School of Science and Technology

Intelligent Battery Repositioning & Rotor speed adjustments for Drone Stability

A Machine Learning Control Strategy for
Rotor Malfunctions

by

Abubaker Shabbir

Project for the Degree of MSc

In Robotics, AI and Autonomous Systems MSc

Supervisor: Prof. Abdelhafid Zenati

London

15 October 2025

Dedication

Dedicated, with deepest gratitude to Allah SWT, the Most Gracious and Most Merciful, whose blessings and guidance have been our greatest strength; to our beloved parents for their unwavering love and prayers; to our esteemed teachers for their wisdom and support; and to our respected project supervisor for his invaluable mentorship.

Project Title: Intelligent Battery Repositioning & Rotor speed adjustments for Drone Stability

Student: Abubaker Shabbir

Supervisor: Prof. Abdelhafid Zenati 15 October 2025

Abstract

Current advancements in secondary flight control within the drone industry involve the implementation of flaps, slats, spoilers, & trim settings. For redundancy, the industry has implemented a Fault-Tolerant Control (FTC) algorithm, which allows a drone to redistribute power if one rotor fails, followed by an emergency landing. Unfortunately, the FTC algorithm suffers from hardware flaws, susceptibility to noise, & a sole focus on adjusting the remaining rotor speeds.

The project's goal involves creating a state-of-the-art XY core head to allow for battery manoeuvrability in a drone. Alongside this mechanism, the project will involve testing both in simulation & real-world environments. The model is powered by machine learning via an Nvidia Jetson Nano Developer Kit, which will actively decide battery location adjustments &, simultaneously, rotor speed in order to restore the drone to its centre of gravity should a rotor fail.

The FTC algorithm implemented in drone rotor power distribution will be altered & improved in reference to noise & adapted for autonomous characteristics, as well as integrated with the battery manoeuvrability. The training will involve the drone self-adjusting the rotor speeds & battery position in the most efficient manner.

The project further entails the full design of a drone, including respective tests for its aerodynamic efficiency & structural integrity. While a pre-made drone could be utilized for testing, achieving tailored results as well as accommodation of the intrinsic requirements for the mechanisms to be implemented necessitate significant adjustments; hence, the entire design is conducted from scratch.

For further testing & scalability, a universal user interface & a controller are also created, allowing for training model execution & providing manual switch controllers for each rotor.

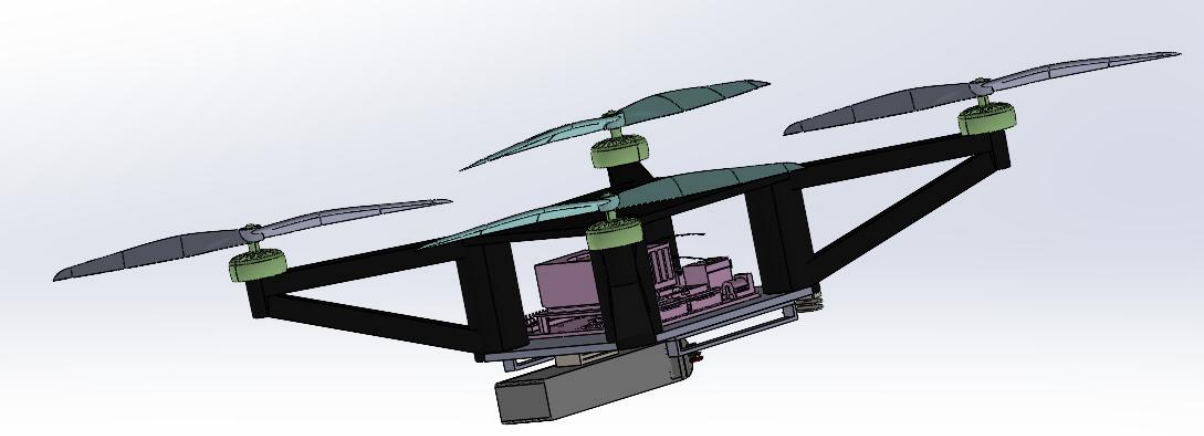


Table of Contents

Nomenclature

CoG – Centre of Gravity
FTC – Fault-Tolerant Control
ML – Machine Learning
UI – User Interface
UAV – Unmanned Aerial Vehicle
IC – Inertial Contributions
RPM – Revolutions Per Minute (min^{-1})
P – Power (watts)
T – Torque (Nm)
DL – Disk Loading (Nm^{-2})
V –Voltage (V)
A – Current (A)
V_i – Induced Velocity (ms^{-1})
PID – Proportional-Integral-Derivative
T_{HOVER} – Flight Time at 80% Battery usage

Ω – Angular Velocity of rotor (rad/s)
F – Force (N)
m – mass (Kg)
g – Acceleration due to gravity (ms^{-2})
K_P – Proportional Gain
K_i – Integral Gain
K_d – Derivative Gain
C_{ARM} – Drones Arm Bending Stress (Pa)
DoF – Degree Of Freedom
P_M – Mach Number for Propeller
FoS – Factor Of Safety
C – Speed of sound
P_T – Propeller Tip
MoI– Moment Of Inertia
C_{RATE} – Battery charge/discharge w.r.t time

Defining Mathematical Symbols

∑Drone Mass (M_T) = 1471.68 (g)
∑Drone Volume (V_T) = 827.46 (cm^3)
∑Drone Weight (W_T) ≈ 14.4 (N)
X CoG (X_{CG}) ≈ 10.14 (cm)
Y CoG (Y_{CG}) ≈ 15.30 (cm)
Z CoG (Z_{CG}) ≈ 24.36 (cm)
X IC (I_{xx}) ≈ 2.28 x 10^{-5} (kg m^2)
Y IC (I_{yy}) ≈ 2.9 x 10^{-5} (kg m^2)

Z IC (I_{zz}) ≈ 5.10 x 10^{-5} (kg m^2)
DL ≈ 444 Nm^{-2}
V_i ≈ 12.1 ms^{-1}
Mach Number ≈ 0.28 (subsonic)
$\Delta x_{batt\_shift\_intial}$ ≈ 3.91 (cm)
$\Delta y_{batt\_shift\_intial}$ ≈ -3.45 (cm)
C ≈ 343 ms^{-1}
g ≈ 9.81 m/s^2

List Of Figures

List Of Tables

Since the first powered flight in 1903 by the Wright brothers, aircraft have continuously evolved, both in their electronic systems & in their ability to remain operational under failure conditions. (The Wright Brothers | National Air & Space Museum, n.d.) Redundancy is the act of designing systems to maintain functionality when components fail; it has, in nature, been a cornerstone of this evolution. (*Doubling down on Safety: Understanding Our Approach to Redundancy in Autonomous Vehicles*, n.d.)

In the modern era, this principle extends to UAVs, where ensuring flight stability after a fault is critical. For instance, drones have a heavy usage in the present generation we live in:

1) Emergency Response & Disaster Management – this can be anything from delivering supplies to performing a search & rescue. ((2) The Evolution of Drones Across Sectors & Their Cybersecurity Risks: Best Practices & National Security Concerns for India | LinkedIn, n.d.)

2) Law Enforcement & Security – drones can monitor large gatherings, giving security a wider perspective, instead of multiple fixed camera points. (*AirHub - How Drones Usage Transform the Security Industry*, n.d.)

3) Infrastructure Inspection – Uk's aviation regulator published a new ruling as per following: "drones will be deployed for long distance inspection of infrastructure such as power lines, wind turbines as well as site security" (*Infrastructure Inspections with Drones Made Easier under New Rules | UK Civil Aviation Authority*, n.d.)

4) Agriculture – given the growing population of earth, there is a constant question in how enough food will be produced to suffice its needs, hence for many agricultural industries, the answer lies in precision agriculture with the aid of drones spraying fertiliser in places required, in which farmers may find a "burden". (*How Drones Are Used in Agriculture - Toll Uncrewed Systems*, n.d.).

Figure 12 Drones usage in the agriculture industry

5) Logistics & Delivery – Many freight industries are starting to deploy drones in their warehouses in support of their inventory management & surveillance. (*The Role of Drones in Revolutionising Warehouse Processes - Minster WMS*, n.d.)

Figure 13 Drone usage in the logistics industry

All in all, from these 5 sectors in which drones are being employed in, redundancy is most crucial in law enforcement & emergency response, given its direct impact on life. However, despite which one offers a more concerning redundancy, one must discuss what redundancy measures are currently in the market of drones & the difficulties in employing these…

1.1 Redundancy

Below will address redundancy measures implemented on specific usage drones.

1) Passive Fault-Tolerant Control [PFTC] (Ke et al., 2022)

PFTC is a control mechanism which handles events in which a rotor detection is present. If a rotor is to fail, the PFTC treats it as an external disturbance & counteracts it via robust control laws to bring back stability by embedding fault tolerance into the controller's design itself. The study states this methodology of treating a rotor failure as an external disturbance is a less complex & delayed process, disregarding fault detection & any form of diagnosis stage.

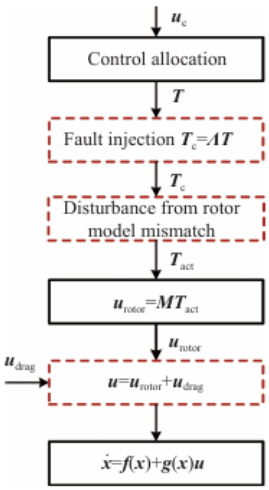


Figure 14 Data flow in the uniform Passive Fault-Tolerant Control (PFTC) method showing how rotor faults are modelled as lumped disturbances & compensated without fault detection

The above visually highlights that the method does not identify the specific fault; instead, all faults, as well as the mismatches, are grouped together as a single disturbance. The PFTC controller then proceeds to compensate for the disturbance directly, skipping fault detection entirely.

2) Model Predictive Control (MPC) / Nonlinear MPC (Nan et al., 2021)

MPC is a control strategy which utilises a *dynamic optimisation approach*. It operates by predicting the future behaviour the drone faces over a set period (known as the prediction horizon) & continuously resolving the optimisation problem as new data becomes available. This allows the controller to choose the best control actions based on both the current state & predicted future states.

Nonlinear MPC is applied in the event of a rotor failure, where it re-optimises the control inputs in real time to compensate for the fault. This enables a more precise & stable recovery compared to fixed-control methods, as it adapts the control commands to match the updated system dynamics after the failure.

In contrast to PFTC, which reacts to faults as disturbances without predicting future behaviour, MPC actively plans ahead within its prediction horizon, which can result in smoother & more efficient fault recovery, though at the cost of higher computational demand.

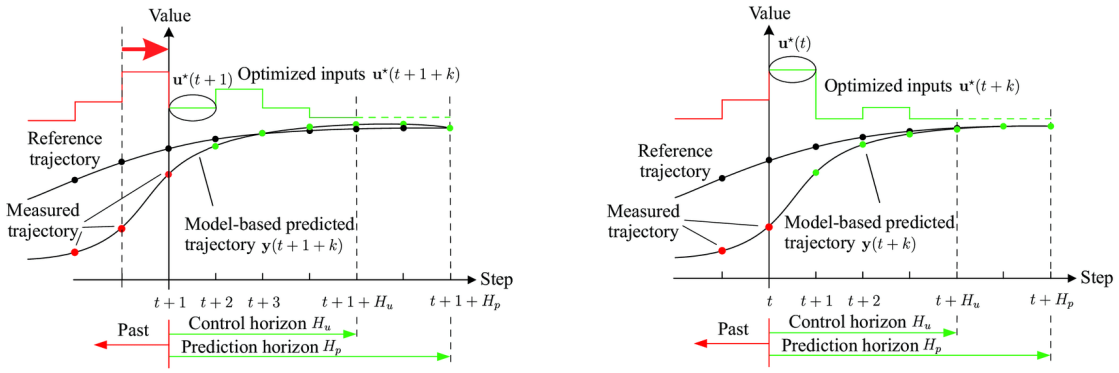


Figure 15 MPC's dynamic optimisation & continuous re-planning (1: Functionality of MPC & the Receding Horizon Principle | Download Scientific Diagram, n.d.)

Fig.15, above, showcases the entire framework of MPC in practice. This can be extracted from the 3 points the graphs share:

I) Prediction Horizon, H_P; MPC predicts the system's future behaviour over a set time.

II) Optimised Control Sequence: It computes an optimal series of future inputs.

III) Receding Horizon; Only the first control input is applied, & the process repeats at the next time step, causing a shift in the horizon forward.

3) Sliding-Mode Control [SMC] & Adaptive FTC (Fourlas & Karras, 2021)

SMC is a mechanism operated by a mathematical condition: the predefined surface.

Unlike MPC predicting the future over a ‘prediction horizon’, SMC doesn’t look ahead; instead, it forces the system onto a predefined surface & keeps it there until the goal is reached. In contrast to PFTC, which relies on a fixed robust design with no online reconfiguration (behaviour under a fault is predetermined via design), SMC with adaptive elements is able to actively tune itself during flight, in order to match a fault condition.

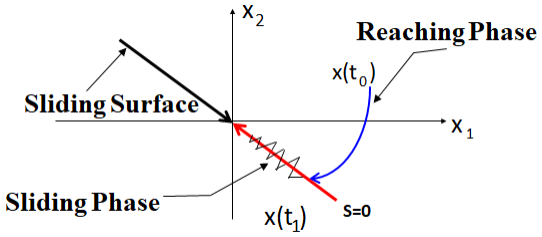


*Figure 16 Sliding Phases of the SMC (*Sliding Phases of the SMC | Download Scientific Diagram*, n.d.)*

In order for one to understand the framework of the SMC fig.5., a few assumptions need to be declared:

- The diagram represents a state space model, which simply means the system is represented via mathematical representations via the usage of state variables (system inputs, which define the system's state) to describe behaviour over time.

- x_1 & x_2 are state components (describe the system’s internal state at a specific point in time.

In relation to drones, x_1 can be the attitude error & x_2 the angular velocity; SMC laws in this case would adjust the motor outputs to drive these two states to the ‘sliding surface’ & as a result maintain stability, despite rotor faults.

SMC operation can be split into two stages:

Stage One – Reaching Phase: System starts at $x(t_0)$ & moves towards the sliding surface. Here, the controller would adjust the rotor speeds so both attitude & angular velocity match a surface condition set.

Stage Two – Sliding Phase: Once the system reaches (s=0) the surface, it moves towards the origin, known to be the target point. As it moves towards the target, it experiences chattering (zig-zag line), which is caused by the controller rapidly switching back & forth, in order to keep the system on the surface.

Here, the relation between angular velocity & attitude error changes smoothly from fixed to zero, ultimately allowing the drone to become both levelled & stable.

Additional redundancy strategies exist in the literature, including: Reconfigurable Actuation & Control Allocation High-Redundancy Actuation (HRA) *(Mousaei et al., 2023),* Modular Aerial Robotic Systems (MARS) *(Huang et al., 2025),* & Fault

Detection via IMU/Vibration & Emergency Landing Triggers *(Cabahug & Eslamiat, 2022)*.

These methods often involve specialised hardware architectures, modular mechanical designs, or are primarily aimed at fault detection rather than active control. While each is valuable in its own context, their case studies & implementation requirements differ significantly from the goals of this dissertation, which focus on real-time control adaptation; as such, these approaches are noted here for completeness, hence are not examined in detail.

1.2 Potential Redundancy Methods

Much of the fault-tolerant control research reviewed above focuses primarily on adjusting rotor speeds in response to a fault. While this is effective to an extent, the approach alone may not fully optimise stability, especially when dealing with severe rotor failures or asymmetrical load distributions; this was seen in the study by Jiaxin Ke explored above.

Given that the battery is often the heaviest single component of a multirotor UAV, the ability to reposition it dynamically offers a significant advantage for restoring balance & controlling the CoG. If it were to be combined with rotor speed adjustments, this could form a more robust redundancy strategy.

Moreover, many of the existing works only make limited use of artificial intelligence for decision-making, given its recent discovery in the aviation industry. Incorporating AI-driven control strategies could enable more adaptive & precise responses to faults via learning optimal battery position & rotor speed configurations in real time.

Another potential avenue is the use of extendable & retractable rotor blades, presented below:

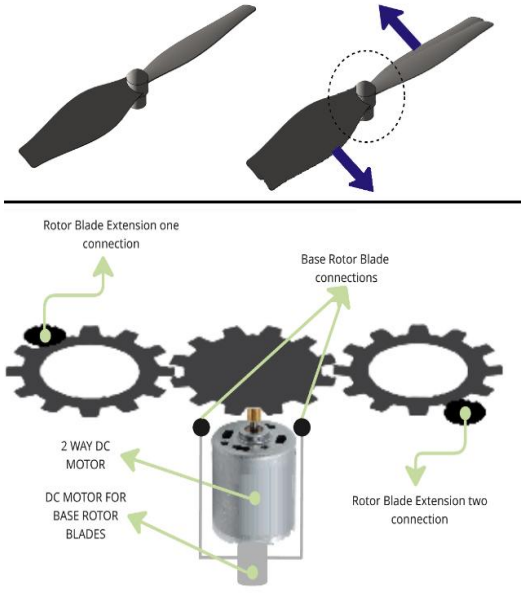


Figure 17 Extendable/retractable rotor blade, a stability potential avenue

Fig.6 presents a 3-cog system with connection to the extending rotor blades sitting on top of the base rotor blades. When extension/retraction is required, the dual DC motor will activate the opposing motor, which in turn allows for the cogs to turn; this is done ideally for a short burst/duration. As a result, the rotor blades' surface will extend (top rotor blade turns out of phase with base rotor blade).

This could dynamically adjust the rotor surface area & therefore directly impact the thrust output, providing additional stability control.

However, this method has not yet been widely deployed in operational drones, due to two main barriers:

-*Mechanical complexity* – The mechanisms required to adjust blade geometry mid-flight add weight, require high precision, & must operate reliably under varying loads.

-*Vibration challenges* – Moving parts introduce unwanted vibrations, which, as a result, interfere with sensors, control loops, & the overall structural integrity; resulting in a chain of instability.

The challenges have been evaluated in an article by Tyto Robotics, which states: "…5000 RPM, a 14-inch imbalanced prop produced **33% more vibration velocity** compared to a balanced one; at high RPMs, the effect intensifies dramatically."

Given the rotor is in direct contact with the environment & it requires wind for its operation, if one were to go ahead with the rotor extension/retraction innovation, one would find it rather difficult to find a solution to avoid vibration.

In contrast, given the battery doesn't need to be in direct contact with the wind, nor is that wind required for its operation, hence design tweaks can be conducted to reduce vibration & perturbation.

All in all, the project aims to contribute to the next stage of redundancy innovation within the drone industry *by autonomously integrating intelligent battery repositioning with adaptive rotor speed control*, providing an advanced fault-tolerant approach to rotor malfunctions.

1.3 Necessary Software in Place

Multiple uses of software are used in the project, as one could imagine, the complexity of this novel requires certain checks (aerodynamic, structure, etc.) prior to AI implementation.

[a] The initial grounding software in use is SolidWorks edition 2021. This model is the latest which can be used, as it allows for compatibility with other software in use; explained further below, Simscape (d).

The model is the hub platform, which will allow for one to model the entire design. A huge question, which arises when reviewing this project: Why not use a pre-made done and implement a moving

battery mechanism? Given this model's differential qualities to other drones, quantities like CoG, play a huge role in stability as well as tweaking structures to stretch out the benefits of testing. This is discussed further in depth in 2.3 Mathematical Modelling.

[b] The next software in play is EduPack. It is known mainly for its ability to understand the depths of a material to a microscopic level; however, in this project, the comprehensive database of material data is to be utilised, given that SolidWorks doesn't have an extensive industry-recognised database. Accurate data on a material like poison ratio, yield strength, etc. can be derived, which is then crucial in the studies conducted, in the next software in use…

[c] After full design (including material selection), the most crucial step is ensuring that the model is both aerodynamically & structurally fit for flight, as it gives a source of design proof. This step is absolutely critical, where tweaks need to be applied in sync to material selection & design choices, in hopes of building a functioning concept. SolidWorks plugins rather than separate CFD & FEA will be implemented to study for design validation.

SolidWorks allows for both CFD & FEA analysis via their flow simulation add-on/plugin. The plugin will allow for the following tests, for both aerodynamic validation & structural integrity validation:

Table 5 Structural & Aerodynamic Testing Checklist

Structural Validation Tests	Aerodynamic Validation Tests
Arm Bending / Deflection under Rotor Thrust	Thrust Balance in Hover
Stress Distribution (Factor of Safety)	Drag Force at Low Forward Speeds
Frame Mounting & Battery Rail Stress	Induced Velocity Check
Modal Analysis	Pressure & Flow Distribution
	Battery Shift Scenarios

[d] The main software which will be used to perform RL/ML for the intended AI training is MATLAB 2025a; more specifically, it's Simulink>Simscape add-on/plugin. Returning to software [a], version 2021, rather than any later model, was specifically downloaded due to the need for a Simscape Multibody link plugin (only supported in SolidWorks editions 2021). The plugin allows for a safe export passage from SolidWorks to Simulink, without losing any constraints (battery movement restrictions, assembly mate constraints). (*Install the Simscape Multibody Link Plugin - MATLAB & Simulink*, n.d.)

A contender to MATLAB for this project's AI training is Unreal Engine 4.27 + AirSims. However, given AirSims has not been in development since 2017, "has been archived as a research project", it didn't seem to be a safe deployment, given the lack of official support and software/plugins incompatibilities.(*Aerial Informatics and Robotics Platform - Microsoft Research*, n.d.)

[e] The final software/plugin is optional for simulation; however, given the intention is to build the model both physically and via simulations, having a custom controller in place is mandatory. PX4 Autopilot (via PX4 Support Package) allows for a MATLAB sync and, therefore, is most suitable since the project's training also takes place there. The controller must be able to control all rotors manually, regarding their speed and their health status (on/off).

2 The Design Phase

Chapter Abstract

This chapter outlines the process of designing a custom-built drone, which integrates the novel battery repositioning mechanism, driven by an X-Y core head, influenced by a 3d printer mechanism. The entire structure will be influenced & backed up via mathematical computations. In addition to these aerodynamic tests, structural integrity tests will be conducted to ensure full proof of the drone's design. Most importantly, the tests will validate the platform before fault-tolerant control trials, keeping in focus the project's ultimate goal: evaluating how effectively the AI-based training model can restore stability following rotor failure.

Having come to a conclusion in what must be achieved, attempting to retrofit these systems into an off-the-shelf drone would require extensive modifications to the frame, arms, & mounting points. At that stage, the amount of re-engineering would approach that of building a new drone entirely, without the benefits of designing the structure around the specific requirements of this novel redundancy approach. Hence, the project going forward will involve the creation of a unique frame from scratch, which ensures the following:

-Optimal weight distribution for both routine flight & post-fault recovery.

-Custom arm lengths & motor spacing in order to balance thrust as well as leverage during battery movement.

-Sufficient mounting space for the battery repositioning mechanism & supporting electronics (Jetson developer kit, receiver)

-Aerodynamic shaping tailored to the intended operational envelope, ensuring any added mechanisms don't cause huge disruptions to the drone's desired movement.

Figure 18 Freefly Alta X, drone inspiration

Although the drone is unique, its overall dimensions & general layout are inspired by existing multirotor designs, e.g., the Freefly Alta X. This, in turn, ensures that the prototype retains proven foundations when it comes to aerodynamic proportions & structural balance.

However, due to this build integrating a novel battery repositioning mechanism, its aerodynamic & structural behaviour cannot be assumed to match existing models & may result in a small addition in disturbance, which in a scaled model, may be lethal. Therefore, dedicated aerodynamic & structural integrity tests will still be conducted for sufficient validation.

2.1 Parts List

Refer Appendix 1 for brief specifications of parts breakdown as well as a schematic for visual understanding

Below presented is a breakdown of the parts, alongside their respective function/purpose & quantity that will be required:

Table 6 Parts Quantity & Function List

Part	Quantity	Function
Frame	X1	- Mounting point for all 4 arms & Jetson Nano Developer Kit. - XY Core Head has repositing mechanism (X-Y core head) for battery.
Arm	X4	- Placed at such distance to avoid physical contact with DC motor & body frame. -Mounting point for DC motors.
Rotor Blade	X4	- Production of thrust & testing mechanism.

Battery	X1	- Power all electronics on-board (Jetson, receiver, MOSFETS, DC motors, stepper motors)
Jetson Nano Developer Kit	X1	- Brain of the model, controller & management of all electronics & holder of the training ai model.
Receiver	X1	- Communicator with controller via radio frequencies.
Stepper Motors	X2	- Crucial motor required for the X-Y core head to function. - Allows for batteries vertical & horizontal movement via belt movement.
MOSFETs irlz44n	X4	- Used in testing phase, to allow for user to manually switch of rotors; in order to observe/test the effectiveness of the training model.
Custom Controller	X1	-Allows user to communicate with the drone; controls the drones movement & via the MOSFET the status of the rotors (ON/OFF)

2.2 Material Selection

Having discussed the primary focus of this project, the justification for material selection will be kept concise. The goal of this section is to provide a practical & functionally sound basis for the physical components, with materials chosen for their balance of strength, weight, & ease of manufacturing via 3D printing. The following analysis compares viable material options, as well as justifying the final choice for each individual 3D-printed part. The parts that are not 3D-printed have their corresponding materials used, which have been provided in the BS8888 CAD drawings in the appendix page.

Table 7 Material Selection for Drones constituent parts

Part	**Material Selected**
Central Frame Plates & Arms	Torayca T300 grade with Epoxy Resin
Justification	
Provides an exceptional strength-to-weight ratio & high rigidity, which is essential for a stable foundation, in order to mount all components as well as in resisting vibrations.	

Linear Rails & Shafts	Alumina (Al_2O_3 ceramic)
Justification	
Selected for high stiffness & dimensional stability. Also ensures the smooth & precise linear motion of the battery mechanism under load.	
Propellers	Torayca T300 grade with Epoxy Resin
Justification	
Chosen for its light weight & high stiffness, in hopes of minimising the rotational inertia & prevent blade flex; which is almost critical in producing the most efficient thrust generation.	
Motor Casings & Shafts	Aluminium Alloy & Steel
Justification	
Aluminium is used for the casing due to its excellent heat dissipation as well as it's light weight aspect. Hardened steel is also implemented for the motor shafts to resist bending & wear.	

2.3 Mathematical Modelling Preparation for AI Training

Several computations for a drone are dependent on the acknowledgement of the weight of the drone's constituent parts, hence the table below. Via the SolidWorks feature on defining a geometry in terms of weight, once the model is modelled and assigned the materials from Table 2, one is able to collect weight data, specifically on the 3d printed parts, given that the non 3d parts, which are provided, have a weight within the product specifications.

A summary of all computations can be found on pg.12, defining mathematical symbols.

-Mass, Volume, and Total Weight

The determination of mass & volume for each individual component is essential in establishing the drone's overall weight as well as validation in the model's physical properties. The total mass provides the basis for the computation of the gravitational force acting on the system, which leads one to understand the thrust requirements for flight.

Accurate knowledge of mass distribution also underpins CoG analysis, which is critical to this project's stability concept, as the battery must be repositioned to restore balance in the event of rotor failure; this can be seen in Appendix 4.

Table 8 Constituent parts mass & Volume Table

Part	Mass (g)	Volume (cm^3)
Frame	1107.48	614.43
Set of 4 Rotors & Propellers	121.96	45.08
Static Rail	21	5.32
Battery Sled	191.74	155.18
Rail Sled	29.50	7.45

Total Mass (g)	Total Volume (cm^3)
1471.68	827.46

Therefore, weight can be computed as following:

$$\mathbf{W = mg} \quad \textbf{(1)}$$

$$W = 1.47168 \times 9.81 \approx 14.4 \text{ N}$$

Note **that** the total mass was obtained by summing component masses from the SolidWorks mass table and datasheets.

-Average density check

$$\bar{p} = \frac{m_{total}}{V_{total}} \quad \textbf{(2)}$$

$$\frac{1471.68}{827.46} \approx 1780\ kg/m^3$$

The inclusion of an average density check further ensures the plausibility. Given value 1780 kg/m^3 against typical values for lightweight engineering materials implemented in design, lies within this expected range (appendix 5) …thereby validating the physical plausibility of the model prior to initiating the AI training. (Ashby, 2005)

Table 5 Drones Overall Centre of Gravity (CoG)

Part	CoG Axis (cm)		
	X_i	Y_i	Z_i
Frame	10.65	14.85	24.30
Set of 4 Rotors & Propellers	10.72	10.36	24.18
Static Rail	10.72	18.62	24.32
Battery Sled	6.69	20.14	24.83
Rail Sled	10.72	18.62	24.38

The CoG was determined from component coordinates:

$$x_{cg} = \frac{\sum m_i x_i}{\sum m_i} \quad y_{cg} = \frac{\sum m_i y_i}{\sum m_i}$$

$$z_{cg} = \frac{\sum m_i z_i}{\sum m_i} \tag{3}$$

$$x_{cg} \approx 10.14\ cm \quad y_{cg} \approx 15.30\ cm$$

$$z_{cg} \approx 24.36\ cm$$

The computations for CoG are another reason to build this specific drone from scratch, in addition to its bespoke elements. The drone has been specifically designed to provide an advantage in scenarios where one or more rotors fail. The advantage is that battery movement upon rotor failure is more effective…

Usually, many drones have CoG towards the bottom end, as it generally increases passive stability. However, if the battery can shift in the event of a rotor's failure, making the shift more effective requires having the CoG away from the manoeuvring mechanics (X-Y core head), which is almost essential. (Bouabdallah et al., 2004)

Here is a schematic of how this design differentiates from others...

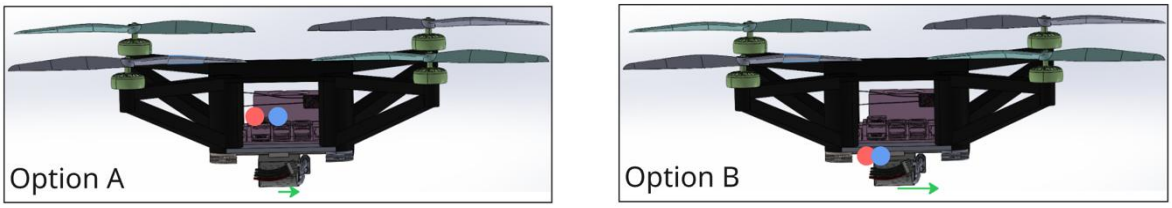


Figure 19 CoG Design Reasoning Illustration

Initially, it is important to note that the above is only for illustration purposes and does not entail any mathematical input other than being an annotated schematic. The diagram presents how this project's design is inclusive to other drones in the market; in that most of them follow option A, due to the 'pendulum effect'.

Drone with option A exhibits passive stability in that it acts as an upside-down pendulum. If the drone is to tilt, gravity would naturally pull the drone back to stability; this, therefore, gives the pilot a sense of autopilot. However, this would be a disadvantage given the project's battery manoeuvring mechanism, as when moving the battery, one would expect the drone to become stable almost instantly, given that it is a major weight component. However, given the pendulum effect discussed, this would act in opposition to it, so one may see CoG gradually drift back, but the process will be considerably slower as represented in the schematic (Green arrow/battery shift shorter in option A but causes faster CoG drift in differential to option B).

Moments of Inertia (MOI)

Via the usage of the overall drones CG values, one can use this to compute the individual components' contributions to I_{xx}, I_{yy}, I_{zz}. Below is the rundown, in computing for the frame (for the rest of the components, refer to Appendix 4):

-Given the following defining parameters of the frame, allow for the computation of its MOI contributions:

Table 6 Frame Defining Parameters

Frame Defining Parameters	
Mass (Kg)	1.10748
CoG Coordinates (m) (X_i, Y_i, Z_i)	0.1065, 0.1485, 0.2430
Global Drone CG (m) (X_{cg}, Y_{cg}, Z_{cg})	0.1014, 0.1530, 0.2436

-Axis Offsets (distance from drones CoG to frame):

-X Axis Offset

$$\Delta \boldsymbol{x} = \boldsymbol{x_i} - \boldsymbol{x_{cg}} \quad \textbf{(4)}$$

0.1065 – 0.1014 = 0.0051

-Y Axis Offset

$$\Delta \boldsymbol{y} = \boldsymbol{y_i} - \boldsymbol{y_{cg}} \quad \textbf{(5)}$$

0.1485 – 0.1530 = -0.0045 (m)

-Z Axis Offset

$$\Delta \boldsymbol{z} = \boldsymbol{z_i} - \boldsymbol{z_{cg}} \quad \textbf{(6)}$$

0.2430 – 0.2436 = -0.0006 (m)

- Inertia Contributions (for frame):

-X Axis Inertia

$$\boldsymbol{I_{xx}} = \boldsymbol{m}(\Delta \boldsymbol{y^2} + \Delta \boldsymbol{z^2}) \quad \textbf{(7)}$$

1107.48 $((-0.0045)^2 + (-0.0006)^2)$

$\approx 2.28 \times 10^{-5}$ (kg m^2)

-Y Axis Inertia

$$\boldsymbol{I_{yy}} = \boldsymbol{m}(\Delta \boldsymbol{x^2} + \Delta \boldsymbol{z^2}) \quad \textbf{(8)}$$

1107.48 $((0.0051)^2 + (-0.0006)^2)$

$\approx 2.9 \times 10^{-5}$ (kg m^2)

-Z Axis Inertia

$$\boldsymbol{I_{zz}} = \boldsymbol{m}(\Delta \boldsymbol{x^2} + \Delta \boldsymbol{y^2}) \quad \textbf{(9)}$$

1107.48 $((0.0051)^2 + (-0.0045)^2)$

$\approx 5.10 \times 10^{-5}$ (kg m^2)

The moments of inertia describe how the drone's mass is distributed & the resistance it has to rotate along its x, y and z axes. For this specific project, computing these values is important primarily for the following two main reasons.

[a] First, they provide the AI system with realistic initial parameters, meaning the training process does not need to "learn" the drone's dynamics from scratch, which makes convergence faster & more accurate. (Pounds et al., 2006)

[b] Secondly, it allows one to assess the design itself. If inertia is too high around an axis, the drone will face difficulty in correcting its orientation. Likewise, if inertia is too low, the drone will have the tendency to become overly sensitive to disturbances.

Therefore, this addresses the absolute need for accurate MOI values, as it will support efficient AI training & engineering decisions in the design phase.

Propeller Disc Loading:

Disc loading is simply the weight of the drone divided by the rotor disc area. In layman's terms, it's how hard each propeller must work to keep the drone in the air, without dropping in altitude. A High disc loading entails the four rotors requiring more power to keep up with the demand and thereby are defined as less efficient, while low disc loading improves lift efficiency and stability.

Given the following: diameter ≡ 8 inch ≡ 0.2032 (m), radius = 0.1016 (m)

The propeller disc area is the following:

$$\mathbf{A} = \boldsymbol{\pi}\,\mathbf{r^2} = \pi\,(0.10106)^2 = 0.032429\ \mathrm{m}^2 \quad \textbf{(10)}$$

Therefore, the disc loading can be computed as per below:

$$\mathbf{DL} = \frac{\mathbf{W}}{\mathbf{A}} = \frac{14.4}{0.032429} \approx 444\ \mathrm{Nm}^{-2} \quad \textbf{(11)}$$

The computed disc loading presents itself with a rather high value, ~444 N/m², in comparison to typical values ranging from 100–250 N/m², which entails this project's drone design requiring more power per unit rotor area. While this reduces aerodynamic efficiency, it is an intentional trade-off in this project. By using smaller propellers & accepting higher disc loading, the overall frame size has been reduced, making the drone compact as well as more manoeuvrable. (Newman, 2007)

Most importantly, it entails an increase in the demand that the AI will be training in hopes of maintaining stability, which aligns with the project's aim of "testing active fault-tolerant control", rather than maximising passive efficiency.

Induced Velocity in Hover (Momentum Theory):

The induced velocity represents the downward airflow required for the propellers to generate sufficient lift, in order to balance the drone's weight.

(12)

$$\mathbf{v_i} = \sqrt{\frac{\mathbf{W}}{\mathbf{2\rho A}}} = \sqrt{\frac{14.4}{2 \times 1.225 \times 0.04}}$$

$$\approx 12.1\ \mathrm{ms}^{-1}$$

This design presents itself with a rather high induced velocity of 12.1 m/s. In reflection of the previously identified high disc loading, this was expected; thereby, propellers must accelerate a greater volume of air in order to sustain the necessary hover state during flight.

From a design perspective for this project, it entails a strong trade-off, in that, although higher induced velocity reduces propulsive efficiency, it also makes the system more responsive to rapid thrust adjustments. Hence, it is of benefit to move the battery mechanism (powered by AI decision making) during rotor failure scenarios. Thus, given endurance may be compromised compared to low-disc-loading drones, the configuration supports the project's focus on "active stability & fault-tolerant control". (Newman, 2007)

Propeller Tip Speed & Mach Number:

$$\boldsymbol{\omega} = \frac{\mathbf{2\pi\ RPM}}{\mathbf{60}} \frac{(2\pi)(9000)}{60} \tag{13}$$

$$= 942.48\ \mathrm{rads}^{-1}$$

$$\mathrm{V_{tip}} = 942.8 \times 0.1016 = 95.76\ \mathrm{ms}^{-1} \tag{14}$$

$$\mathbf{Mach} = \frac{\mathbf{V_{tip}}}{\boldsymbol{c}} = \frac{95.76}{343} \approx 0.28 \tag{15}$$

In acknowledging, that propeller speeds vary with throttle/load, the computations for 9000RPM (maximum operating speed) and 7000RPM (typical hover) are tabulated below:

Table 7 RPM for respective Mach

RPM	ω (rad/s)	Tip Speed (m/s)	Mach
7000	733.04	74.50	0.22
9000	942.48	95.76	0.28

The propeller tip speed & Mach number tabulated above is crucial, since it defines the aerodynamic regime the drone operates in.

Figure 20 Understanding Mach Number State Ranges (Mach Number, *n.d.*)

"Designers aim to keep the tip Mach number below ~0.3–0.4 in hover and low-speed flight". Given this project's drone holds at Mach at 0.28, it sits within the subsonic state; therefore, compressibility limits ensure that the aerodynamic forces remain predictable.

This makes it easier for the AI to learn accurate control behaviours. It also avoids unexpected losses in thrust efficiency or stability issues that could arise, and lead to a complicated AI testing environment or potential real-world testing.

Battery Relocation for CG Trim (Quantified):

The frame's CoG is effectively the geometric centre of the structure, given that its the location the motors & arms are symmetrically placed.

Hence, alignment to the global drone CG with this point ensures thrust is evenly distributed across all four rotors, Jetson nano and other parts mounted onto the frame. The AI training model hugely benefits from this as an initial supply fed to the controller as a source of stable references. Additionally, from this, the balance point will allow AI to learn how to shift/reposition the battery in its XY core head, in hopes of restoring stability during rotor failures.

$$\Delta x_{batt\_shift\_intial} = \frac{m_{tot}\,(x_t - x_{cg})}{m_{batt}} \qquad \frac{1.47168\,(10.65-10.14)}{0.19174} \approx 3.91\ \text{cm} \qquad \textbf{(16)}$$

$$\Delta y_{batt\_shift\_intial} = \frac{m_{tot}\,(y_t - y_{cg})}{m_{batt}} \qquad \frac{1.47168\,(14.85-15.30)}{0.19174} \approx -3.45\text{cm} \qquad \textbf{(17)}$$

As per the computation above, the battery must shift by approximately +3.9 cm in the x-direction and -3.5 cm in the y-direction, in order to align with the global CoG with the frame's CoG.

These are realistic travel distances for a rail-based sledge and validate that small lateral movements of the battery can produce meaningful changes in CG position.

Therefore, this confirms that the mechanism entailing two servo valves & non-electrical support pivots is viable for active stability control, without requiring excessive mechanical displacement.

2.4 The XY Core Head

As per the introduction, drones have been implemented with a safety mechanism to distribute thrust in hopes of maintaining stability. The project's novelty comes into play when we build on these models with a moving battery.

A famous phrase by Guy Kawasaki, as follows: "Ideas are easy. Implementation is hard", comes into play here. One must assess that moving the battery has an actual influence on moving the drone CoG. One must also consider that the weight and complexity this adds to the drone framework are less of a drawback than the potential advantages the mechanism brings. Most importantly, a factor to consider is whether the design is feasible when scaled. (*Ideas Are Easy, Implementation Is Hard*, n.d.)

As per the computations conducted in 2.4 mathematical justification, the 'XY Core Head ' has been considered a plausible design choice, at the drone's current scale, given that a small battery step can influence a good deal of CoG shift.

Going back into the complexity, the following represents a schematic explaining how the XY core head operates:(*CoreXY Kinematics – 3D Distributed*, n.d.)

For one to understand how it operates, the schematic is used as an illustration:

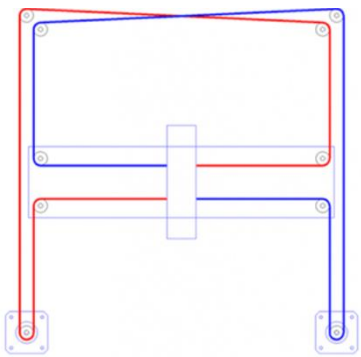


Figure 21 XY Core Head Layout

Regarding the setup, the requirements are the following tabulated sub-parts:

Table 98 XY Core Head Sub Parts

XY Core Head Sub Parts	
1	(x10) Guide Chains
2	(x2) Rubber Belts
3	(x2) Stepper Motors
4	(x1) Static Rail
5	(x1) Sled Rail (Y Directional)
6	(x1) Sub Sled Rail (X Directional)

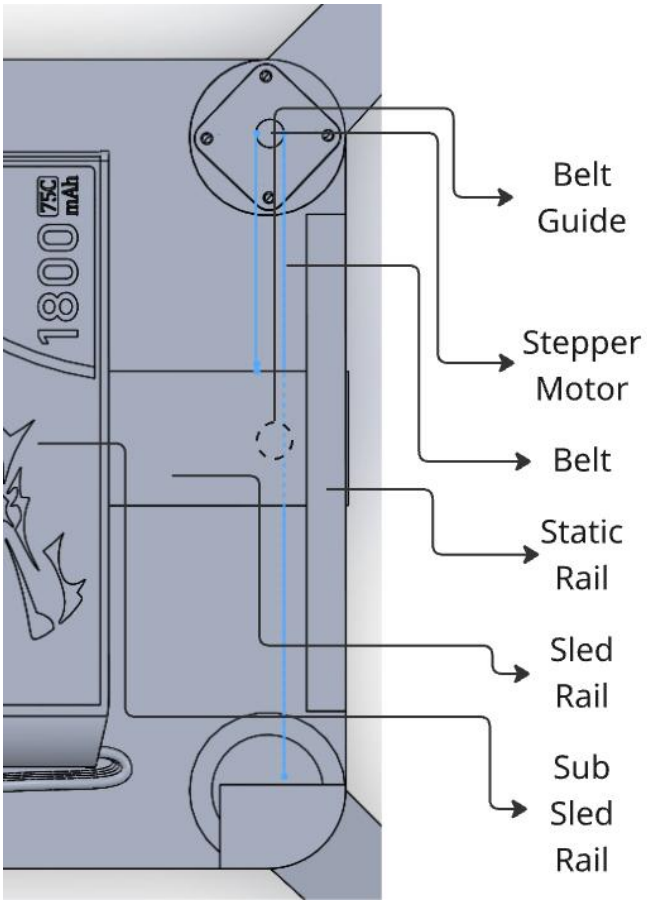


Figure 22 Annotated XY Core Head Parts on Custom Design

As can be seen from the tabulated XY core head sub parts, model complexity has clearly increased, and likewise the susceptibility to maintenance/wear & tear. However, given this to be an emergency protocol (rotor fails), the mechanism won't be under constant stress during the length of the common flight.

The XY Core Head, given its title, allows for both x and y directional movement. Here presented is how the stepper motors must function in order to reflect a manoeuvre to a certain location within the restricted area:

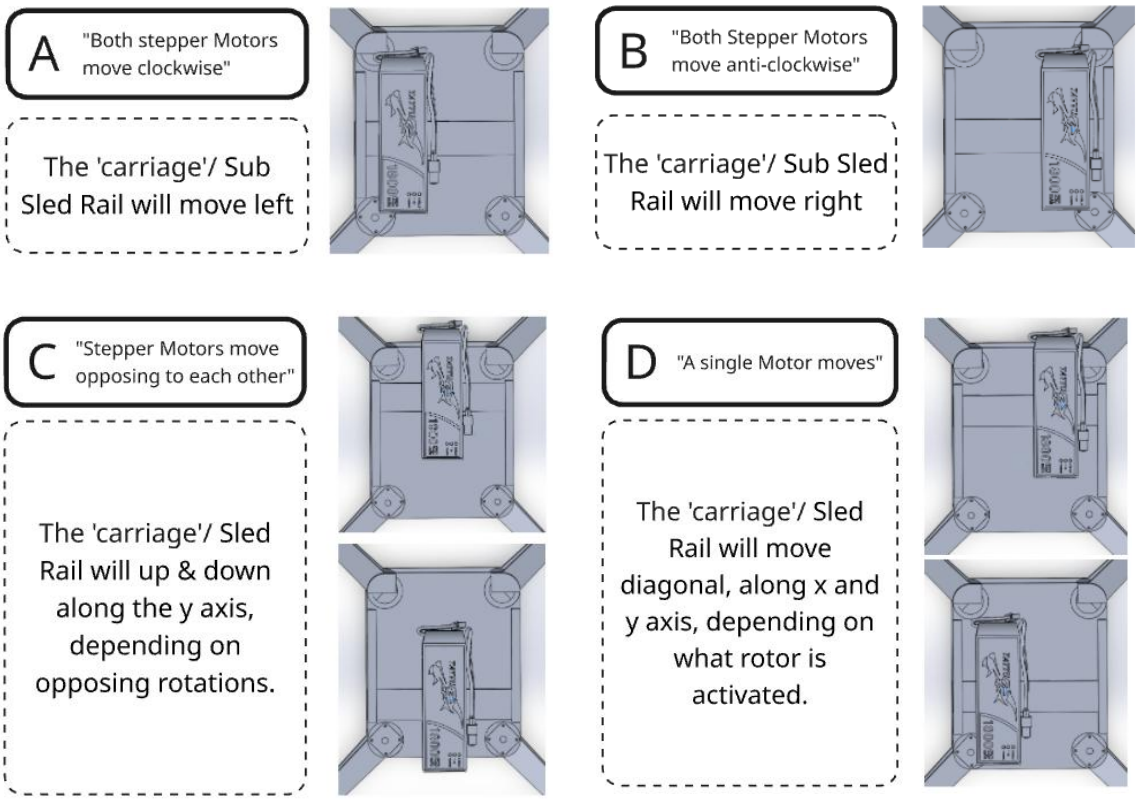


Figure 12 XY Core Head Stepper Motor Functions

The design has been precisely implemented onto the design, with assurance that each belt guide is in line with its corresponding as shown below:

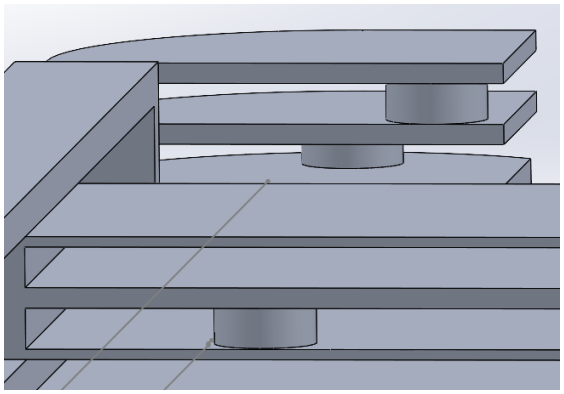


Figure 13 Precise XY core head supports Design with sufficient tolerance

Given that the XY core head will add a level of mathematical complexity to the AI model, the current model for simulation has been configured such that each stepper motor is allocated one axis from x and y.

3.0 Quadcopter Physics

Chapter Abstract

The following chapter dives into the Simulink setup, including tuning the thrust, pitch, yaw & roll controls of the drone as well as exporting the constraints in place from the Simulink environment. The goal at this specific part is to achieve full control over the drone, as well as simultaneously ensuring that the drone is able to remain stable at hover and during a given torque to all four rotors.

Prior to initialising the Simulink setup, one must ensure all CoM align with the respective drone frames, as visible in fig.14.

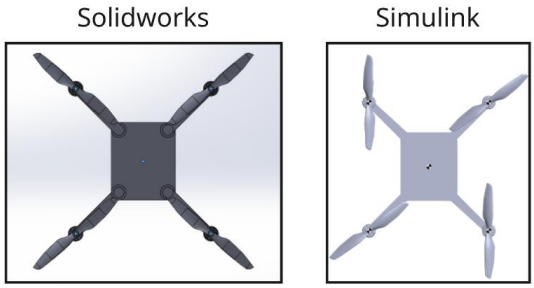


Figure 14 Drone CoM checks on SolidWorks & Simulink

This ensures there aren't discrepancies during the import phase. Another crucial factor to check is that the axes are such that z points upwards in both files, as mathematical computations (e.g., inertia) are referred to a specific X, Y & Z axis.

For a drone controller to have full control over the drone, the mandatory controls it requires are the ability to pitch, yaw, roll and most importantly, control its altitude fig.15.

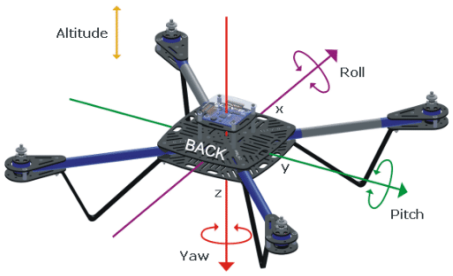


Figure 15 Yaw, Pitch, Roll & Thrust

Having control over, however, isn't enough, due to physics. If one were to input a specific thrust control, the drone would have to tackle gravity and other forms of resistance from the environment, e.g., air resistance. Hence, giving control doesn't mean an instant desired output.

This is where PID tuning comes into play. A PID is a form of 'feedback controller' which will have the job of continuously comparing, one the desired state input from

the user, two the actual state of the drone. This way it's able to compute for an error (desired state – actual state = error)

Once it has an error computed, the system may change the thrust produced to decrease the computed error, in hopes of achieving the 'desired output'.

Hence, PID tuning over yaw, pitch, roll and altitude controllers is mandatory, which is referred to in the discussion of the Simulink environment setup.

3.1 Simulink Environment

Refer to the appendix to view the entire environment as a whole.

Below showcases the drone Simulink environment, split into the following six sub-sections:

Table 910 Simulink Environment Breakdown

Simulink Environment Breakdown	
1	World & Environment
2	Drone Body Dynamics
3	Attitude Sensor & Converter
4	Altitude Controller
5	Attitude Control Loops
6	Motor Mixer & Motor Outputs

3.1.1 – World & Environment

The following area defines the simulation environment. It establishes the global reference frame, applies gravity (in the -z axis), and ensures forces are correctly transferred between both the world & the drone body.

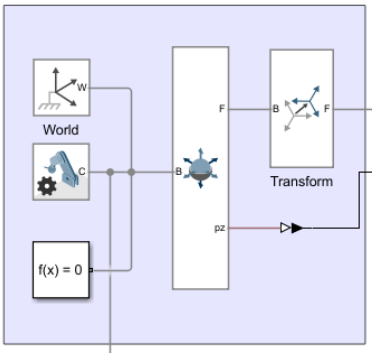


Figure 16 World & Environment Simulink

3.1.2 – Drone Body Dynamics

The quadcopter frame & its respective four rotors were imported into Simulink using the 'SolidWorks Multibody Export feature'. This ensures the geometry, mass properties, and joint constraints (translations, rotations, and rigid connections) are

transferred exactly as designed they are, during the custom design phase. The result is a rigid-body assembly where the drone frame holds the four rotors in their correct positions and orientations, as shown in the schematic, fig.17.

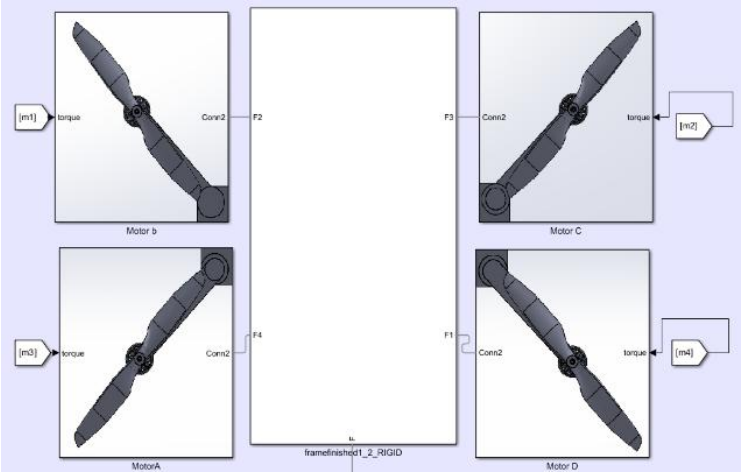


Figure 17 Propeller, under mask sub-system

Each respective rotor is represented via a masked subsystem feature, allowing for torque inputs to be applied individually, without overcomplicating the home environment.

Inside each mask (fig.17), the subsystem consists of:

- A Revolute Joint; this enables a rotational motion to the propeller itself. (visuals)

- A Torque input; this drives the propeller's angular velocity (ω).

- A Propeller model block, which converts angular velocity (ω) into thrust (T) & drag torque (Q).

Drag torque is then subtracted from a torque command that we, the user, apply. This is a crucial step as it motions a more realistic environment; as the rotor spins, it produces thrust and a form of aerodynamic drag, resisting rotation, and torque drag.

A crucial element, leading to torque drag, is the rotor rotation direction.

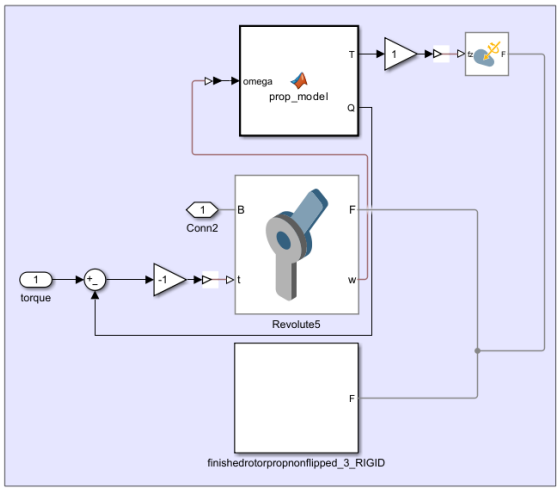


Figure 18 Propeller, under mask sub-system

The rotors must behave in the following manor, to avoid any potential drone yaw:

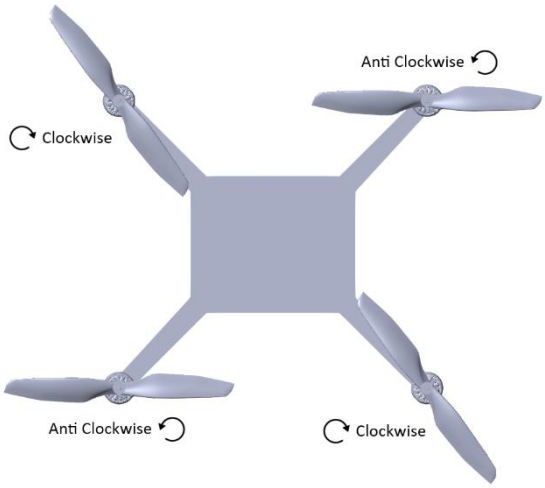


Figure 19 Drone Rotor Clockwise Directions

The diagonals must have the same rotation, in comparison to their opposing diagonals, allowing for the drag torques to cancel out. On Simulink, this was achieved via simple gain blocks of +1 & -1 post the torque commands.

All in all, the setup allows each motor to be independently controlled while still respecting the physical constraints of the imported CAD model, as well as ensuring accurate simulation of forces & torques.

3.1.3 – Attitude Sensor & Converter

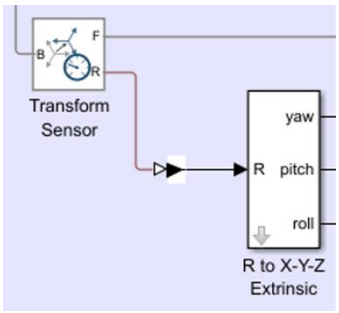


Figure 20 Attitude Sensor & Converter

Moving on, the project heads to the control side of the environment. This particular region is where the drone's altitude in terms of yaw, pitch & roll is monitored. Via the 'transform sensor' block, Simulink allows for the measurement of the relative orientation of the drone body (B) w.r.t the world reference frame (F). The output is a rotation matrix; this fully describes the drone's 3D orientation. However, given that feeding a controller with a 3x3 rotation matrix is a little impractical, this is converted into Euler angles.

Below presents the element of impracticality:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \textbf{(18)}$$

Above is a 3x3 rotation matrix, which can also be represented as:

$$R = [\hat{x}_b \quad \hat{y}_b \quad \hat{z}_b] \quad \textbf{(19)}$$

It can be represented as per above, given each column in a rotation matrix is a unit vector along the body's x, y & z.

Now, having described the reduced rotation matrix formula, one can demonstrate the constraints below:

a) Unit Length Constraint; states each vector axis must have a length of 1, hence

$$\|\hat{x}_b\| = \|\hat{x}_b\| = \|\hat{x}_b\| = 1 \quad \textbf{(20)}$$

This itself presents three constraints.

b) Orthogonality Constraint; states all axis must be perpendicular, hence

$$\hat{x}_b \cdot \hat{y}_b = \hat{y}_b \cdot \hat{z}_b = \hat{z}_b \cdot \hat{x}_b = 0 \quad \textbf{(21)}$$

The following demonstrates an additional three constraints.

c) Right-hand Rule Constraint; states the third axis must be the cross product of the first two

$$\hat{z}_b = \hat{x}_b \cdot \hat{y}_b \quad \textbf{(22)}$$

All in all, these constraints reduce the nine matrix entries down to the three independent variables that define the drone's attitude. The right-hand rule in particular removes ambiguity by distinguishing between a proper rotation (determinant +1) and a mirror reflection (determinant = -1); the right hand ensures only +1 determinant is in use, therefore, the drone is presented only its true rotation and not mirrored (left hand constraint).

Thus, while the rotation matrix is a mathematically complete representation, it is redundant for control. Hence, by converting it into Euler angles (yaw, pitch, roll), the controller obtains three intuitive scalar signals that can be directly compared to desired reference angles:

-Yaw (ψ, about Z-axis):

$$\psi = arctan2(r_{21}, r_{11}) \quad \textbf{(23)}$$

-Pitch (θ, about Y-axis):

$$\theta = -arcsin(r_{31}) \quad \textbf{(24)}$$

-Roll (φ, about X-axis):

$$\phi = arctan2(r_{32}, r_{33}) \quad \textbf{(25)}$$

3.1.4 – Altitude Controller

Following on from the attitude sensing stage, where yaw, roll & pitch are extracted, the next focus lies on the drone's actual stability. At this specific point, as per the schematic presented:

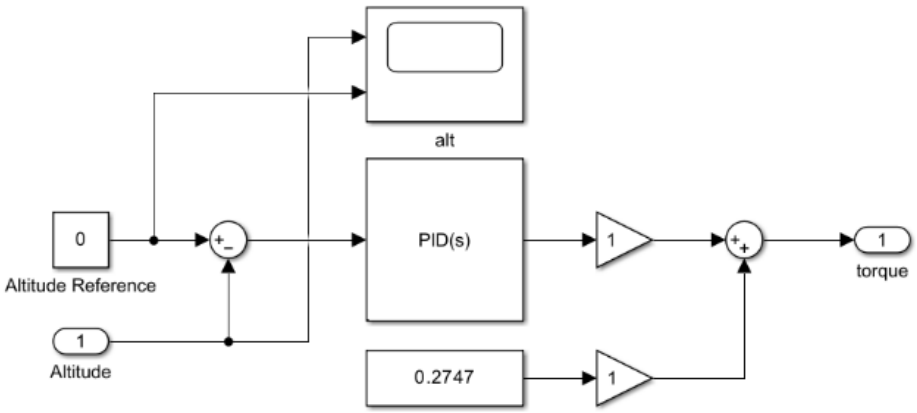


Figure 21 Altitude PID Tuned Controller

The project focuses on ensuring the stability of the drone's vertical motion. The sole objective of this controller, alongside stability, is that a user-defined height can be executed, taking into account any gravity or other forms of resistance the drone may face during the travel phase.

The process initiates via comparison of altitudes from a user-defined constant block to the exact altitude from the transform block discussed earlier. The subtraction of both these allows for a computation of altitude error:

$$e_z = z_{ref} - z \quad \textbf{(26)}$$

This error is then utilised with the controller's respective PID controller, acting as a source of data; it can be referred to as a feedback loop provided to the PID, for it to adjust accordingly.

Lastly, a summation block is used to all the drone to hoover if altitude command is to be 0. The drones hoover was computed simply via the formula:

$$T_{hover} = mg \tag{27}$$

It also ensures that the drone doesn't generate the thrust from 0, instead, a hover baseline.

3.1.5 – Attitude Control Loops

Moving from the drone's altitude control mechanism, the project focuses on regulating the drone's rotational stability.

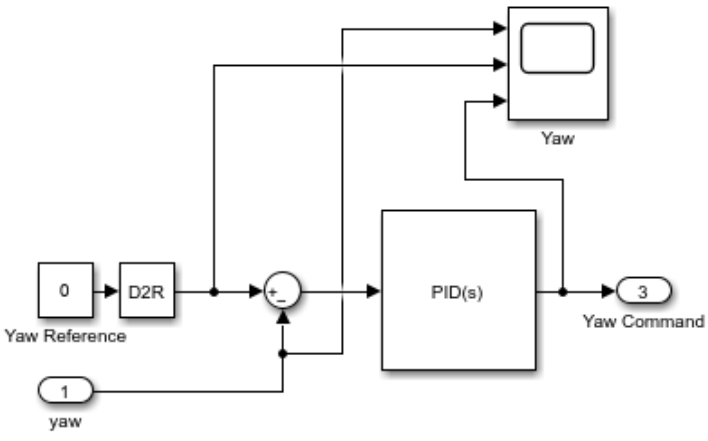
The schematic below presents 3 control loops, which are responsible for ensuring the drone is able to achieve, as well as maintain, the commanded yaw, roll & pitch. Similar to the altitude mechanism, pitch, yaw and roll respective errors are computed:

$$e_{\psi} = \psi_{ref} - \psi \ (Yaw\ Error) \tag{28}$$

$$e_{\theta} = \theta_{ref} - \theta \ (Pitch\ Error)$$

$$e_{\phi} = \phi_{ref} - \phi \ (Roll\ Error)$$

Each error signal is then fed into its respective tuned PID controller, where it is adjusted to produce corrected torque outputs.



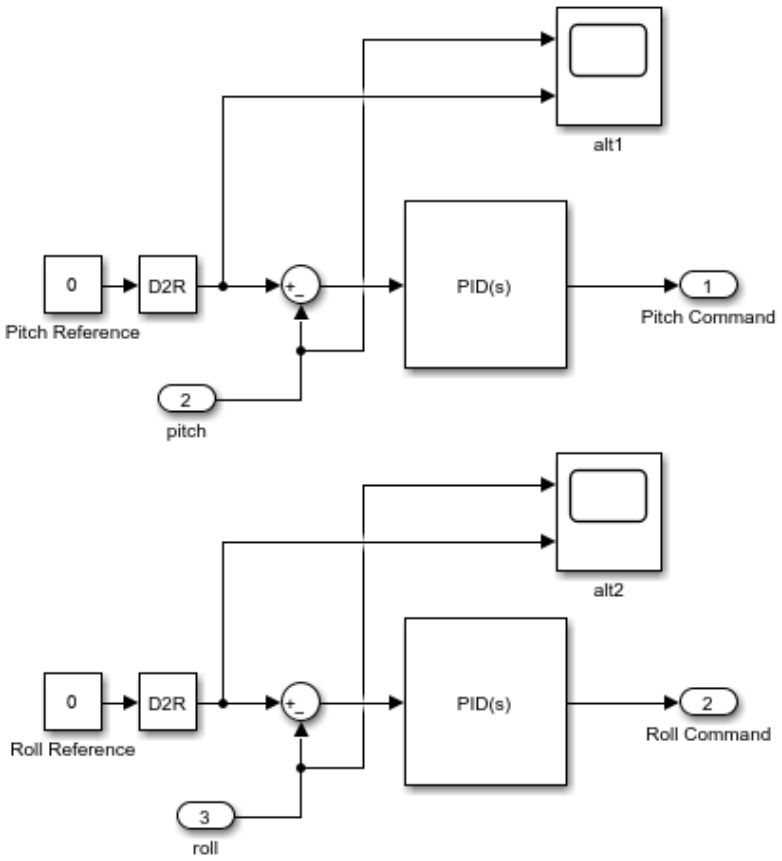


Figure 22 Attitude, Pitch, Yaw & Roll PID Controllers

3.1.6 – Motor Mixer & Motor Outputs

The final part of this environment chain is the Motor mixer. This is where torque commands from altitude & attitude controllers get distributed to all 4 rotors, depending on what positive & negative role is (based on import axis from SolidWorks).

It is governed by the formula:

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} = \begin{bmatrix} 1 & +1 & +1 & +1 \\ 1 & +1 & -1 & -1 \\ 1 & -1 & +1 & -1 \\ 1 & -1 & -1 & +1 \end{bmatrix} \begin{bmatrix} T \\ u_\phi \\ u_\theta \\ u_\psi \end{bmatrix} \quad (29)$$

Roll, pitch, and yaw corrections

4 motor commands

Total thrust command

The motor mixer is configured via an MATLAB function block feature, presented as a flow chart below:

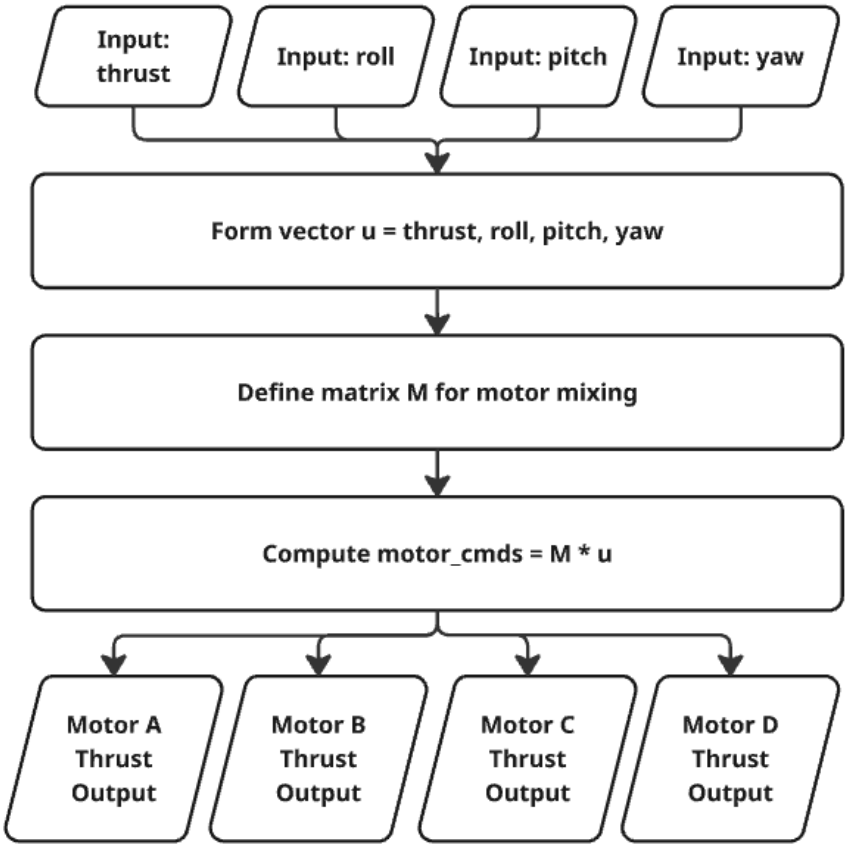


Figure 23 Motor Mixer Systematic Mechanism Flowchart

The usage of 'go to' & 'from' blocks is utilised (torque outputs to motor torque inputs), to ensure the environment is as clear as possible.

3.2 – Thrust & Altitude PID Tuning

In order for the drone to be stable as well as responsive during flight, the project implements & tunes PID controllers for each of the four key axes mentioned above: altitude (vertical thrust control), pitch (forward/backward tilt), roll (side-to-side tilt), and yaw (rotational heading control).

Each axis has been tuned independently for the custom drone through manually adjusting k_p, k_i & k_d.

Each controller followed a standard PID structure, demonstrated in the following formula:

$$u(t) = k_p e(t) + k_i \int e(t)dt + k_d \frac{de(t)}{dt} \quad \textbf{(30)}$$

Where e(t) is the error between one of four axes reference & actual value.

Regarding the tuning strategy, the following approach was conducted for the axes of the four controllers in play:

Table 1011 PID Tuning Strategy

Step 1: Proportional-Only Control
K_p was initially tuned. The tuning starts of with setting derivative & integral to 0. K_p is then gradually increased via closely monitoring the scope of the reference axes vs actual axes value; until a reasonable rise time is achieved. Over tuning, specifically a this part, would result in an overshoot and hence when an overshoot was observed, K_p was slightly reduced.
Step 2: Introduce Derivative Term
K_d is now introduced to improve damping & further reduce overshoot. K_d was likewise, increased gradually, while one simultaneously observing the systems responsiveness & stability.
Step 3: Introduce Integral Term
K_i was only configured given a ‘steady state error’ was to be visible. They are also known to reduce ‘sluggish responses’.

Below presented are the initial tuning axes, altitude conducted alongside a graph & its respective tuning parameters. The parameters have both with/without battery, w.r.t two testing phases (chapter 4).

Table 11 Attitude PID Tuning Gain Values

Gain	Value with battery	Value without battery
K_p	0.025	0.0315
K_i	0	0.0032
K_d	0.03	0.02
N	100	100

‘N’, tabulated above as a gain parameter, stands for ‘filter coefficient’; it also refers to the filter bandwidth. It is a parameter set in the prevention of noise amplification as a byproduct of tuning K_d. Given the nature of the drone, being susceptible to noise during flight, the project utilises a ‘low-pass filtered derivative’. This ensures there is less filtering and therefore closer to the pure derivative, resulting in more responsiveness & noise sensitivity.

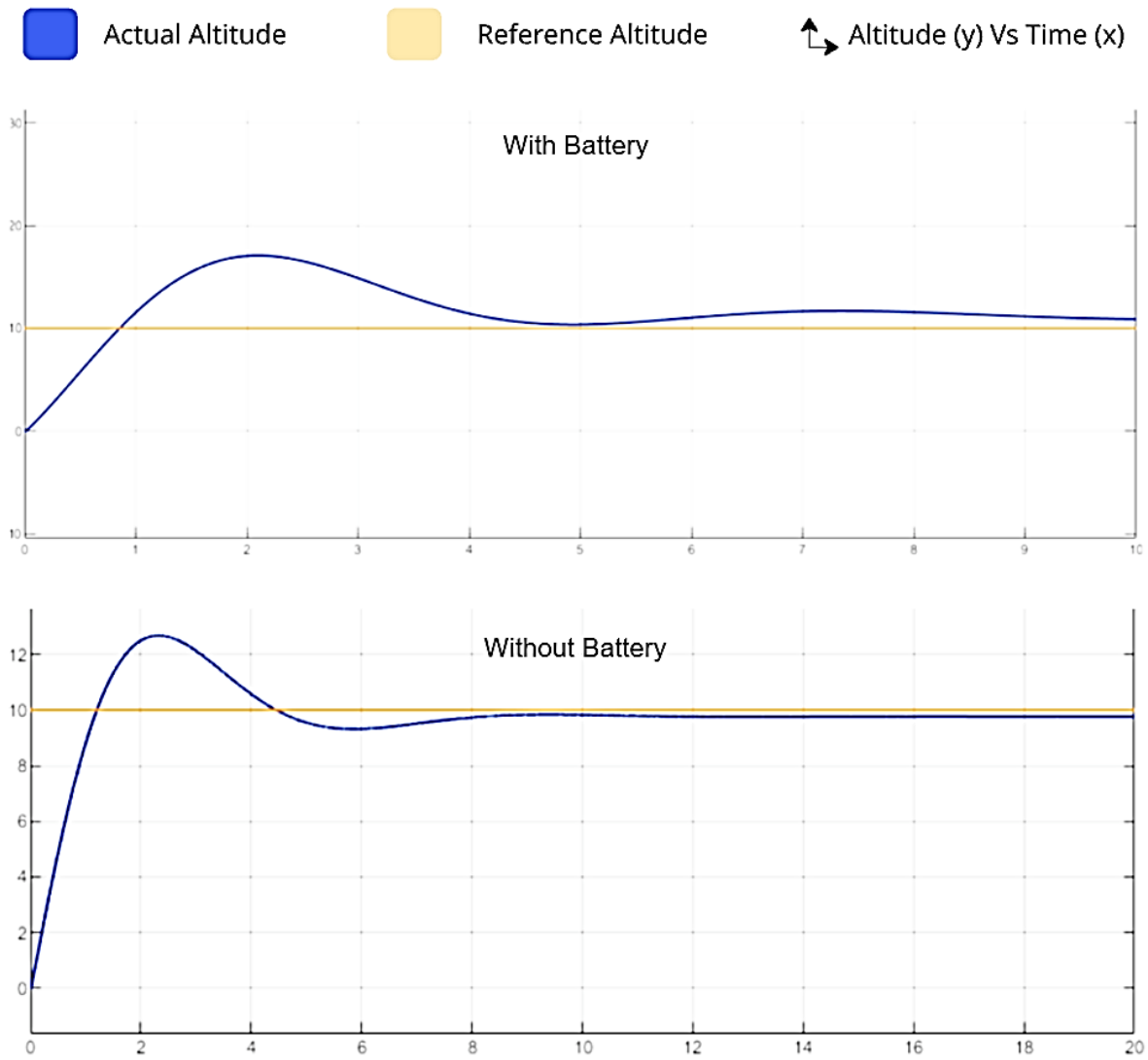


Figure 24 Altitude with 10m command PID Tuning View Graph with & without battery

Table 12 Altitude PID Tuning Analysis

Axis
Altitude
Scope Response Summary
Blue line rises quickly toward the blue setpoint, with a small overshoot & no oscillation. Final value matches close to reference.
Interpretation
Tuned for smooth, reliable vertical control with minimal amount off error.

3.3 – Yaw PID Tuning

Table 13 Yaw PID Tuning Gain Values

Gain	Value with battery	Value without battery
K_p	0.3	0.3
K_i	0.002	0.01
K_d	0.02	0.005
N	100	100

Table 14 Yaw PID Tuning Analysis

Axis
Yaw
Scope Response Summary
Yellow trace gradually follows the step input without overshoot or any form off ripple. Settles cleanly, close to the reference.
Interpretation
Prioritised stability as well as heading accuracy. Small integral term removes offset & steady state error.

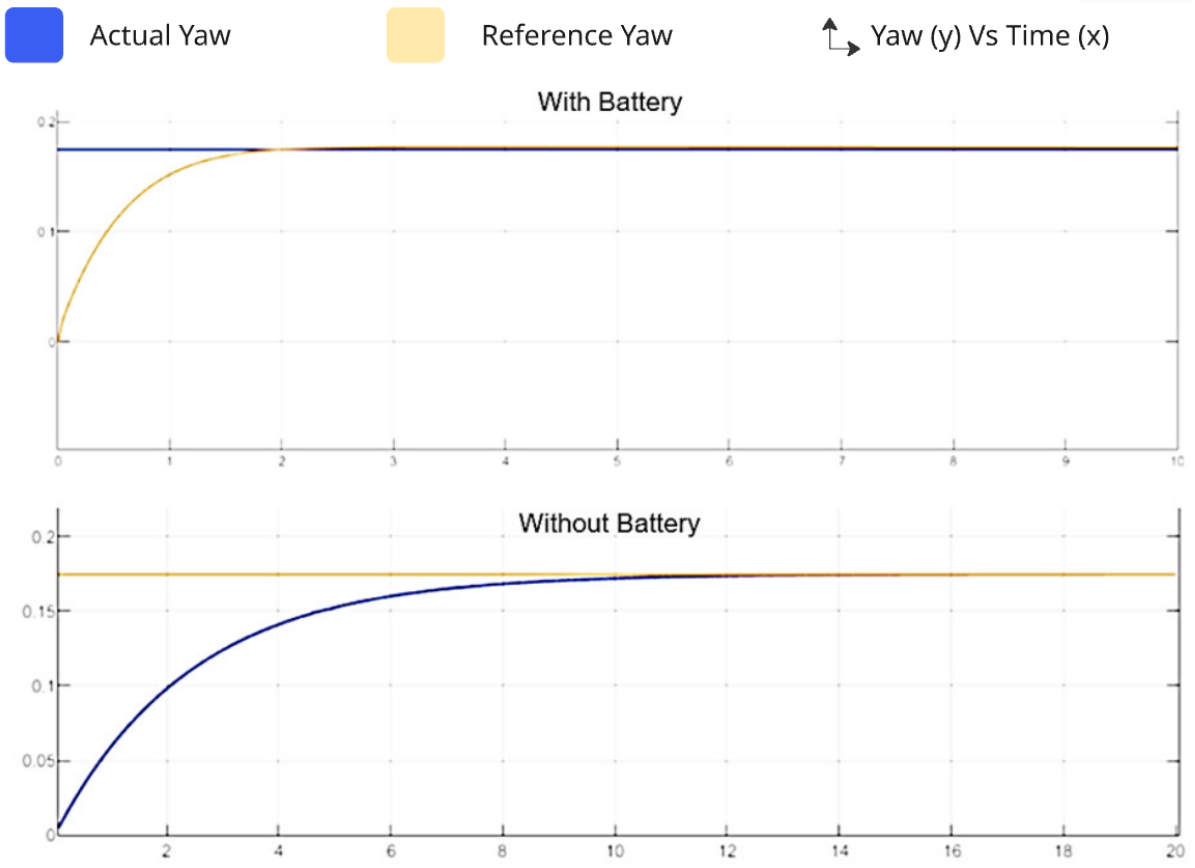


Figure 25 Yaw with 10m command PID Tuning View Graph with & without battery

3.4 – Pitch PID Tuning

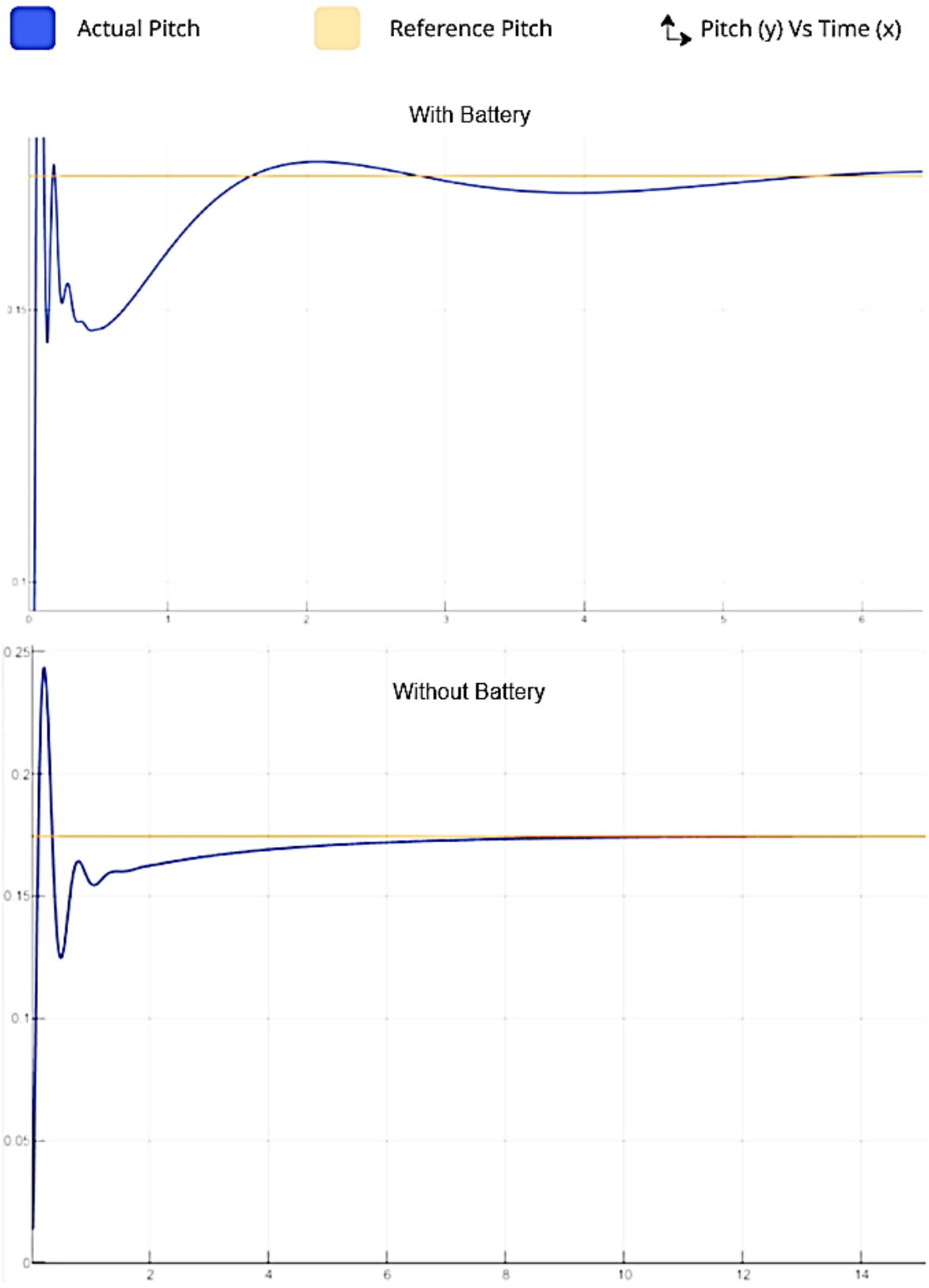


Figure 26 Pitch with 10m command PID Tuning View Graph without battery

Table 15 Pitch PID Tuning Gain Values

Gain	Value	Value without battery
K_p	0.35	0.2
K_i	0.15	0.3
K_d	0.03	0.05
N	100	100

Table 16 Pitch PID Tuning Analysis

Axis
Pitch
Scope Response Summary
Blue line responds aggressively with an overshoot as well as some brief oscillations. Eventually locks onto the reference.
Interpretation
Tuned for responsiveness; some overshoot tolerated to preserve agility.

3.5 – Roll PID Tuning

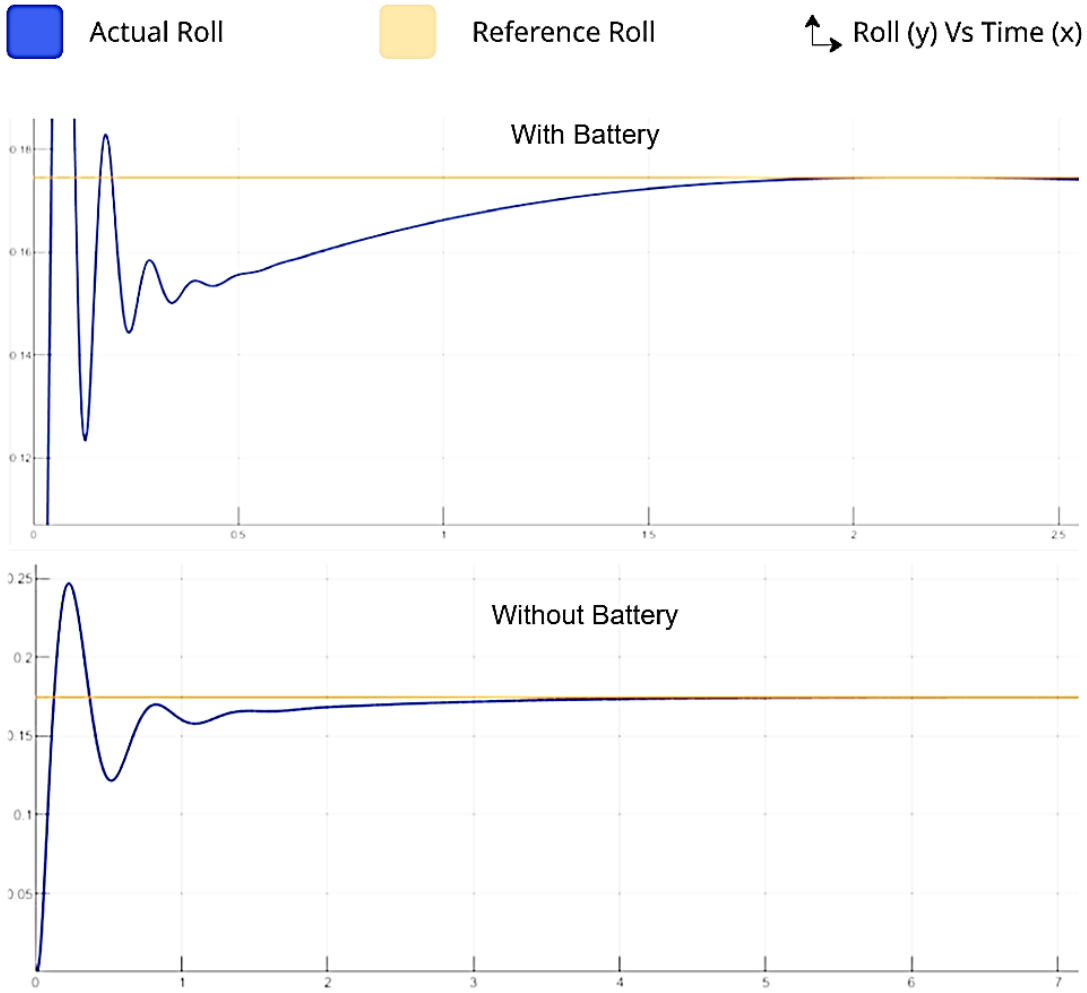


Figure 27 Roll with 10m command PID Tuning View Graph without battery

Table 17 Roll PID Tuning Gain Values

Gain	Value	Value without battery
K_p	0.35	0.38
K_i	0.3	0.4
K_d	0.03	0.04
N	100	100

Table 18 Roll PID Tuning Analysis

Axis
Roll
Scope Response Summary
Likewise, to pitch. Quick rise & overshoot, followed by damped oscillations. Tracks the reference after settling.
Interpretation
Balanced tuning for dynamic performance as well as a stable roll recovery.

Below is a schematic, presenting Yaw, Pitch & Roll commands at a set integer of 10, during the drones hoover phase:

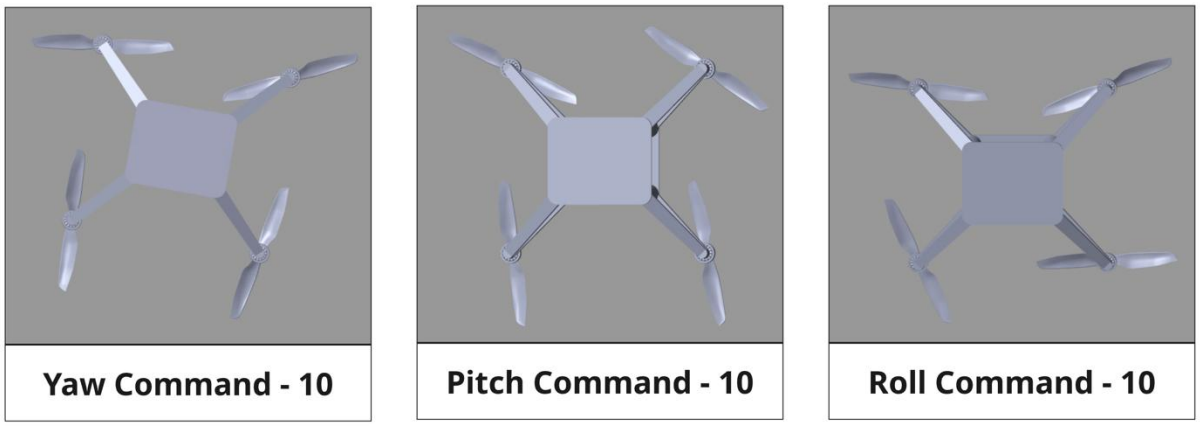


Figure 28 Thrust, Yaw & Roll Visual Schematic

3.6 The DDPG Agent

DDPG Overview

The Deep Deterministic Policy Gradient (DDPG) agent was selected as the learning algorithm for this project's RL framework, given its suitability for continuous control systems. As a model-free, off-policy, actor–critic algorithm, DDPG effectively combines the strengths of Deterministic Policy Gradient (DPG) as well as a Deep Q-Network (DQN)**.** This, in turn, allows it to learn directly from simulated flight data without any explicit mathematical modelling prior to training the quadcopter. This is also mandatory for this project's application, as the model is non-linear & under-actuated in its quadrotor dynamics nature, making analytical modelling of fault conditions extremely challenging. (Lillicrap et al., 2015)

For one to understand DDPG, one must understand the two main branches, being DPG & DQN, specific to DDPG only.

3.6.1 DPG

DPG: Here, the training attempts to directly adjust the policy, where a policy is simply how the RL agent picks actions per epoch. It involves a variant which is not probabilistic, but deterministic (as per the name), which in turn means the policy outputs exactly one action. This deterministic nature allows one to derive the 'deterministic policy gradient theorem'. The gradient allows the RL agent to determine the change required to improve the model. The great advantage of DPG, in comparison to its predecessors like stochastic policy gradient (SPG), is the following:

i) Lower Variance; methods like SPG entail a probabilistic nature. Similar to DPG, SPG executes an output from one action, but in the form of a sample. However, that action has the tendency to be noisy and therefore have a high variance; this, in result, means the gradients will fluctuate widely with different action samples, ultimately slowing down training. DPG eliminates this high variance by removing the element of sampling the action; instead, it gives off a deterministic action (computable & set).

ii) Higher Efficiency; given the samples are less noisy, as discussed above, it leads to fewer samples being required. Therefore, in cases where data collection is expensive & time-consuming, which is the case with this project's goals, DPG allows for a more efficient execution.

iii) Suitability for continuous state actions; the quadcopter currently works in a continuous action space, which results in an infinite number of actions. This is another issue that DPG eliminates. As highlighted below:

The project encompasses Q learning as its algorithm selection. The algorithm's main objective is to find the best action in each state, in order to maximise the reward, per epoch. It's also model-free, meaning it doesn't require any manual computation over the environment; instead, it learns purely from experience/training phase. Below introduces where the environment comes into play in the model.

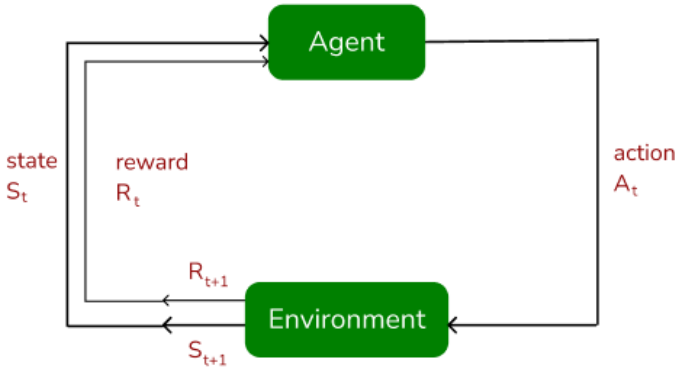


Figure 38 The Agent - Environment Loop

As per Fig. 38, the agent & environment seem to be in a continuous loop. The environment is the world that the quadcopter/agent interacts with. Its key task is to determine the conditions during the quadcopter/agent interaction with the world, in reference to what manoeuvres it can perform, what it can observe (source of feedback) and how reward is measured.

Now, given the fundamental understanding of Q learning, one can move on to its process and how DPG provides advantages to the algorithm in play. Q learning involves the creation of a Q table; this is simply a table that maps each state to a possible action as well as a Q value, a numerical estimate of the total future reward an agent can expect to receive by taking a specific action in a particular state. However, here is where the problem arises: given that the environment is presented in a 'continuous action space' (as described above), it results in an infinite amount of actions; therefore, creating the Q table turns into an impossibility.

Here is where DPG presents its advantage. Instead of creating a Q table, it bypasses the table entirely and instead creates a 'Q-function' from its neural network. This is used when an action space is too large for tabulated representation and works via approximating Q values from a discrete/deterministic set.

Having discussed the fundamentals behind how DPG operates, one can dive deeper into understanding the mechanics of DPG and how exactly the agent/quadcopter improves its actions via the Q function. (Silver et al., 2014)

3.6.1.1 Actor & Critic Networks

DPG introduces two neural networks, the actor & critic. In simple terms, the 'actor' decides which action to take & the 'critic' acts as a source of feedback, in telling the actor how good its action was during the last epoch and how much it should potentially adjust its policy in the next epoch.

Below, custom fig.39 describes how both neural networks merge with the environment. As shown below, the Actor Network Policy receives the drone's state vector, which includes roll, pitch, yaw, altitude, and velocity components. Based on this input, the Actor outputs thrust commands for the three remaining healthy rotors (A, C, and D) during a motor failure 'B' scenario. These thrust values are then applied to the Simulink Environment, which goes ahead into modelling the drone's 6-

DOF body dynamics, rotor allocation, battery repositioning, as well as onboard sensors.

The environment then returns an updated state & a reward signal, indicating how well the drone maintained stability & altitude. This information is processed via the Critic Network, which estimates the Q-value of the Actor's chosen action. The Critic computes a Temporal Difference (TD) error as an output, representing the gap between the expected & actual rewards. This TD is then used to update both respective networks accordingly for the next epoch, where the Critic refines its internal value estimates, while the Actor adjusts its policy to ensure future rewards increase further.

Through this continuous feedback loop, the Actor–Critic architecture enables the quadcopter to learn stable control policies in a continuous action space**,** ultimately improving the recovery behaviour following rotor failure 'B'.

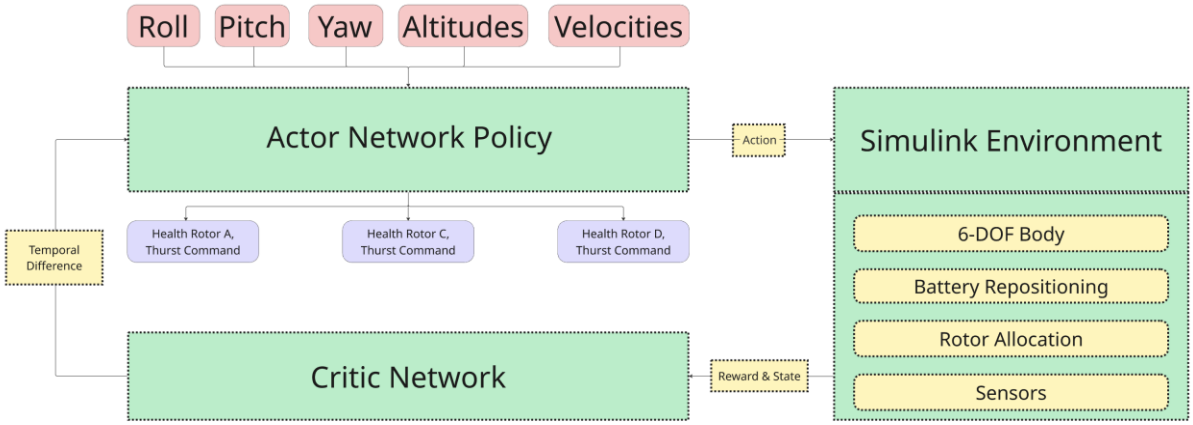


Figure 39 Actor & Critic Loop

3.6.1.1 DQN

Having now understood how DPG forms the foundation for deterministic, continuous policy learning through its actor–critic structure (discussed above), it is equally as important to recognise the stabilisation techniques that DDPG entails from DQN.

Given DPG focuses on how actions are generated as well as refined in continuous spaces, DQN involves mechanisms that make this learning process more stable and efficient, primarily via the use of 'replay buffers' (1) & 'target networks' (2)**.** Both aim at preventing instability as well as overfitting during RL.

- Replay Buffer and State Transitions (1)

Another technique inherited from DQN is the 'experience replay buffer'. This stores all the interaction data between the agent & environment. Each time the RL agent interacts with the quadcopter, it produces a state transition, which contains four essential elements that form the foundations of its learning storage:

i) s - Current State (observation) - roll, pitch, yaw, altitude, velocities, etc.

ii) a - Action Taken - thrust outputs for the three healthy rotors

iii) r - Reward Received - a numerical score based on how stable the drone was

iv) s' - Next State - the new drone state after applying that thrust

This structure allows the agent to recall and learn from a collective amount of past experiences, in comparison to relying solely on the most recent epoch.

- Target Networks & Stability (2)

In order to improve training stability, **'**target networks' are utilised. These target networks were maintained as slow-moving copies of the primary actor & critic. In comparison to updating immediately with every epoch, the target networks are set to evolve gradually via a soft-update factor of 0.001. This, in turn, ensures smooth convergence & prevents oscillations during RL. Conceptually, this can be compared to a 'step-block' in Simulink, where small incremental updates lead to a stable overall set trajectory (defined in block settings).

3.6.1.2 DDPG Crucial Extensions, Alongside DPG: Ornstein–Uhlenbeck Noise (1), Random Sampling (2)

2 additional mechanisms were employed in order to further enhance exploration, leading to training stability in the long run…

- Ornstein–Uhlenbeck Noise (1)

DPG is deterministic. This means the actor always outputs the same action for a given state; in this case, being commanded to thrust the three rotors. In order to explore new behaviours during training, DDPG adds OU noise to the actor's actions. This, in turn, allows the drone to try slightly different thrust combinations during learning, in hopes of discovering better solutions. After training stabilises, the noise is turned off so the drone acts deterministically, once again.

The noise produces a smooth, correlated variation, referred to as 'correlated noise'. In comparison, the noise produced can be purely random. However, if random fluctuations were in play, the drone would receive unstable signals per epoch. Therefore, via the usage of correlated noise, the setup is able to represent the normal flight of a drone more closely (where thrust, torque, & angular acceleration changes are gradual, not instantaneous.

- Random Sampling (2)

In order to further enhance learning reliability, the utilisation of a 'replay buffer' is accessed through random sampling. Instead of feeding sequential data back into the training process, the agent randomly selects batches of past epochs.

It’s similar to OU noise; however, both random sampling & OU noise occur at different phases in the DDPG process. Our noise is executed during the actor network phase; it helps the actor discover better thrust combinations by adding different levels of noise each epoch. Whereas, Random sampling occurs in the critic training process, ensuring ‘decorrelated training’ via random past experience selections.

All in all, the randomisation will assist in breaking the temporal correlation (near consecutive experience). As a result, it reduces the risk of overfitting to recent data; the model learns the training data too well, so that when it comes to unseen data, it’s unable to predict a suitable action. (Fujimoto et al., 2018)

4.0 The Training Ground

Chapter Abstract

In this chapter, the project explores the 2 phases of training, executed on the custom drone. This involves initially training it to adjust rotor thrust outputs, given one rotor fails; the desired outcome would involve the drone being able to sustain a malfunctioning rotor via holding its altitude.

The next phase of training involves the battery moving in sync with the rotor speed adjustments. The desired outcome, in this specific phase, would involve both the drone being able to maintain altitude & simultaneously be close to perpendicular with respect to the ground. (via shifting CoG).

Prior to initiation of any of the 2 phases above, the fundamental environments from PID above need to be altered & tailored…

4.1 Training Environment – Phase 1

Phase One Objective:

> Produce a fully functional Simulink environment to train a custom-built drone in the event of a rotor failure. The environment must be able to detect a rotor failure, and upon that failure, switch from PID controls to an RL agent. The RL agent must be able to adjust the remaining rotor thrust to maintain drone altitude.

As described above, the environment required for phase one is built via the foundations of the previous environment (appendix 6), which allowed the drone to be controlled via 4 PID-tuned controllers: altitude/thrust, pitch, yaw & roll controllers. Hence, below will describe the additional elements, as well as tweaks to the original environment that were conducted.

[1] Propeller Motor Failure Logic

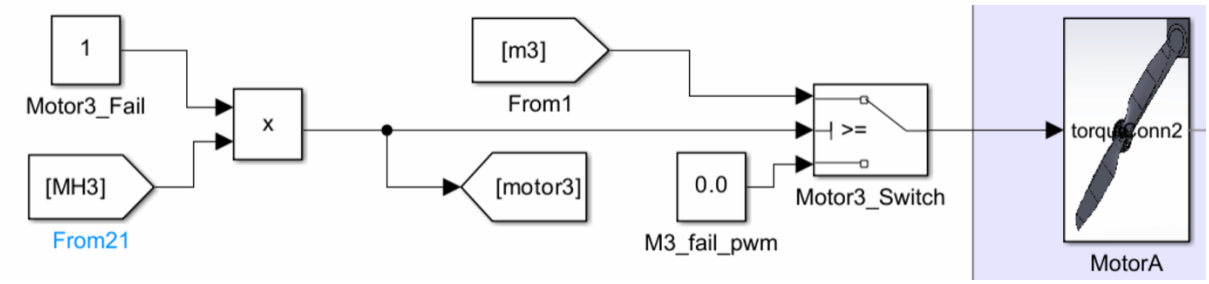


Figure 29 Propeller Motor Failure Logic

To enable rotor-failure training, a motor-fault logic block was added to the environment, as shown in fig.28. Its purpose is to enable/disable torque transmission to a specific motor. In return, this allows the environment to replicate the sudden failure of a rotor during flight. When the failure flag is active, the torque signal to the corresponding motor is set to zero; as a result, it forces the PID controls switch into RL agent controls. Once the controls are switched from PID, the RL agent's role is to stabilise the drone, using only the remaining three rotors.

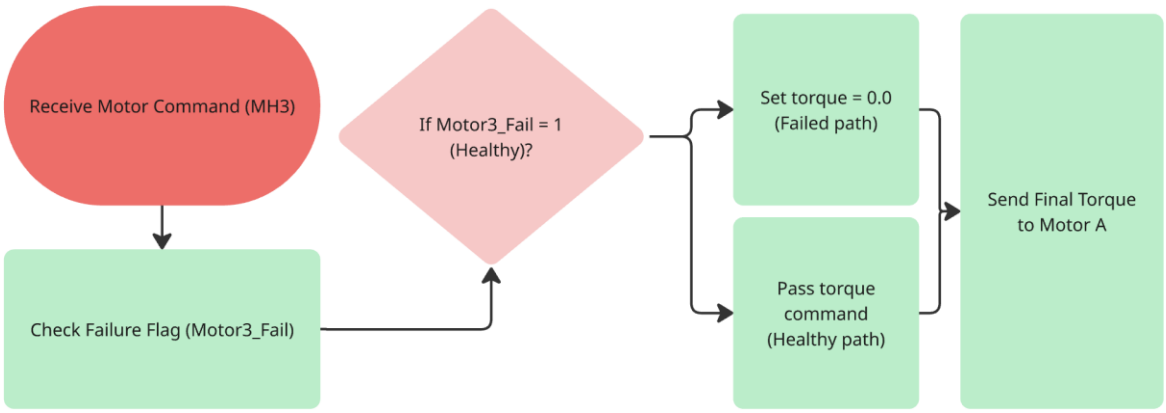


Figure 30: The rotor health check mechanism

The flowchart above in fig.29 illustrates the work behind the propeller health check. Note, the failure is triggered via a script, which is executed during the training phase. Therefore, depending on when a specific altitude is reached in normal flight, the drone will then experience a rotor failure, resulting in a torque of 0 N (set torque = 0.0).

[2] The PID to RL Agent Switch

The next key element inserted into the environment is the switch, which switches from PID to RL agent, given that a motor health is set at 0, in other words, a rotor failure has occurred. Figure 30., represents the switch alongside the RL agent environment:

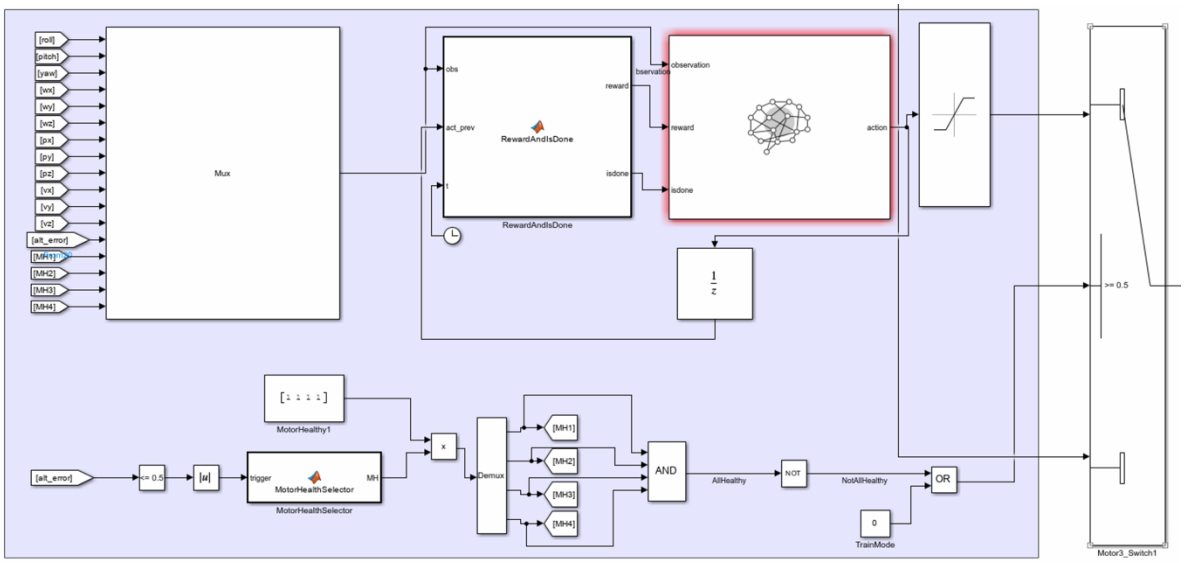


Figure 31 RL Agent & Switch Simulink Environment

The switch consists of 2 ports and a condition under which it is activated. The bottom switch goes into the PID mechanics. At the same time, the top switch leads to a saturation block, in line with the RL agent mechanics. The saturation block here allows for a smooth transition between PID & AI, ensuring the model isn't prone to collapsing, as well as to avoid a singularity error, caused when a drone loses its ability to continue with its flight due to it losing a degree of freedom, which is susceptible in the event of rotor failure.

[2.1] Switch 2 – RL Agent

Via the reinforcement learning block add-on, in MATLAB Simulink, one can insert an RL agent block, which allows a user to define an environment, configure the agent's behaviour and generate actions based on a set reward system. Currently visible in the RL agent block, there are inputs: observation, reward & isdone. It has a single output 'action', which is fed directly into the saturation block, discussed above.

The observation input requires a user to provide information regarding its current state from a simulation perspective. This leads to its function, of utilising this information, to decide on a suitable action to take for the next iteration/epoch. Specific to the project environment, a mux block is fed into this system, consisting of positions, velocities, motor health status, roll, pitch, yaw and the altitude error. An absolutely crucial element visible is a 'unit delay' block, outputted from the RL agent. This allows the agent to store the previous value, which leads to its ability in computations regarding penalising large/sudden changes in control, as well as encouraging smoother, energy-efficient behaviour. Without the presence of the unit delay block, the system will use the current action in play, which simply destroys the element of having a feedback loop, mandatory in RL.

The isdone as well as reward are defined in a neighbouring MATLAB function block, presented below in fig.30, in the form of a flow chart.

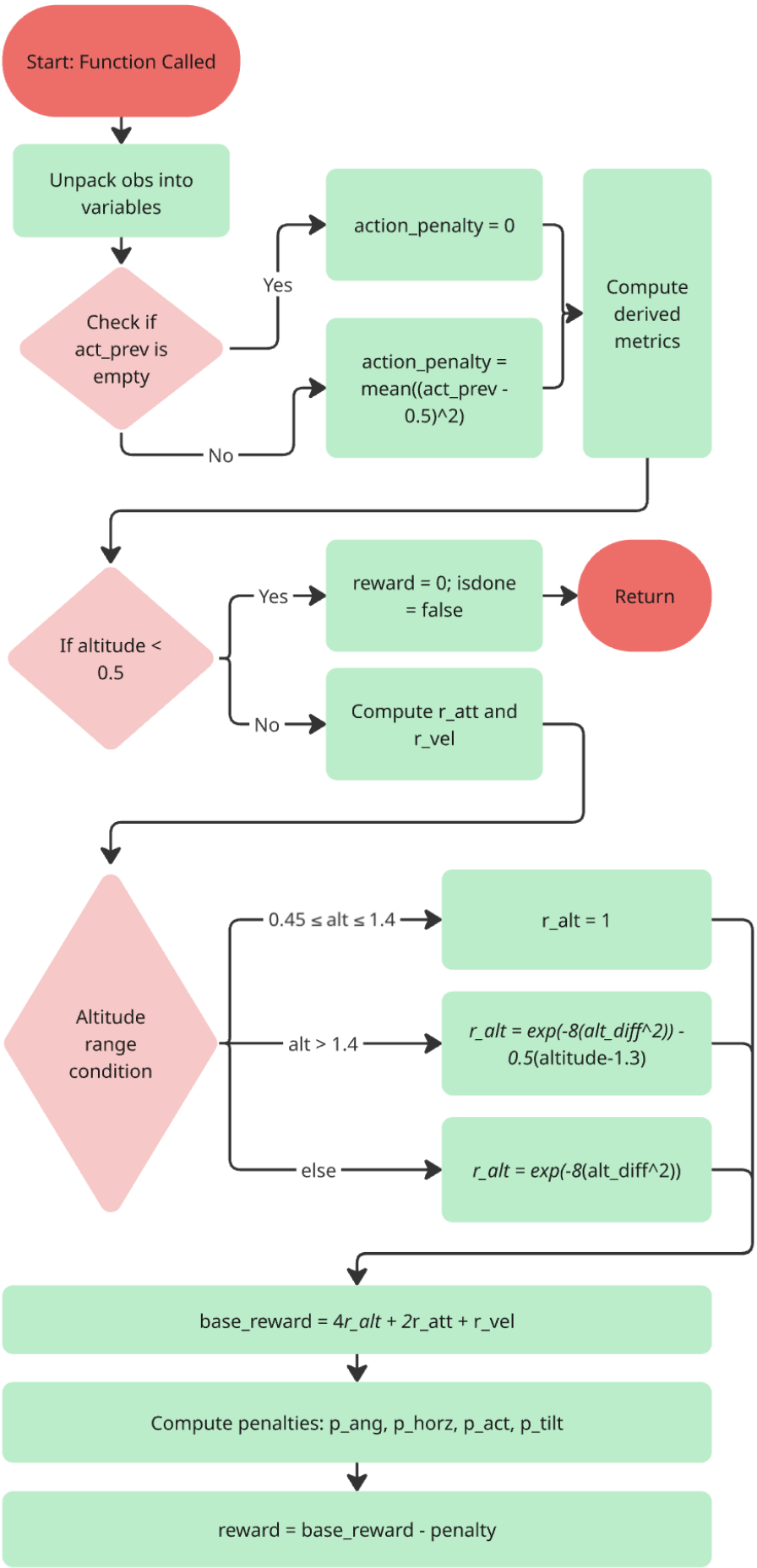
[2.2] RL Agent Reward System MATLAB function block

Prior to the reward function flowchart, one must understand the core uses of the isdone function as well as the reward element itself.

Reward signal works as a key identifier in feeding the agent with how good/bad a specific epoch is, through a set numerical penalty set. Therefore, the goal of the RL agent is to maximise the reward, iteration after iteration…

The isdone signal is referred to as a 'logical flag' which indicates when a specific simulation epoch has come to an end. When an isdone signal is set to 'true', the environment will reset & will lead to the RL agent adjusting weight gains for the next episode, based on the last epoch's performance.

Below represents the logic behind the reward function block:


Start: Function Called
Unpack obs into variables
Check if act_prev is empty
Yes
No
action_penalty = 0
action_penalty = mean((act_prev - 0.5)^2)
Compute derived metrics
If altitude < 0.5
Yes
No
reward = 0; isdone = false
Return
Compute r_att and r_vel
Altitude range condition
0.45 ≤ alt ≤ 1.4
alt > 1.4
else
r_alt = 1
r_alt = exp(-8(alt_diff^2)) - 0.5(altitude-1.3)
r_alt = exp(-8(alt_diff^2))
base_reward = 4r_alt + 2r_att + r_vel
Compute penalties: p_ang, p_horz, p_act, p_tilt
reward = base_reward - penalty

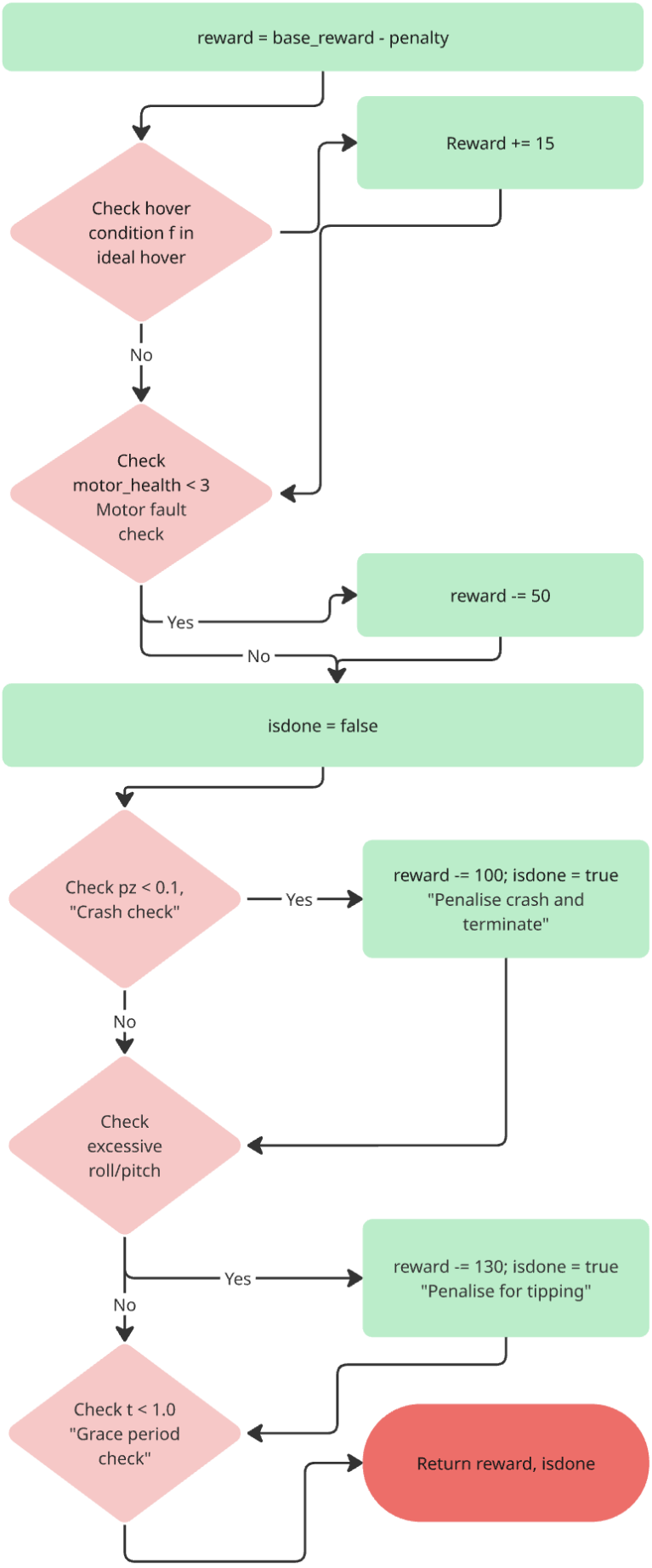


Figure 32 Reward system function block

As per flowchart fig.31, the reward function governs the agent's learning behaviour via quantifying flight stability, altitude control, & fault recovery performance.
The algorithm evaluates each simulation step through a sequence of conditional checks covering altitude, velocity, attitude, and motor-health status.

Initially, observation variables are unpacked, and penalties for abrupt control actions are computed. The function then commutates the intermediate rewards for maintaining a desired altitude as well as low angular velocities.
If the altitude were to drop below the safe threshold (0.5 m), the episode resumes without reward accumulation, hence encouraging recovery rather than early termination.

Subsequent branches handle graded altitude conditions:

- Nominal range (0.45 – 1.4 m): full reward.
- Above range: exponential decay penalty.
- Below range: soft penalty encouraging lift correction.

A base reward is formed via weighting altitude, attitude, and velocity terms, from which angular, horizontal, and action penalties are subtracted.

Final conditions enforce safety:

- Hover stability gains a bonus (+15).
- Rotor failure incurs a penalty (−50).
- Crashes ($p_z < 0.1$) or excessive roll/pitch trigger heavy penalties and episode termination (−100 / −130).

This layered reward structure ensures the agent progressively learns, in hopes of maintaining stable hover, adapting thrust redistribution after rotor loss, & avoiding unsafe flight attitudes.

Below represents a summarised formulation of the reward function, alongside an abbreviation for support, tabulated:

Table 1219 Support abbreviation table for summarised reward formula

r_{alt}	Reward for maintaining altitude close to target
r_{att}	Reward for keeping stable attitude (roll, pitch, yaw)
r_{vel}	Reward for low velocity error
P_{ang}	Penalty for large angular deviations
P_{tilt}	Penalty for excessive tilt
P_{horz}	Penalty for horizontal displacement
P_{act}	Penalty for high control effort
R_t	Total reward function

$$R_t = 4r_{alt} + 2r_{att} + r_{vel} - (P_{ang} + P_{horz} + P_{act} + P_{tilt}) \quad \textbf{(31)}$$

Note: The respective weighting coefficients (4, 2, 1) in equation 31 were empirically chosen to prioritise altitude regulation over attitude & velocity damping.
This, in turn, ensures that during fault recovery, the agent initially learns to maintain lift, prior to optimising lateral or angular performance.

Below represents equation 32 with conditional adjustments R_{t_c}, stated without the flowchart fig.:

$$R_{t_c=} R_t + 15\ (if\ hover\ stable) - 50\ (if\ rotor\ fault\ detected) \\ -100\ (if\ crash) - 130\ (if\ tipping) \quad \textbf{(32)}$$

[2.3] RL Agent Switch Trigger

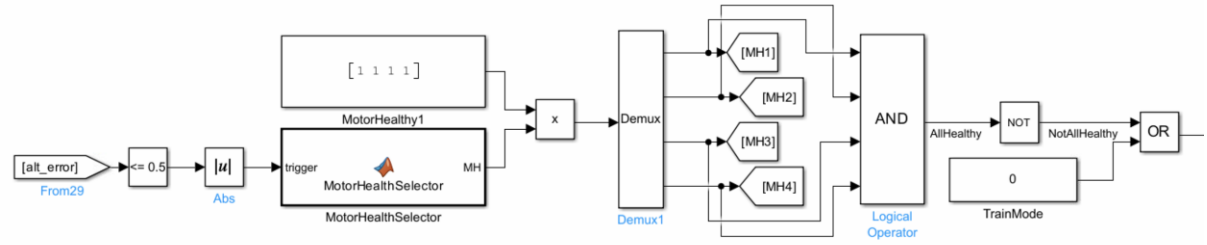


Figure 33 Switch Condition

The above presents the switch condition mechanics, controlling the transition between PID controls to the RL agent, given that a rotor fails. It continuously evaluates the 'system's altitude error' & 'motor-health states', to decide when the RL controller replaces PID controls.

Initially, the sub-system presents itself with the following stream:

[alt_error] → [Abs] → [Constant (0.5)] Here, the absolute altitude error is compared with a 0.5 m threshold. If the deviation is to exceed this value, it in turn indicates a significant instability, often corresponding to a motor fault onset.

Moving on, the present is a 'Motor Health Selector' MATLAB function block. For simplicity, the project hasn't got a detector in place to detect a rotor failure. Instead, the project focuses on specifically disabling motor 2; this applies to both phases 1 and 2. However, given that the project involves the future ability for scaling, the logic employed allows for checking all rotor health.

Hence, the logic of the function block:

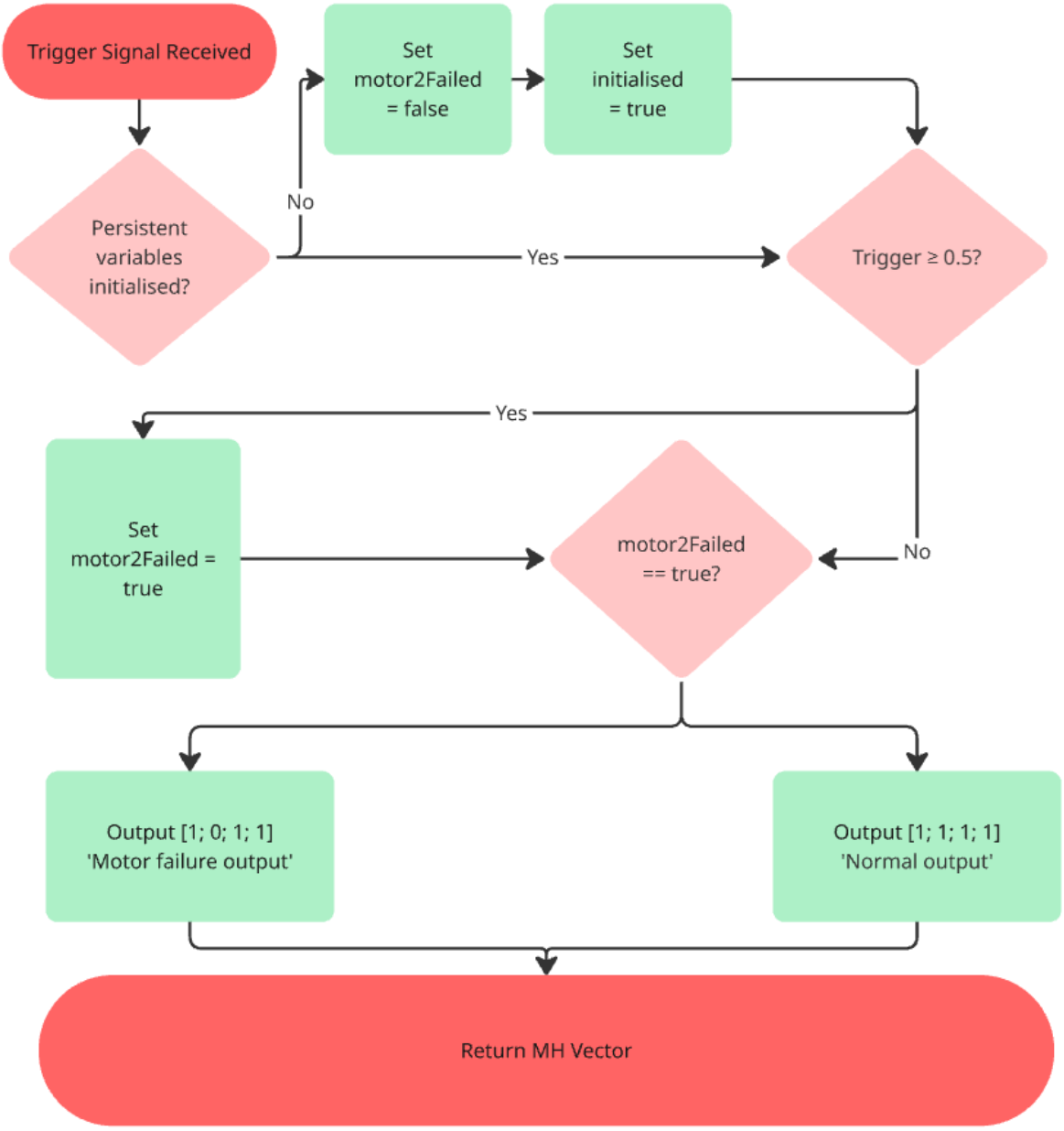


Figure 34 Motor Health Selector, MATLAB function Block, Logic

The flowchart in fig.33 presents the internal flow of the MotorHealthSelector function. When a trigger signal is received, the block first checks if the persistent variables have been initialised. If not, it sets the failed motor (e.g., motor 2) to false & initialised = true, defining the starting condition. Once initialised, the trigger value is continuously monitored. If the trigger is to reach a value >0.5, the function sets motor2Failed = true, in return, permanently flagging Motor 2 as failed. Depending on the nature of the flag, the block then outputs one of two motor-health vectors: [1 ; 1 ; 1 ; 1] for normal operation or [1 ; 0 ; 1 ; 1] once failure occurs, where '1' is healthy and '2' is failed. Below represents the motor vector representation:

$$[motor\_1, motor\_2, motor\_3, motor\_4] \quad (33)$$

The following loops, ultimately ensuring the failure state across time steps persists, providing a consistent & repeatable fault event that activates the RL control phase.

Alongside the Simulink environment logic, the MATLAB workspace included three key script files supported by a short README explaining how to initiate training; refer to the appendix to view the respective GitHub repository.

The create_env_and_agent.m script is responsible for constructing the RL environment as well as defining the DDPG agent architecture. It specifies both the observation and action spaces. These build the actor and critic neural networks, configure noise and learning parameters, and link the Simulink model to MATLAB through the rlSimulinkEnv interface.

The completedfinished_DataFile.m file, automatically generated from Simscape Multibody, contains all the imported physical parameters of the drone model, such as rigid-body transforms, mass properties, and joint definitions. All in all, this ensures the simulation accurately reflects the SolidWorks geometry and dynamics, defined on SolidWorks.

Finally, the resume_training_motor2.m script acts as a manager of the training process itself; it loads the latest saved agent checkpoint and then, via user request, resumes training from the most recent successful weights, and saves progress in batches of 100 episodes; therefore, as mentioned above, it removes the element of error mid-way during the 3000 total epochs.

4.2 Phase One: Rotor Speed Change

This phase focuses on training the reinforcement learning (RL) agent to stabilise the quadrotor after a rotor malfunction via intelligently adjusting the speed of the remaining healthy rotors. The objective is to enable the agent to have the ability to restore balance & maintain altitude after a simulated failure of Motor-2 during flight.

In order to ensure crash-free training, the model was trained in epochs of 100 episodes instead of one continuous session. This strategy was chosen because Simulink-based reinforcement learning environments can occasionally crash due to solver instability or high system stiffness. The solver may fail either due to its incapacity to recompute the next epoch, or the episode involves the drone reaching a singularity error instantly (loses a degree of freedom). Via division of the training into smaller batches:

- The model could be safely stopped as well as resumed after each epoch.
- The agent’s weights were saved more frequently. In return, it prevents total data loss in the event of a crash.
- The stability of the solver was maintained, avoiding accumulation of numerical errors.
- Progress was monitored at consistent intervals, allowing for the tuning of reward thresholds & training options.

Each episode represented a full simulation run of the drone starting from rest, taking off, experiencing a rotor failure at *t = 1.0 s*, and then attempting to recover using learned control actions, as represented in the storyboard below:

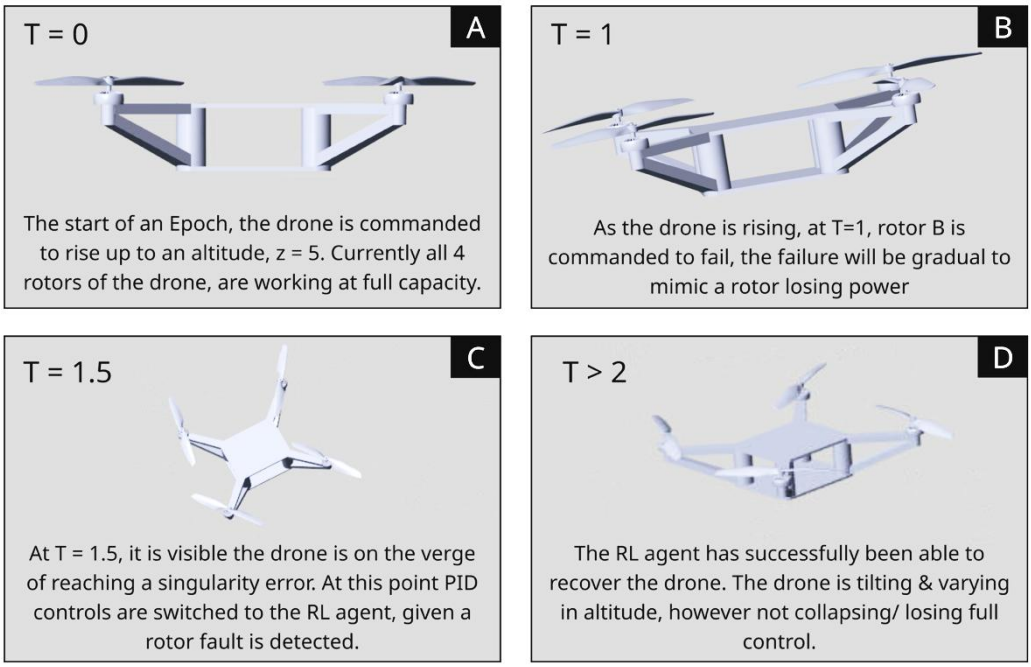


Figure 35 Storyboard of Phase 1, Completed AI Training

The termination/isdone logic penalised the agent if altitude dropped below 0.1 m or if roll/pitch angles exceeded ±70°, ensuring that the agent learned to maintain both stability and sufficient lift.

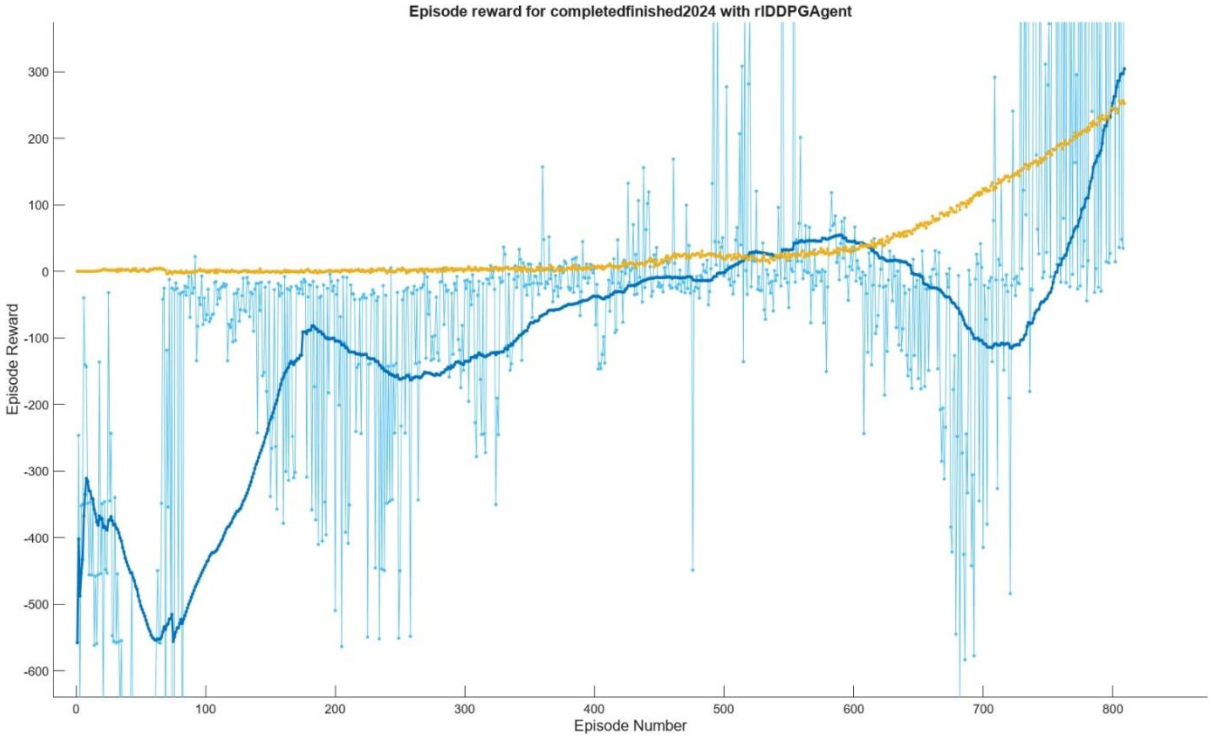


Figure 36 Phase One Training graph

The reward curve shown in fig.35 indicates that training in smaller epochs led to safer and more consistent convergence, with high-reward peaks corresponding to successful recovery episodes. The average reward increased gradually as the agent explored effective thrust-redistribution strategies.

Over 50 initial episodes, the agent demonstrated mixed performance with intermittent spikes in reward values (up to ~3000), corresponding to stable recovery after rotor failure. The average episode reward (~800) showed a general upward trend, signifying progressive learning. Early negative rewards (around –60 to –100) reflect initial instability and crashes before the policy began to generalise.

In essence, upon a single rotor failure, the rotor diagonal to the failure must increase thrust severely, to counteract the tilt caused on the failure side. However, this comes with the side effect of yaw, given that the remaining two rotors spin in the same direction. To counteract this slightly, the diagonal drone must gradually accelerate and decelerate, ensuring the tilt as well as the yaw can be reduced to the bare minimum. Likewise, the opposing diagonals will have to vary speed to assist in allowing the drone to reach its target with the least yaw.

One may have the misconception that the diagonal of the failed motor must decrease, as increasing thrust would work in favour of the tilt caused by failure. However, the correct response is the opposite: the diagonally opposite rotor must increase thrust to create a combined roll–pitch counter-torque around the CoM. Which ultimately raises the drooping quadrant. As discussed above, the remaining two rotors oscillate to cancel the resulting yaw and to maintain total lift, which is evident through the constant fluctuation in the off-diagonal motors A (green line) & C (red line); in fig.36.

In support of this, further graphical evidence was extracted via scope blocks in the Simulink environment:

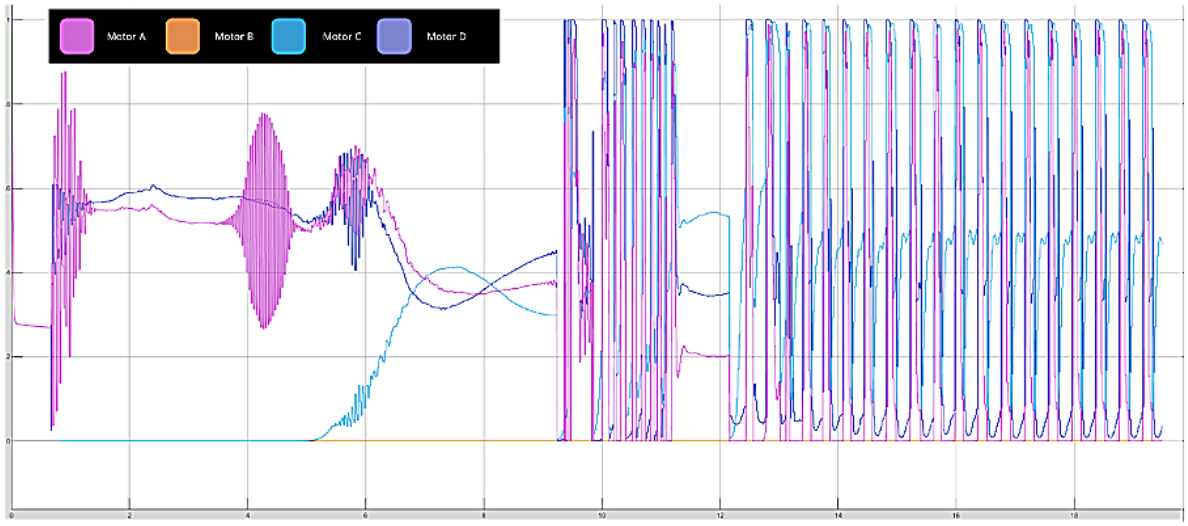


Figure 37 Snapshot of rotors 1-4 performance, during phase training

The plot presented above in fig.36 illustrates the individual motor speed responses following a rotor failure event. The orange trace (Motor B) rapidly drops to zero,

hence confirming a simulated motor failure. Immediately after this loss, the diagonal motor (Motor C, blue) increases its output to counter the roll and pitch imbalance created on the failure side. However, because this compensation generates a strong yaw moment, the remaining pair of rotors (Motors A and D) engage in rapid alternating fluctuations, visible as high-frequency oscillations. The fluctuations are visible to cancel the yaw torque while maintaining altitude dynamically. The pronounced oscillations between roughly *t = 9 s to 15 s* show the controller's continuous attempts to stabilise both attitude and heading using only three functioning rotors. As the system adapts, the oscillations gradually settle, indicating that the reinforcement-learning policy is redistributing thrust in a coordinated yet highly responsive manner. One could argue that if the model were further trained, the fluctuations would be less evident; however, given that the model was trained for a long duration and met the expected criteria, the model had proven itself a success.

Furthermore, given the validity of phase 1, the following phase 2 can be trained, using the present phase's weighting as a foundation.

4.3 Training Environment - Phase Two

Phase Two Objective:

Produce a fully functional Simulink environment to train a custom-built drone in the event of a rotor failure. The environment must be able to detect a rotor failure, and upon that failure, controls must switch from PID controls to an RL agent. The RL agent must be able to both balance out thrusts with the remaining healthy rotors & move the battery to re-adjust CoG.

From sub-chapter 4.1, the project demonstrates a fully working environment for phase one, which involves RL to adjust remaining rotor speeds, given that one rotor fails. Here, the project moves onto RL for both rotor speed adjustments as well as battery location adjustments. The idea is that the rotor speeds, as executed in sub chapter 4.2, will allow the drone to hold an altitude; however, a tilt will be present, as evident in the supporting schematics (sub chapter 4.2). Therefore, to eliminate the element of tilt, the RL agent in phase 2 involves battery movements, which aim to change the CoG.

The following elements, in addition to the entire environment setup, during phase one, were set up, to perform phase 2:

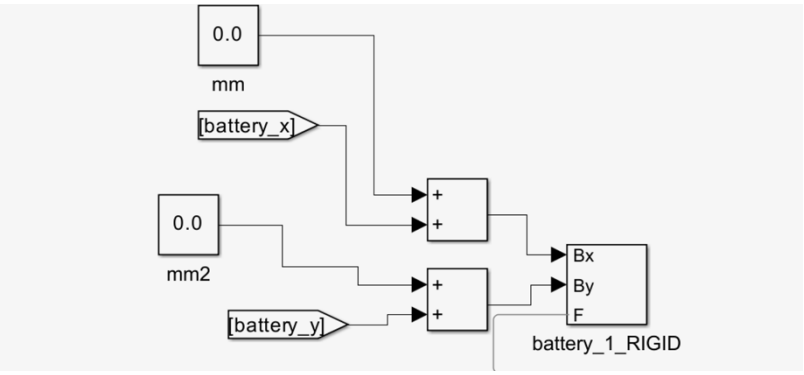


Figure 38 The Battery Frame Connection & Mechanics

Fig.38 highlights two important aspects of the battery integration. One, connect with the solid body (battery), with the central frame of the drone. Two, summation boxes which can alter the initial battery starting points, in addition to RL agent commands (via usage of go to/from block). Prior to setting up this environment, the battery was re-imported from SolidWorks, after CoG battery alignment, with frame alignment.

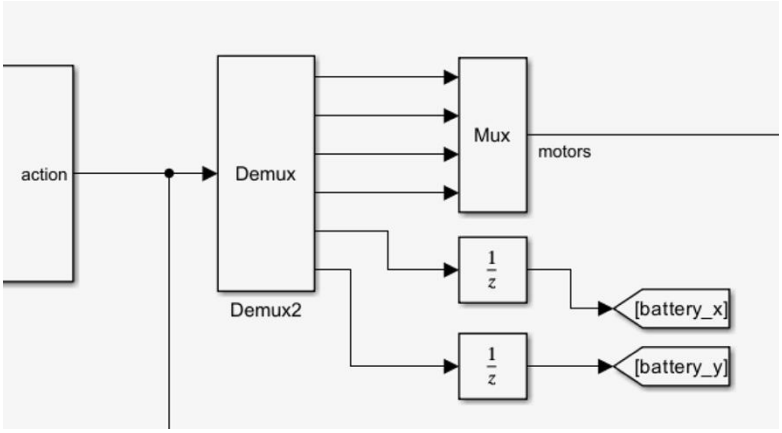


Figure 39 The Battery RL Agent Switch Merge

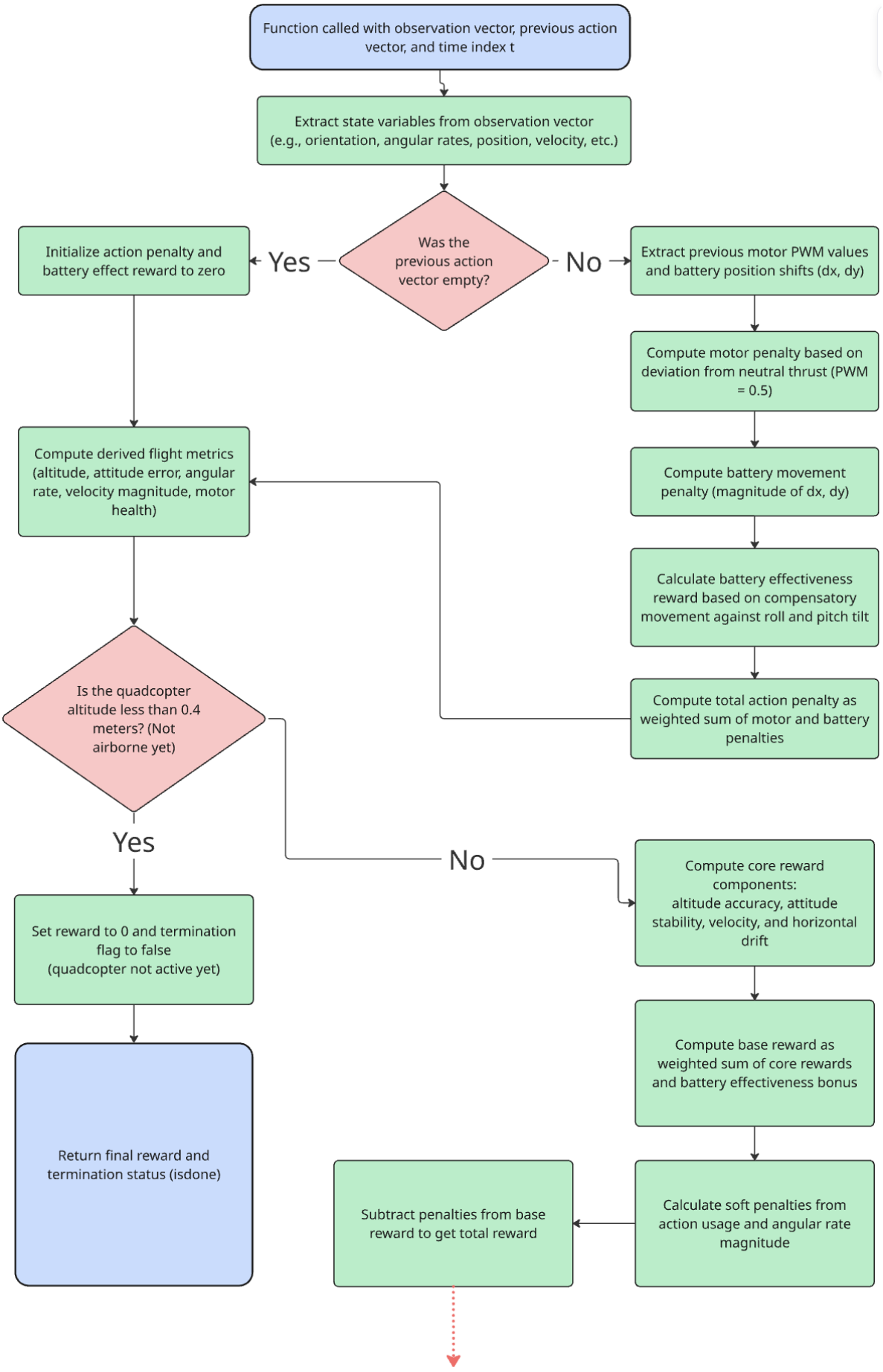
Fig.39 presents the RL merge between the battery and the initial phase 1, rotor adjustment sub-system. It also features the go-to blocks, which lead to from-to blocks in fig.38. Lastly, the environment here has 2-unit delay blocks before sending out movement commands to the battery on the x & y axes, respectively. These delay blocks allow for a more realistic system representation, via tackling three significant aspects of latency:

1) Communication latency: Time required for the command to travel from the RL agent to the battery movement actuators.

2) Execution latency: The Time the actuators take to react to commands from the RL agent.

3) Sensor Latency: Time taken for the battery to move to its new position and then be sensed via the system to provide an element of feedback.

The remaining element that differentiates phase 2 from phase 1 is the reward system. The following figure.40 presents a flow chart for the new reward system in play:


Function called with observation vector, previous action vector, and time index t
Extract state variables from observation vector (e.g., orientation, angular rates, position, velocity, etc.)
Was the previous action vector empty?
Yes
No
Initialize action penalty and battery effect reward to zero
Extract previous motor PWM values and battery position shifts (dx, dy)
Compute motor penalty based on deviation from neutral thrust (PWM = 0.5)
Compute battery movement penalty (magnitude of dx, dy)
Calculate battery effectiveness reward based on compensatory movement against roll and pitch tilt
Compute total action penalty as weighted sum of motor and battery penalties
Compute derived flight metrics (altitude, attitude error, angular rate, velocity magnitude, motor health)
Is the quadcopter altitude less than 0.4 meters? (Not airborne yet)
Yes
No
Set reward to 0 and termination flag to false (quadcopter not active yet)
Return final reward and termination status (isdone)
Compute core reward components: altitude accuracy, attitude stability, velocity, and horizontal drift
Compute base reward as weighted sum of core rewards and battery effectiveness bonus
Calculate soft penalties from action usage and angular rate magnitude
Subtract penalties from base reward to get total reward

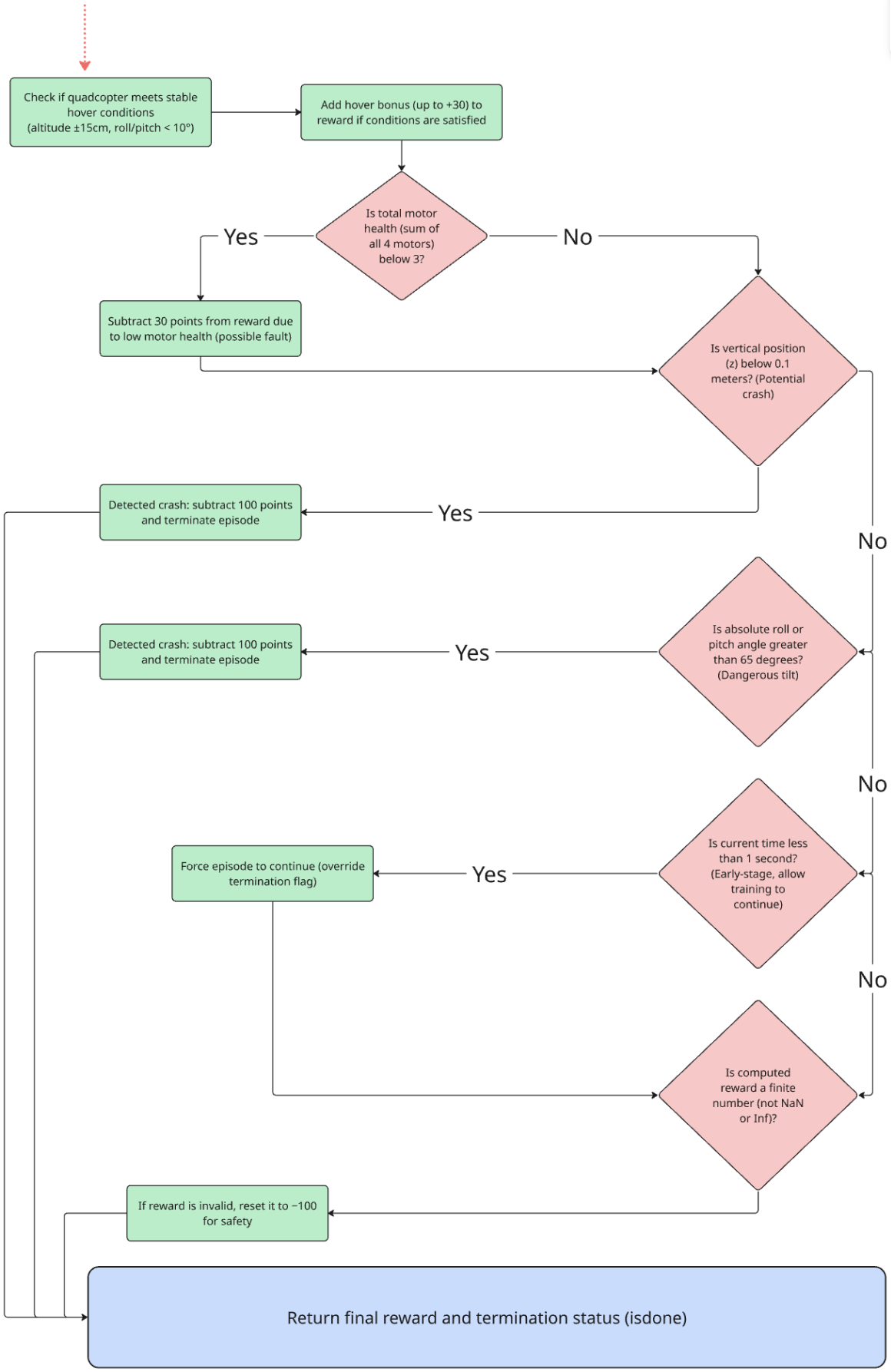


Figure 40 Phase 2 Reward Mechanism Flow Chart

The flowchart above (fig.40) represents the updated reward logic, executed during Phase 2. Structurally, it remains largely consistent with Phase 1's reward mechanism, in maintaining the core components such as altitude reward, attitude stability, velocity, and horizontal drift control. However, the key addition in this phase is the inclusion of a battery repositioning mechanism**.** This introduces both a reward and a penalty related to battery movement.

Specifically, the battery reward term (batt_eff) is designed to encourage corrective movements of the battery, in order to counteract roll and pitch disturbances. If the battery is to move in the opposing direction of the tilt (e.g., shifting left when rolling right), it generates a positive stabilising effect. In return, this will contribute positively to the total reward, crucial for improving flight recovery during fault conditions.. Conversely, the battery penalty **(**p_batt**)** is scaled based on the magnitude of movement**.** Excessive or erratic repositioning is penalised to discourage unnecessary shifts that have a tendency to destabilise the quadcopters' CoG too aggressively.

As one can appreciate, the addition of the battery adds a whole new level of complexity; however, having trained data from phase one brings an advantage to the initial training of phase 2; rotors know how much they need to adjust. It is also important to note that the advantage also comes with a drawback, in that the phase one's design didn't encompass the battery in the design, hence phase 2 involves a drone with a different weight as well as respective CoG position. Ultimately, training duration is expected to be much larger than phase one, given that phase two has more state outputs in play.

4.4 Phase Two: Integrated Battery & Rotor Speed Training

Finally presented are the results, as well as a storyboard on how the drone recovers now regarding:

I) Reaching a desired altitude & maintaining it, when rotor B fails.

II) Maintaining an altitude, as well as reducing tilt (little to nothing)

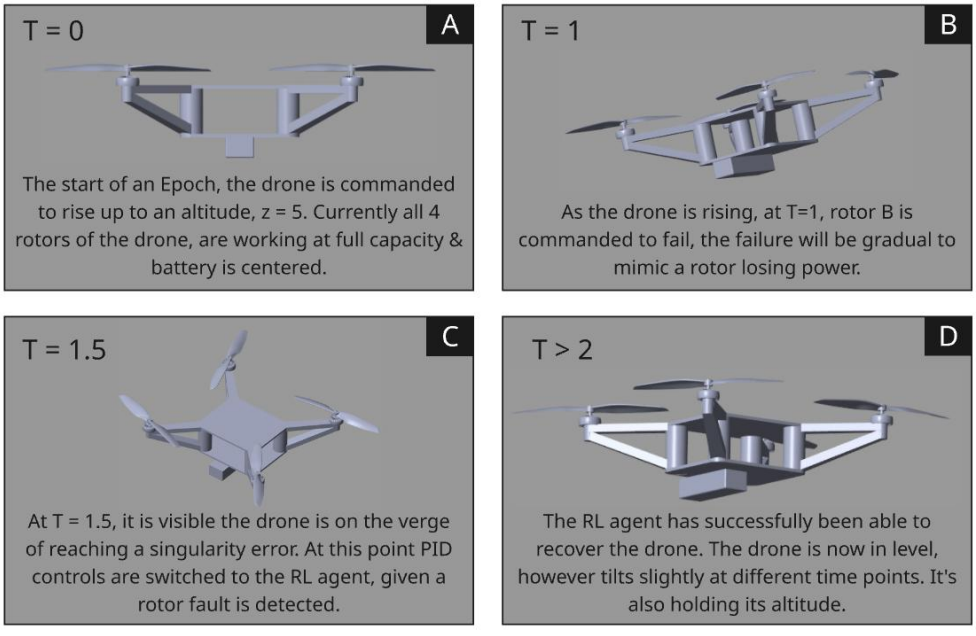


Figure 41 Storyboard Phase 2

As per the above, it is visible that the drone is able to meet an altitude requirement, with minimum tilt. Therefore, validating that collectively have rotor distribution as well as battery/CoG movement, will allow stabilising the drone, giving it enough time to either perform an emergency landing, or potentially carry on with its flight. Given the complexity of the training has increased immensely, given there are possible state actions, one could expect an increase in both episodes and initial penalties, as visible in fig.42:

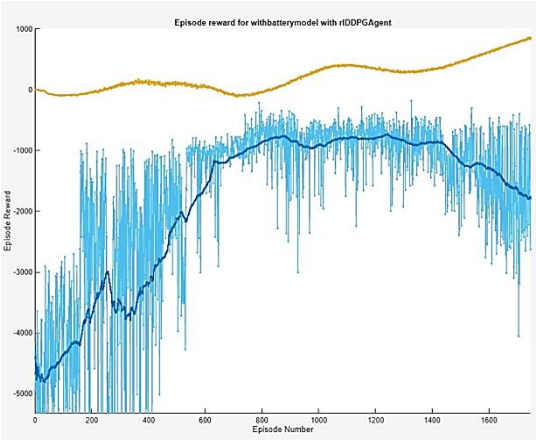


Figure 42 Initial Training Phase 2 Graph

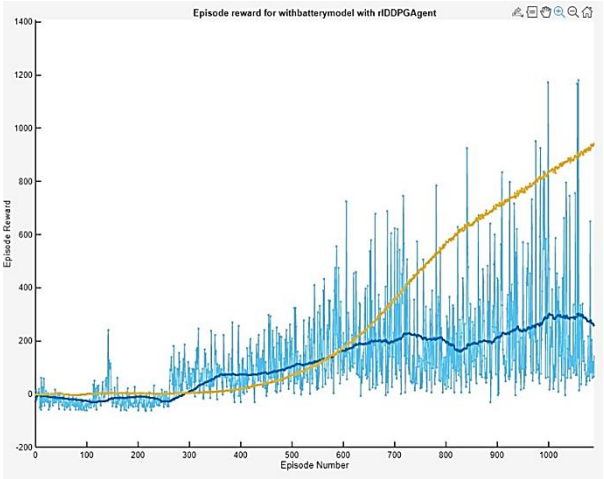


Figure 43 After 5 sets of 2000 epoch Phase 2

As per Fig.42, the graph clearly shows that the model is improving and penalizing correctly, resulting in a positive gradient. As one could appreciate, the RL agent has many state actions to choose from and therefore a constant positive gradient is always possible at the start of a training period.
Fig 43: is the training graph after 2000 epochs. A similar system to phase 1 was implemented here, where the training set was set to a maximum of 1000-epoch, depending on time limitation, training can be conducted per day and mainly as a safe passage in case the solver was to reach 'min step size' problem or a singularity error; as it allows one to hold most recent weights, ensuring the resume of the training is at its most efficient point in time. Evidently, on this training set, the RL agent is starting to receive a reward rather than a penalty, suggesting the limits are within the range set in the reward system.

Below in fig.44. presents a zoomed in training set graph, where the rewards start to balance out, and the rewards have been roughly balanced. One can surely continue improving the training of the model by reducing the magnitude to the limits and applying larger penalties.

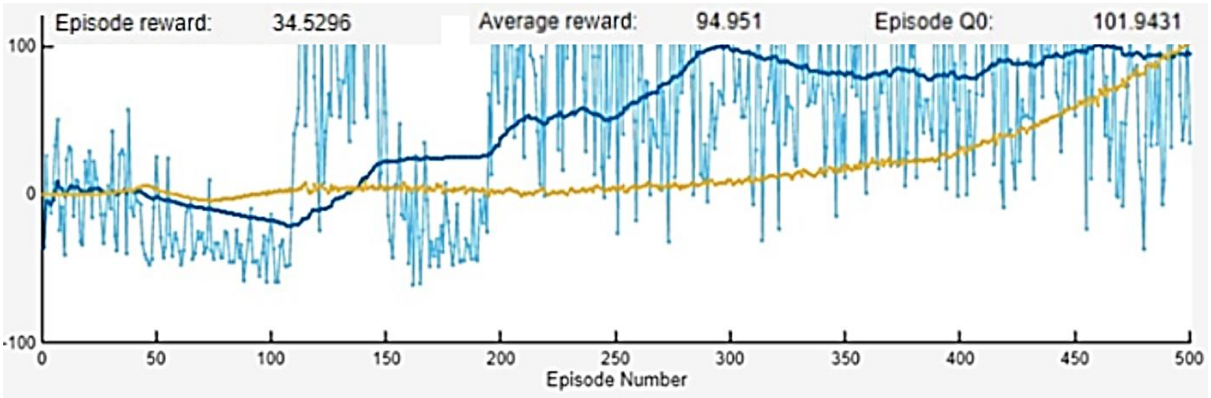


Figure 44 Rewards Starting to balance out Phase 2

5. Conclusion

Dubai city's introduction of aerial taxis signifies a global shift toward advanced air mobility, where redundancy & stability are becoming key critical design priorities. This project directly underlines that the frontier is demonstrated by showing how intelligent fault-tolerant control can be achieved through coordinated rotor-speed adjustment and adaptive centre-of-gravity (CoG) management via battery shifting. (Time Out Dubai, n.d.)

In Phase 1, the reinforcement learning (RL) model was trained to redistribute thrust across the remaining rotors following a single rotor failure. The agent successfully maintained altitude. Though slight vibrations were present, and a tilted orientation were observed, the drone was able to hold its altitude. This overall indicated that thrust redistribution alone could sustain flight, however, not achieve a non-tilted orientation…

In Phase 2, the integration of battery repositioning transformed the response. By enabling active CoG alteration through battery movement, the drone not only maintained altitude but also achieved a stable, level posture, confirming the theoretical prediction that mass redistribution, along with rotor speed adaptation, provides a superior redundancy mechanism.

Furthermore, the AI model's scalability enables retraining under different payload configurations. While a movable battery is ideal for smaller prototypes, larger or scaled delivery drones could instead employ parcel-shifting mechanisms**,** using similar control logic to adjust internal load distribution; given moving batteries themselves in larger-scale drones, it isn't the safest nor efficient option. This, therefore, expands the project's relevance from drone safety to broader fault-tolerant logistics systems**,** supporting safe operation of passenger and parcel UAVs in complex urban environments such as Dubai's envisioned air taxi network.

View the following GitHub Repository to view

1) Videos on phase one and two
2) PID Custom Drone Controller
3) SolidWorks design files
4) Phase One Simulink environment & respective scripts
5) Phase Two Simulink environment & respective scripts

https://github.com/Abz299/RotorFailureDroneWith_XY_CORE_HEAD

6. Validation

[1] Asar Iqbal - *Lean Engineer at Phinia*

Validation of material selection for XY-core head

[2] Abraham Mathews - *Associate Design Engineer at GKN Aerospace*

Validation during the design phase

[3] Huzaifa Muhammad - *Engineering IP at Bentley Motors*

Validation during the structural testing phase

- The rail sledge, wall thickness is relatively thin, and therefore, a bending analysis is required via a point load on the wall thickness, to see if it can sustain the battery load.

- Y sledge & X sledge seem to be thin in dimensions (especially in the slot regions, therefore making aluminium difficult to implement; would materials like aluminium be possible for manufacturing? Alternative material like PLA, would be preferable. (Aluminium may be possible if the model is scaled slightly up)

- Need for tolerances in each CAD drawing. (To ensure manufacturing is done to the correct standard)

- During the phase of battery movement, the material must have a tendency not to fatigue & therefore, given the nature of the element, for safety purposes, the material choice is crucial.

- The 3d Printer nozzle size must match the scale of the smallest features required.

7. Suggestions for future work

Future work should focus on introducing dynamic fault & environmental conditions to enhance the robustness of the learning agent.

While the current Simulink environment simulates a single fixed motor failure under ideal gravity-only conditions (9.81 acting on the -z axis/downwards), further testing could incorporate the following:

[1] Multiple Motor Failures & Recoveries:

Simulate scenarios where more than one rotor fails or restarts mid-flight to test adaptability and how the drone behaves in response to varying health states.

[2] Dynamic Disturbances & Weather Effects:

Integration of wind gusts, turbulence, or varying air densities, both linear & angular, will, in return, allow for an accurate representation of outdoor environments rather than ideal laboratory/Simulink mechanical explorer conditions.

[3] Physically test the model:

Given the design made was both fully aerodynamically & structurally tested, the drone can be 3d printed, with the acknowledgement that the drone will be able to perform close to simulation flight (given environmental conditions align). Primarily, the model XY core heads would need to be tested to ensure that during flight conditions, they're able to move with little to no resistance from environmental effects.

Such extensions would transform the simulation from a controlled fault-tolerant testbed into a fully dynamic resilience platform, enabling evaluation of the RL controller's performance under diverse, real-world conditions.

Gantt Chart 01/08/2025 Intelligent Battery Repositioning & Rotor speed adjustments for Drone Stability

Sub Tasks	August — Design, material testing, & controller tuning								September — Training for rotor redistribution & battery repositioning								October — Data analysis & Validation Meetings			
	1 - 4	5 - 8	9 - 12	13 - 16	17 - 20	21 - 24	25 - 28	29 - 1	2 - 5	6 - 9	10 - 13	14 - 17	18 - 22	23 - 26	27 - 30		1 - 4	5 - 8	9 - 12	13 - 15
Literature review on redundancy, FTC, MPC, SMC papers																				
Draft theory section on UAV fault tolerance (1.1 – 1.2)																				
Initial SolidWorks frame & arm design																				
Add XY-core head battery mechanism																				
Material selection & property validation in EduPack																				
Aerodynamic validation (Flow Simulation – hover tests)																				
Structural tests (arm bending, stress distribution)																				
Compute CoG & inertia values																				
Export CAD to Simscape using Multibody Link																				
Simulink environment setup (World, Dynamics, Sensors)																				
Configure altitude, pitch, roll, yaw control loops																				
Begin PID tuning – altitude loop																				

Gantt Chart 01/08/2025 Intelligent Battery Repositioning & Rotor speed adjustments for Drone Stability

Sub Tasks	August — Design, material testing, & controller tuning								September — Training for rotor redistribution & battery repositioning							October — Data analysis & Validation Meetings			
	1 - 4	5 - 8	9 - 12	13 - 16	17 - 20	21 - 24	25 - 28	29 - 1	2 - 5	6 - 9	10 - 13	14 - 17	18 - 22	23 - 26	27 - 30	1 - 4	5 - 8	9 - 12	13 - 15
Continue PID tuning – pitch, yaw, roll loops																			
Collect scope data for PID tuning graphs																			
Build RL environment base (phase 1)																			
Implement motor-failure logic & PID via RL switch																			
Define reward function & termination logic																			
Phase 1 Training: RL learns rotor redistribution																			
Save weights in 100-episode batches																			
Analyse phase 1 results (altitude stable, tilted hover)																			
Extract and annotate motor-speed scope plots																			
Begin Phase 2 environment build (battery integration)																			
Configure battery CoG shift logic (XY-core control)																			
Phase 2 Training: rotor + battery repositioning																			

Gantt Chart 01/08/2025 Intelligent Battery Repositioning & Rotor speed adjustments for Drone Stability

Sub Tasks	August								September							October			
	1 - 4	5 - 8	9 - 12	13 - 16	17 - 20	21 - 24	25 - 28	29 - 1	2 - 5	6 - 9	10 - 13	14 - 17	18 - 22	23 - 26	27 - 30	1 - 4	5 - 8	9 - 12	13 - 15
Evaluate Phase 2 graphs (altitude + stability restored)																			
Compare Phase 1 vs Phase 2 performance																			
Draft discussion for Chapter 4 (Training Ground)																			
Write Conclusion (incl. Dubai flying taxi link)																			
Write Validation & Future Work sections																			
Final editing, figure numbering, references																			
Proofread, format and generate PDF for submission																			
Meeting with Abraham Mathews																			
Meeting with Huzaifa Muhammad																			
Meeting with Asar Iqbal																			
Publish training videos on youtube																			
Finalise the Github Repo																			

Design, material testing, & controller tuning

Training for rotor redistribution & battery repositioning

Data analysis & Validation Meetings

References

1: Functionality of MPC and the receding horizon principle | Download Scientific Diagram. (n.d.). Retrieved August 10, 2025, from https://www.researchgate.net/figure/Functionality-of-MPC-and-the-receding-horizon-principle_fig2_318233447

(2) The Evolution of Drones Across Sectors and Their Cybersecurity Risks: Best Practices and National Security Concerns for India | LinkedIn. (n.d.). Retrieved August 9, 2025, from https://www.linkedin.com/pulse/evolution-drones-across-sectors-cybersecurity-risks-best-abhirup-guha-wi7oc/

10PCS/LOT IRLZ24N IRLZ34N IRLZ44N TO-220 TO220 IRLZ24 IRLZ34 IRLZ44 Transistor - AliExpress 502. (n.d.). Retrieved August 14, 2025, from https://www.aliexpress.com/item/1005006228628494.html?src=google&pdp_npi=4%40dis!GBP!5.23!1.72!!!!!%40!12000036374535632!ppc!!!&snps=y&snpsid=1&src=google&albch=shopping&acnt=742-864-1166&isdl=y&slnk=&plac=&mtctp=&albbt=Google_7_shopping&aff_platform=google&aff_short_key=_oDeeeiG&gclsrc=aw.ds&&albagn=888888&&ds_e_adid=&ds_e_matchtype=&ds_e_device=c&ds_e_network=x&ds_e_product_group_id=&ds_e_product_id=en1005006228628494&ds_e_product_merchant_id=5551326180&ds_e_product_country=GB&ds_e_product_language=en&ds_e_product_channel=online&ds_e_product_store_id=&ds_url_v=2&albcp=22435797343&albag=&isSmbAutoCall=false&needSmbHouyi=false&gad_source=1&gad_campaignid=22432265180&gbraid=0AAAAA99aYpds7tdIzMX9y0QngOBg39JJM&gclid=CjwKCAjw7_DEBhAeEiwAWKiCC2ByCJPbitNy2XOJ0IaLEELUSPMqhKz5TO6N_lYCdaYC-1ler0zqbRoCkOwQAvD_BwE

Aerial Informatics and Robotics Platform - Microsoft Research. (n.d.). Retrieved September 20, 2025, from https://www.microsoft.com/en-us/research/project/aerial-informatics-robotics-platform/

AirHub - How Drones Usage Transform the Security Industry. (n.d.). Retrieved August 9, 2025, from https://www.airhub.app/resources/news/the-role-of-drones-in-security-enhancing-efficiency-safety-and-data-security

Ashby, M. F. (2005). Materials Selection in Mechanical Design Third Edition | tuan nguyen thanh - Academia.edu. *Elsevier*. http://www.academia.edu/8200323/Materials_Selection_in_Mechanical_Design_Third_Edition

Bouabdallah, S., Murrieri, P., & Siegwart, R. (2004). Design and control of an Indoor micro quadrotor. *Proceedings - IEEE International Conference on*

Robotics and Automation, *2004*(5), 4393–4398. https://doi.org/10.1109/ROBOT.2004.1302409

CoreXY Kinematics – 3D Distributed. (n.d.). Retrieved September 21, 2025, from https://3ddistributed.com/corexy-3d-printer/corexy-kinematics/

Doubling down on safety: understanding our approach to redundancy in autonomous vehicles. (n.d.). Retrieved August 9, 2025, from https://www.volvoautonomoussolutions.com/en-en/news-and-insights/insights/articles/2024/jul/doubling-down-on-safety.html

Dubai flying taxis: The route, cost, travel time and when we'll get them explained | Time Out Dubai. (n.d.). Retrieved October 12, 2025, from https://www.timeoutdubai.com/news/dubai-flying-taxi-guide

Fourlas, G. K., & Karras, G. C. (2021). A Survey on Fault Diagnosis and Fault-Tolerant Control Methods for Unmanned Aerial Vehicles. *Machines 2021, Vol. 9, Page 197*, *9*(9), 197. https://doi.org/10.3390/MACHINES9090197

FrSky r-xsr 2.4Ghz receiber. | 3D CAD Model Library | GrabCAD. (n.d.). Retrieved August 16, 2025, from https://grabcad.com/library/frsky-r-xsr-2-4ghz-receiber-1

FrSky R-XSR EU-LBT 16CH Ultra Mini Receiver | HobbyRC UK. (n.d.). Retrieved August 14, 2025, from https://www.hobbyrc.co.uk/frsky-r-xsr-eu-lbt-16ch-ultra-mini-receiver-with-sbus-cppm-and-telemetry

How Drones Are Used in Agriculture - Toll Uncrewed Systems. (n.d.). Retrieved August 9, 2025, from https://tolluncrewedsystems.com/blog/how-drones-are-used-in-agriculture/

HQProp 8X4.5 Black Glass Fiber Reinforced Nylon 1xCW - Flying Tech. (n.d.). Retrieved August 14, 2025, from https://www.flyingtech.co.uk/product/hqprop-8x4-5-black-glass-fiber-reinforced-nylon-1xcw/

Ideas Are Easy, Implementation Is Hard. (n.d.). Retrieved September 21, 2025, from https://www.forbes.com/2004/11/04/cx_gk_1104artofthestart.html

IFlight XING 2806.5 1300KV 1800KV 2-6S Brushless Motor for 7-8 inch FPV Frame Propeller RC FPV Racing Drone - AliExpress 26. (n.d.). Retrieved August 14, 2025, from https://www.aliexpress.com/item/1005003495601761.html?src=google&albch=search&acnt=479-062-3723&isdl=y&aff_short_key=UneMJZVf&albcp=21520180446&albag=165797549415&slnk=&trgt=dsa-

1642801257570&plac=&crea=726704056236&albad=726704056236&netw=g&device=c&mtctp=&memo1=&albbt=Google_7_search&aff_platform=google&albagn=888888&isSmbActive=false&isSmbAutoCall=false&needSmbHouyi=false&gad_source=1&gad_campaignid=21520180446&gbraid=0AAAAAD2chG4fTNRvkoDBZDnTgwyhSLaH1&gclid=CjwKCAjwp_LDBhBCEiwAK7FnkkMrT6w9EfWT0ioCkWzLBg25uV3OWhR4jBSBt1uHqYIxklUF9N45KhoCJGcQAvD_BwE

Infrastructure inspections with drones made easier under new rules | UK Civil Aviation Authority. (n.d.). Retrieved August 9, 2025, from https://www.caa.co.uk/newsroom/news/infrastructure-inspections-with-drones-made-easier-under-new-rules/

Install the Simscape Multibody Link Plugin - MATLAB & Simulink. (n.d.). Retrieved September 20, 2025, from https://uk.mathworks.com/help/smlink/ug/installing-and-linking-simmechanics-link-software.html

IRLZ44N Mosfet transistör | 3D CAD Model Library | GrabCAD. (n.d.). Retrieved August 16, 2025, from https://grabcad.com/library/irlz44n-mosfet-transistor-1

Jetson Orin Nano Super Developer Kit | NVIDIA. (n.d.). Retrieved August 14, 2025, from https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit/

Ke, C., Cai, K.-Y., & Quan, Q. (2022). Uniform Passive Fault-Tolerant Control of a Quadcopter with One, Two, or Three Rotor Failure. *IEEE Transactions on Robotics*, *39*(6), 4297–4311. https://doi.org/10.1109/TRO.2023.3297048

Mach Number. (n.d.). Retrieved September 20, 2025, from https://www.grc.nasa.gov/www/k-12/airplane/mach.html

Nan, F., Sun, S., Foehn, P., & Scaramuzza, D. (2021). Nonlinear MPC for Quadrotor Fault-Tolerant Control. *IEEE Robotics and Automation Letters*, *7*(2), 5047–5054. https://doi.org/10.1109/LRA.2022.3154033

NEMA 8 Hollow Shaft Steppermotor | 3D CAD Model Library | GrabCAD. (n.d.). Retrieved August 14, 2025, from https://grabcad.com/library/nema-8-hollow-shaft-steppermotor-1

Newman, S. J. (2007). Principles of Helicopter Aerodynamics – Second edition J.G. Leishmann Cambridge University Press, The Edinburgh Building, Shaftesbury Road, Cambridge, CB2 2RU, UK. 2006. 826pp. Illustrated. £65. ISBN 0-521-85860-7. *The Aeronautical Journal*, *111*(1126), 825–826. https://doi.org/10.1017/S0001924000087352

Pounds, P., Mahony, R., & Corke, P. (2006). *Modelling and control of a quad-rotor robot*. https://researchportalplus.anu.edu.au/en/publications/modelling-and-control-of-a-quad-rotor-robot

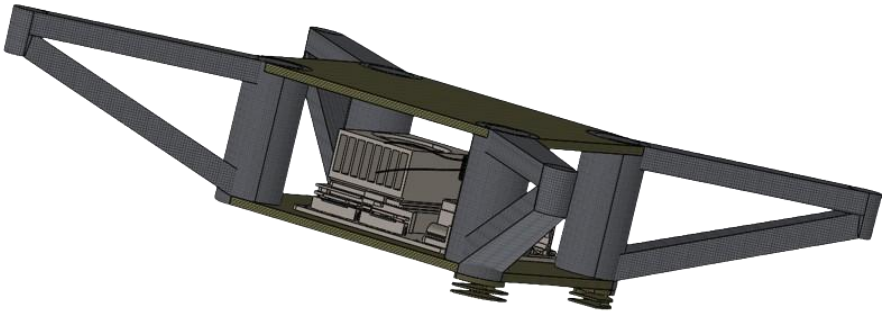
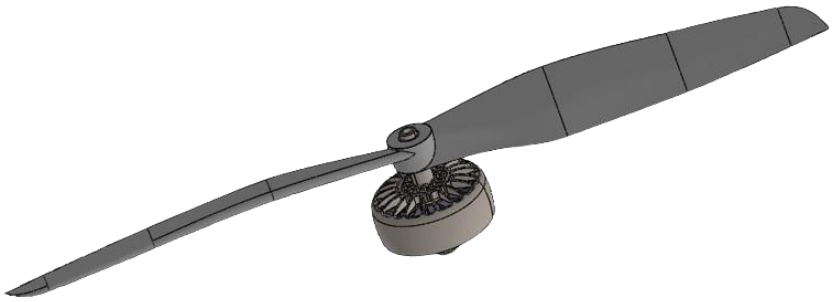
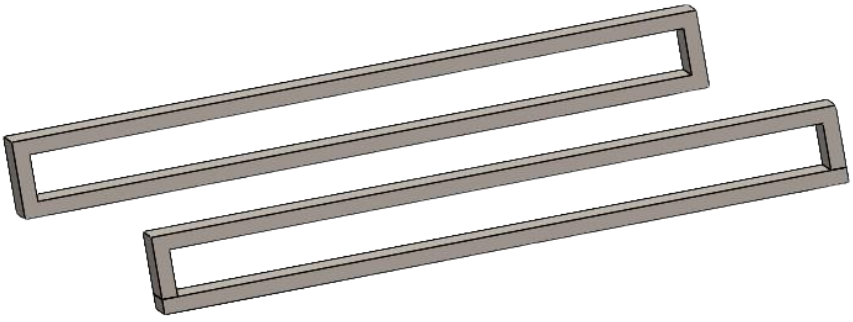
Sliding Phases of the SMC | Download Scientific Diagram. (n.d.). Retrieved August 10, 2025, from https://www.researchgate.net/figure/Sliding-Phases-of-the-SMC_fig4_347294517

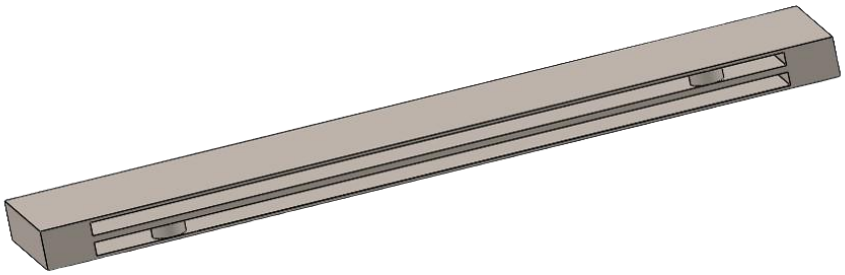
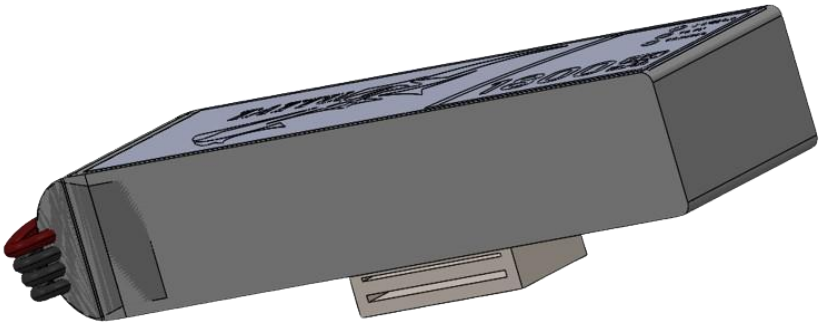
Tattu 3s 1800mah 11.1v 75c 3s1p Lipo Battery Pack With Xt60 Connector: Amazon.co.uk: Electronics & Photo. (n.d.). Retrieved September 17, 2025, from https://www.amazon.co.uk/Tattu-1800mah-11-1v-Battery-Connector/dp/B013I9T3RA

The Role of Drones in Revolutionising Warehouse Processes - Minster WMS. (n.d.). Retrieved August 9, 2025, from https://minsterwms.com/the-role-of-drones-in-revolutionising-warehouse-processes/

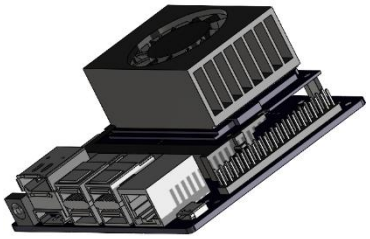
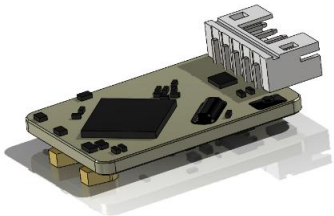
Appendices

Appendix 1 - Parts List Assembly for Entire Drone

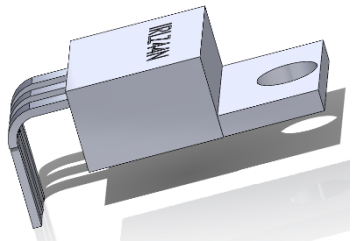
Part Name	Schematic	Specification
Frame		
Rotor Blade (*HQProp 8X4.5 Black Glass Fiber Reinforced Nylon 1xCW - Flying Tech*, n.d.) **DC Motor** (*IFlight XING 2806.5 1300KV 1800KV 2-6S Brushless Motor for 7-8 Inch FPV Frame Propeller RC FPV Racing Drone - AliExpress 26*, n.d.)		
Static Rail		

Rail Sled	
Battery Sled	

Appendix 2 - Electronic Drone Part Specifications

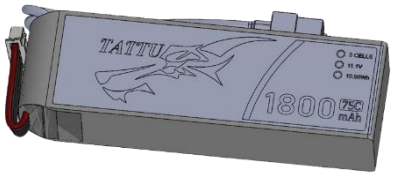
Jetson Nano Developer Kit (*Jetson Orin Nano Super Developer Kit | NVIDIA*, n.d.) 
Receiver (*FrSky R-XSR EU-LBT 16CH Ultra Mini Receiver | HobbyRC UK*, n.d.)(*FrSky R-Xsr 2.4Ghz Receiber. | 3D CAD Model Library | GrabCAD*, n.d.) 
Stepper Motors (*NEMA 8 Hollow Shaft Steppermotor | 3D CAD Model Library | GrabCAD*, n.d.)

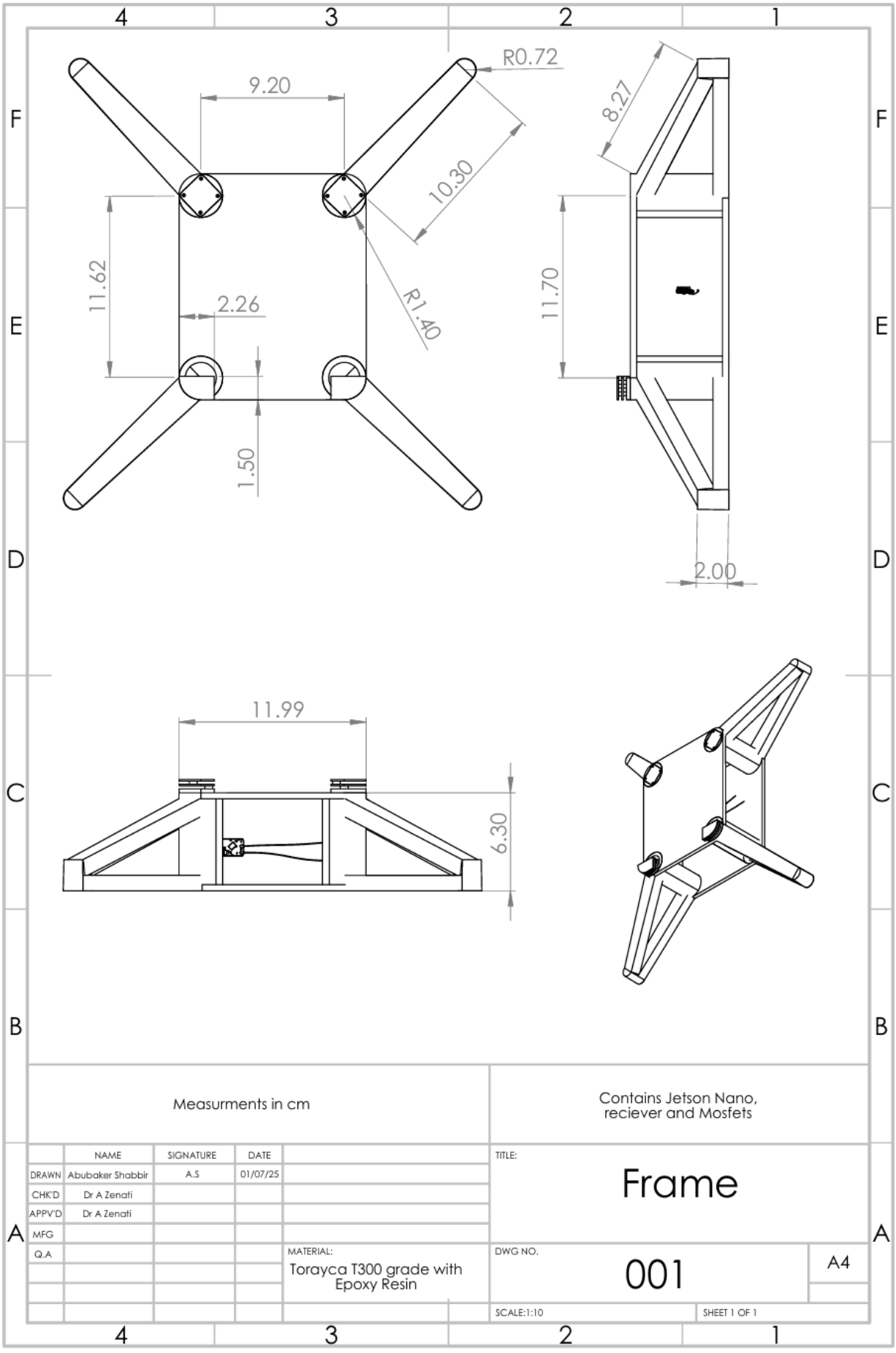
MOSFETs irlz44n (*10PCS/LOT IRLZ24N IRLZ34N IRLZ44N TO-220 TO220 IRLZ24 IRLZ34 IRLZ44 Transistor - AliExpress 502*, n.d.)(*IRLZ44N Mosfet Transistör | 3D CAD Model Library | GrabCAD*, n.d.)

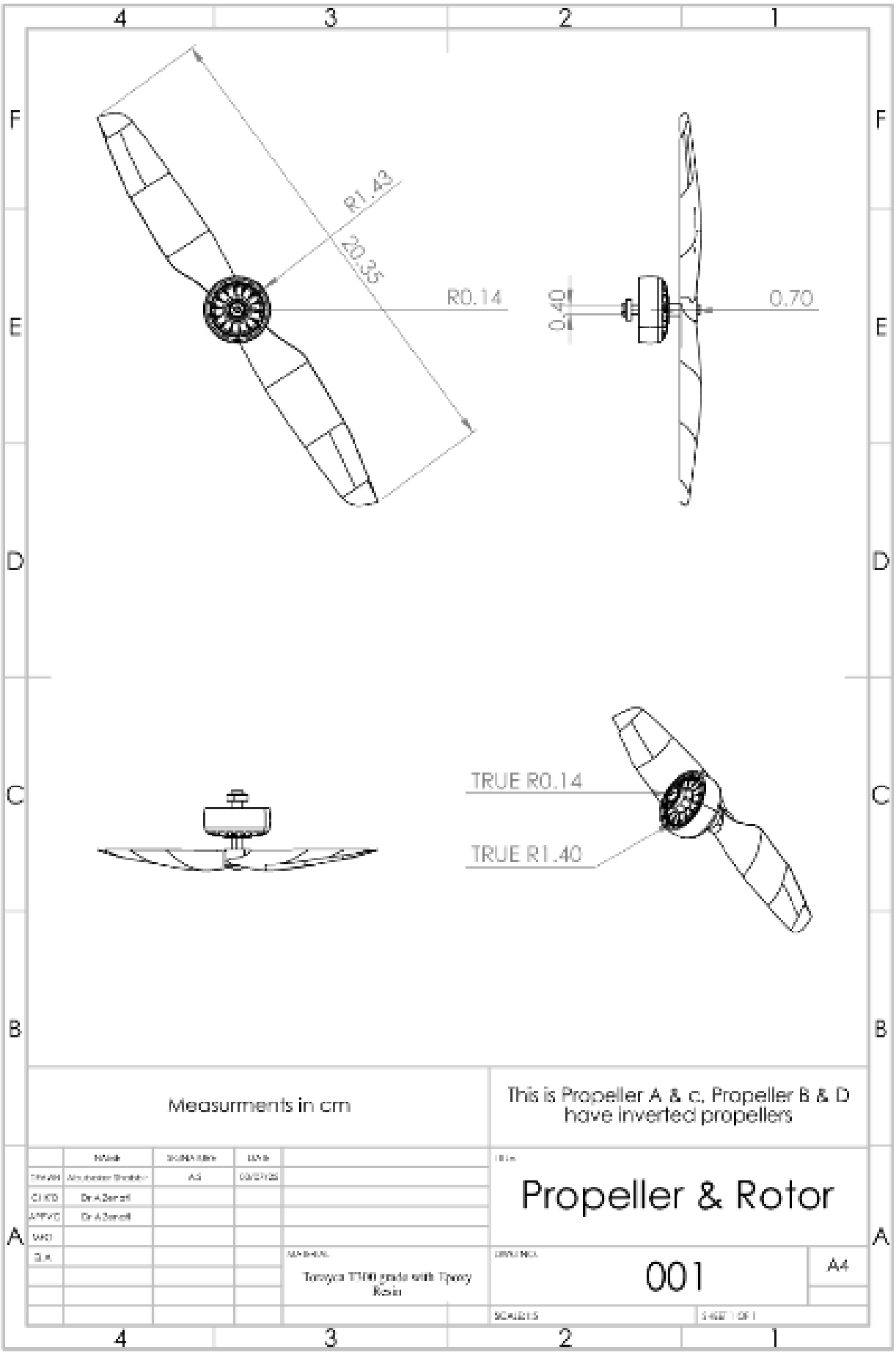


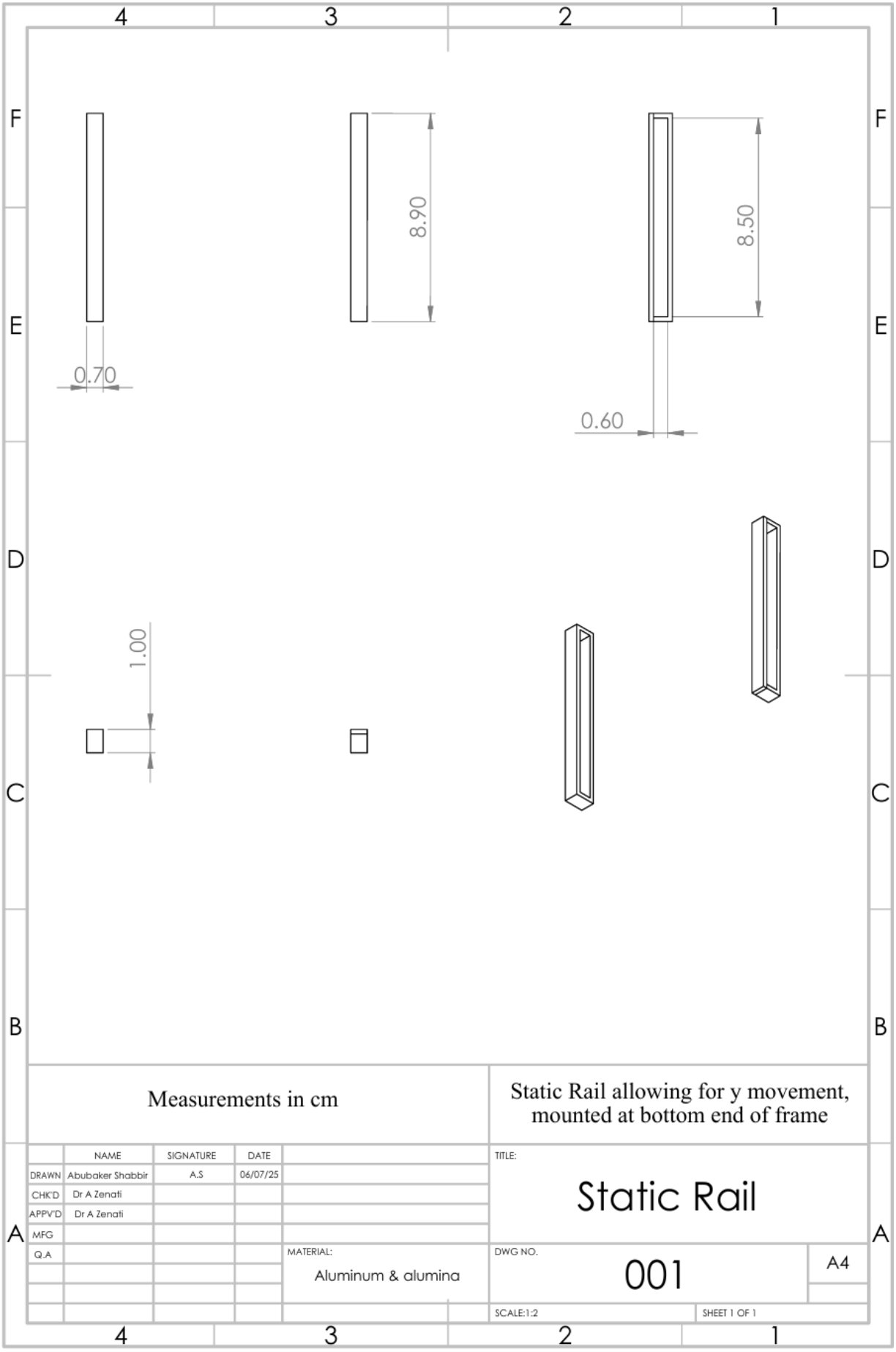
DC Motor (*IFlight XING 2806.5 1300KV 1800KV 2-6S Brushless Motor for 7-8 Inch FPV Frame Propeller RC FPV Racing Drone - AliExpress 26*, n.d.)

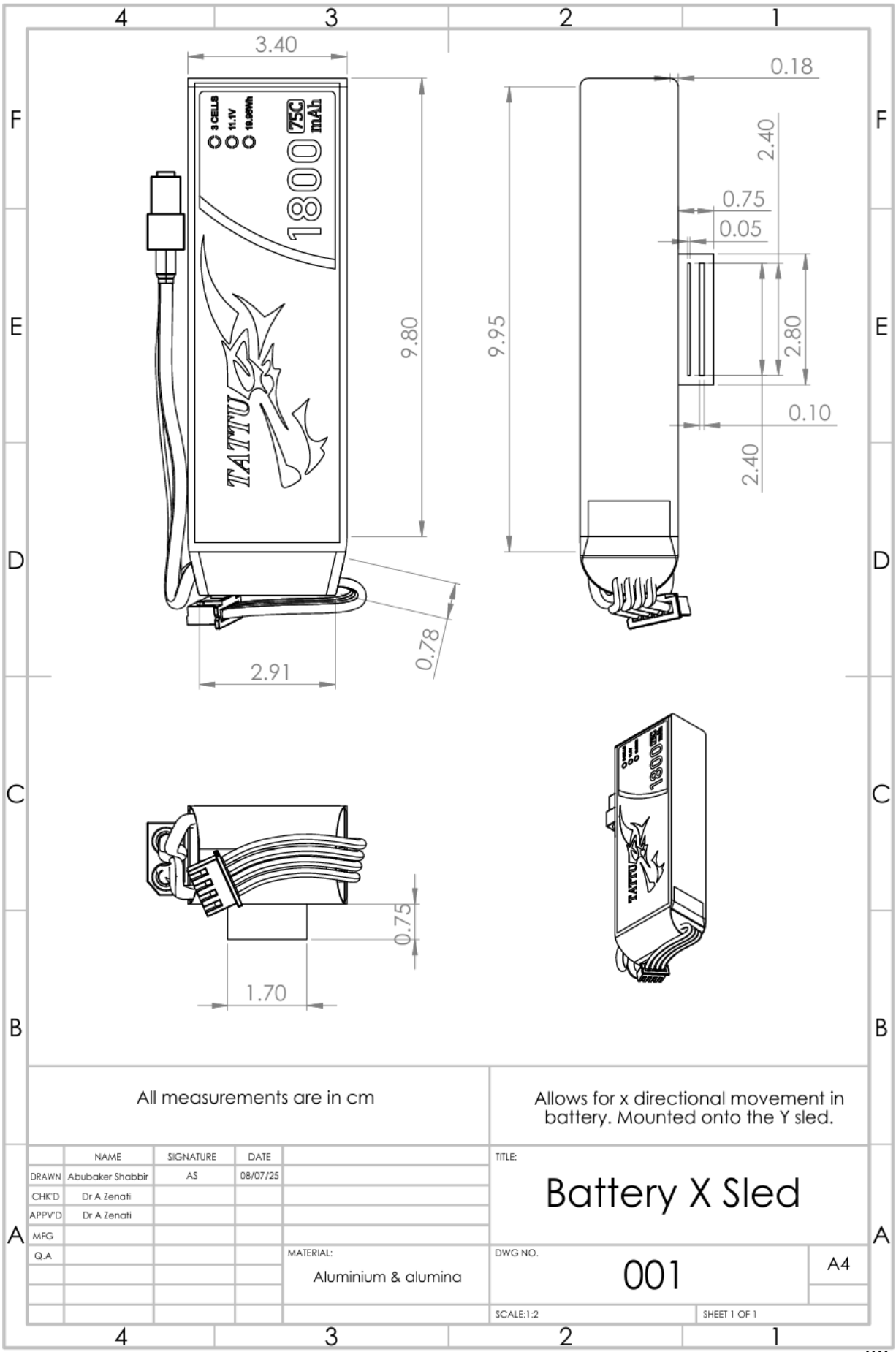
Battery (*Tattu 3s 1800mah 11.1v 75c 3s1p Lipo Battery Pack With Xt60 Connector: Amazon.Co.Uk: Electronics & Photo*, n.d.)




4
3
2
1
F
E
D
C
B
A
R0.72
9.20
10.30
8.27
11.62
2.26
R1.40
11.70
1.50
2.00
11.99
6.30
Measurments in cm
Contains Jetson Nano, reciever and Mosfets
NAME
SIGNATURE
DATE
DRAWN
Abubaker Shabbir
A.S
01/07/25
CHK'D
Dr A Zenati
APPV'D
Dr A Zenati
MFG
Q.A
TITLE:
Frame
MATERIAL:
Torayca T300 grade with Epoxy Resin
DWG NO.
001
A4
SCALE:1:10
SHEET 1 OF 1


4
3
2
1
F
E
D
C
B
A
R1.43
20.35
R0.14
0.40
0.70
TRUE R0.14
TRUE R1.40
Measurments in cm
This is Propeller A & c, Propeller B & D have inverted propellers
Propeller & Rotor
001
A4
Tenyca T700 grade with Epoxy Resin


4
3
2
1
F
E
D
C
B
A
8.90
8.50
0.70
0.60
1.00
Measurements in cm
Static Rail allowing for y movement, mounted at bottom end of frame
NAME
SIGNATURE
DATE
DRAWN
Abubaker Shabbir
A.S
06/07/25
CHK'D
Dr A Zenati
APPV'D
Dr A Zenati
MFG
Q.A
MATERIAL:
Aluminum & alumina
TITLE:
Static Rail
DWG NO.
001
A4
SCALE:1:2
SHEET 1 OF 1



All measurements are in cm

Allows for x directional movement in battery. Mounted onto the Y sled.

	NAME	SIGNATURE	DATE
DRAWN	Abubaker Shabbir	AS	08/07/25
CHK'D	Dr A Zenati		
APPV'D	Dr A Zenati		
MFG			
Q.A			

TITLE: Battery X Sled

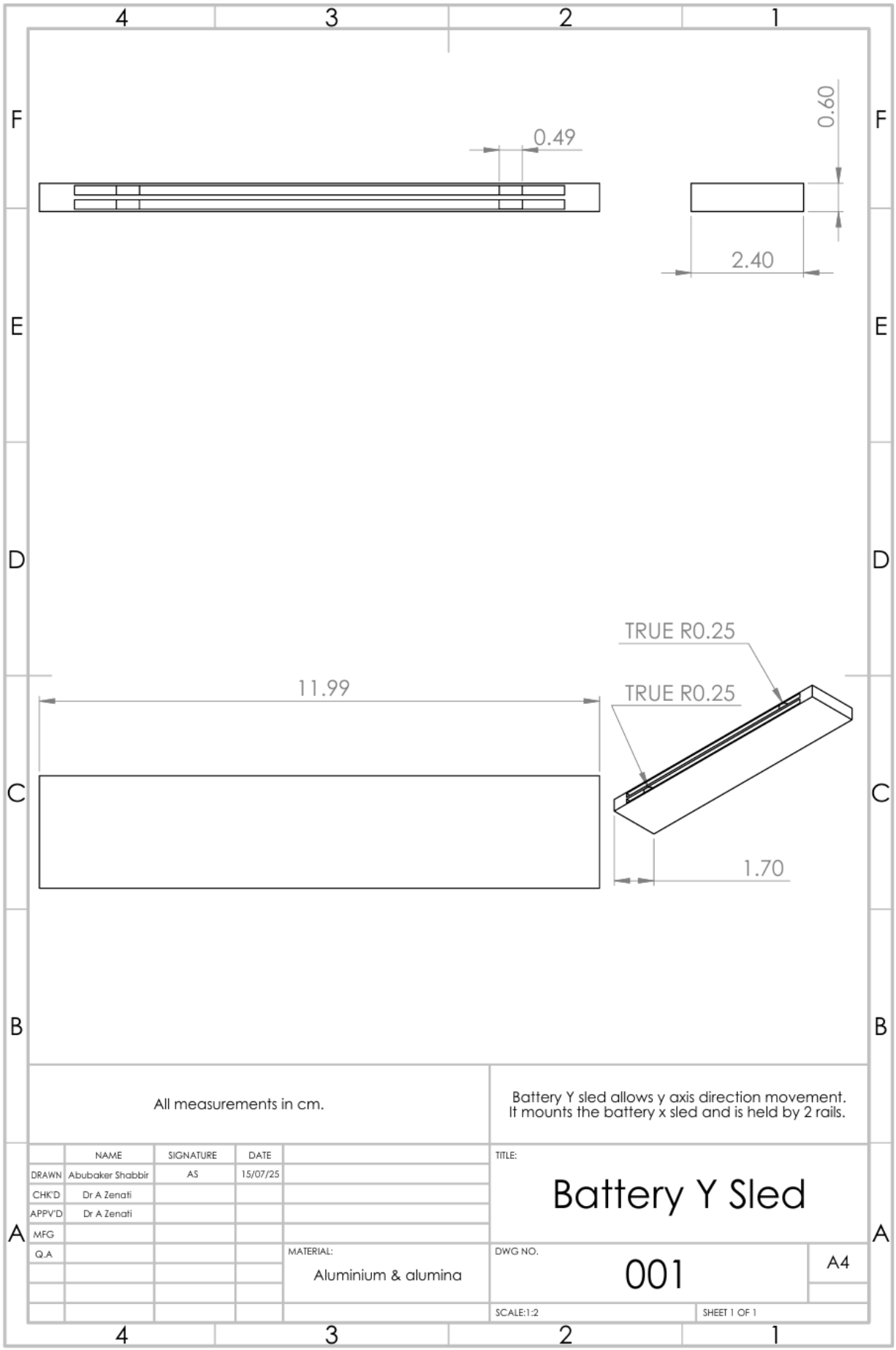
MATERIAL: Aluminium & alumina

DWG NO. 001

A4

SCALE:1:2

SHEET 1 OF 1



All measurements in cm.

Battery Y sled allows y axis direction movement.
It mounts the battery x sled and is held by 2 rails.

	NAME	SIGNATURE	DATE
DRAWN	Abubaker Shabbir	AS	15/07/25
CHK'D	Dr A Zenati		
APPV'D	Dr A Zenati		
MFG			
Q.A			

MATERIAL: Aluminium & alumina

TITLE: Battery Y Sled

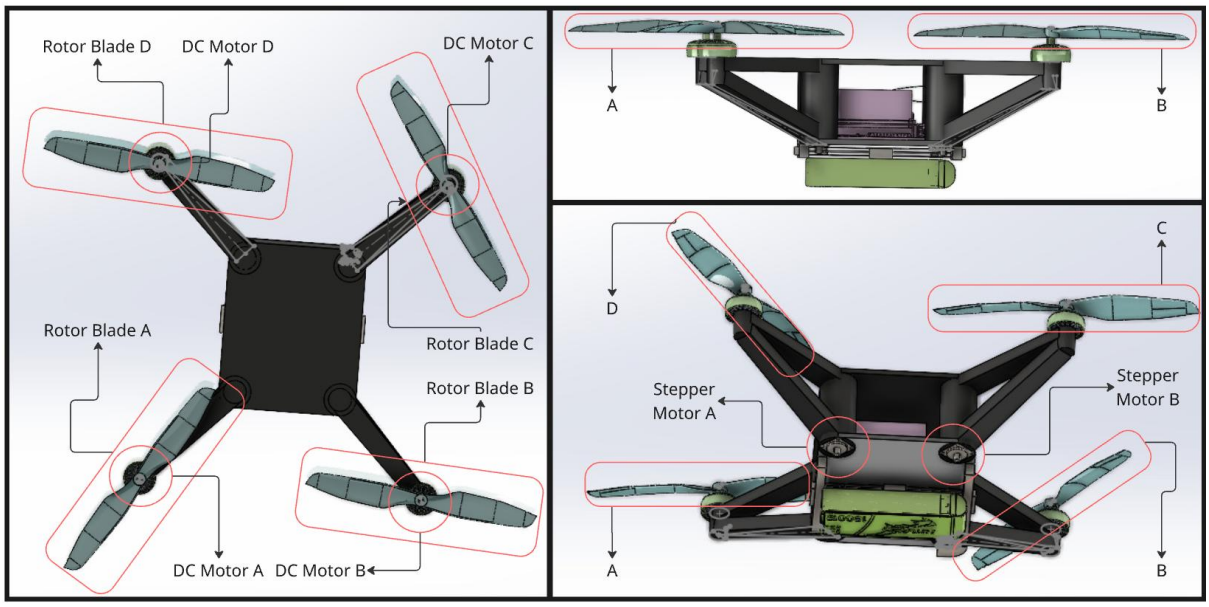
DWG NO. 001

A4

SCALE:1:2

SHEET 1 OF 1

Appendix 4 – CoM based on 3 points (training data for AI)



CoM from Coordinate perspective - Drone Midpoint

Part	CoM x	CoM y	CoM z
Base Frame	0.45	-0.15	1.17
XY Core Head, Top Frame	-0.52	-0.09	-1.67
Battery & Hub	-0.03	0.08	-3.55
Jetson Nano Developer Kit	0.02	-0.69	0.02
Receiver	-0.37	-2.62	1.93
Dc Motors			
DC Motor A	13.83	-12.41	5.15
DC Motor B	13.82	12.22	5.15
DC Motor C	-13.66	12.56	5.15
DC Motor D	-13.65	-12.77	5.15
Support Arms			
Arm A	7.60	-6.43	2.30
Arm B	7.59	6.25	2.30
Arm C	-7.52	6.17	2.00
Arm D	-7.51	-6.36	2.00
Rotor Blades			
Rotor Blade A	13.85	-12.40	6.95
Rotor Blade B	13.83	12.23	6.95
Rotor Blade C	-13.65	12.56	6.96
Rotor Blade D	-13.63	-12.77	6.95
Stepper Motors			
Stepper Motor A	5.79	-4.69	-0.25
Stepper Motor B	5.78	4.51	-0.25

CoM from Coordinate perspective - Drone Static Midpoint

Part	CoM x	CoM y	CoM z
Base Frame	0.52	-0.05	0.12
XY Core Head, Top Frame	-0.44	0.01	-2.73
Battery & Hub	0.05	0.17	-4.60
Jetson Nano Developer Kit	0.10	-0.60	-1.04
Receiver	-0.29	-2.53	0.87
Dc Motors			
DC Motor A	13.90	-12.32	4.09
DC Motor B	13.90	12.31	4.09
DC Motor C	-13.58	12.66	4.09
DC Motor D	-13.58	-12.67	4.09
Support Arms			
Arm A	7.67	-6.34	1.24
Arm B	7.67	6.34	1.24
Arm C	-7.44	6.27	0.95
Arm D	-7.44	-6.26	0.95
Rotor Blades			
Rotor Blade A	13.92	-12.31	5.90
Rotor Blade B	13.92	12.32	5.90
Rotor Blade C	-13.56	12.67	5.90
Rotor Blade D	-13.56	-12.66	5.90
Stepper Motors			
Stepper Motor A	5.86	-4.60	-1.30
Stepper Motor B	5.86	4.60	-1.30

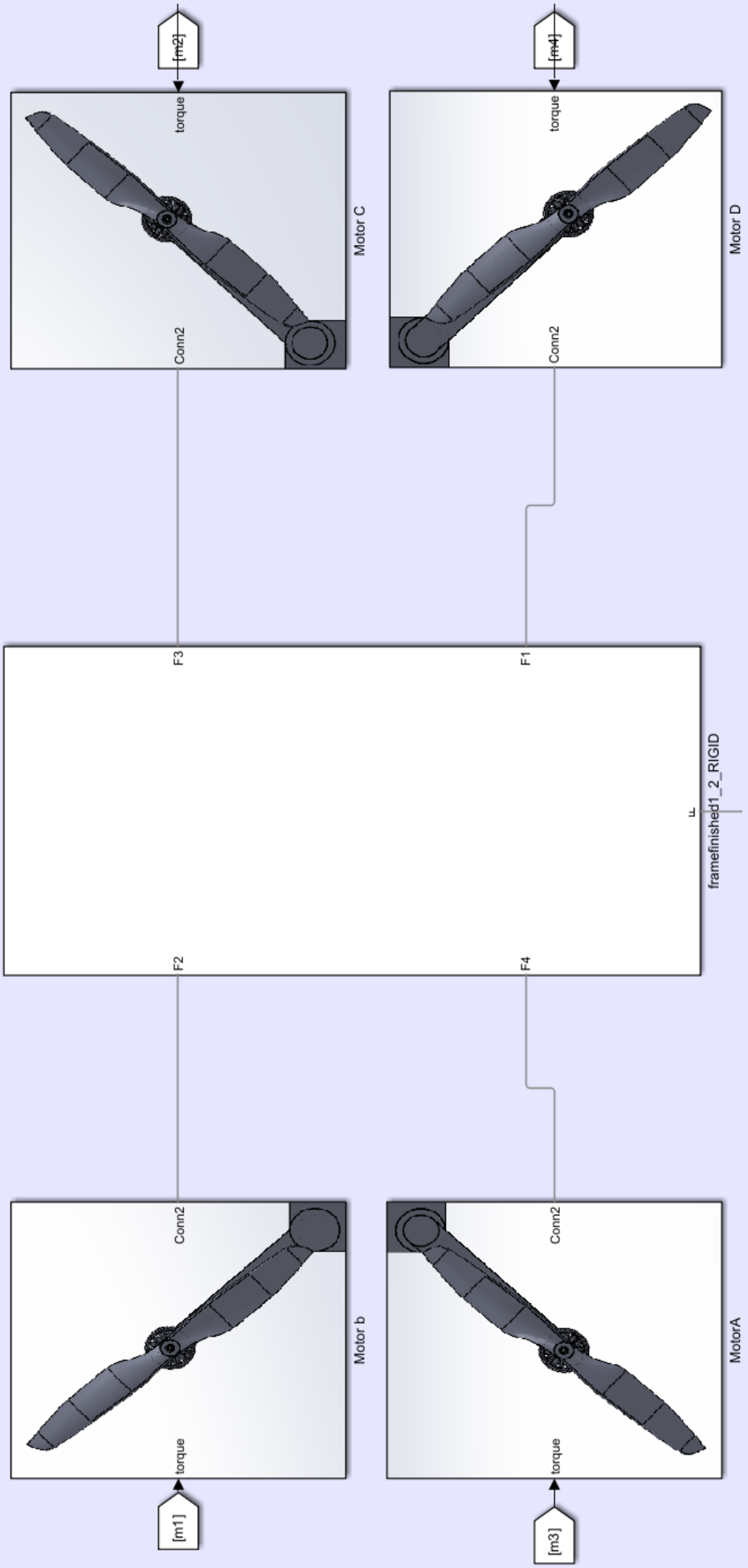
CoM from Coordinate perspective - Drone Moving Midpoint

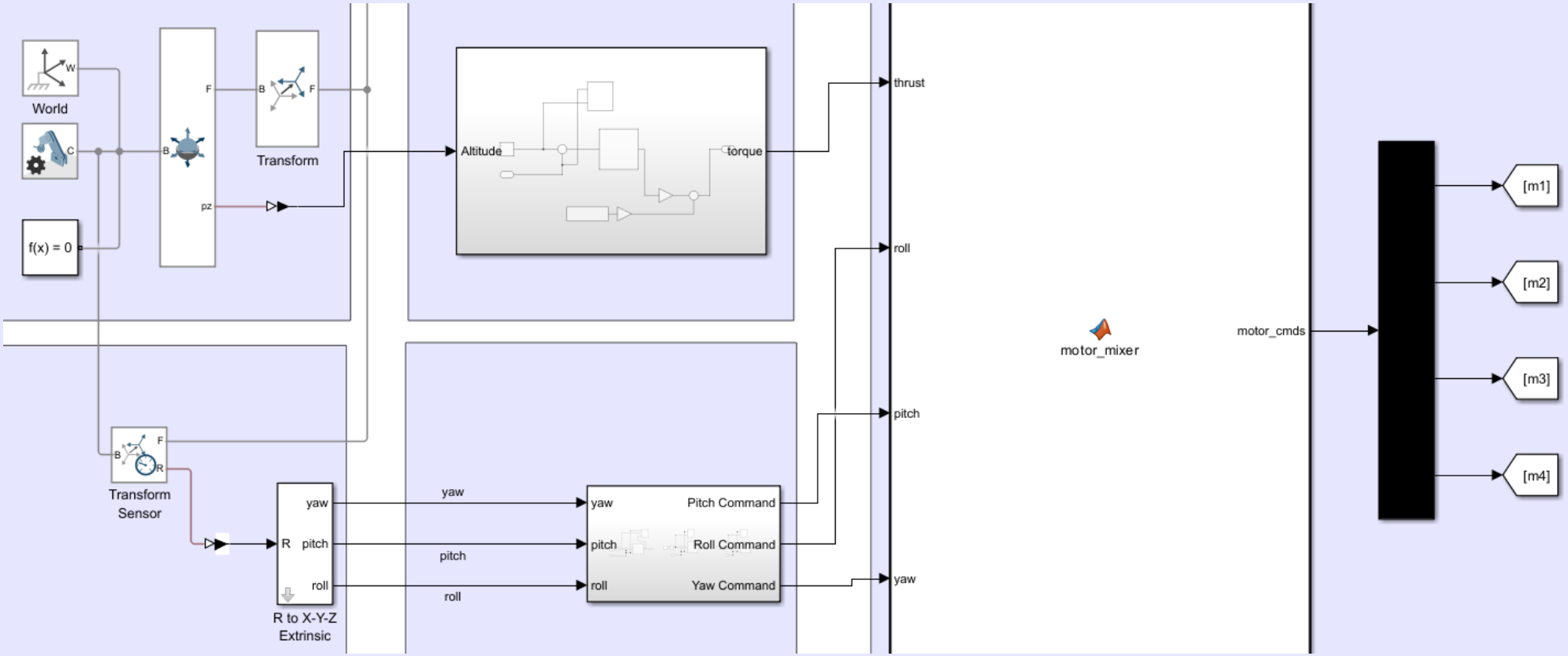
Part	CoM x	CoM y	CoM z
Base Frame	0.45	-0.14	4.45
XY Core Head, Top Frame	-0.52	-0.09	1.60
Battery & Hub	-0.03	0.08	-0.28
Jetson Nano Developer Kit	0.03	-0.69	3.29
Receiver	-0.33	-2.63	5.20
Dc Motors			
DC Motor A	14.02	-12.19	8.42
DC Motor B	13.62	12.44	8.42
DC Motor C	-13.86	12.34	8.42
DC Motor D	-13.45	-12.99	8.42
Support Arms			
Arm A	7.70	-6.31	5.57
Arm B	7.49	6.37	5.57
Arm C	-7.62	6.05	5.28

Arm D	-7.42	-6.48	-5.27
Rotor Blades			
Rotor Blade A	14.04	-12.18	10.23
Rotor Blade B	13.64	12.45	10.23
Rotor Blade C	-13.85	12.35	10.23
Rotor Blade D	-13.43	-12.98	10.23
Stepper Motors			
Stepper Motor A	5.86	-4.60	3.03
Stepper Motor B	5.71	4.60	3.03

Appendix 5 – Drone part, Inertial Contributions

Part	**Inertial Contributions Kg m^2**		
	I_{xx}	**I_{yy}**	**I_{zz}**
Frame	2.28×10^{-5}	2.90×10^{-5}	5.10×10^{-5}
Set of 4 Rotors & Propellers	2.98×10^{-4}	4.47×10^{-6}	3.01×10^{-4}
Static Rail	2.30×10^{-5}	7.04×10^{-7}	2.37×10^{-5}
Battery Sled	4.54×10^{-4}	2.33×10^{-4}	6.78×10^{-4}
Rail Sled	3.31×10^{-5}	9.86×10^{-7}	3.41×10^{-5}
Total	**8.30×10^{-4}**	**5.58×10^{-4}**	**1.08×10^{-3}**


[m2]
torque
Conn2
Motor C
[m4]
torque
Conn2
Motor D
F3
F1
F2
F4
framefinished1_2_RIGID
Conn2
torque
Motor b
Conn2
torque
MotorA
[m1]
[m3]


World
Transform
f(x) = 0
Altitude
torque
thrust
roll
pitch
yaw
motor_mixer
motor_cmds
[m1]
[m2]
[m3]
[m4]
Transform
Sensor
R to X-Y-Z
Extrinsic
Pitch Command
Roll Command
Yaw Command