L.I.F.T. Platform: A Hybrid Flying and Driving Robot for Safe Exploration of Hazardous Environments

Capstone Group #28

Eric Pham, Felix Hu, Jeremy Chen, Junyu Lu, Max Freeman

Advised by: Professor Francesco Borrelli and Fionna Kopp

Executive Summary

Exploring the world is a complicated challenge for both humans and robots. For humans, the challenges often lie in the accessibility of a space or the risks that accessing it poses. While the use of robots like drones and rovers can help reduce these risks, those robots are typically limited to a single mode of movement, limiting their utility across different environments. For example, wheeled robots may face difficulties navigating up staircases, while drones may face difficulties maneuvering through narrow spaces inside buildings. To enable robotic exploration of unknown environments, multimodal robots—that is, robots capable of more than one form of locomotion—are essential.

Our project aims to solve these issues through the Land and Integrated Flight Transition Platform (L.I.F.T. Platform), a hybrid robotic system capable of both driving and flying. While similar multimodal robots exist, their designs are often complex and rely on the use of different drivetrains for both flying and driving—that is, they use separate motors to actuate the wheels and propellers. This approach can add complexity to the robots, increasing cost and weight and making them more difficult to maintain in the field.

To directly address these limitations, the L.I.F.T. Platform uses a single drivetrain that actuates both flight and driving, reducing mechanical complexity. It is designed to be rugged and cost-effective, enabling operation across diverse environments. In addition, its structure is designed out of fully 3D-printed modular parts, streamlining assembly and repairs. These features make the L.I.F.T. Platform a versatile solution for a variety of applications, including search-and-rescue, last-mile delivery, infrastructure inspection, and reconnaissance.

Our project seeks to validate this design concept through three core technical demonstrations: the design and assembly of a physical prototype, the implementation of autonomous driving using Model Predictive Control, and the verification of the flight capabilities of the system through a short tethered flight test.

Contents

1. Project Scope, Context, and Impact

1.1 Navigating the Unknown: The Need for Versatile Autonomous Systems

Exploring unknown or hazardous environments is an inherently challenging task due to the unpredictable and extreme conditions it brings. These conditions often make human access impossible and render the use of traditional robotic systems impractical.

Hazardous environments impose significant limitations on what can be achieved by humans alone, with personnel being exposed to potential physical harm and operational windows being constrained by human endurance. For instance, following the Fukushima nuclear disaster in 2011, human responders faced the risk of exposure to high levels of radiation, limiting their ability to conduct investigations on-site and, as a result, slowing their response to the disaster [1]. These added constraints increase the cost and logistical support required to complete missions. In many cases, such as when access is needed to very confined spaces, human entry is not just difficult, it is impossible.

While the use of autonomous robotic systems can help reduce risk to human life, they also encounter critical limitations. Most notably, single-mode robots, such as drones or ground rovers, are specialized for either flight or terrestrial navigation. Drones perform well in surveying large areas from the air, for instance, but they face limited payload capacity, potentially short endurance, and difficulty accessing confined locations. Wheeled rovers offer larger payload capacities and longer endurance, but often struggle with uneven terrain and cannot traverse steep cliffs or gaps. As a result, completing a single objective can require deploying multiple specialized robotic systems, which increases complexity and cost.

1.2 Industry Landscape and Limitations of Existing Systems

One approach to solving many of the shortcomings of single-mode robots lies in the development of multimodal robotic systems—that is, robots capable of more than one form of locomotion. This versatility allows them to switch modes depending on their objectives. For instance, by driving long distances over open ground and then switching to fly over an obstacle that is blocking the path ahead. While there have been a number of research projects on robots of this nature to date,

each design suffers from various drawbacks. In order to illustrate this point, we will consider the design of three existing multimodal robots: the M4 robot from Caltech [2], Tsinghua University's Multimodal Land-Air Robot [3], and the LaMSA robot from Virginia Tech [4].

1.2.1 Caltech M4 Multimodal Robot

The M4 robot, shown below in Figure 1, adopts a dual drivetrain system—using separate motors to power the wheels and propellers. Its design enables a total of eight different modes of locomotion, offering a high level of environmental adaptability [2]. It features a relatively compact form factor and a carbon-fibre shell construction, increasing its resistance to crashes and impacts. However, its design is relatively complex, increasing costs and complicating the prospect of in-field repairs and maintenance.

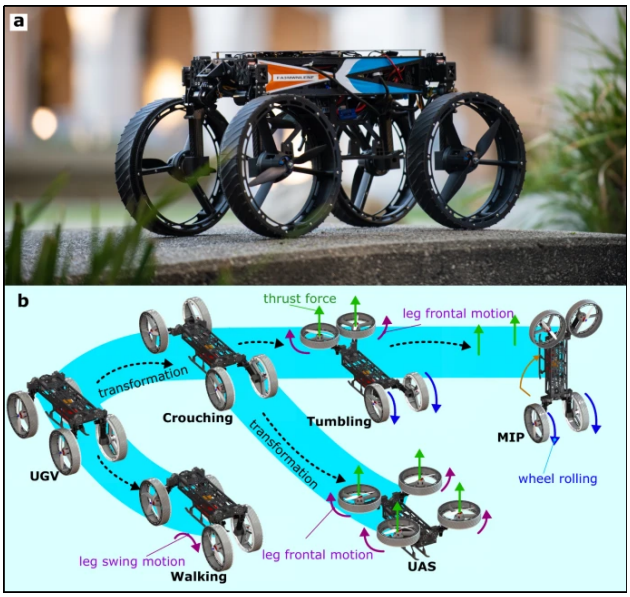

Figure 1: Image of the Caltech M4 robot along with illustrations of each of its modes of locomotion. [2]

1.2.2 Tsinghua University Multimodal Land-Air Robot

The Multimodal Land-Air Robot, like the M4, features a dual drivetrain system. It employs an automatic folding arm mechanism, with the propellers folding out from the main body, as illustrated in Figure 2 below. This approach also allows for an integrated suspension system to be built into the chassis, improving the ability of the robot to drive over uneven terrain [4]. However, the platform is extremely heavy, weighing in at 25kg, and features a large footprint of 1.25m by 1.25m

when flying, limiting its portability and its ability to access confined spaces. In addition, like the M4, its reliance on multiple actuators and its relatively complex design complicate repairs and maintenance.

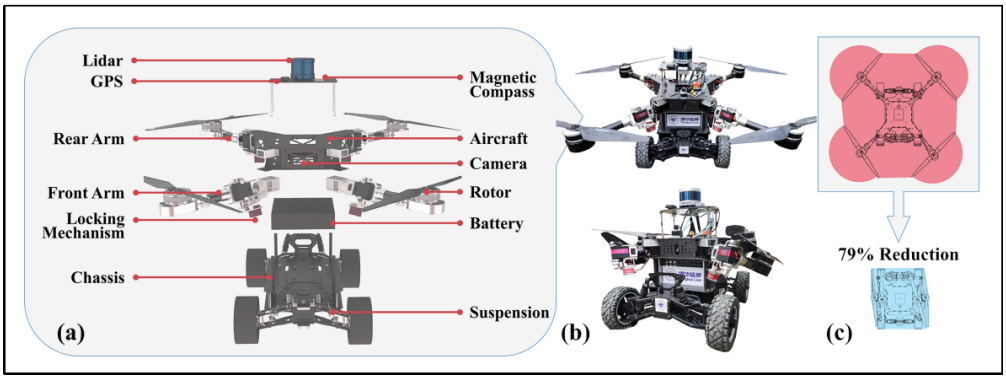

Figure 2: Image of the Multimodal Land-Air Robot from Tsinghua University, showcasing the differences in its form factor between the driving and flying mode. [3]

1.2.3 Virginia Tech LaMSA Robot

In contrast, Virginia Tech's LaMSA robot takes a very different approach. Its design uses a shape memory alloy (SMA), a material that is capable of returning to its original shape after being heated. By strategically applying heat to these SMAs, the shape of the alloy can be changed, lifting or lowering the wheels depending on what is needed, as shown in Figure 3 below. Using SMAs allows for extremely rapid morphing, achieving mode switching in just 50 milliseconds [4]. While this design simplifies the transformation mechanism significantly, the scale of the design is very small, limiting battery size and thus the length of a mission, as well as payload capacity for sensors and cameras.

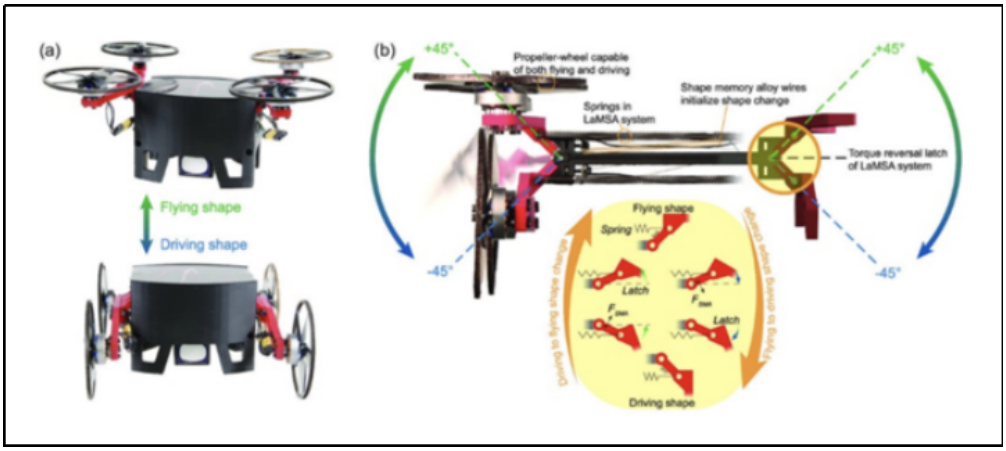

Figure 3: Image of the Virginia Tech's LaMSA, showcasing the differences in its form factor between the driving and flying mode. [4]

1.3 The L.I.F.T. Platform: A Unified Solution for Aerial and Ground Mobility

Our project aims to overcome the limitations of traditional single-mode robotic systems while advancing a new approach to multimodal mobility that prioritizes simplicity, cost-effectiveness, and field readiness. Central to our concept is using a single integrated drivetrain that actuates both flight and ground movement.

As shown above, existing land-air multimodal robots frequently rely on the use of multiple independent drivetrains [5]. While this can allow for optimized performance within individual locomotion modes, it significantly increases weight, mechanical complexity, and maintenance overhead. Most notably, in these systems, the inactive components of one mode become dead weight during the other. This can have a detrimental impact on endurance and performance, especially during flight, where weight is a critical constraint.

To overcome these challenges, we have developed the Land and Integrated Flight Transition (L.I.F.T.) Platform—a hybrid robotic system that uses the same motors to drive both the wheels and propellers, simplifying the design, lowering costs, and increasing the feasibility of in-field maintenance and repairs.. In ground mode (left side of Figure 4), the platform operates as a rugged off-road vehicle with a high-clearance chassis and long wheelbase. In flight mode (right side), it transforms into a quadcopter with vertical takeoff and landing capability. The rationale behind the design choices is discussed in greater detail in Section 2.

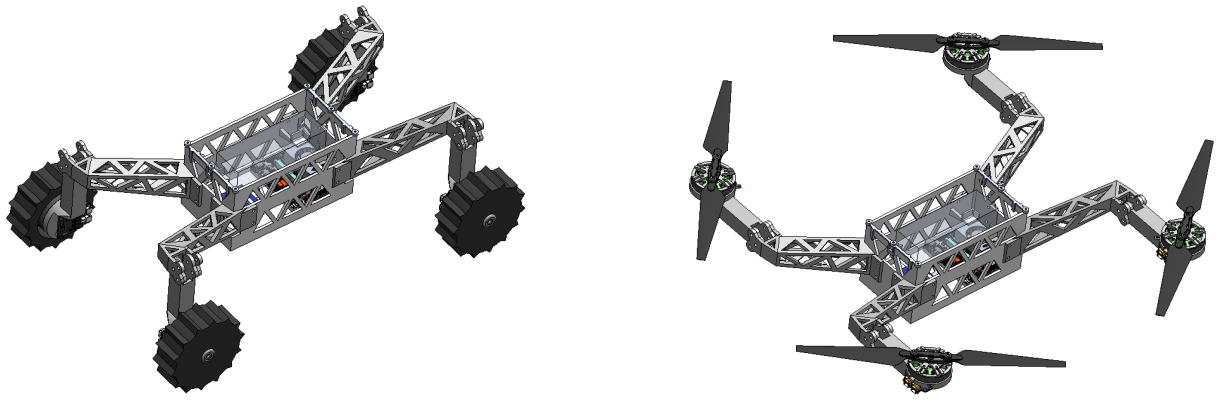

Figure 4: CAD render showing the L.I.F.T. Platform in both its driving (left) and flying (right) configurations.

1.4 Assessing Real-World Impact: Search-and-Rescue as a Case Study

The L.I.F.T. Platform offers potential applications across a variety of sectors, including last-mile delivery, infrastructure inspection, and military reconnaissance. In this section, we present search-and-rescue as a representative use case to illustrate the platform's advantages in scenarios that demand rapid deployment, adaptability, and cost-effectiveness.

Disaster zones are often hazardous and unpredictable, making it both risky and inefficient to rely solely on human responders. As a result, robotic systems are increasingly being integrated into disaster relief operations to help assess damage, locate survivors, and reduce human exposure to life-threatening conditions. Examples include Boston Dynamics' Spot, which was deployed at the Fukushima nuclear disaster site in 2011 [6], and the RS3 robotic firefighting vehicle used by the Los Angeles Fire Department [7]. Beyond physical danger, disaster relief efforts are often hindered by logistical bottlenecks and communication breakdowns among the many agencies involved, making it difficult to gain a clear situational overview and coordinate effectively [8]. Robotic systems can play a key role in addressing these challenges by providing timely, on-the-ground data to inform rescue efforts.

However, disaster environments are not only dangerous but also highly complex to navigate, featuring rubble, large obstacles, confined spaces, and varied terrain. These environments pose significant challenges to traditional single-mode robotic systems, which may become stuck or be unable to complete their missions. This creates a strong case for multimodal robots, which can

dynamically adapt their method of locomotion to suit the environment—flying over impassable obstacles and driving through tight or cluttered spaces.

The L.I.F.T. Platform builds upon these capabilities by providing dual-mode mobility, low cost, and a simplified, modular construction that allows for in-field repairs. In flight mode, it can rapidly survey large areas to assess damage and locate potential survivors, while its driving mode allows for efficient ground traversal over longer distances and entry into confined spaces, such as collapsed buildings. Moreover, the L.I.F.T. Platform's relatively low cost reduces barriers to access and makes it feasible to deploy multiple units in a single area, allowing for simultaneous data collection from different vantage points. Its modular 3D-printed design further supports rapid field repairs, minimizing downtime and reducing the need for specialized equipment.

By combining versatility, affordability, and rapid deployment, the L.I.F.T. Platform contributes a promising design approach to the field of disaster response robotics, offering a robust solution for improving efficiency and safety in search-and-rescue missions.

2. System Design and Prototyping

2.1 Design Objectives and Constraints

Designing a single drivetrain robot capable of both flying and driving poses several challenges. In this section, we discuss the design constraints that guided the development of our prototype for the L.I.F.T. Platform. These constraints fall into three key categories:

1. **Physical Constraints:** The platform must be compact enough to navigate tight spaces, lightweight enough to fly efficiently [9], and rugged enough to handle uneven and potentially hazardous terrain. Designing for modularity and ease of repair was also a crucial consideration.

2. **Electrical Constraints:** As a single drivetrain system, the motors must perform across two entirely different operating regimes: high torque for driving and high speed for flight.

3. **Logistical Constraints:** All development had to be completed within a nine-month timeline using primarily 3D-printed parts and off-the-shelf components.

In light of these constraints and the relatively short timeline for the project, we focused our development efforts on validating and de-risking our design concept in three core areas:

1. **Prototype Development:** Establishing the feasibility of a low-cost single drivetrain multimodal robot by using 3D-printed, field-serviceable components.

2. **Ground Autonomy:** Demonstrating autonomous navigation in both simulations and hardware tests using Model Predictive Control (MPC).

3. **Flight Demonstration:** Verifying structural readiness and flight stability through tethered joystick-controlled tests.

The most technically challenging of these constraints was selecting a motor that could meet the demands of both locomotion modes. The following section explores this challenge and the design decisions it drove in greater detail.

2.2 Balancing Torque and Speed Is Critical to Single-Motor Multimodal Performance

Motor selection presents a unique challenge in the design of multimodal robots, as the flight and ground modes demand fundamentally different performance characteristics from the motors. In the case of flight, high rotational speeds with low torque are desirable to enable effective lift with smaller propellers, while the driving mode requires slower speeds with high torque to ensure traction on rough terrain. For a system with multiple drivetrains, this tradeoff can be avoided. However, in our case, where a single drivetrain powers both flight and driving, motor selection is critical. Since torque and speed are inversely related in electric motors (see Figure 5), identifying a motor that performs well in both modes is inherently challenging.

We ultimately selected the MJBots MJ5208 brushless motor—a 330 KV motor with 1.7 Nm peak torque and a maximum rotational rate of 7500 RPM—as the primary actuator for both flight and ground modes. While this motor is well suited for driving due to its high torque, its relatively low KV rating (which determines the rotational rate per volt of input) limits its rotational speed, making it less ideal for flight with smaller propellers. We also considered alternative motor options that offered a better balance of torque and speed, which could have improved flight efficiency without compromising ground performance. However, these alternatives introduced integration risks and would have required the development of low-level motor interfaces. In contrast, the MJ5208 was already well supported by existing infrastructure within the MPC Lab, allowing us to reduce risk and focus our efforts on system-level integration and validation.

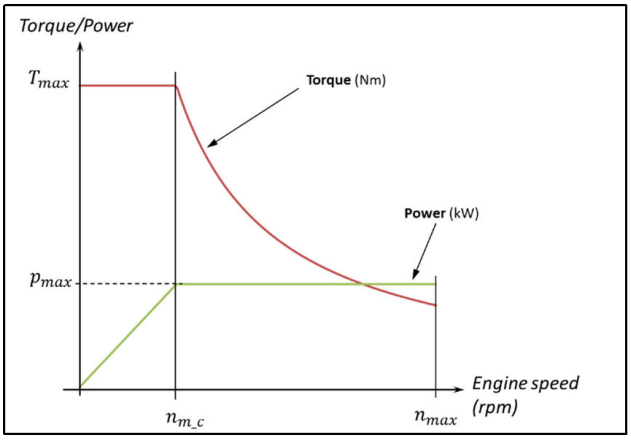

Figure 5: Plots showing the relationships between torque and power vs rpm for a standard electric motor. [10]

To compensate for the lack of speed, we implemented two key design choices. First, we

increased our input voltage to the motors. Since the rotational rate of an electric motor scales with voltage, we decided to move from a 16.8V 4S LiPo battery to a 25.2V 6s LiPo battery, increasing the achievable speed of rotation. Second, we chose to use relatively large 15” diameter propellers to maximise thrust output. Together, these design decisions were crucial to achieving a viable thrust-to-weight ratio for flight without compromising performance on the ground.

2.3 System Architecture Selection & Tradeoff Analysis

At the beginning of the project, we considered several different vehicle architectures to meet our development objectives. These candidate designs varied primarily in how they reconfigured their mechanical structure to support both locomotion types. After completing our initial brainstorming, we narrowed our focus to three primary options, shown in Figures 6-8 below. The first design was the smallest and simplest of the three. It used a slightly faster 1250 KV motor to support the use of smaller propellers, achieving a more compact form factor. Mode transition was achieved by manually reorienting the motors along sliding tracks on the arms. The second design, which was ultimately used as the basis for our final design, used a single folding arm to switch between configurations. Finally, the third design used a two-point geared rotation method that allowed the arms to be folded into the main body. While this enabled a highly compact stowed configuration, it added significant mechanical complexity.

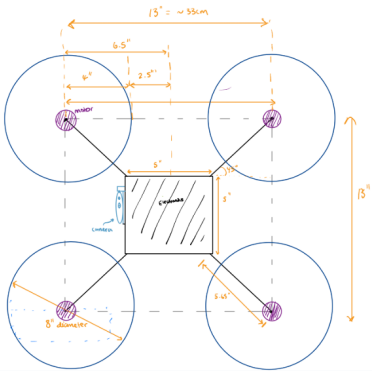

Figure 6: Sketch of Design 1, showing its simple rigid body design.

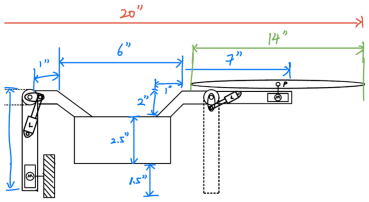

Figure 7: Sketch of Design 2, showing its folding arm mechanism.

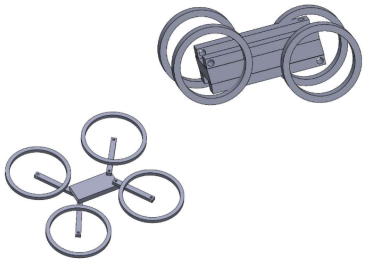

Figure 8: Simple CAD of Design 3, showing the arms fold into the body.

To systematically compare these architectures, we used a Pugh Matrix (Table 1), evaluating each design against a range of criteria aligned with our design goals. The first set of factors—such

as mass, size, and endurance—are quantitative and were measured or calculated directly from design specifications. The second set, such as ease of fabrication and novelty of the design, were qualitatively assessed and assigned scores based on group consensus, with 1 representing the best performance and 3 the worst. Based on this evaluation, we selected Design 2 as the most balanced and practical option for further development.

Table 1: Pugh Matrix used for evaluating competing design decisions.

Criteria	**Unit**	**Design 1** Rigid Body	**Design 2** Single Folding Hinge	**Design 3** Double Folding Hinge
Mass of System	kg	2.630	3.44	3.3
Maximum Size of System	m×m	0.2×0.2	0.48×0.48	0.48×0.48
Minimum Size of System	m×m	0.2×0.2	0.15×0.25	0.25×0.48
Maximum Speed (Driving)	m/s	7.58	6.1	7.1
Number of Moving Parts		0	5	9
Endurance Time (Driving)	hr	2	1.2	1.3
Endurance Time (Flight)	min	18	10	12
Assembly Time		2	1	3
Ease of Fabrication		1	2	3
Rigidity of Structure		1	2	3
Ease of Sandproofing		2	2	2
Cost		2	2	1
Modularity		3	3	3
Speed of Conversion		3	1	2
Novelty of Design		3	2	1
Sum of evaluation		**17**	**15**	**18**

2.4 Thrust Testing and Motor Performance Analysis

Validating the performance of our motors was a critical step in our design process for the L.I.F.T. Platform. Because the MJ5208 motors lacked thrust data in drone configurations, we conducted a number of thrust tests with different combinations of input voltages and propeller sizes to assess lift capability and inform our system's mass budget. Our tests included both 13-inch and 15-inch propellers powered by a 24V DC power supply, as well as a 15-inch setup driven by a 6S LiPo battery to simulate flight conditions. An image of our testing setup can be seen below in Figure 9.

The results of these tests, shown in Figure 10, showed that the 15-inch propellers paired with the 6S battery produced the highest thrust, achieving a maximum thrust of approximately 15N, or 1.5kg-f, at 5400 RPM. With validated thrust data, we were able to confirm that the motor could meet our performance needs, allowing us to move forward with our design. This data was also crucial for implementing our flight controller (discussed in greater detail in Section 4), where we performed a polynomial fit to accurately model the relationship between motor RPM and generated thrust. This fitted equation was then integrated into our flight controller to enable more precise thrust prediction and improve vehicle stability during flight.

Figure 9: Photo showing our thrust testing stand with a motor and propellers attached.

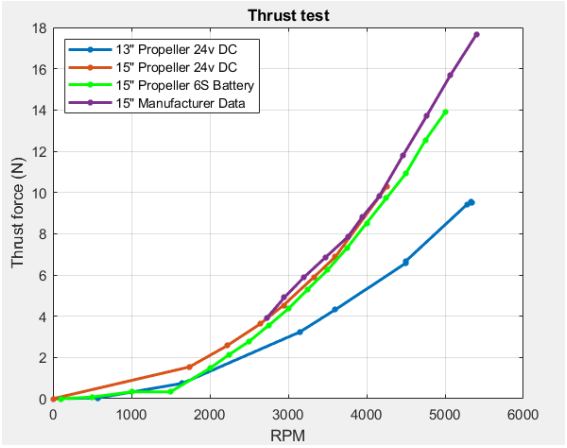

Figure 10: Thrust vs RPM performance for different tests

With this data, we were able to establish a mass budget for our system (Table 2), which defined the upper bound of system mass and guided our structural optimization efforts.

Table 2: Full system mass budget, with a final thrust-to-weight ratio of 1.6:1.

Category	Part	Mass (g)	Qty	Total Mass (g)
System Structure				
	Body Frame (ABS)	230	1	230.0
	Long Arm (ABS)	61	4	244.0
	Short Arm / Motor Mount (ABS)	40	4	160.0
	15x5.5” Propellers + Fasteners	26	4	104.0
Electronics & Power				
	NVidia Jetson Board w/ SSD	177	1	177.0
	MJ5208 BLDC Motor	189	4	756.0
	Motueus r4.11 Motor Controller	15	4	60.0
	Arduino Nano	5	1	5.0
	Power Distribution Board	50	1	50.0
	6s 6000 mAh LiPo Battery	772	1	772.0
Miscellaneous				
	Harnessing (16 AWG) + Connectors (XT30)	85	1	85.0
	Fasteners (Nuts, Bolts, Pins, etc.)	100	1	100.0
			Total Estimated Mass (g)	**2743.00**
			Total Mass + 15% Contingency (g)	**3154.45**
			Maximum Thrust (g)	**5200**
			Thrust-to-Weight Ratio	**1.648**

2.5 Mass Optimization of Structure

Mass optimization is an incredibly important process when designing any vehicle capable of flight, as excess weight directly affects the endurance and maneuverability of the system. With our mass budget completed, we were able to begin the process of mass optimization, seeking to reduce system mass without compromising structural integrity. This was achieved through a combination of different methods. First, targeted cutouts were added to both the supporting arms and the main

chassis. Second, where feasible, previously multi-part assemblies were consolidated into single components, reducing mass and design complexity. Third, electronic components that were initially arranged on a single level were stacked across two levels, allowing for more effective use of the volume of the center box and reducing unnecessary mass. Finally, the entire system is manufactured using 3D printing, allowing for targeted optimization of printing density and wall thickness based on the loads placed upon each part. Figures 11 and 12 show before-and-after comparisons of the structural design, highlighting the 17% weight reduction achieved through these optimizations.

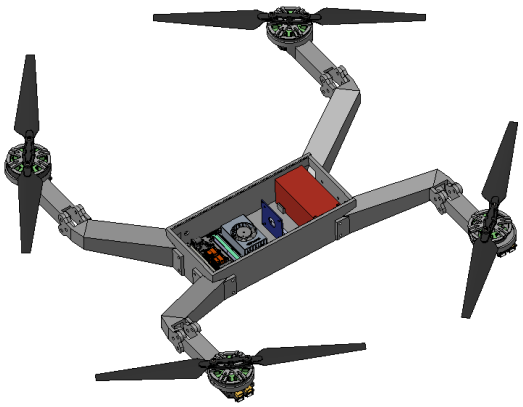

Figure 11: Original design without mass reduction.

Figure 12: Optimized design with reduced weight.

To ensure the structure continued to meet strength and load-bearing requirements after weight reduction, we conducted Finite Element Analysis (FEA) on both the arms and central chassis of the system. The analyses were done through SolidWorks Simulation with fixed points at the arm-to-body connections, and modeled force from thrust acting on the ends of the arms. The results of this analysis, shown in Figures 13 and 14, confirmed that the mass-optimized components could withstand expected loading conditions. For Arm 1, we paid particular attention to its deformation, as excessive deflection could misalign the thrust vector from the propellers and compromise flight stability. Our FEA simulations predicted a tip displacement of $1.8mm$, corresponding to less than 1.2% of its total length, well within acceptable limits for stable operation. We also evaluated the structural integrity of the main chassis under peak loading conditions. The analysis shows that the maximum Von Mises stress is $3.49 \times 10^5 Pa$, significantly below the ultimate tensile strength of ABS material ($4 \times 10^7 Pa$), indicating a strong safety margin.

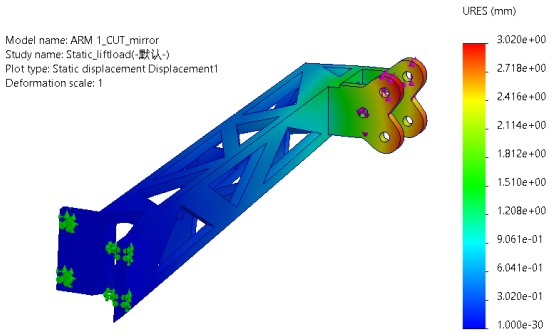

Figure 13: Arm I FEA result.

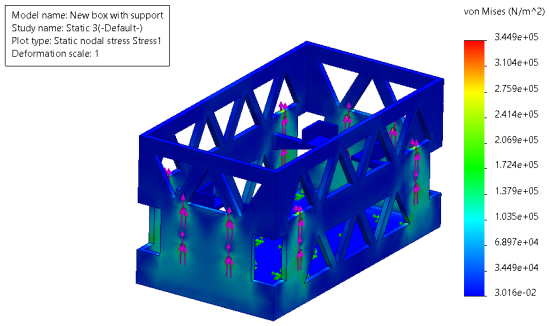

Figure 14: Central Body FEA result.

2.6 System Architecture and Final Design Overview

The final design of the L.I.F.T. Platform integrates all critical subsystems required for both aerial and ground mobility. An exploded view of the complete assembly is shown in Figure 15.

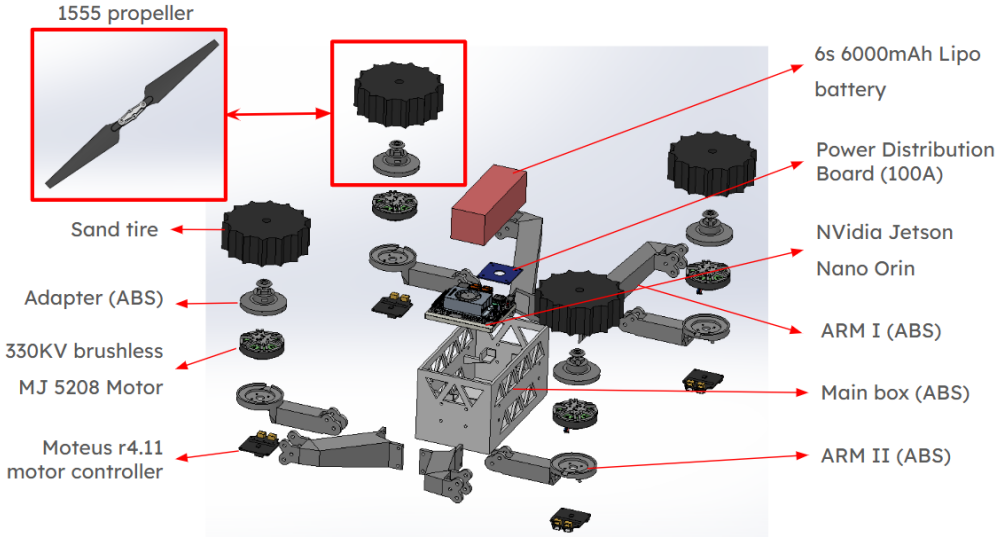

Figure 15: Exploded view of the L.I.F.T. Platform with all components labelled.

The core subsystems of the L.I.F.T. Platform include:

1. **Drivetrain System:** The drivetrain consists of four 330KV brushless electric motors that drive the system in both ground and flight modes.

2. **Mode-Specific Locomotion Components:** In flight mode, the motors spin large 15-inch propellers to provide vertical lift. In ground mode, the same motors are fitted with four large off-road wheels with flanges that maintain traction on uneven terrain.

3. **Control System:** The control module includes four motor control boards, an NVIDIA Jetson for onboard computation, and an Arduino Nano that processes IMU readings. This system handles real-time sensor data acquisition and motor command execution.

4. **Power Supply Module:** Power is provided by a 6000mAh 6S LiPo battery, which feeds into a power distribution board that supplies power to each motor. A voltage regulator steps down the battery voltage to 19V for the NVIDIA Jetson.

5. **Structural Frame and Morphing Mechanism:** The physical frame is composed of three main components: Arm I, Arm II, and the central chassis. These form the core load-bearing structures, ensuring stability, durability, and ease of repair. The morphing mechanism, created by combining geometric design and binding barrel screws, enables the robot to seamlessly transition between driving and flight modes via a simple rotating motion.

3. Driving Control System

3.1 MPC Provides a Predictive, Constraint-Aware Framework for Vehicle Control

One of the core technical goals for our project was to demonstrate that our ground vehicle was capable of autonomously navigating a known environment by using Model Predictive Control (MPC). At its core, MPC is a control strategy that determines the optimal control inputs to apply to a system by making predictions about the system's future behavior over a fixed time horizon. At each timestep, it solves an optimization problem that minimises a cost function subject to constraints. By carefully defining both the cost function and the constraints, we can formulate an optimization problem that accurately captures both the desired objective and the limitations of the system. This allows the controller to plan ahead, while respecting physical constraints, like the vehicle dynamics or obstacle locations.

3.2 Formulating the Control Problem

We frame the ground navigation task as a constrained optimization problem solved at each timestep. The objective is to minimize distance from a desired setpoint while satisfying physical and environmental constraints. At a high level, our optimization problem can be formulated as follows:

$$
\begin{aligned}
\min_{q_k, u_k} \quad & (q_N - q_f)^\top P (q_N - q_f) + \sum_{k=0}^{N-1} \left[(q_k - q_f)^\top Q (q_k - q_f) + u_k^\top R u_k \right] \qquad (1) \\
\text{subject to:} \quad & q_0 = q(0), \\
& q_{k+1} = f(q_k, u_k), \\
& q_k \in \mathcal{Q}, \\
& u_k \in \mathcal{U}, \\
& q_k \notin \mathbb{O},
\end{aligned}
$$

where q_k is the robot's state at time step k (position and orientation); u_k is the control input at time step k (wheel angular velocities); $f(\cdot)$ defines the system dynamics; P, Q, and R are weighting matrices that shape the behavior (e.g., penalize deviation, input magnitude, etc.); $\mathcal{Q}$,

$\mathcal{U}$, and $\mathbb{O}$ are sets that describe the feasible state space, input bounds, and obstacle positions, respectively; and N is our prediction horizon.

Our state vector, q_k, consists of three parameters: x, y, and θ, which describe the vehicle's position and heading in the global frame. The control input, u_k, is comprised of two parameters: u_L and u_R, which describe the angular velocities of the left and right wheels measured in radians per second. This configuration is illustrated in Figure 16.

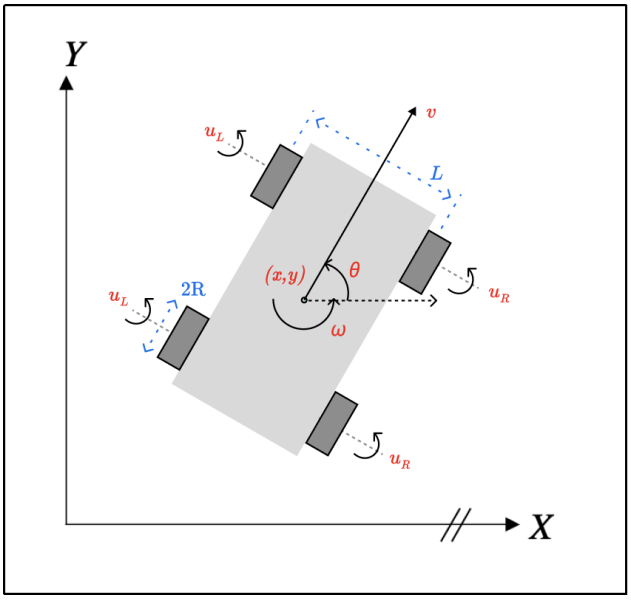

Figure 16: Diagram of the ground vehicle with labeled coordinates and dimensions.

3.3 Vehicle Dynamics Are Captured Through a Kinematic Skid-Steer Model

One of the most crucial aspects of effectively implementing MPC is accurately modeling the system dynamics. In our case, our ground vehicle is a four-wheeled skid-steer vehicle capable of independently controlling its left and right wheels. This allows the robot to turn in place by counter-rotating the left and right wheels.

We initially attempted to model the vehicle dynamics through a full dynamic model that used wheel torques as control inputs. The model aimed to fully capture all of the forces acting on the vehicle by working up in several stages. First, wheel angular velocities were computed based on the applied torques and the wheel dynamics. These angular velocities were then used to compute wheel-ground contact forces using the Pacejka Tire Model. The resulting contact forcces could then be mapped to net forces and moments acting on the vehicle body. Finally, these forces could

be used to calculate the vehicle's linear and angular accelerations.

While this model offered the potential of a high-fidelity representation of the system's physical behavior, it introduced significant complexity and was highly sensitive to uncertain parameters like friction coefficients and Pacejka Model parameters. After encountering various issues with the numerical stability of the model, we opted to transition to a simpler model based on a two-wheel differential drive mobile robot. This model offered significantly improved stability and lower complexity, making it better suited for real-time control and more robust within the scope of our project.

Our simplified motion model is derived from standard kinematic formulations for differential-drive robots [11]. In this formulation, the robot is modeled as a rigid body moving on a plane with two independently driven wheels of radius R mounted symmetrically on either side of its centerline. The key assumptions are: (1) planar motion, (2) no slip at the wheels (pure rolling), and (3) rigid body dynamics. The derivation of the model is described below.

Let $\dot{\phi}_l$ and $\dot{\phi}_r$ represent the angular velocities of the left and right wheels, respectively. Under the no-slip assumption, the linear velocities of each wheel are:

$$v_l = R\dot{\phi}_l, \quad v_r = R\dot{\phi}_r$$

The linear velocity v of the robot's center (midpoint between the wheels) is the average of the two:

$$v = \frac{v_r + v_l}{2} = \frac{R}{2}(\dot{\phi}_l + \dot{\phi}_r)$$

The angular velocity ω of the robot about its yaw axis is the difference in wheel speeds divided by the wheelbase L:

$$\omega = \frac{v_r - v_l}{L} = \frac{R}{L}(\dot{\phi}_r - \dot{\phi}_l)$$

Let the robot's pose in the world frame be represented by (x, y, θ). The time derivatives of x and y are given by projecting the linear velocity along the robot's heading direction:

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = \omega$$

Substituting the expressions for v and ω yields the final continuous-time kinematic model:

$$\dot{x} = \cos(\theta)\left(\frac{R}{2}(\dot{\phi}_l + \dot{\phi}_r)\right)$$
$$\dot{y} = \sin(\theta)\left(\frac{R}{2}(\dot{\phi}_l + \dot{\phi}_r)\right)$$
$$\dot{\theta} = \frac{R}{L}(\dot{\phi}_r - \dot{\phi}_l)$$

This kinematic model provides a balance between simplicity and predictive fidelity, forming the basis for our Model Predictive Control (MPC) framework. It enables accurate real-time state prediction while avoiding numerical instability.

3.4 Collision Avoidance Using OBCA

One of the key challenges in optimization-based trajectory planning is the question of how to encode obstacle avoidance constraints without sacrificing real-time performance or robustness. Many existing approaches rely either on linear approximations of obstacles, which may be inaccurate, or use mixed-integer formulations, which can significantly increase run-time and make real-time computation infeasible [12].

In order to address these issues, we implemented our collision avoidance constraints using the Point-Mass Optimization-Based Collision Avoidance (OBCA) framework [12]. This approach allows for obstacle-avoidance constraints to be reformulated into smooth, differentiable constraints that can be handled efficiently by traditional solvers, enabling us to generate dynamically feasible, collision-free trajectories in real time. The OBCA approach introduces a new optimization variable, $\lambda_k^{(m)}$, and reformulates the generic obstacle avoidance constraint $q_k \notin \mathbb{O}$ into the following set of constraints [12]:

$$\left(A^{(m)}p_k - b_k^{(m)}\right)^\top \lambda_k^{(m)} - d_{\min} > 0, \tag{2}$$

$$\left\|\left(A^{(m)}\right)^\top \lambda_k^{(m)}\right\| \leq 1, \tag{3}$$

$$\lambda_k^{(m)} \geq 0, \tag{4}$$

$$\forall k = 0, 1, \ldots, N$$

$$\forall m = 1, 2, \ldots, M$$

where $A^{(m)}$ and $b^{(m)}$ are the matrices that define the obstacle given by: $\mathbb{O}^{(m)} = \{y \in \mathbb{R}^n : A^{(m)}y \leq b^{(m)}\}$; p_k describes the robot's current position, $[x, y]$; $\lambda_k^{(m)}$ is the dual variable associated with obstacle $\mathbb{O}^{(m)}$ at time step k; and d_{min} is a design parameter that sets the minimum allowable distance between the vehicle and the obstacle. In our case, d_{min} was set to $1.5\times$ the robot's width to ensure a sufficient buffer.

3.5 MPC Validation Through Simulation and Hardware Testing

3.5.1 Simulation-Based Testing: Validating Control Logic in Ideal Conditions

With our model implemented, we were able to begin testing its performance under different conditions. The first step of our validation process was to conduct software-in-the-loop testing using a custom simulator. This simulator modeled ideal operating conditions—i.e., no model mismatch or noise—and allowed us to verify the basic operation of the controller. These tests have confirmed that the controller is generally functioning as intended, successfully planning and tracking paths in the presence of obstacles, as shown in the examples in Figure 17 below.

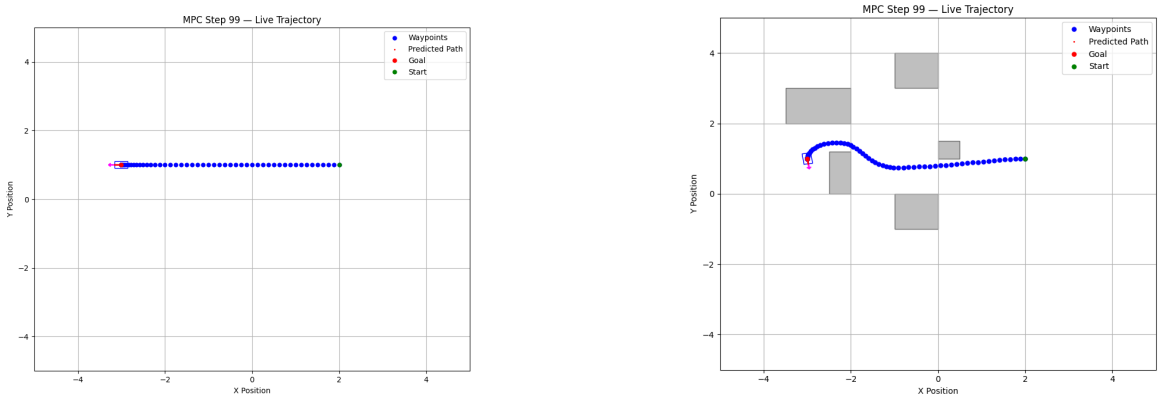

Figure 17: Plots showing the planned trajectory of the robot both with and without obstacles.

That said, we also observed several issues that require refinement. Namely, as a result of the non-convex nature of the problem formulation, the controller occasionally became trapped in local minima, stopping "near" the goal state instead of actually reaching it, or generating suboptimal trajectories with unnecessary motion, as shown in Figure 18.

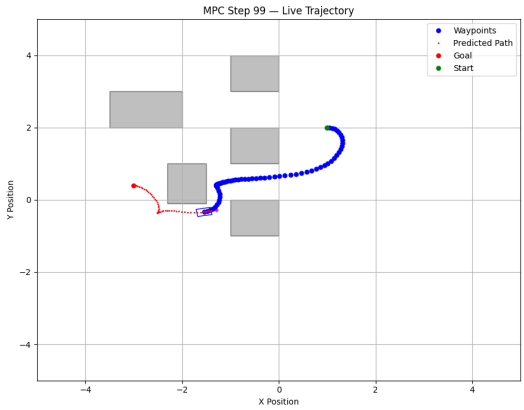

Figure 18: Plot showing a suboptimal planned trajectory by the robot.

3.6 Hardware Testing: Verifying Real-World Performance

To enable hardware testing, we implemented a ROS2-based control architecture that integrates OptiTrack motion capture for state feedback and supports real-time control. A summary of the ROS2 network and data flow is provided in Figure 19.

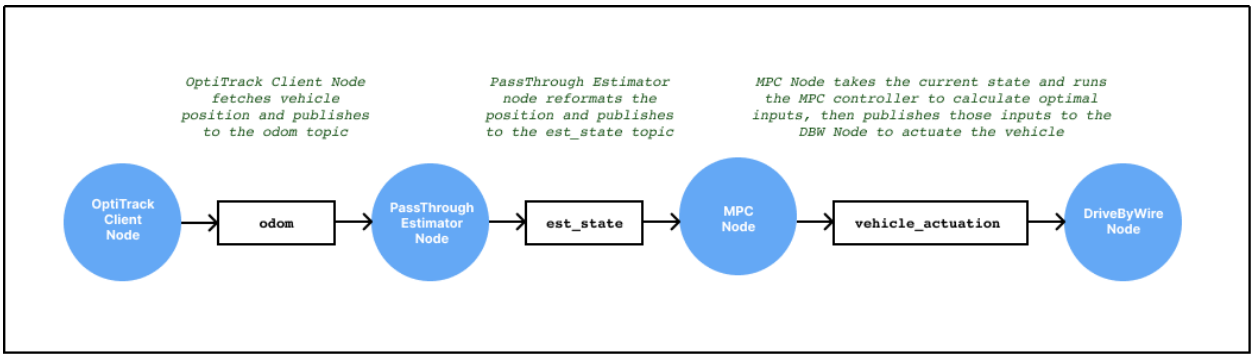

Figure 19: Graphical representation of our ROS2 network, showing the flow of information between nodes and topics.

Our hardware tests (an example of which is shown in Figure 20) confirmed that our controller could effectively model and control the behavior of our vehicle, planning and navigating a simple environment in real-time, both with and without obstacles. However, the controller still suffered from many of the setbacks of the simulated behaviours we saw, including sensitivity and suboptimal solutions.

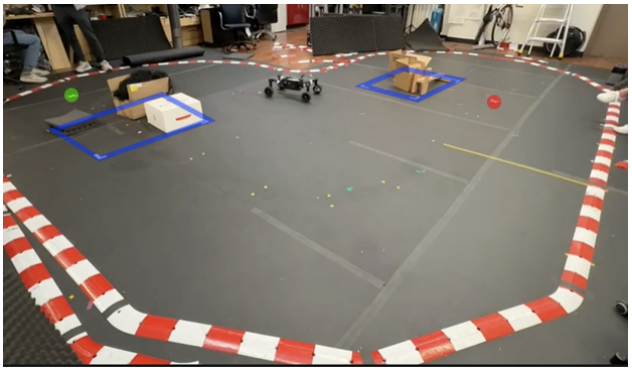

Figure 20: Photograph of the hardware test in progress with obstacles. Overlays show obstacle locations (blue), starting point (green), and goal position (red).

Our hardware tests provided valuable insight into the viability of the simplified two-wheel differential drive dynamic model used in the controller. As shown in Figure 21, the simulated and actual trajectories generally align well, indicating that the model captures the system's overall behavior on a high level. By comparing the evolution of the state variables over time (Figure 22), we can get a stronger sense of the accuracy of the model. While both simulations and hardware runs successfully reach the setpoint, the hardware does so roughly 50% more slowly and with significant divergence along the way. This suggests that although the model is sufficient for demonstrating

basic real-time control, it fails to capture certain dynamic effects, most notably during turning. These become especially apparent near the end of the trajectory, where the controller overestimates the vehicle's ability to make sharp turns, leading to overshoot. This behavior is consistent with the significant wheel slip observed during turns, which violates the no-slip assumption underlying the model.

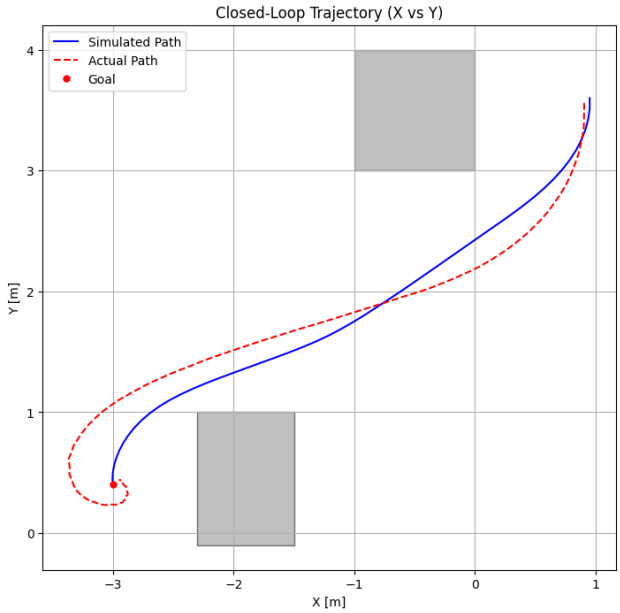

Figure 21: Plot showing closed-loop trajectory planned by the controller, comparing simulated and actual results.

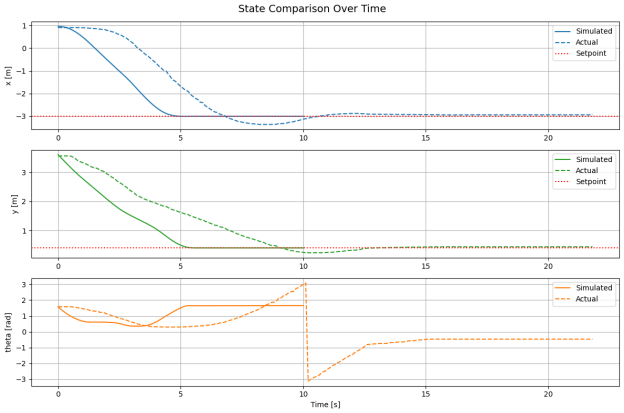

Figure 22: Plots showing the evolution of state variables throughout the trajectory, comparing simulated and actual results.

4. Flight Control

4.1 Flight Control System Architecture

Our flight control system is designed to enable stable and responsive quadcopter flight by integrating sensor data, estimating the vehicle's orientation, and controlling motor commands through a cascaded control strategy. This section introduces the architecture and main components of the system, highlighting how data flows from sensors to actuators (see Figure 23, below).

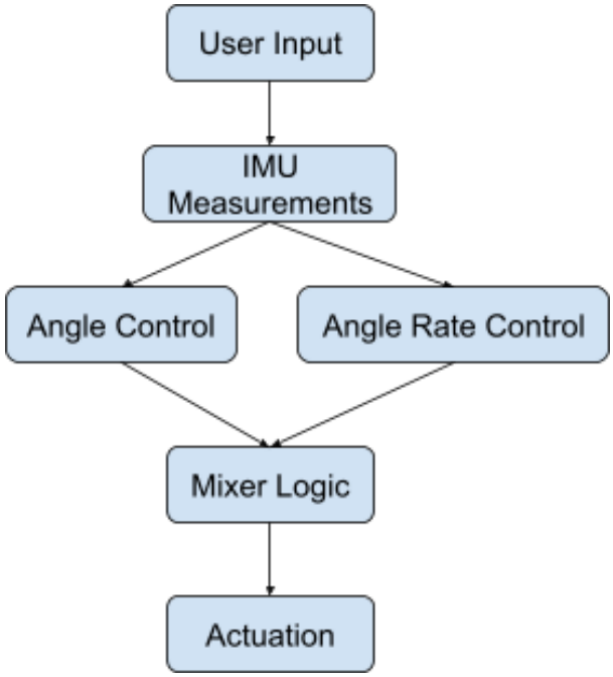

Figure 23: Flight Control Logic Chart

4.1.1 Sensor Data Processing

The quadcopter is capable of estimating its orientation (roll, pitch, and yaw) using onboard attitude sensors such as an Inertial Measurement Unit (IMU). In our robot, we use the Arduino Nano 33 BLE Sense Board, which includes an LSM9DS1 IMU module featuring a 3D accelerometer, a 3D gyroscope, and a 3D magnetometer. However, for this project, we relied on the OptiTrack motion capture system due to its ease of integration and superior reliability.. This allowed us to bypass the need for developing more complex sensor fusion algorithms, which would otherwise be required

to obtain accurate and low-noise orientation estimates from the onboard IMU.

The OptiTrack system onsists of multiple motion tracking cameras installed around the perimeter of the lab space. Reflective markers placed on the robot (shown in Figure 24) allow the cameras to track its position and orientation in real time.

Figure 24: Reflective Markers for Optitrack Motion Capture

One important aspect of working with a motion capture system like this is that the raw sensor data often includes small errors or biases that accumulate over time. To address this, we apply a simple calibration routine at startup. During this process, the quadcopter is left stationary, and we collect a series of sensor readings over a fixed time window. Since the quadcopter is not moving, the expected value for the readings should be zero and any non-zero value can be assumed to be a result of some bias. For each axis the bias can then be computed as:

$$\text{bias}_i = \frac{1}{N} \sum_{k=1}^{N} \text{sensor}_i[k] \tag{5}$$

This estimated bias can then be subtracted from subsequent readings:

$$\text{sensor}_i^{\text{corrected}} = \text{sensor}_i^{\text{raw}} - \text{bias}_i \tag{6}$$

This calibration step ensures that the initial position and orientation of the vehicle are effectively zeroed, improving the consistency and reliability of downstream state estimates.

4.1.2 Cascaded Flight Control Design

Our flight controller adopts a cascaded control architecture, which separates the stabilization task into two nested loops: an outer loop for angle control and an inner loop for angular rate con-

trol. This layered structure improves system stability and simplifies controller tuning by isolating different dynamics at each level [13].

The outer loop is responsible for determining how fast the quadcopter should rotate in order to reach the desired orientation. In our system, we assume the target angles are zero (i.e., the quadcopter should maintain a level hover), but the structure supports arbitrary target angles if desired. The controller compares the estimated orientation with the desired angles. The error is scaled by a proportional gain (implemented as a time constant) to compute the desired angular velocities: p_{des}, q_{des} and r_{des}:

$$p_{\text{des}} = -\frac{1}{\tau_{\text{roll}}} (\Phi_{\text{est}} - \Phi_{\text{des}}) \tag{7}$$

$$q_{\text{des}} = -\frac{1}{\tau_{\text{pitch}}} (\theta_{\text{est}} - \theta_{\text{des}}) \tag{8}$$

$$r_{\text{des}} = -\frac{1}{\tau_{\text{yaw}}} (\psi_{\text{est}} - \psi_{\text{des}}) \tag{9}$$

where: Φ, θ, and ψ represent roll, pitch, and yaw, respectively; and the τ values are time constants set in the code—essentially acting as proportional gains in a PID controller.

The inner loop receives the desired angular rates from the outer loop and compares them with the actual angular rates (obtained from the OptiTrack system). It then computes the required angular accelerations to reach those desired rates in the same fashion as in the outer loop:

$$p'_{\text{des}} = -\frac{1}{\tau_{\phi}} (\Phi_{\text{meas}} - \Phi_{\text{des}}) \tag{10}$$

$$q'_{\text{des}} = -\frac{1}{\tau_{\theta}} (\theta_{\text{meas}} - \theta_{\text{des}}) \tag{11}$$

$$r'_{\text{des}} = -\frac{1}{\tau_{\psi}} (\psi_{\text{meas}} - \psi_{\text{des}}) \tag{12}$$

These angular accelerations are then used to compute the torques that must be applied to rotate the quadcopter at the desired rates:

$$\tau_{\phi} = J_{xx} \cdot p'_{\text{des}} \tag{13}$$

$$\tau_{\theta} = J_{yy} \cdot q'_{\text{des}} \tag{14}$$

$$\tau_{\psi} = J_{zz} \cdot r'_{\text{des}} \tag{15}$$

These torque values can then be sent to the motor mixing block, which distributes the forces among the four motors based on quadcopter geometry and direction of rotation.

4.1.3 Mixer Theory for Quadcopter in X Configuration

The mixer is the final stage of the flight control system, responsible for converting the desired total thrust and body torques (roll, pitch, and yaw) into individual thrust commands for each motor. This transformation is necessary because a quadcopter cannot directly control these forces independently—each motor contributes to all three rotational axes and the overall lift, depending on its location and rotation direction [14].

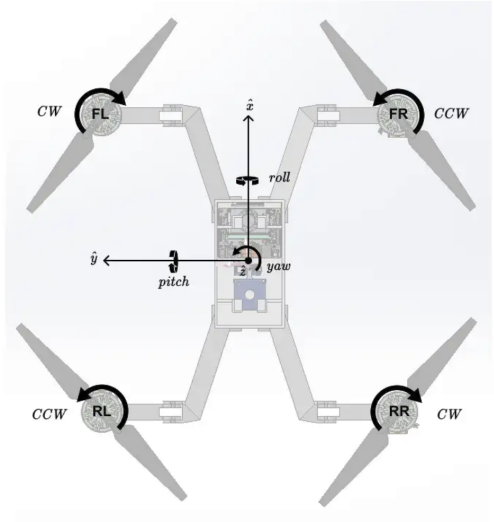

Figure 25: Quad X Configuration.

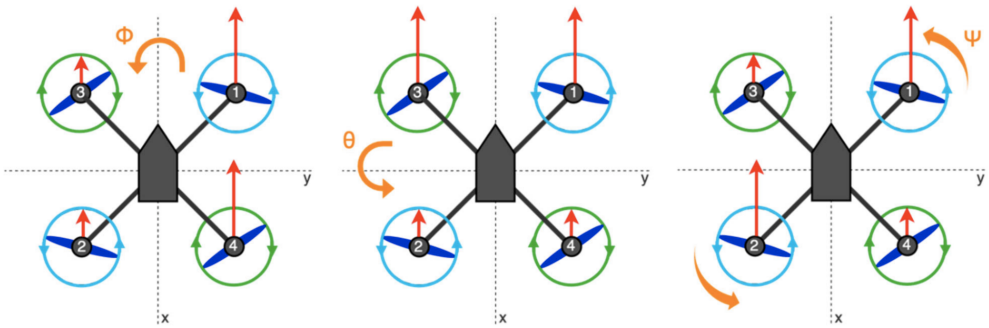

Figure 26: Effect on Drone's Behavior due to Different Motor Outputs. [15]

The mixer computes individual motor thrusts by solving a linear system that maps the desired total force and moments to the motor outputs. The input to the mixer includes: F_z, the total vertical thrust; τ_ϕ, the roll torque; τ_θ, the pitch torque; and τ_ψ, the yaw torque. For an X configuration, the

mixing matrix accounts for each motor's contribution to these quantities based on arm length, l, and torque constant, k (see the equation below [16]).

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} F_z \\ \tau_\phi / l \\ \tau_\theta / l \\ \tau_\psi / k \end{bmatrix} \tag{16}$$

Each motor's thrust is then converted to a motor speed and PWM command using motor-specific calibration curves modelled during our preliminary thrust testing.

4.2 Tethered Joystick Testing: Verifying Manual Control Over Flight Movements

Given the shortened project timeline, our primary goal in testing the flight controller was not to achieve high-performance flight, but to validate the mechanical readiness of the L.I.F.T. Platform for flight in general. To that end, we adopted an incremental testing strategy. Initial tests were performed without propellers to verify motor direction, control mapping, and emergency stop functionality. For live testing, the drone was tethered using equal-length bungee cords secured to fixed anchor points, physically constraining its flight altitude to 30–50 cm (Figure 27). A large safety net separated the test zone from the operator area, minimizing risk in the event of instability or uncontrolled motion. This testing approach allowed us to validate the flight controller's basic functionality in a controlled environment and identify potential issues like incorrect control mapping, motor response delays, or improper parameter settings.

4.3 Flight Testing Results

Our tethered flight tests successfully demonstrated that the L.I.F.T. Platform is mechanically capable of autonomous lift-off and that the onboard flight control system is capable of commanding thrust and initiating hover behavior. Upon startup, the quadcopter generated sufficient thrust to lift off the ground without manual input, confirming that the sensing, actuation, and control pipelines were functioning at a basic level. A snapshot from these tests is shown in Figure 28.

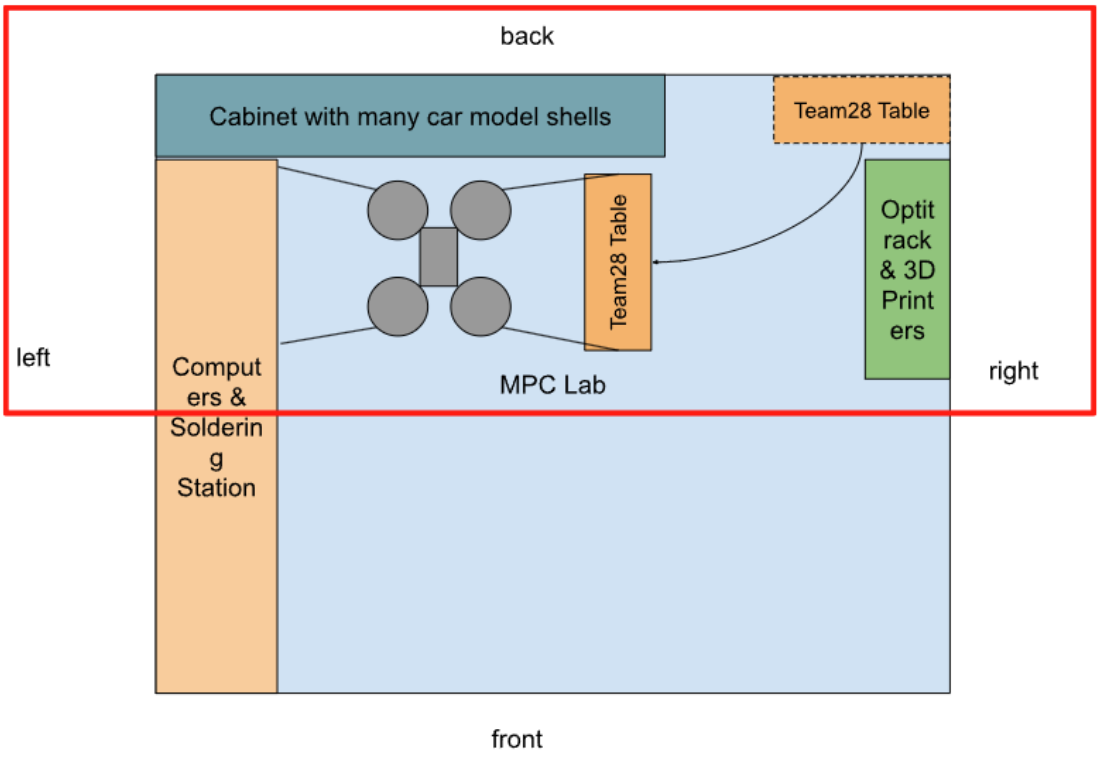

Figure 27: Schematic diagram of the test area.

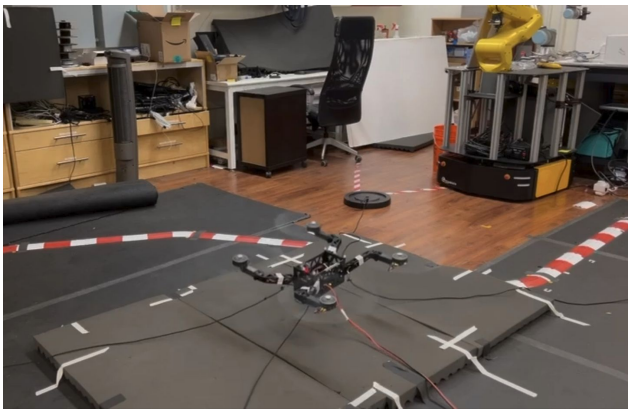

Figure 28: Photograph of tethered lift-off, showing quadcopter airborne before instability.

However, the system failed to maintain a stable hover. After initial liftoff, the vehicle exhibited increasing roll and pitch instability, ultimately leading to a crash into the foam padding below. Fortunately, no structural damage occurred, and all systems remained operational after the incident.

Analysis of the flight log data provides insight into the cause of instability. As shown in Figure 29, roll and pitch angles remained close to zero during the initial seconds of flight but

began diverging rapidly after lift-off. Notably, the roll angle reached approximately 0.15 radians (8.6 degrees), while pitch decreased sharply, indicating a loss of balance.

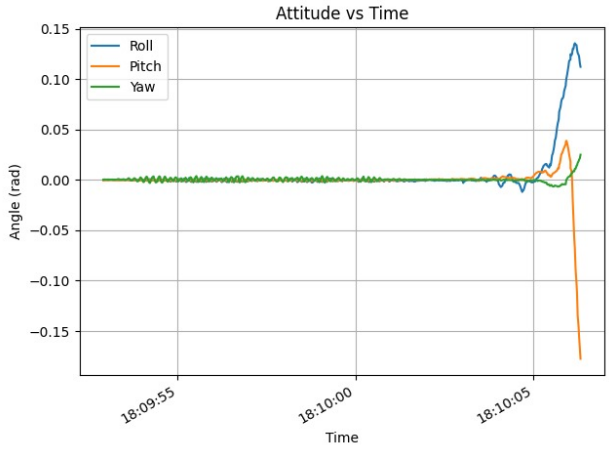

Figure 29: Attitude (roll, pitch, yaw) vs. Time during tethered flight test.

This behavior is further explained by the corresponding angular accelerations shown in Figure 30. Both roll and pitch experience sharp spikes in angular acceleration, with pitch reaching nearly -0.35 rad/s² immediately prior to the crash. These spikes suggest that the controller responded too aggressively to small disturbances, likely due to overly high gain values and inadequate damping in the control loop.

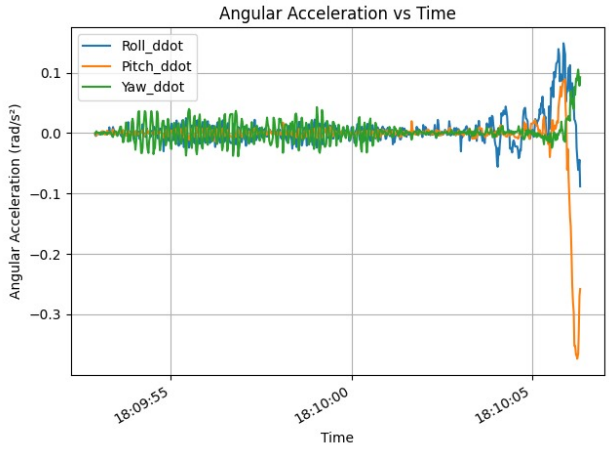

Figure 30: Angular acceleration vs. Time during tethered flight test.

To summarise, we believe the instability resulted from several compounding factors:

1. **Aggressive gain settings**, which led to overshoot and oscillatory response.

2. **Unmodeled asymmetries** in motor thrust or frame balance, causing uneven actuation.

3. **Simplified dynamics in the controller**, which did not fully account for motor delays or nonlinear effects near lift-off. Crucially, inspection of flight data revealed high lag times between commanded motor speeds and actual motor speeds.

Overall, while fully stable autonomous hover was not achieved, the testing successfully met the primary goal of demonstrating that the platform is capable of generating sufficient thrust for flight, that the structure can survive lift-off stresses, and that the control pipeline can command basic flight behavior. Given the time constraints of the project, achieving robust flight stability was beyond the scope of the final phase, but the work lays a strong foundation for future tuning and system improvements.

5. Conclusion

Navigating complex environments will increasingly require multimodal robotic solutions. While several such systems exist, their reliance on independent drivetrains adds cost, complexity, and mass to their designs.

The L.I.F.T. Platform offers a practical and streamlined approach to multimodal robotics by using a single drivetrain for both ground and aerial locomotion. Over the course of this project, we successfully demonstrated the feasibility of this design approach across three key areas: the physical assembly and integration of the system, its ability to perform autonomous driving using MPC, and its basic flight capabilities through tethered lift-off.

Looking ahead, the system provides a strong foundation for future development, including improving flight stability, enabling autonomous mode transitions via an actuator, and testing in more complex environments. With continued refinement, the L.I.F.T. Platform has the potential to support real-world use cases where adaptability, cost-efficiency, and rapid deployment are essential.

References

[1] K. N. et al., "Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots," *J. Field Robot.*, vol. 30, no. 1, pp. 44–63, 2013.

[2] R. N. A. R. E. Sihite, A. Kalantari and M. Gharib, "Multi-modal mobility morphobot (m4) with appendage repurposing for locomotion plasticity enhancement," *Nat. Commun.*, vol. 14, no. 1, 2023.

[3] X. Z. et al., "A multi-modal deformable land-air robot for complex environments," arXiv preprint arXiv:2210.16875, 2022.

[4] B. Wilcox and M. Bartlett, "Morphing drone through shape change for multimodal vehicles," Virginia Tech Intellectual Properties, Dec. 12 2024, [Online]. Available: https://vtip.technologypublisher.com/technology/55600.

[5] G. Y. et al., "Review of hybrid aquatic-aerial vehicle (haav): Classifications, current status, applications, challenges and technology perspectives," *Prog. Aerosp. Sci.*, vol. 139, p. 100902, 2023.

[6] B. Dynamics, "Spot in fukushima daiichi," Case Studies, 2022, [Online]. Available: https://bostondynamics.com/case-studies/spot-in-fukushima-daiichi/.

[7] L. A. F. Department, "Lafd debuts rs3: First robotic firefighting vehicle in the united states," News, Oct. 13 2020, [Online]. Available: https://lafd.org/news/lafd-debuts-rs3-first-robotic-firefighting-vehicle-united-states.

[8] L. N. V. Wassenhove, "Humanitarian aid logistics: Supply chain management in high gear," *J. Oper. Res. Soc.*, vol. 57, no. 5, pp. 475–489, 2006.

[9] QuadPartPicker, "How to estimate and calculate drone flight characteristics," QuadPartPicker News, May 20 2024, [Online]. Available: https://news.quadpartpicker.com/how-to-estimate-and-calculate-drone-flight-characteristics/.

[10] P. L. et al., "Multi-objective evolutionary design of an electric vehicle chassis," *Sensors*, vol. 20, no. 13, p. 3633, Jun. 2020, [Online]. Available: https://www.mdpi.com/1424-8220/20/13/3633.

[11] A. P. Allen, "Instantaneous center of curvature kinematics," https://www.cs.columbia.edu/~allen/F17/NOTES/icckinematics.pdf, 2017, lecture Notes, Columbia University, Department of Computer Science, Fall 2017.

[12] A. L. X. Zhang and F. Borrelli, "Optimization-based collision avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, May 2021.

[13] A. N. S. Bouabdallah and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, 2004, pp. 2451–2456.

[14] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton, NJ, USA: Princeton Univ. Press, 2012.

[15] P. K. J. Novotňák, Z. Szőke and M. Šmelko, "Quadcopter modeling using a system for uav parameters measurement," *Drones*, vol. 8, no. 7, p. 280, 2024.

[16] C. Robotics, "Motor mixer theory," Cookie Robotics, 2025, [Online]. Available: https://cookierobotics.com/066/. Accessed: Apr. 13, 2025.