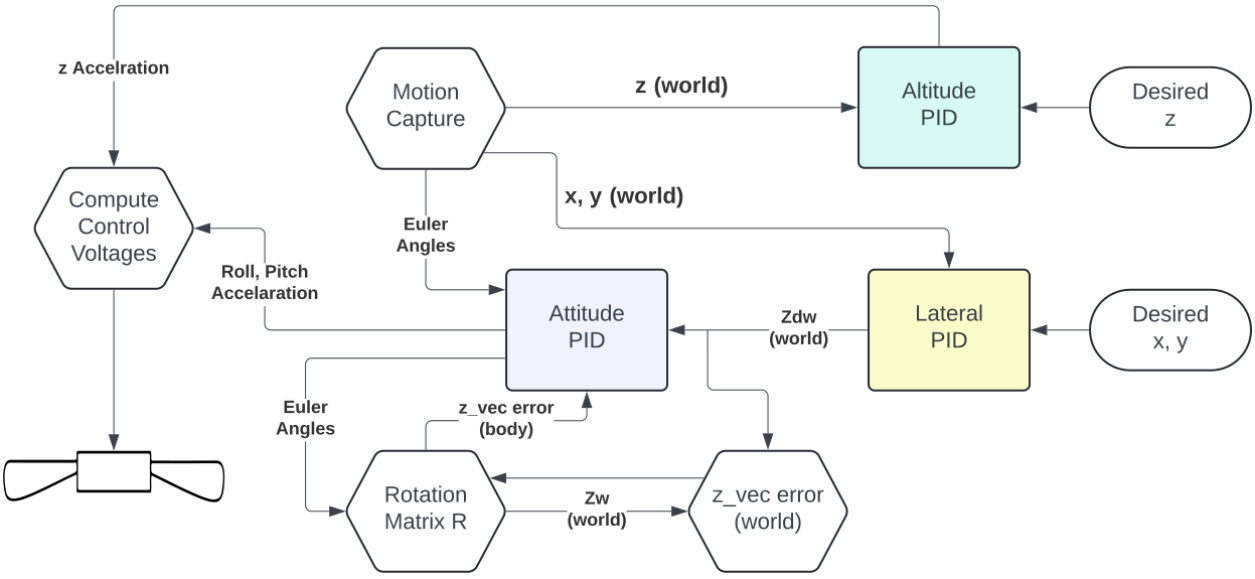

Figure 3.2: Description of the Cascaded PID controller implemented in the system.

- **Altitude PID controller**: Receives the z position and maintains the desired vertical position.
- **Lateral PID controller**: Computes the necessary orientation of the body axis (specifically, the thrust vector direction) in the world frame that would achieve the desired x, y position. This desired body orientation, expressed as a vector in the world frame, is then passed to the attitude PID controller.
- **Attitude PID controller**: Computes the error between the desired and current body orientations, transforms this error into body frame using the rotation matrix, and determines the corresponding roll and pitch accelerations required to align the system accordingly. If the lateral controller is switched

off, the attitude controller maintains the default neutral hovering attitude. Notably, yaw control is not implemented in this structure.

The outputs of these PID loops are then used to compute the actuator command voltages. This cascaded approach allows the control system to decouple translational and rotational objectives, simplifying controller design and tuning.

3.2.2 PID Controller Implementation

The underlying PID controller implemented in this framework adheres to the standard structure comprising proportional, integral, and derivative components. However, rather than differentiating the error signal, this implementation computes the derivative of the system output. This design choice is intended to reduce transient output spikes that can occur due to abrupt changes in the error term, such as when step inputs are introduced through new setpoints. By differentiating the output instead of the error, the controller generates smoother commands, thereby protecting the actuator from potentially harmful voltage spikes and enhancing overall system robustness.

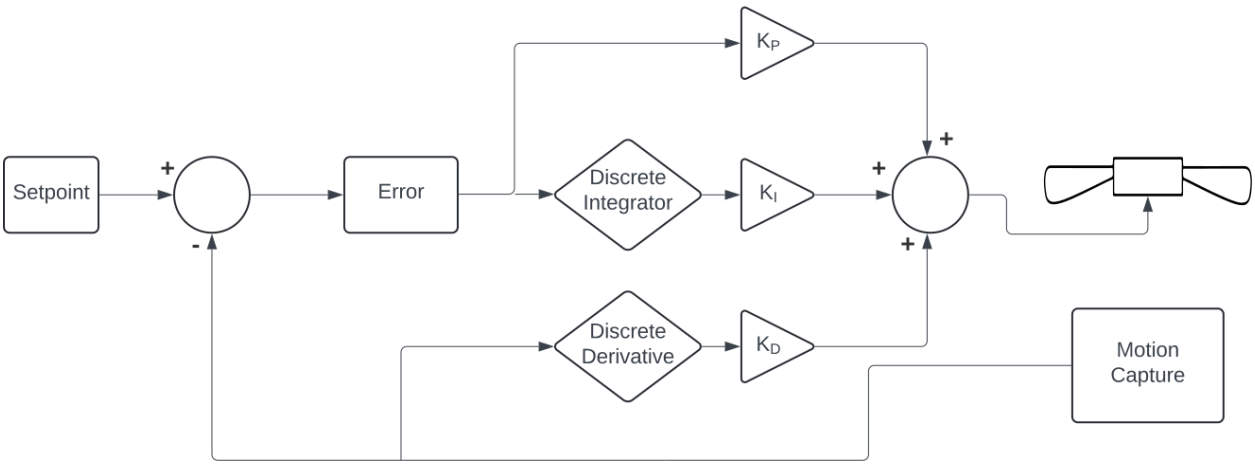

Figure 3.3: Description of the developed underlying PID controller.

Chapter 4

Data Logging Architecture

4.1 Motivation and Design Challenges

Traditional data logging solutions typically involve buffering data in random-access memory (RAM) and subsequently writing the collected information to log files upon completion of an experiment or operational cycle. While this approach is effective in systems with ample memory and relaxed real-time constraints, it is incompatible with the resource limitations inherent to embedded platforms. In such environments, RAM capacity is often highly constrained, making it infeasible to store large volumes of data directly in memory. Furthermore, instantaneous or real-time data transfer is not viable, as the bandwidth of interfaces such as UART is insufficient and may introduce significant processing overhead. Logging operations that interrupt or contend with higher-priority control or signal-processing tasks can furthermore jeopardize system responsiveness and reliability.

4.2 Queue-Based Logging Solution

To address these challenges, a queue-based data logging system has been implemented, leveraging both software and hardware resources—specifically, the DMA (Direct Memory Access) peripheral and a queue structure with a first-in-first-out (FIFO) policy. As depicted in the accompanying diagram, the logging architecture integrates several functional components:

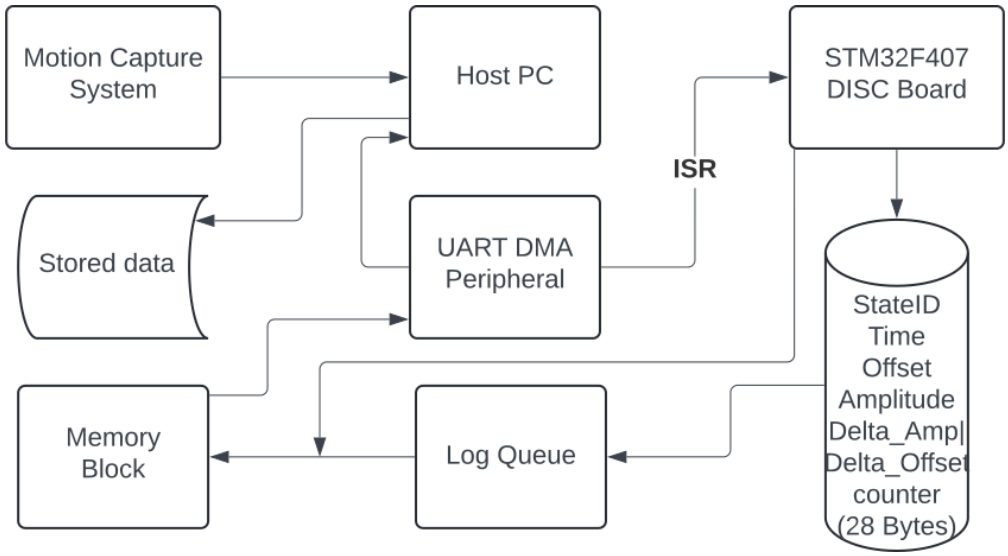

Figure 4.1: Description of the developed Queue-based Logging solution.

4.2.1 Data Collection and Enqueueing:

Relevant data—including state identifiers, time, offset, amplitude, and control-related variables—is gathered on the STM32F407 Discovery board. This data is structured in compact records (28 bytes each). Instead of direct RAM accumulation, incoming data entries are enqueued into a log queue and processed when CPU time is available. This decouples the data generation rate from the data transmission rate, allowing higher-priority tasks to proceed without blocking on slow peripheral transfers.

4.2.2 DMA-Based UART Transmission:

Once buffered, log entries are transmitted to the host PC via the UART peripheral using DMA. This hardware-assisted transfer minimizes CPU usage, as data is moved directly from memory to the UART interface with only minimal processor involvement, coordinated via interrupt service routines (ISR).

4.3 Advantages

By streaming data out as it is generated and avoiding accumulation in RAM, overall memory usage remains low. The queue-based, DMA-assisted design ensures that the primary control and actuation tasks are not hindered by logging activity, preserving real-time system performance.

Chapter 5

Conclusion

The proposed flight control system has successfully demonstrated the generation of smooth, continuous waveform signals required for the precise actuation of the piezoelectric actuators. The system effectively incorporated real-time pose feedback at 240 Hz, allowing the control framework to operate at high update rates necessary for insect flight. Additionally, the lightweight logging infrastructure continuously recorded state estimates and control signals throughout flight experiments without interfering with signal generation or control performance. In revisiting the objectives established at the outset of this work—namely, the real-time generation of actuator signals, reliable integration of high-rate feedback, high-frequency control, and efficient onboard logging—each goal has been achieved. The system's performance in hardware-in-the-loop experiments demonstrates its readiness as a foundation for deploying onboard control for piezo-actuated insect-scale flight. The following section discusses the future directions for onboard autonomous flight, followed by photos and plots of the developed system.

5.1 Future Work

Building on the architecture presented in this thesis, the next step is to experimentally demonstrate stable hovering of the RoboFly using the developed embedded flight control system. Transitioning from reliance on powerful desktop computers to an embedded system for flight control and actuation marks a significant step toward control autonomy. Demonstrating stable hovering with the developed system will serve as a key milestone in this progression, validating the feasibility of autonomous operation on highly constrained insect-scale platforms.

A key direction for further research is the implementation of optimal control algorithms on the embedded system. Leveraging more advanced control techniques will enable improved agility, efficiency, and robustness, especially as flight maneuvers become increasingly complex.

In parallel, ongoing developments in onboard state estimation will be critical for sensor autonomy. Advancements in compact, lightweight, and power-efficient sensing technologies will enable us to move away from dependence on external motion capture systems. With onboard sensors providing real-time state estimation, the RoboFly platform will be able to operate untethered and independently, paving the way for fully autonomous insect-scale flight.

5.2 Photos & Plots

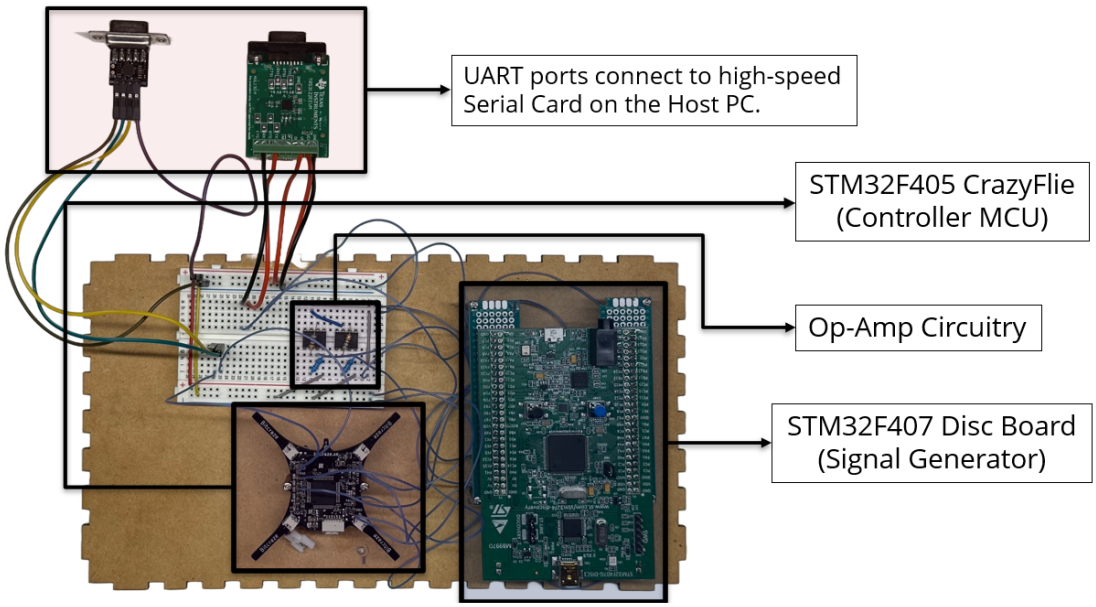

Figure 5.1: Description of the developed system.

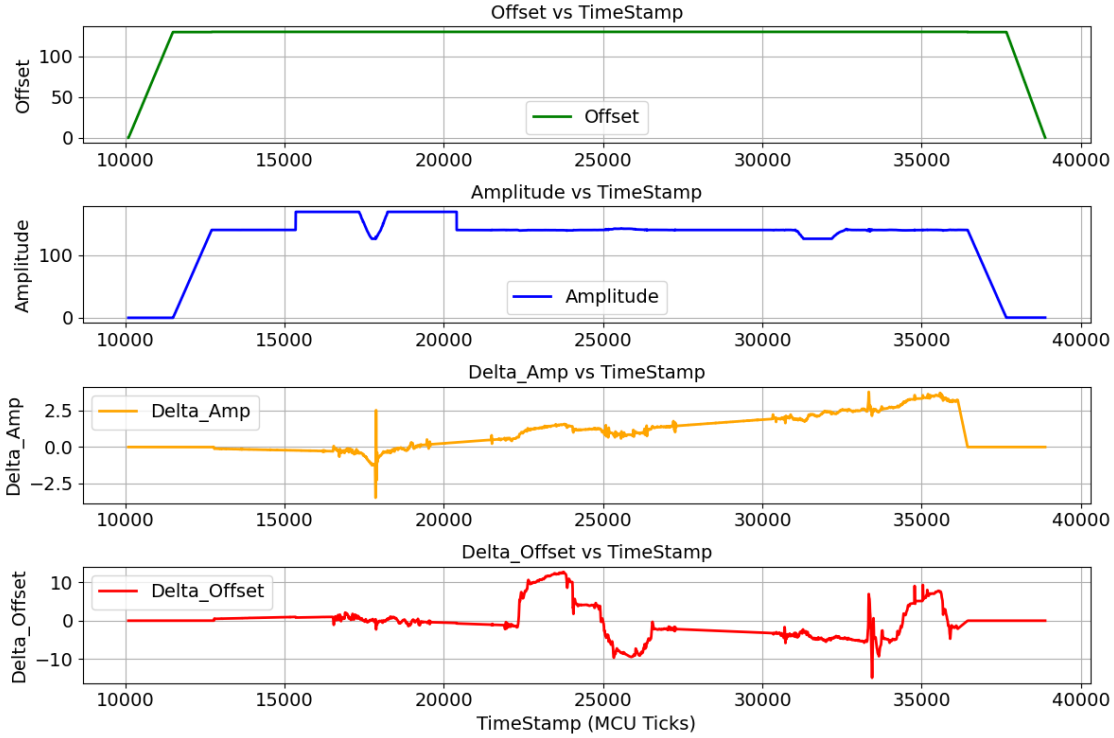

Figure 5.2: Plot of the waveform signal parameters from the logged data.

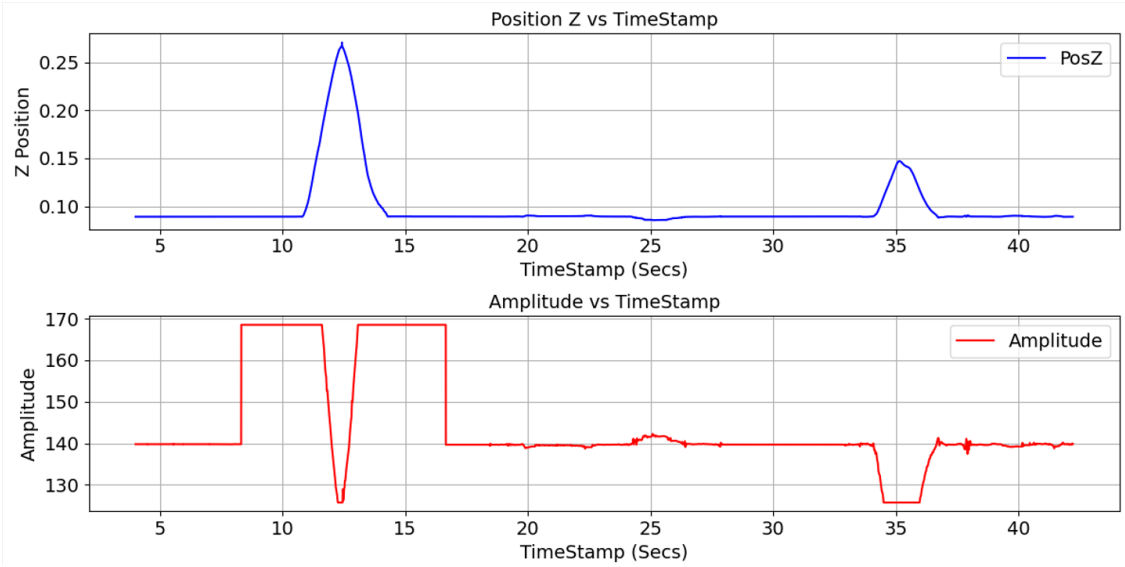

Figure 5.3: Plot of Altitude (z) and Amplitude vs time.

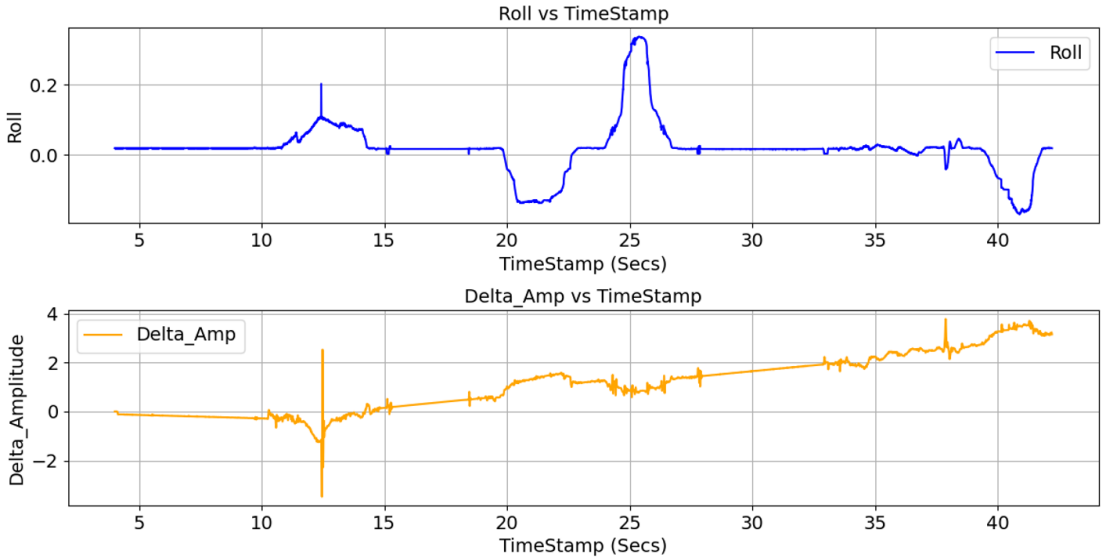

Figure 5.4: Plot of Roll and Delta Amplitude vs time

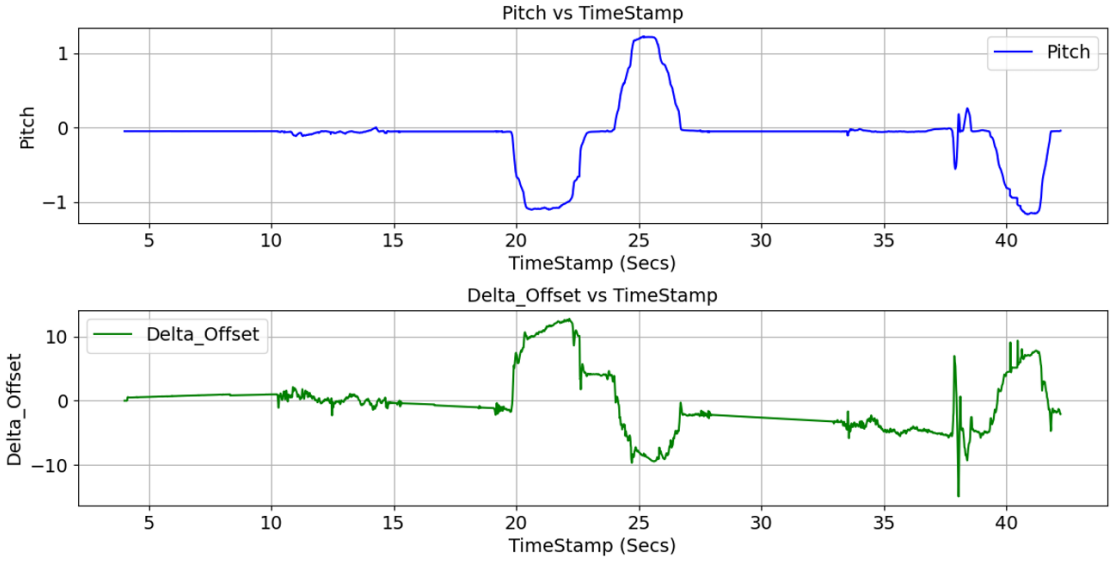

Figure 5.5: Plot of Pitch and Delta Offset vs time.

Bibliography

[1] R. Wood, "Design, fabrication, and analysis of a 3dof, 3cm flapping-wing mav," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 1576–1581.

[2] K. Ma, P. Chirarattananon, S. Fuller, and R. Wood, "Controlled flight of a biologically inspired, insect-scale robot," *Science (New York, N.Y.)*, vol. 340, pp. 603–7, 05 2013.

[3] S. Xie, C. Richter, K. Peterson, J. Li, and S. Chung, "A review on flapping-wing robots: Recent progress and challenges," *The International Journal of Robotics Research*, vol. 43, no. 4-5, pp. 567–595, 2024. [Online]. Available: https://journals.sagepub.com/doi/10.1177/02783649251343638

[4] S. Fuller, M. Karpelson, A. Censi, K. Ma, and R. Wood, "Controlling free flight of a robotic fly using an onboard vision sensor inspired by insect ocelli," *Journal of the Royal Society, Interface / the Royal Society*, vol. 11, 08 2014.

[5] C. Zhu, X. Kong, Y. Zhou, L. Zhang, and Y. Sun, "Hybrid locomotion for flapping-wing aerial robots with passive hopping legs," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Abu Dhabi, UAE, 2024, pp. 11 234–11 241. [Online]. Available: https://arxiv.org/abs/2503.00805

[6] M. Graule, P. Chirarattananon, S. Fuller, N. Jafferis, K. Ma, M. Spenko, R. Kornbluh, and R. Wood, "Perching and takeoff of a robotic insect on overhangs using switchable electrostatic adhesion," *Science*, vol. 352, pp. 978–982, 05 2016.

[7] R. S. F. et al., "The berkeley microrobotic fly project," Online, available: https://eecs.berkeley.edu/research/berkeley-microrobotic-fly.

[8] T. Qin, W. Jin, Z. Zhao, Y. Sun, and Y. Sun, "An insect-scale flapping-wing robot capable of 9 dynamic flying primitives via wing–tail coordination," *Nature Communications*, vol. 16, no. 1, p. 4809, 2025. [Online]. Available: https://www.nature.com/articles/s44172-025-00480-9

[9] L. Xu, Q. Wang, J. Wang, H. Dong, and M. Sun, "Tumbler: A robust and agile insect-scale flapping-wing robot with self-righting and gust recovery capability," *Science Robotics*, vol. 9, no. 76, p. eadh1935, 2025. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.adh1935

[10] J. James, V. Iyer, Y. Chukewad, S. Gollakota, and S. Fuller, "Liftoff of a 190 mg laser-powered aerial vehicle: The lightest wireless robot to fly," 05 2018, pp. 1–8.

[11] Y. P. Talwekar, A. Adie, V. Iyer, and S. B. Fuller, "Towards sensor autonomy in sub-gram flying insect robots: A lightweight and power-efficient avionics system," in *Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022. [Online]. Available: https://depts.washington.edu/airlab/wordpress/wp-content/uploads/2022/04/talwekar_adie_iyer_fuller_sensor_suite_icra2022_compressed.pdf

[12] S. Fuller, Z. Yu, and Y. P. Talwekar, "A gyroscope-free visual-inertial flight control and wind sensing system for 10-mg robots," *Science Robotics*, vol. 7, no. 72, p. eabq8184, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abq8184

[13] Z. Yu, J. Tran, C. Li, A. Weber, Y. P. Talwekar, and S. Fuller, "TinySense: A lighter weight and more power-efficient avionics system for flying insect-scale robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Atlanta, USA, May 2025, p. to appear, may 19–23. [Online]. Available: https://arxiv.org/abs/2501.03416

[14] L. Ristroph, G. Ristroph, S. Morozova, A. J. Bergou, S. Chang, J. Guckenheimer, Z. J. Wang, and I. Cohen, "Active and passive stabilization of body pitch in insect flight," *Journal of the Royal Society Interface*, vol. 10, no. 85, p. 20130237, 2013. [Online]. Available: http://doi.org/10.1098/rsif.2013.0237

[15] D. Dhingra, K. Kaheman, and S. B. Fuller, "Modeling and lqr control of insect sized flapping wing robot," *npj Robotics*, vol. 3, no. 1, p. 6, 2025. [Online]. Available: https://doi.org/10.1038/s44182-025-00022-7

[16] K. J. Å ström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ: Princeton University Press, 2008.

[17] K. J. Åström, "Limitations on control system performance," *European Journal of Control*, vol. 6, no. 1, pp. 2–20, 2000.

[18] J. S. Freudenberg and D. P. Looze, "Right half plane poles and zeroes and design tradeoffs in feedback systems," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 555–565, 1985.

[19] A. Tagliabue, Y. Hsiao, U. Fasel, J. N. Kutz, S. L. Brunton, Y. Chen, and J. P. How, "Robust, high-rate trajectory tracking on insect-scale soft-actuated aerial robots with deep-learned tube mpc," *arXiv preprint arXiv:2209.10007*, 2022. [Online]. Available: https://arxiv.org/abs/2209.10007

[20] U. Fasel, Y. Hsiao, A. Tagliabue, J. N. Kutz, S. L. Brunton, Y. Chen, and J. P. How, "Learning-based trajectory tracking control for insect-scale flapping-wing robots," *arXiv preprint*, 2024. [Online]. Available: https://arxiv.org/abs/2411.15130

[21] K. Nguyen, S. Schoedel, A. Alavilli, B. Plancher, and Z. Manchester, "Tinympc: Model-predictive control on resource-constrained microcontrollers," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[22] R. J. Wood, "The first takeoff of a biologically inspired at-scale robotic insect," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 341–347, 2008. [Online]. Available: https://ieeexplore.ieee.org/document/4479958

[23] S. B. Fuller, N. N. T. Ojikutu, V. K. S. Jandhyala, T. V. S. T. Tran, and R. J. Wood, "Controllable compact electronic linear piezoelectric actuator for insect-scale flapping-wing robots," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 2, pp. 510–521, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8606451

[24] K. Tanaka, Y. Tanaka, and H. Liu, "Electromagnetic actuation with flexible transmission for flapping-wing micro air vehicles," *Bioinspiration Biomimetics*, vol. 7, no. 3, p. 036012, 2012. [Online]. Available: https://iopscience.iop.org/article/10.1088/1748-3182/7/3/036012

[25] Z. Ren, S. Kim, X. Ji, W. Zhu, F. Niroui, J. Kong, and Y. Chen, "A high-lift micro-aerial-robot powered by low-voltage and long-endurance dielectric elastomer actuators," *Advanced Materials*,

vol. 34, no. 7, p. 2106757, 2022. [Online]. Available: https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/adma.202106757

[26] M. Jebelli and K. J. Ma, "Electrostatic actuation for centimeter-scale untethered insect-inspired robots," *Nature Communications*, vol. 14, no. 1, p. 5438, 2023. [Online]. Available: https://www.nature.com/articles/s41467-023-36024-8

[27] P. G. Ifju and M. R. Waszak, "Development of micro air vehicles: Lightweight, adaptive, and autonomous," in *AIAA Paper 2000-0436*, 2000. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2000-436

[28] M. Sitti, "Mobile microrobotics," *Nature*, vol. 458, pp. 1121–1129, 2009. [Online]. Available: https://www.nature.com/articles/nature08106

[29] J. James and S. Fuller, "A high-voltage power electronics unit for flying insect robots that can modulate wing thrust," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2021, p. 7212–7218. [Online]. Available: https://doi.org/10.1109/ICRA48506.2021.9561869

[30] Y. Chukewad, A. Singh, J. James, and S. Fuller, "A new robot fly design that is easier to fabricate and capable of flight and ground locomotion," 10 2018.