Team DRONA – Project ARJUNA

Team A Project Technical Report to the 2026 IREC

Mahathi Subramanian[1], Arya Chatterjee[2], Shaunak Saraf[3], Ansh Sinha[4], Dev Sharma[5], Jenna Ann Sonali Sreshta Shetty[6], Yash Patidar[7], Tejaswinee Gunjal[8], Krishna Gund[9], Badampudi Mytreya[10], Keshav H Nambiar[11], Sai Srinivas[12], Sudarsan D Naidu[13], Shalik Raj[14]

Manipal Institute of Technology, Manipal, Karnataka (576104), India

---

This report presents the technical specifications of Team Drona's rocket, Arjuna, developed for the 2026 sub-scale 10k SRAD category. The project targets an apogee of 10,000 feet while carrying a non-deployable payload and employs a dual-event parachute system for complete recovery. The rocket is powered by an N3316 solid propellant SRAD motor and features a glass fibre composite fuselage reinforced with Aluminium 6061 components, including bulkheads, centring rings, stringers, fins, and a thrust plate, to ensure structural integrity. The avionics system consists of a COTS altimeter and GPS, and Sanjaya, a fully SMT-based SRAD system that provides redundancy by enabling apogee detection, live video telemetry, recovery, and precise altitude control.The payload conforms to the shape of a 3U cubesat. It intends to demonstrate precise control of a ball over a movable platform and compare the visual coordinates from the Raspberry PI camera with the coordinates from the touchscreen to demonstrate the reliability of visual data under high vibrational environment.

---

[1] Management Team Member
[2] Management Team Member
[3] Payload Team Member
[4] Payload Team Member
[5] Payload Team Member
[6] Payload Team Member
[7] Avionics Team Member
[8] Avionics Team Member
[9] Avionics Team Member
[10] Avionics Team Member
[11] Propulsion Team Member
[12] Structures Team Member
[13] Aerodynamics Team Member
[14] Propulsion Team Member

Nomenclature

AGL	Above Ground Level
AOA	Angle of Attack
CFD	Computational Fluid Dynamics
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
MaxQ	Maximum dynamic pressure
RK4	Fourth-order Runge–Kutta integration
RPi4-8GB	Raspberry Pi 4 with 8 GB RAM
TD	Tender Descender
A	Area
a	Acceleration
a_∞	Speed of sound
a_{dist}	Lateral disturbance acceleration (m/s^2)
C	Design vector
C_D	Drag coefficient
C_{dis}	Coefficient of discharge
d	Diameter
D	Drag force
$f(x)$	Uncontrolled dynamics
F	Force
F_d	Deployment force
F_{sl}	Shock loading force
g	Acceleration due to gravity
g_{eff}	Effective axial acceleration (m/s^2)
G	Shear strength
h	Altitude
K	Controller gain
K_p, K_i, K_d	PID control gains
m	Mass
$\dot{m}$	Mass flow rate
Ma	Mach number
P	Pressure
P_0	Total pressure
P_c	Pressure in avionics bay
P_{atm}	Atmospheric pressure
q	Dynamic pressure
r	Radius
Re	Reynolds number
R	Specific gas constant
$s(x)$	Sliding surface function
t	Time
T_s	Sampling period
u	Control input (air brake extension ratio)
v	Velocity
v_d	Parachute deployment velocity
v_t	Parachute terminal velocity
V	Volume
V_∞	Freestream velocity
x	State vector / Ball position (context dependent)
y	Ball position in plate frame
$\dot{x}, \dot{y}$	Ball velocity

Contents

I. System Architechture

Arjuna is a 2.9m tall rocket with a diameter of 15cm weighing about 44.7kg houses a 3U CubeSat Payload powered by a N-class SRAD motor.

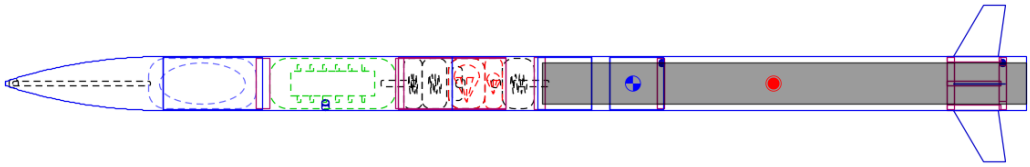

Figure 1. OpenRocket Design

The following softwares have been used for the design, simulation and analysis of the rocket:

- Fusion 360
- Ansys 2023 R1
- OpenRocket
- FinSim
- RASAero
- Nakka XLS SRM

II. Aerostructures

A. Nosecone

Conical, Tangent Ogive, Parabolic, Power (0.75), Power (0.5), LV Haack and LD Haack nosecones were considered initially. As skin friction drag is dominant in subsonic regime, a nosecone with least wetted area was preferred. The shape of the nosecone also limits the amount of payload that can be placed inside the nosecone which results in increase in skin friction drag in the form of lengthier body tube. The geometric calculations yielded the graph in Figure 2.

Ogive, Parabolic and Conical nosecones were shortlisted from this analysis. To consider transonic wave drag at Mach 0.85, CFD was done on the three nosecones with a fineness ratio of 3:1. It was found that Ogive nosecone was the best choice.

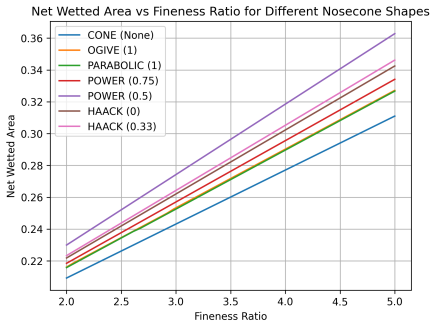

Figure 2. Net surface area vs fineness ratio for various nosecone shapes.

To determine the optimal fineness ratio, CFD was done for fineness ratios of 2.8, 3.0, 3.2, and 3.4 of Tangent Ogive nosecone. As the fineness ratio of nosecone increases, there is more space for payload to be placed inside the nosecone reducing the fuselage length which reduces the skin friction drag but at some point, skin friction drag starts to increase again because of increasing surface area of nosecone. The optimal fineness ratio was found to be 3.2.

Table 1. Drag Coefficients of different Nosecone Shapes

Shape	Drag coefficient
Conical	2.9947
Parabolic	0.3702
Ogive	0.3632

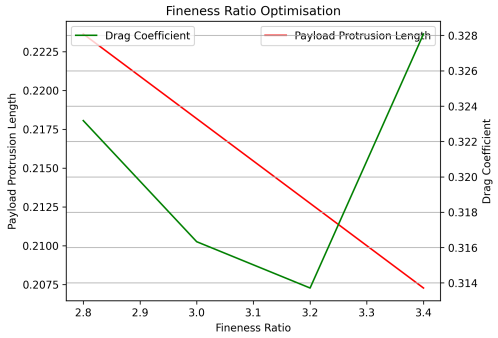

Figure 3. Drag coefficient vs Fineness ratio of an Ogive Nosecone

B. Body Tube

The body tube thickness was chosen to be 2mm considering compressive stresses and buckling strength of the GFRP used. Additionally, bending analysis was done to find possible maximum bending stresses at MaxQ.[16] The maximum bending stress was found to be within the limits of the GFRP.

A dynamic gust analysis was done with one degree of freedom to know the maximum angle of attack possible at each mach number. The gust is modelled using a 1-cosine gust[24] with a gust length of 300m and gust windspeed as 23 knots.[?]

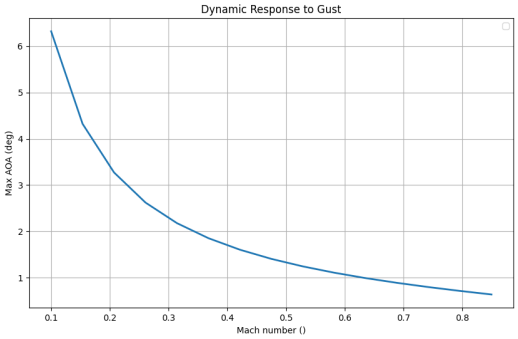

Figure 4. Maximum angle of attack in a gust at different Mach numbers.

In this simulation, effects of aerodynamic damping is neglected as we are only interested with maximum angle of attack. The simulation is started from Mach 0.1 as for any speed below Mach 0.1, there is no enough lift to correct the lift resulting in very high angle of attacks. The rocket is modelled as a cylinder to obtain moment of inertia. At MaxQ, an angle of attack of 1° was taken for bending analysis.

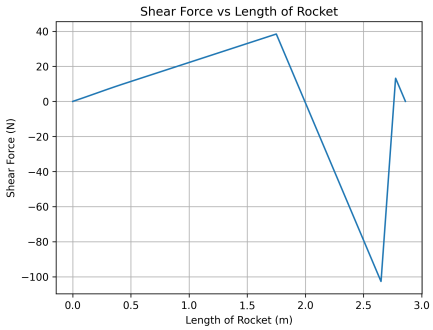

Figure 5. Shear force along the axis of the rocket

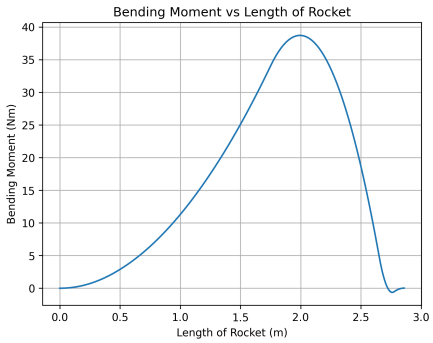

Figure 6. Bending moment along the axis of the rocket

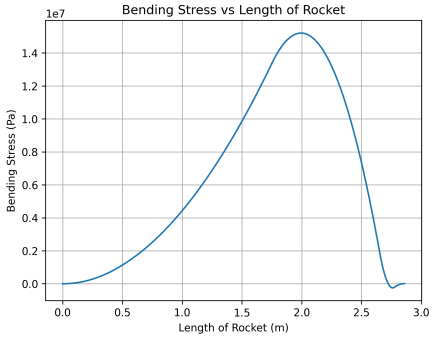

Figure 7. Bending stress on the body tube along the axis of the rocket

Arjuna consists of three body tubes. The Payload-Avionics bay and the Recovery bay body tube has been made of GFRP while the Motor bay has been made of CFRP.

C. Fins

Fin attachment in Arjuna is done using a fin can setup with trapezoidal fins having square cross section. The fins are attached to the stringers using TIG welding.

The dimensions chosen for fin are from OpenRocket to obtain best performance. Height, Root chord, Tip chord, Sweep length were decided to obtain the minimum stability caliber of 1.45 as per the requirement provided in the Problem Statement. Thickness of the fin was decided based on fin flutter velocity of 569.85 m/s obtained from FinSim to maintain the fin flutter velocity 50% higher than the maximum velocity of 293 m/s (Mach 0.857).

Table 2. Fin Dimensions

Name	Value
Root Chord	12.4cm
Tip Chord	6.1cm
Height	14.7 cm
Sweep length	8.29cm
Thickness	0.3cm

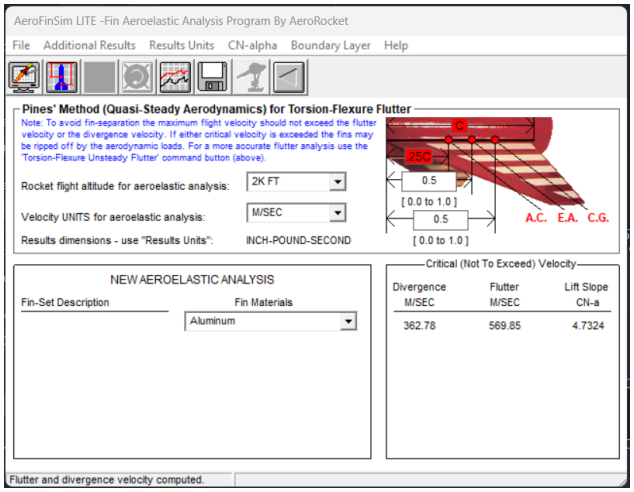

Figure 8. FinSim

D. Pitot Tube

The static port of the pitot tube is placed in the avionics bay where the pressure coefficient along the body tube is almost 0. To make sure the pressure in the avionics bay is equalised with the static pressure as the rocket is travelling, a quasi steady approximation of orifice flow was used. An error of 1.4% was found by using two 5 mm holes which will also be used to place pull pins.

$$\dot{m} = C_{dis} A \sqrt{2\rho |P_{atm} - P_c|}$$
$$\frac{dP_c}{dt} = \text{sgn}(P_{atm} - P_c) \cdot \frac{RT}{V} \dot{m}$$

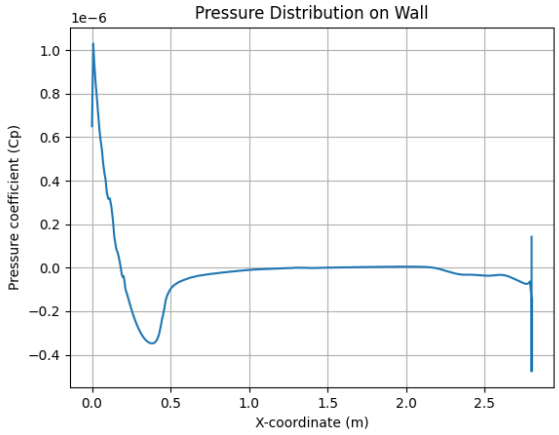

Figure 9. Pressure coefficient distribution along the axis of rocket

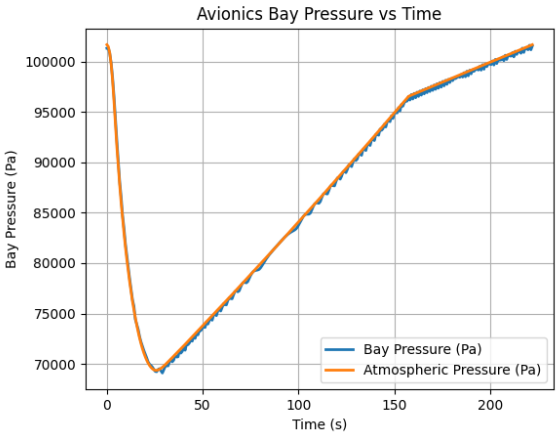

Figure 10. Avionics Bay Pressure

The stagnation port of the pitot tube is integrated with the nosetip. This decision was taken to reduce the additional pressure drag cause by two aerocovers. The dimensions of the stagnation port was designed to make sure that the range of angle of attack upto which the error is under 5%.[21] CFD was run to calculate this error at different Mach numbers at 15 degree angle of attack. The error was found to be under 5% always.

The stagnation pressure is carried through a silicone tube passing through the payload bay reaching the avionics bay undergoing three bends. To take account for the friction in the pipe and the bending losses, 1D transient flow equations were simulated which resulted in an error under 1%.[22] The python code used to run this simulation is given in the Appendix.

E. Airbrakes

The airbrake was implemented using a Slider crank mechanism with two drag plates of size 40x50cm which was sufficient enough to stop the rocket at 10,000 ft if deployed at Mach 0.7 in the coasting phase. CFD was run for different Mach numbers at two extension ratios (100% and 50%). Polynomial regression was done on the drag coefficients obtained of the drag plate as a function of extension ratio and Mach number.

$$C_d = 9.6187 \cdot Ma \cdot u - 19.0139 \cdot Ma^2 \cdot u + 1.1245 \cdot Ma^3 \cdot u + 3.7661 \cdot Ma^2 \cdot u^2 - 3.0457 \cdot Ma \cdot u^3$$

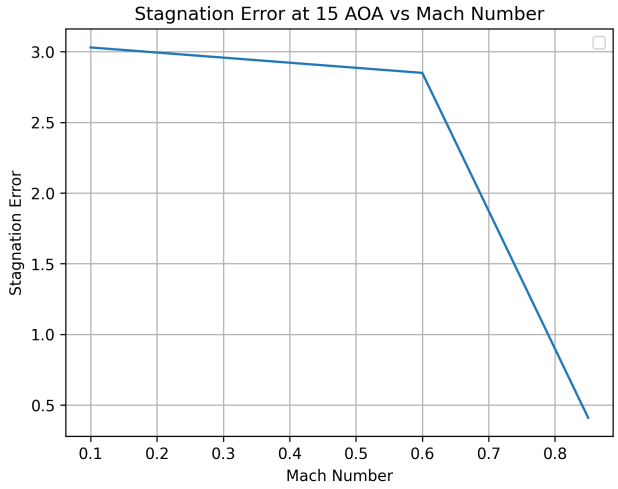

Figure 11. Pitot Stagnation Error at 15 AOA

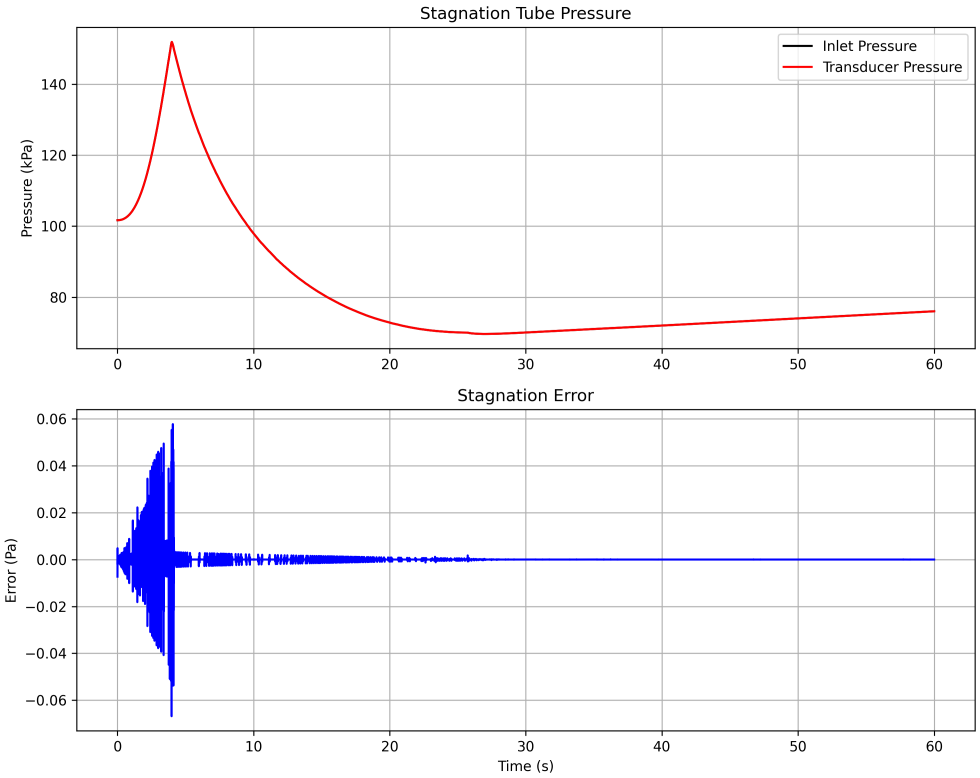

Figure 12. Pitot Stagnation Error

F. Aerocover

The forebody of the aerocover was designed like a 3:1 Tangent Ogive to house the 14mm camera with a 35° inclination. The drag coefficient of the aerocover was found to be 0.004 from CFD. It is positioned opposite to the rail button to reduce the asymmetric moment produced by the aerocover.

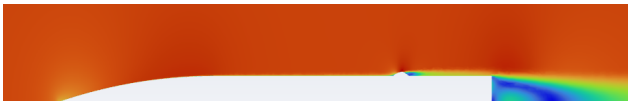

Figure 13. CFD for Aerocover at Mach 0.85

G. Recovery

The rocket is recovered using a single bay dual-event parachute recovery system. This was implemented to reduce the length of the rocket and additional weight of structural elements that would have been for a dual bay deployment. This system consists of a single recovery bay having a joint coupler with avionics bay attached with 4 #6-32 shear pins. The drogue parachute is deployed at apogee using 4F black powder as the ejection charge, and the main parachute is deployed at 1500ft AGL with the Tender Descender TD-2 Mechanism. The recovery bulkhead has both main and redundant charge. Similarly, the Tender Descender also has both main and redundant E-match wires. The main charge is fired using a COTS Altimeter while the redundant one using SRAD Altimeter.

The drogue and main parachutes were selected based on the descent rate, i.e., 20-40 m/s for drogue and under 11 m/s for main. Descent rate calculations were done using this formula,

$$d = \frac{1}{v}\sqrt{\frac{8mg}{\pi\rho C_D}}$$

This resulted in a 36" Compact Elliptical Parachute (C_D of 1.55) for drogue and a 72" Iris Ultra Parachute (C_D of 2.2) for main. The quantity of black powder was chosen so that the pressure developed is enough to break the 4 #6-32 shear pins.[19]

$$F = 4GA_{sp}$$
$$P = \frac{F}{\pi\frac{D^2}{4}}$$
$$m_{BP}(\text{g}) = 454\text{g/lbf}\frac{P(\text{psi})V(\text{in}^3)}{266\frac{\text{inches lbf}}{lbm}3307R}$$

Here, G is shear strength of nylon pins (75 MPa), A_{sp} is the area of shear pin bearing the shear (3.5mm dia), V is volume of the recovery bay (0.007 m^3). This resulted in about 5g of Black Powder assuming an empty recovery bay. For the redundant charge well, 6g of Black Powder was taken.

Charge well thickness was calculated using ASME Boiler Pressure Vessel Code for cylindrical shell,.[?]

$$t = \frac{PR}{SE - 0.6P}$$

The shock loading and deployment force was found to be 1894N and 1872N respectively and was calculated using the following formulae,[20]

$$F_{sl} = mg(\frac{v_t}{v_d})^2$$
$$F_d = C_d \cdot \frac{1}{2}\rho v_d^2 \cdot \pi\frac{d^2}{4}$$

Swivels, Quicklinks and Eyebolts were chosen so that they are able to bear the shock loading and deployment forces.

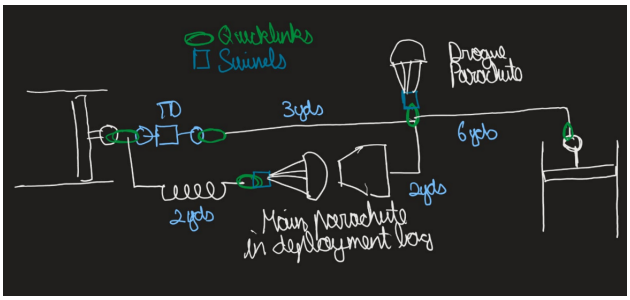

Figure 14. Recovery circuit during Drogue descent

III. Propulsion

The propulsion system of Team Atlas uses an N-class SRAD solid rocket motor which uses coarse KNSB in a ratio of 65:35 (oxidizer/fuel) as the propellant. The theoretical total impulse of the motor is 13642.2 Ns. The average thrust is 3316.1 N. The maximum expected operating pressure (MEOP) is 5.43 MPa. The motor shows a slightly regressive neutral thrust-time curve, and the burn time is 4.11 seconds.

Team Atlas Solid Rocket Motor Specifications

Motor Casing		**Nozzle**	
Mass [g]	9699.3	Mass [g]	3221.4
Internal Diameter [mm]	97	Throat Diameter [mm]	24.5
External Diameter [mm]	114	Nozzle Exit Diameter [mm]	78.5
Material	Al6063 T6	Expansion Ratio	10.27
		Material	AISI 310 Stainless Steel
Forward Closure		**Bates Liners (×6)**	
Mass [g]	825.4	Mass [g]	169.23
Diameter [mm]	97	Length [mm]	162
Thickness [mm]	60	Outer Diameter [mm]	97
Material	Al6063 T6	Inner Diameter [mm]	93
Bottom Liner		**Grain (×6)**	
Mass [g]	261.9	Mass [g]	1547.26
Length [mm]	155	Length [mm]	155
Outer Diameter [mm]	97	Outer Diameter [mm]	93
Inner Diameter [mm]	93	Inner Diameter [mm]	35

A. A. Motor Casing

The motor casing is made of 6063-T6 aluminium, with an external diameter of 114 mm, a height of 1280 mm, and a thickness of 8.5 mm. The casing is designed to withstand a pressure of 10.64 MPa. The MEOP considered while designing the rocket motor was 7 MPa with a factor of safety of 1.5. The maximum pressure achieved according to SRM calculations is 5.43 MPa.

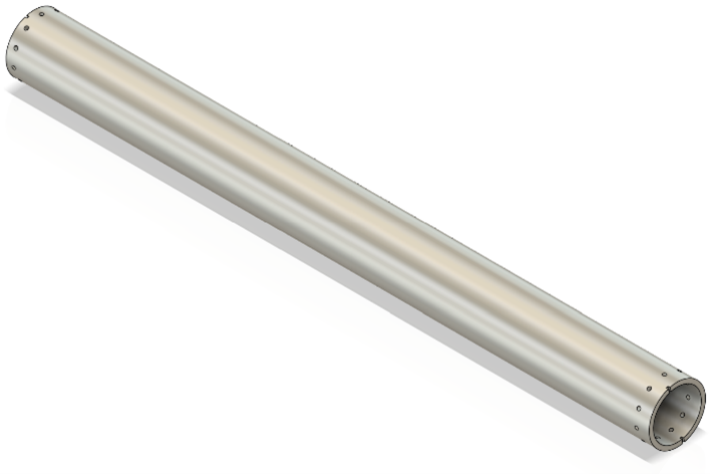

Figure 15. Motor Casing

B. B. Internal Ballistics

The internal ballistics of the motor were simulated using SRM sheet of Richard Nakka using custom burn rate coefficients for coarse KNSB. There are 7 KNSB BATES grains used in the motor, each of 155 mm length with a core diameter of 35 mm. The BATES grain geometry was chosen due to ease of casting setup compared to conocyl or finocyl geometries.

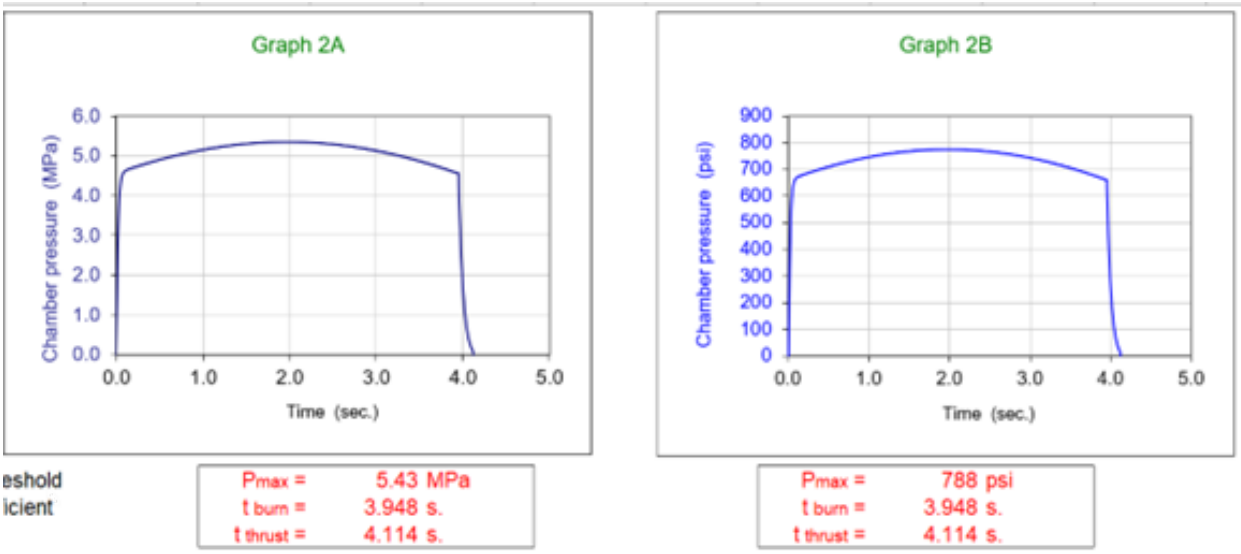

Figure 16. SRM Chamber Pressure Curve and Thrust-Time Curve

The L/D ratio is the ratio between the total length of grain segments and the outer diameter of the grain. The L/D ratio for the given grain segments is 11.67. A value of approximately 12 is used as a threshold for this parameter.

The port-to-throat ratio is the ratio between the area of the port (core diameter) to the area of the throat of the nozzle. The port-to-throat ratio for this motor is 2.04, satisfying the condition that it should lie between 2 and 4.

C. C. Forward Closure

The forward closure is made of Aluminium 6063-T6. Its purpose is to prevent combustion gases from escaping from the forward end of the motor.

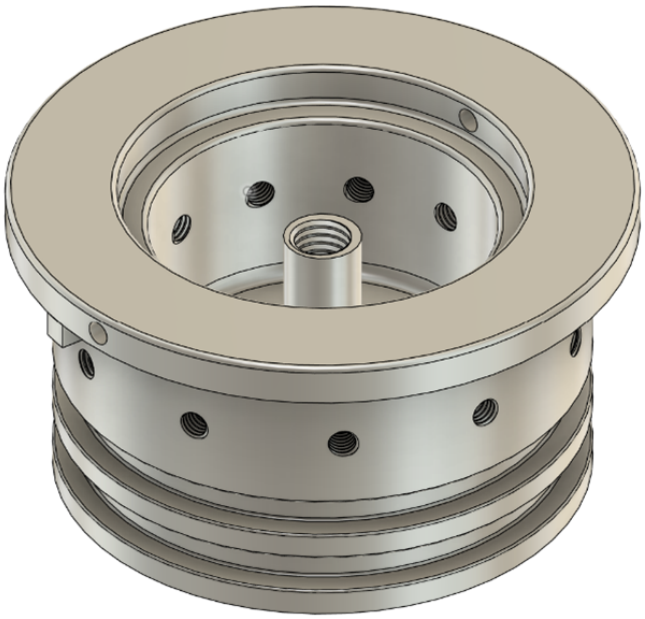

Figure 17. Motor Forward Closure

Two O-rings (Parker 2-340) are used for sealing. The O-ring has an internal diameter of 85.09 mm and a cross-sectional diameter of 5.330 mm. A small amount of AeroL silicon grease is applied to prevent leaks and ensure smooth operation of the O-rings . The O-ring was sized using the Parker O-Ring Selector with appropriate values for squeeze, stretch, gland fill etc.

D. D. Nozzle

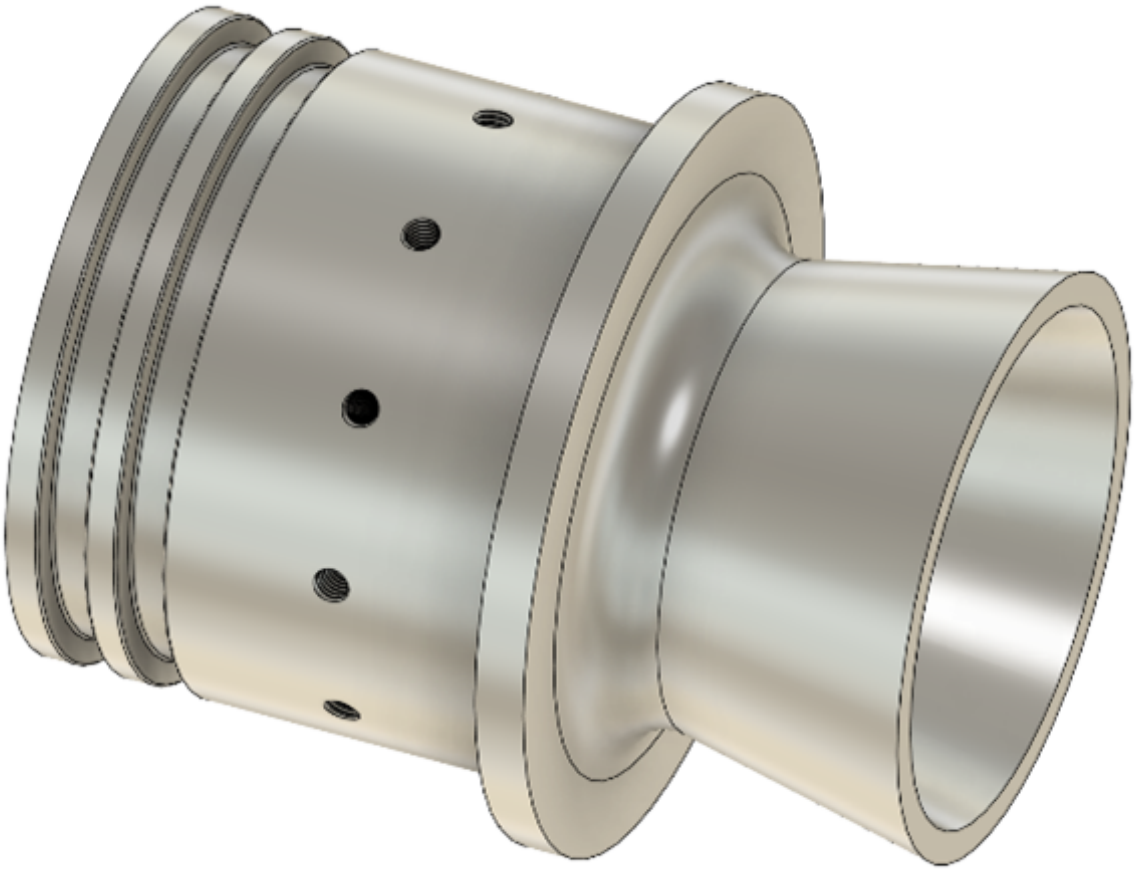

Figure 18. Motor Nozzle

The nozzle is manufactured from AISI 310 stainless steel with a throat diameter of 24.5 mm and an expansion ratio of 10.27.

E. E. Nozzle

The nozzle is made of AISI 310 stainless steel. It has a throat diameter of 24.5 mm and length of 1 mm, a converging half-angle of 35° and a diverging half-angle of 12°, and an exit diameter of 78.5 mm.

The alloy has a yield strength of 136 MPa at 1100°C. Stainless steel is relatively easy to machine into complex shapes such as convergent-divergent nozzles and performs well in short duration burns.

The holder of the nozzle is made out of aluminium 6063 and is used to reduce the overall weight. The nozzle like the forward closure has two O-ring grooves to install O-rings (2-340) to prevent any pressure leaks. The nozzle also has 11 bolt holes through which it will be bolted to the motor casing.

F. F. Thermal Insulation and Liners

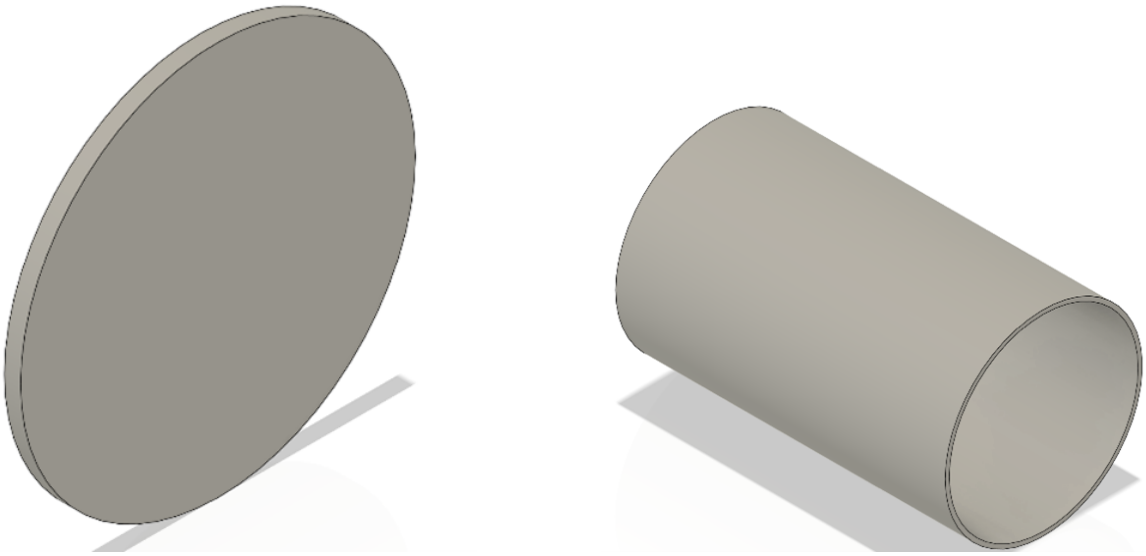

Figure 19. Motor Liner and Head Liner

The thermal insulation is provided by GFRP. A 2 mm thick GFRP tube is laid using 3D printed mandrels. Glass fibre fabric used is 200 GSM bi-directional. GFRP has excellent heat resistance and can withstand the extreme temperatures generated during combustion.

During combustion, glass fiber liners exhibit ablative characteristics. They form a protective char layer that absorbs heat and shields the underlying material from thermal damage.

All liner lengths are extended by 7 mm of the grain length to act as spacers to ensure that the grains are in BATES configuration.

G. G. Propellant Casting

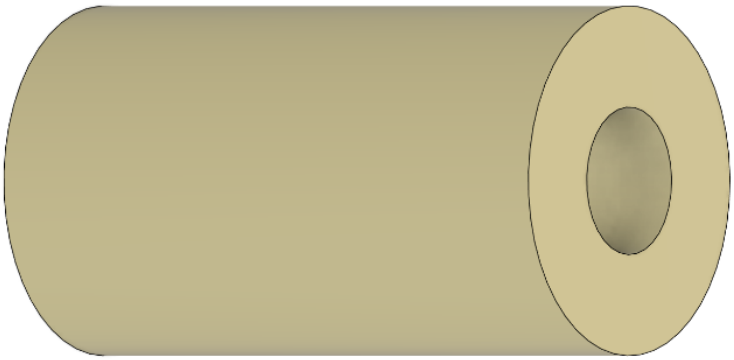

Figure 20. Motor Grain

The grain has outer and inner diameters of 93 mm and 35 mm respectively. There are 7 grains powering the motor.

A 65:35 KNSB formulation is used. It consists of:

- 65% Potassium Nitrate (KN)
- 35% Sorbitol (SB)

The oxidizer-to-fuel mass ratio represents a practical upper limit for casting while maintaining good burn characteristics.

The casting process is done in batches. Since each grain has the same weight, the process is repeated identically for all six grains. The mixture is heated using induction heating and continuously stirred to ensure homogeneity. Once the mixture reaches approximately 120°C, it is transferred to the casting tube.

H. H. Hand Calculations

Design Pressure The design pressure for the casing was calculated using ASME Boiler and Pressure Vessel Code Section VIII (UG-27):

$$P = \frac{SEt}{R + 0.6t} \tag{1}$$

Where:

- R = chamber radius
- t = casing thickness
- S = yield strength
- E = weld joint efficiency (taken as 1)

The calculated design pressure is 10.64 MPa.

With a factor of safety of 1.5:

$$P_{max} = \frac{10.64}{1.5} = 7 \text{ MPa} \tag{2}$$

Forward Closure Thickness The attachment factor C is given by:

$$t = \sqrt{\frac{CP}{SE}} \tag{3}$$

The attachment factor C is 0.33. The minimum thickness calculated is 18.5 mm.

The actual forward closure thickness used is 60 mm, which is well above the minimum requirement.

I. I. Bolting Calculations

Possible failure modes include:

- Bolt shear
- Bolt tear-out
- Casing tensile failure
- Bearing stress

Bolt Shear

$$\tau_{bolt} = \frac{\pi D_{casing}^2 \times MEOP}{4N \times A_{bolt}} \tag{4}$$

The bolt is in single shear mode due to only two layers: closure/nozzle and casing.

Bolt Tear-Out

$$\sigma_{tearout} = \frac{F_{bolt}}{N \times E_{min} \times 2t} \tag{5}$$

Where:

- E_{min} = minimum edge distance (1.5 × bolt diameter)
- Minimum hole spacing = 2.2 × bolt diameter

All calculated stresses are within allowable limits.

Bearing Failure Bearing failure occurs when the force of the bolts pushes against the edge of the hole, causing the casing material to yield.

Yield strength of Al6063-T6 at 100°C is taken as 220.8 MPa.

$$\sigma_{\text{bearing}} = \frac{F_{\text{bolt}}}{N \times d_{\text{bolt}} \times t} \tag{6}$$

Casing Tensile Failure This occurs when the aluminium casing material between bolt holes stretches beyond its breaking point.

$$\sigma_{\text{tensile}} = \frac{\frac{\pi D_{\text{casing}}^2}{4} \times MEOP}{(D_{\text{casing}} - N \times d_{\text{bolt}}) \times t} \tag{7}$$

According to the stress calculations, 11 M6 × 1.25 Class 12.9 alloy steel bolts were selected with a minimum factor of safety of 2.7 against bearing failure.

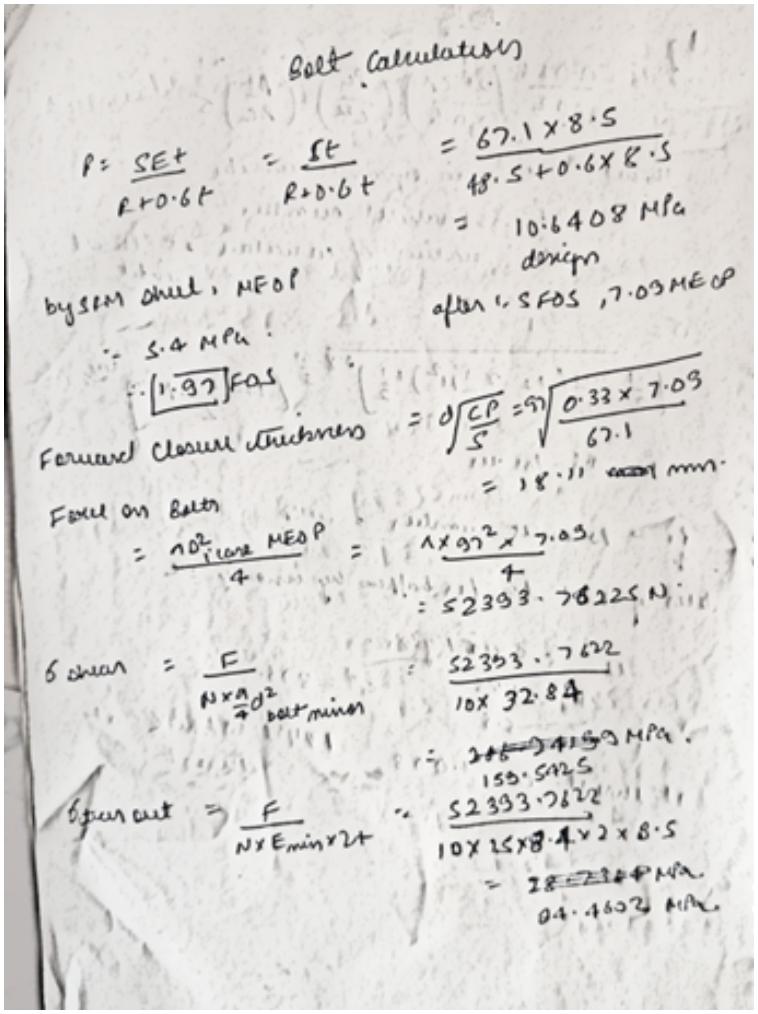

Figure 21. Hand Calculations for Bolt Stresses

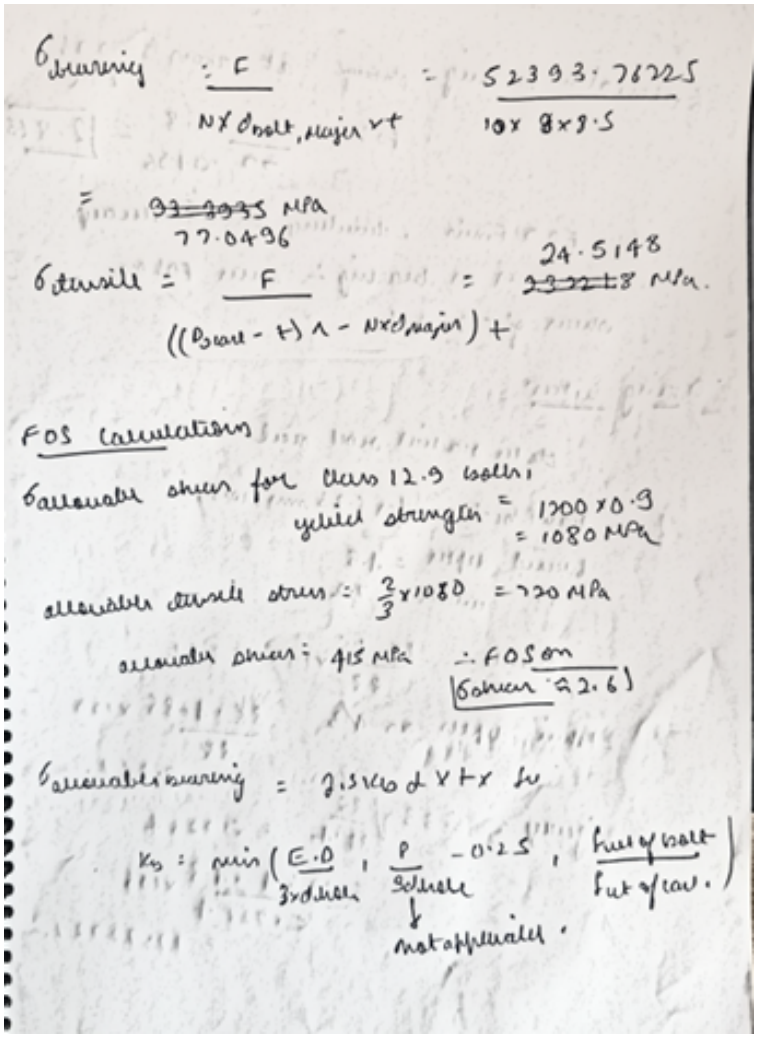

Figure 22. Additional Hand Calculations

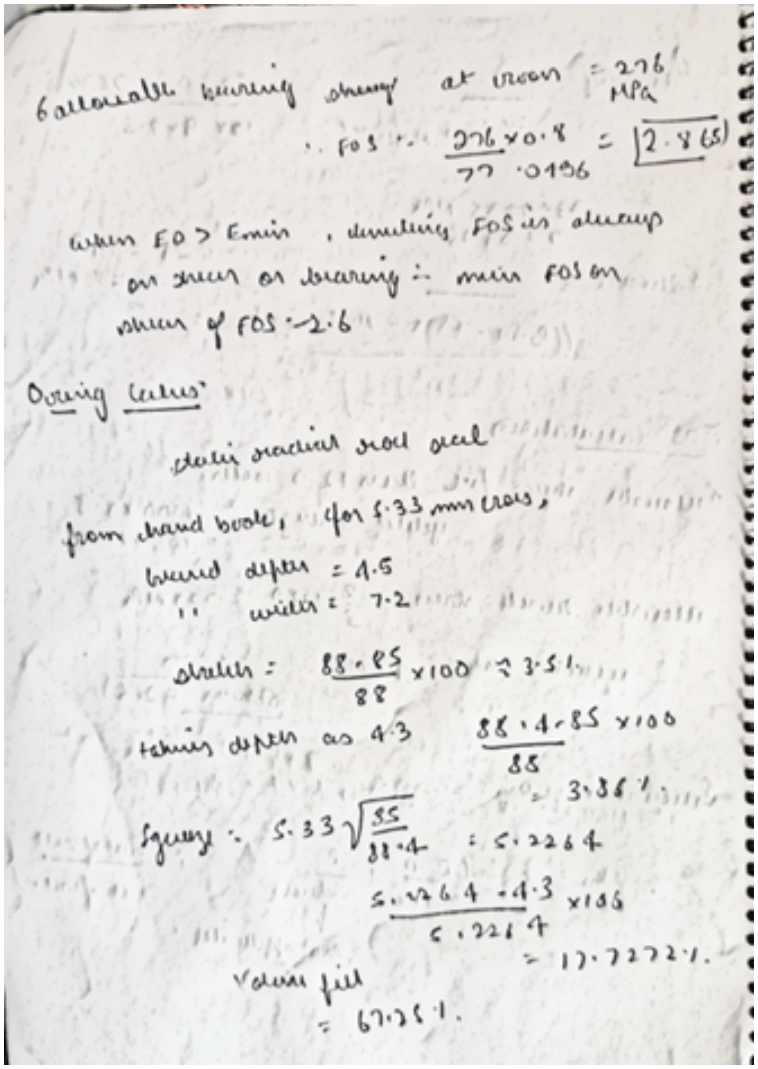

Figure 23. Additional Hand Calculations

J. J. Simulations

The bartz model was used to get the film coefficient values for the convective loads. Isentropic equations were used to get the temperatures from start of nozzles converging to the exit. The temperature values at five diff sections for converging and diverging section were calculated and then hg values were obtained at those then avg was taken and applied for the whole converging section which came out to be around 12600W/m2K for converging, 1500W/m2K for chamber, 17200W/m2K at throat and 6200W/m2K for diverging

$$h = \left[\frac{0.026}{D_t^{0.2}}\left(\frac{\mu^{0.2}c_p}{Pr^{0.6}}\right)\left(\frac{p_c g}{c^*}\right)^{0.8}\left(\frac{D_t}{r_c}\right)^{0.1}\left(\frac{A_t}{A}\right)^{0.9}\right]\sigma \tag{8}$$

$$\sigma = \left[\frac{1}{\left(1+\frac{\gamma-1}{2}M^2\right)^{0.8}}\right] \tag{9}$$

Values of dynamic viscosity and cp were taken from propep to be 5.3*10-5kg/m and 1685J/Kg-K. while the rest of the values were taken from SRM sheet and the wall temps were calculated using the isentropic equation.

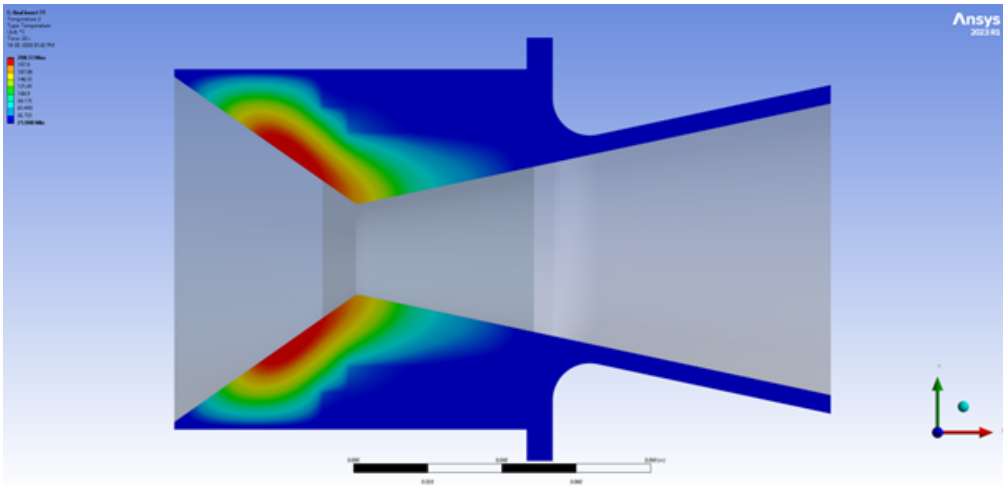

Figure 24. Nozzle Transient Thermal Analysis

NOZZLE TRANSIENT THERMAL The whole nozzle was assumed to be steel 310 and transient thermal was run. Burn time was set to 4.1 seconds followed by a 20 second soak time.Dimensions of the aluminium holder was calculated from this simulation. The mesh has an average skewness of 0.27 with aspect ratio of 4.8

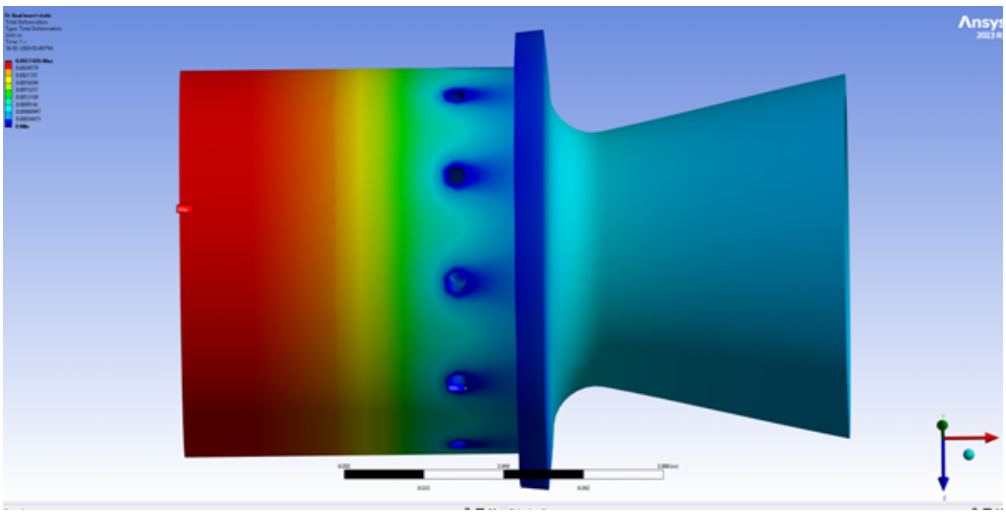

Figure 25. Nozzle Static Structural Analysis

NOZZLE STATIC STRUCTURAL The static structural analysis of the steel nozzle insert along with the holder gave a max deformation of 2.7mm.

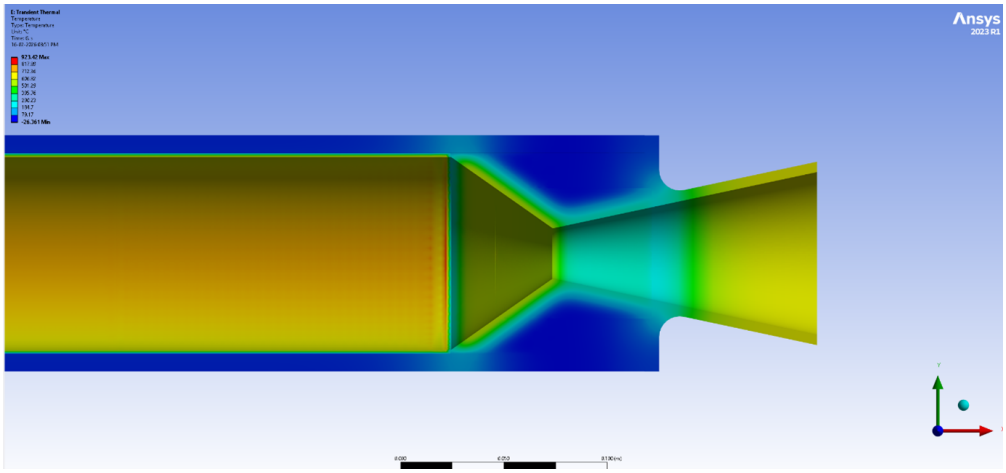

Figure 26. Motor Transient Thermal Analysis

MOTOR TRANSIENT THERMAL The mesh had an average skewness of 0.27 with an aspect ratio of 4.8.

IV. Avionics

The avionics system for Arjuna is named "Sanjaya", which means vision. It comprises two independent configurations: a Commercial Off-The-Shelf (COTS) system and a redundant Student Research and Development (SRAD) system. Both systems are responsible for executing recovery procedures, specifically parachute deployment and tracking of the rocket's position. The SRAD system comprises a microcontroller which processes the acquired data from various sensors during flight. The system runs on an FSM-based architecture that determines the current state of the rocket for recovery events, real-time data logging and transmission to the ground station through telemetry, which is displayed on the plotter, along with that the SRAD system is capable of performing live-video telemetry.

A. COTS Electronics

The Commercial Off-The-Shelf (COTS) electronics system for *Ekalavya* serves as the primary avionics suite, incorporating an Altimeter and GPS the facilitates safe recovery of the rocket and provide official altitude logging. The **EasyMini Altimeter** serves as the COTS altimeter, along with that the **Featherweight GPS** provides flight tracking capabilities, ensuring the acquisition of precise and reliable position data throughout the flight. The specifics and features of these modules will be detailed in a later section.

Table 3. COTS Component Specification

Component	Description	Features
Featherweight GPS	GPS Tracker	• Max Altitude - above 100,000 ft. • Real-time tracking, excellent for high altitude and long-drift recovery.
EasyMini Altimeter	Altimeter	• Barometric Apogee Detection • Can read data up to 30km • 1MB onboard flash

B. SRAD Electronics

The Student Research and Development (SRAD) avionics system, named *Sanjaya*, is implemented on a four-layer Surface Mount Technology (SMT) PCB to enhance signal integrity, electromagnetic compatibility, and structural robustness. The system employs a real-time dual-core processing architecture designed to decouple time-critical sensor acquisition from computationally intensive estimation and communication tasks. The flight computer is built around the Raspberry Pi RP2354B microcontroller.

1. System Objectives

The primary objectives of the SRAD avionics system are summarized in Table 4.

Table 4. SRAD System Objectives

Function	Description
Real-time Apogee Detection	data acquisition of pressure for reliable apogee detection.
Safe Recovery of the Rocket	Reliable triggering of parachute deployment mechanisms.
Data Logging and Telemetry	Continuous onboard logging and real-time transmission for post-flight analysis.
Altitude Control System	Deployment of airbrakes during coast phase for apogee regulation.

2. SRAD Hardware

The components integrated into the flight computer are listed in Table **??**.

3. Ignition and Storage

The SRAD board integrates two MOSFET-based pyro channels for drogue and main deployment. An onboard SD card module provides redundant data storage.

An external 12 MHz crystal oscillator is incorporated to provide a stable and low-jitter reference clock. Compared to internal RC oscillators, the external crystal significantly improves timing precision for USB, SPI, and UART interfaces.

4. System Integration

The complete avionics assembly is packaged to minimize EMI coupling and vibration sensitivity. Figure 27 illustrates the 3D view of the Sanjaya flight computer assembly.

C. Flight Software Architecture

1. Dual-Core Asymmetric Multiprocessing (AMP)

Core 0 operates as the primary computational engine, responsible for synchronized sensor data acquisition, real-time state estimation, and the atomic updating of shared memory buffers. Core 1 continuously accesses these buffers to execute persistent data logging and telemetry downlink tasks without impacting the primary control loop. This separation ensures that the variable execution times inherent to logging and telemetry never impact the deterministic timing of the flight control loop.

2. Core 0: The Flight Stabilization Engine

Core 0 operates on a strict, timer-driven interrupt cycle at 150 Hz. In every 6.67 ms window, it performs synchronized sensor acquisition (IMU, Barometer, Magnetometer), executes the sensor fusion algorithms (EKF and Madgwick), and computes the airbrake control output. This core executes no file system or

Table 5. SRAD Hardware Components and Selection Rationale

Component	Description	Manufacturer	Selection Rationale
RP2354B	Microcontroller	Raspberry Pi	Dual-core architecture (up to 150 MHz, 520 KB SRAM) enables separation of time-critical sensor acquisition from estimation and I/O tasks. High GPIO availability and deterministic execution make it suitable for flight-critical applications.
LSM9DS1	9-DOF IMU	STMicroelectronics	Provides acceleration up to $\pm 16g$, gyroscope up to ± 2000 dps, and sampling rates up to 952 Hz. High data rate supports accurate velocity estimation during boost and coast phases.
DPS368	Barometric Pressure Sensor	Infineon Technologies	Operates over 300–1200 hPa with $\sim$0.002 hPa resolution using a 24-bit ADC, enabling precise altitude estimation required for reliable apogee detection.
MPXV5100DP	Differential Pressure Sensor	NXP	Measurement range suitable for dynamic pressure estimation and Mach number computation, supporting airbrake control logic.
RYLR993	LoRa Telemetry Module	Reyax	Operates at 915$\tilde{\text{M}}$Hz with 22$\tilde{\text{d}}$Bm transmit power and 138$\tilde{\text{d}}$Bm sensitivity, providing reliable long-range communication (tested up to 8$\tilde{\text{k}}$m).
u-blox NEO-M9N	GPS Module	u-blox	High update rate positioning with 2.0 m CEP accuracy and low power consumption, enabling reliable recovery tracking.

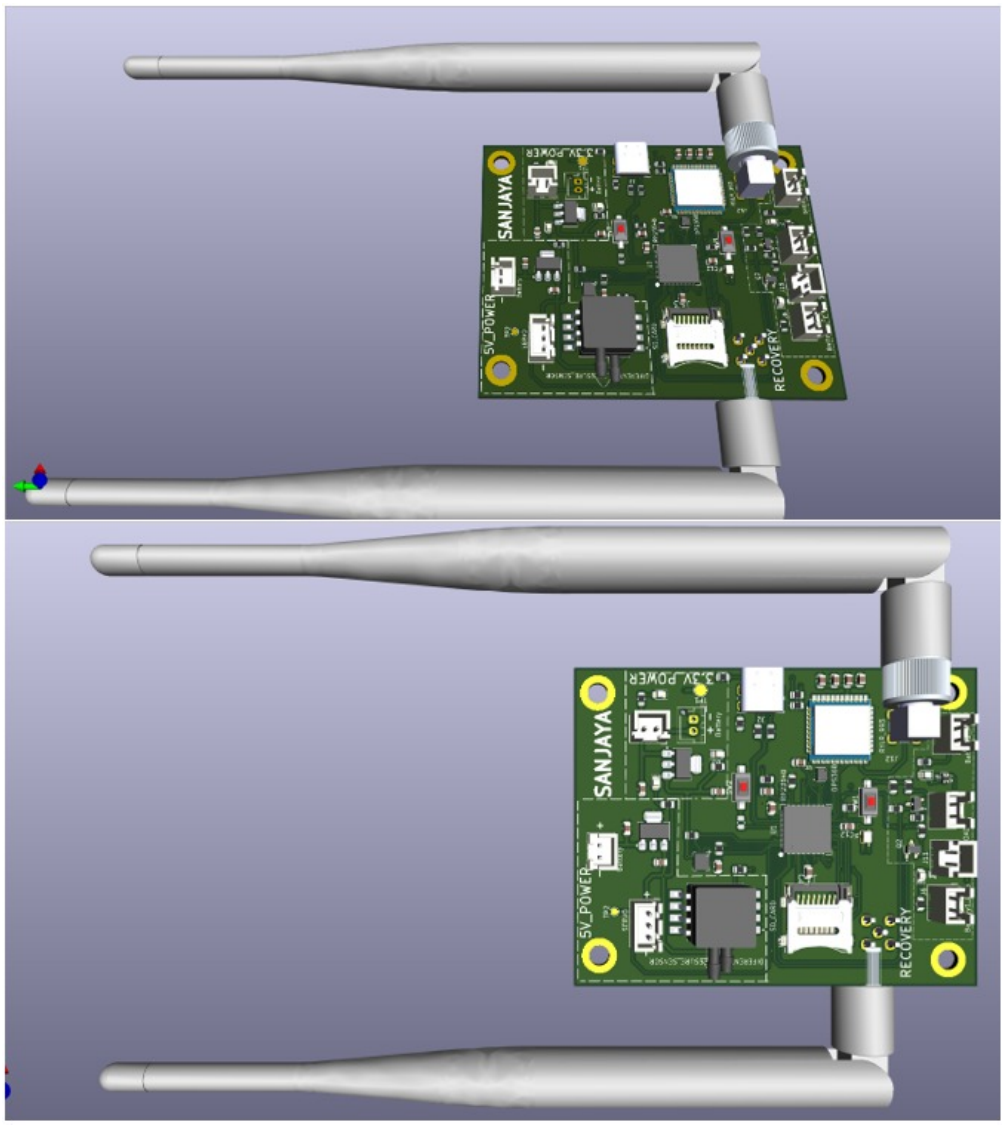

Figure 27. 3D view of the Sanjaya avionics system.

Table 6. Battery Configuration Details

S/N	Battery Name	Description	Function
1	Li-ion 18650	Two 3.6V 2800mAh battery of 13C for powering 3.3V rail and 5V	Powers 3.3V line and 5V line
2	Li-ion 18650 – Pyro battery	Two 3.6V 2800mAh battery of 13C for powering Pyro charges	Provides power to Pyro MOSFETs

radio operations, guaranteeing that the time between sensor read and servo actuation remains constant and minimized.

3. Core 1: Logging and Telemetry

Core 1 is dedicated exclusively to the system's high-latency tasks: persistent logging and remote telemetry. It operates in a continuous, asynchronous loop that retrieves flight data from the shared buffer. It writes this data to the microSD card in binary format for high-speed storage and simultaneously packages key metrics for transmission via the LoRa radio link. By confining these blocking operations to Core 1, the system isolates Core 0 from the unpredictable write latencies of the SD card and the timing constraints of the radio module.

4. Zero-Latency Data Handoff

To transfer data from the real-time domain (Core 0) to the I/O domain (Core 1), the system employs a lock-free Ring Buffer. Upon completing a control cycle, Core 0 pushes a state snapshot into this shared memory queue and immediately returns to its flight tasks. Core 1 consumes these snapshots as resources permit. This "fire-and-forget" mechanism ensures that Core 0 never waits for Core 1, preventing the slower logging and telemetry processes from creating back-pressure on the control loop.

5. Hardware Offloading (DMA)

The system further reduces CPU load by utilizing Direct Memory Access (DMA) for background sensor ingestion. The analog Pitot tube is continuously sampled and averaged by the ADC hardware, while GPS data is buffered via UART DMA channels. This allows the processor to access the most recent averaged airspeed and position data instantly during its execution cycle, without expending clock cycles on managing sensor data transfer.

D. Flight State Machine

The flight computer autonomously manages the mission through an eight-state Finite State Machine (FSM). The transition logic is threshold-based rather than time-based to ensure safety. The states and triggers are detailed in Table 7.

Table 7. Flight State Machine (FSM) Transition Logic

State	Transition Trigger	System Actions
STANDBY	Vertical Accel $> 2.0g$ (5 consecutive samples)	Exit low-power mode, initialize high-rate logging
BOOST	Vertical Accel $< 0.0g$ (5 consecutive samples)	Log Motor Burnout, EKF High-Noise Mode
COAST_HIGH_MACH (Mach > 0.7)	Mach Number < 0.7 (10 consecutive samples)	EKF Coast Mode, Airbrakes Shadowing (Closed)
COAST_MID_MACH ($0.30 <$ Mach < 0.7)	Mach Number < 0.30 (10 consecutive samples)	**Airbrake Authority Active**, Control Loop Enabled
COAST_LOW_MACH (Mach < 0.30)	Altitude decreasing (10 consecutive samples)	Revoke Airbrake Authority (Retract)
DROGUE	Altitude < 457 m AGL (10 consecutive samples)	Fire Drogue Pyro Channel
MAIN	Pressure Variance < 0.5 Pa (20 consecutive samples)	Fire Main Pyro Channel
RECOVERY	None (Terminal State)	Stop logging, Disable High-Power Sensors, Clear Recovery Memory

E. Filter Design

An Extended Kalman Filter (EKF) is implemented onboard to estimate vertical velocity in real time by fusing barometer altitude measurements with accelerometer-derived specific force. The filter formulation is summarized below.

1. State Vector

$$\mathbf{x} = \begin{bmatrix} h \\ v \\ b \end{bmatrix}, \tag{10}$$

where h is altitude, v is vertical velocity, and b is accelerometer bias.

2. Process Model

The discrete-time nonlinear process model is

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, u_k) = \begin{bmatrix} h_k + v_k \Delta t \\ v_k + (u_k - b_k - k\ (v_k|v_k|))\,\Delta t \\ b_k \end{bmatrix}, \tag{11}$$

with

$$k = \frac{\rho\, C_d\, A}{2}. \tag{12}$$

3. Process Jacobian

The Jacobian of the process model is

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\mathbf{x}_k, u_k)} = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 - k\ (v_k|v_k|)\,\Delta t & -\Delta t \\ 0 & 0 & 1 \end{bmatrix}. \tag{13}$$

4. Measurement Model

The measurement is altitude from the barometer:

$$z_k = \mathbf{h}(\mathbf{x}_k) = h_k, \tag{14}$$

with measurement matrix

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \tag{15}$$

and (equivalently) measurement Jacobian vector

$$\mathbf{G}_k = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{16}$$

5. Control Input

The control input is formed from the measured acceleration:

$$u_k = a_{\text{meas},k} - g. \tag{17}$$

6. Adaptive Covariance

An adaptive covariance approach is used for the EKF noise matrices to account for changing trust in measurements and dynamics across the flight state machine (FSM). Covariances are estimated online from a window of past data, reducing the need for static, phase-specific covariance tuning. :contentReference[oaicite:1]index=1

7. Aerodynamic Coefficient Source

The drag coefficient C_d is derived from ORK flight data, converted into a Mach–C_d lookup table (LUT), and evaluated via interpolation between breakpoints during flight. :contentReference[oaicite:2]index=2

8. Assumptions

The following assumptions are made:

- 1D vertical motion.
- Wind effects are negligible.
- No horizontal coupling.
- Mass is assumed constant within boost and coast phases (different constants between phases).
- Accelerometer bias is modeled as a stochastic random walk and treated deterministically constant in the process model, i.e., $\dot{b} = 0$.

F. Air Brakes System Overview

1. Introduction

Air brakes for *Arjuna* are designed to minimize apogee overshoot and reach the target apogee of 10,000 ft. The system estimates vertical velocity using an Extended Kalman Filter (EKF) and predicts apogee by propagating the current state through a coast-phase model. Two extremal cases are evaluated: (i) air brakes fully retracted and (ii) air brakes fully deployed, enabling selection of an appropriate control input. A Smooth Sliding Mode Control (SSMC) law computes the commanded brake deployment during the coasting phase to regulate apogee.

2. SSMC Overview

Smooth Sliding Mode Control (SSMC) was selected due to (i) its strong performance on nonlinear systems and (ii) inherent robustness to modeling uncertainties. Apogee control is influenced by nonlinearities and uncertainties from Mach effects, density variations, and sensor noise; therefore, SSMC is well-suited for this application. The controller continuously compares predicted apogee with the target apogee and adjusts deployment to reduce the apogee error. A smoothing (saturation) mechanism is used to avoid abrupt on–off actuation and to ensure actuator-friendly commands.

3. System Overview

The air-brake subsystem comprises:

- Barometric sensor (DPS368) for pressure-based altitude estimation,
- IMU (LSM9DS1) for acceleration measurement used by the EKF to estimate velocity,
- Differential pressure sensor (MPXV5100DP) to measure pitot differential pressure,
- Aerodynamic drag lookup table (Mach number vs. drag force) derived from CFD simulations.

4. Apogee Prediction

Predicted apogee is computed by forward integrating the current vertical dynamics during the coast phase (no thrust), accounting for gravity and drag trends. A forward Euler integration is used to propagate velocity and altitude:

$$v_{k+1} = v_k + a_k \, \Delta t, \tag{18}$$

$$h_{k+1} = h_k + v_k \, \Delta t, \tag{19}$$

where a_k includes gravity and drag (with air-brake dependent drag). The integration proceeds until $v \to 0$, and the corresponding h is taken as the predicted apogee.

5. Controller Formulation

The controller compares the predicted apogee $\hat{h}_{\text{apo}}$ with the target apogee h_{ref} through a sliding surface $s(x)$ representing apogee error. The control input $u \in [0, 1]$ corresponds to the commanded air-brake deployment fraction. A generic SSMC law can be written as

$$u = g(x) + K \, \text{sat}\left(\frac{s(x)}{\phi} \right), \tag{20}$$

where $g(x)$ captures the nominal control component, K is the gain, ϕ is the boundary-layer thickness, and $\text{sat}(\cdot)$ is the saturation function.

6. Saturation Law

To ensure smooth actuation, the saturation law is defined as

$$\text{sat}\left(\frac{s}{\phi} \right) = \begin{cases} 1, & s > \phi, \\ -1, & s < -\phi, \\ s/\phi, & |s| \leq \phi. \end{cases} \tag{21}$$

This ensures small apogee errors produce proportionally small brake movements, while larger errors yield stronger corrective action without discontinuous commands.

7. Mathematical Model

A coast-phase vertical dynamics model is used to estimate the influence of air-brake deployment on apogee. The motion is represented by

$$\dot{h} = v, \tag{22}$$

$$\dot{v} = -g - \frac{D(v, h, u)}{m}, \tag{23}$$

where m is rocket mass, g is gravitational acceleration, and $D(\cdot)$ is drag force dependent on flight condition and brake deployment. Drag is computed using a CFD-derived lookup table parameterized by Mach number.

If a design vector C is used to define a linear sliding surface, it may be written as

$$s(x) = Cx, \tag{24}$$

where x is the state vector.

8. Mach Calculation

Air-brake effectiveness varies strongly with Mach number. Mach is estimated using pitot differential pressure and isentropic compressible-flow relations. Using measured differential pressure and ambient/static pressure, the Mach number is computed via the standard isentropic formulation:

$$M = f\left(\frac{q}{P}\right), \tag{25}$$

where q is dynamic pressure and P is static pressure, and $f(\cdot)$ denotes the isentropic mapping for the applicable regime.

9. Results

The algorithm was evaluated on seven flights using ORK data. The observed performance was:

- Absolute mean apogee error: 4 m,
- Mean percentage apogee error: 0.1%. result is provided in Fig. 28.

10. Flowchart

A flowchart summarizing the estimation, apogee prediction, and control logic is provided in Fig. 29.

G. Live Video Telemetry Subsystem

The live video telemetry subsystem was designed to transmit real-time onboard video to the ground station during flight operations. For onboard video capture, the CaddxFPV Ratel 2 camera was selected due to its reliable low-latency analog output and robust operation from a regulated 5 V supply. Video transmission is handled by the Flyrobo 1.2 GHz 2 W 8-channel transmitter, chosen for its strong RF link performance, wide input voltage compatibility, and dependable long-range line-of-sight transmission capability.

On the ground station, video reception is achieved using the Flyrobo 1.2 GHz 12-channel receiver, providing enhanced signal reliability through multi-channel compatibility. The hardware configuration of the video telemetry link is summarized in Table 8.

Battery for powering is Pro range A grade IFR 32650 12.8 V 6000 mAH 3C 4S1P LiFePo4

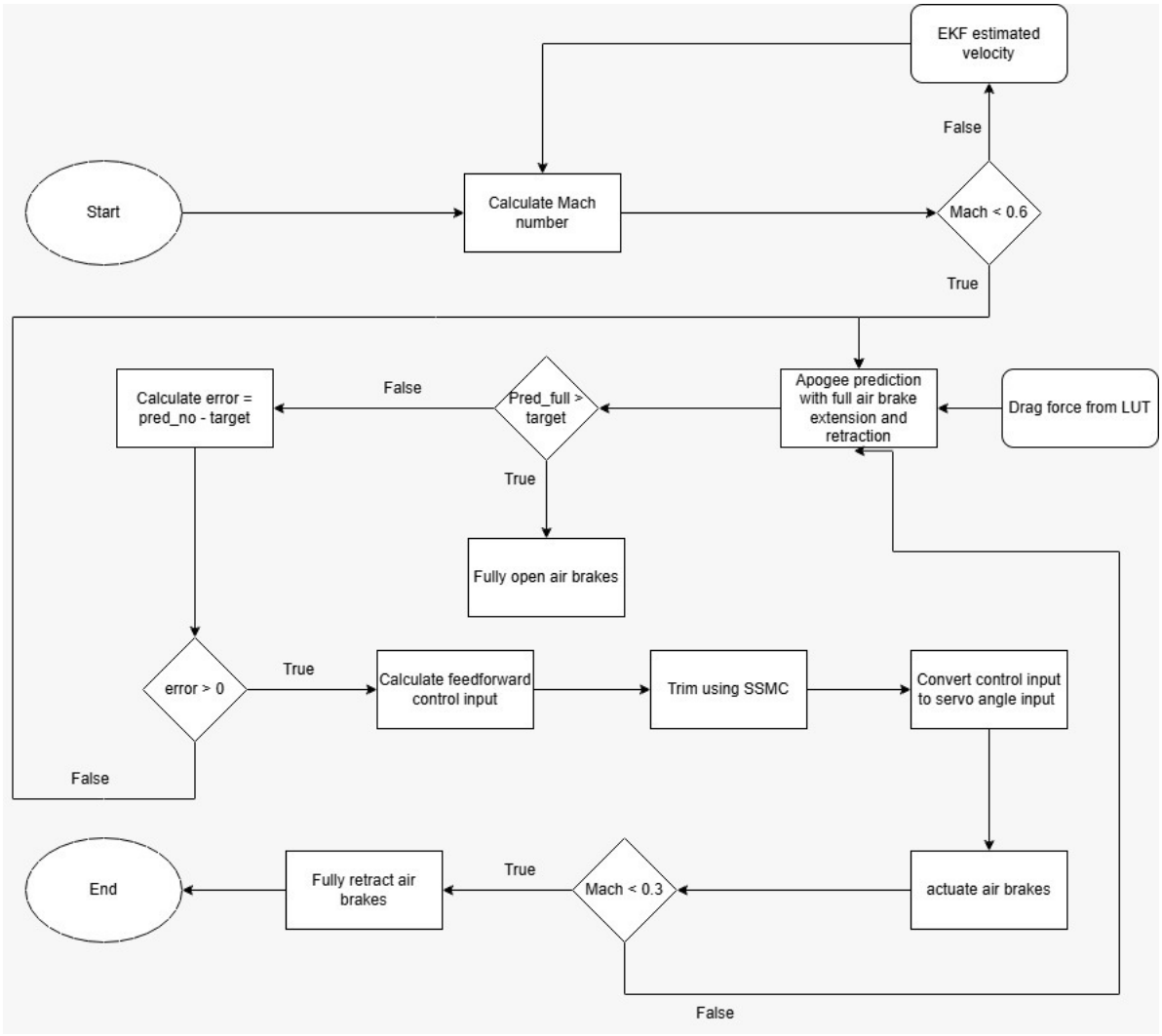

Figure 28. result of air brakes simulation.

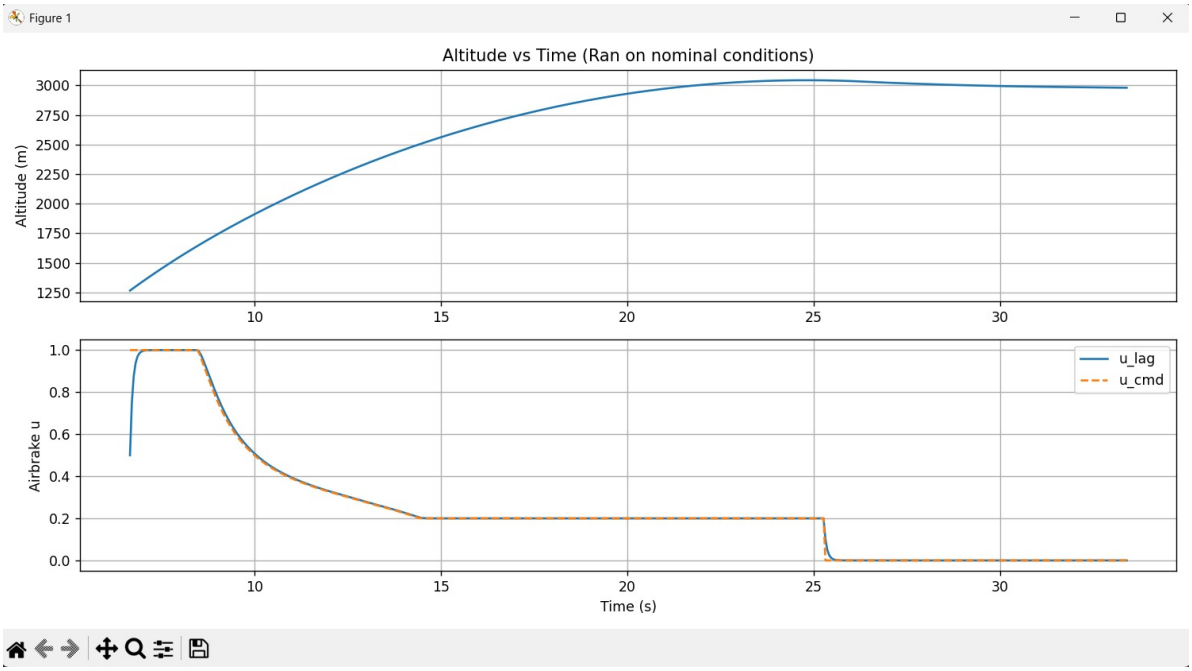

Figure 29. Air-brakes estimation and control flowchart.

Table 8. Live Video Telemetry Configuration

Component	Model	Role / Specification
Camera	CaddxFPV Ratel 2	Low-latency analog video output, 5 V regulated input
Video Transmitter (VTX)	Flyrobo 1.2 GHz 2 W 8CH Transmitter	1.2 GHz RF transmission, compact form factor, long-range line-of-sight capability
Receiver	Flyrobo 1.2 GHz 12CH Receiver	1.2 GHz multi-channel reception for enhanced signal reliability

H. Ignition System

The ignition system named *Arka* comprises a Ground Station and a Launchpad unit both utilize *Arka* constructed on a four-layer SMT PCB. This system features wireless ignition, data logging and data telemetry in real-time. This system utilizes four analog open channels (three active, one backup) for connecting pressure transducer, weight indicator, and thermistors. To enhance the system, both Ground station and Launchpad unit are equipped with SD card storage providing redundancy for logging. Ignition is controlled via MOSFETs and the system runs on FSM-based architecture to ensure safe, secure and smooth operation. The states included in the FSM are SAFE, ARM, LAUNCH and TRAP.

Table 9. Ignition System Component Specification

Model	Function	Key Features	Reason of Choice
ESP32-S3-WROOM-1U-N8R8	Central Processing	1. Xtensa dual-core 32-bit LX7 microprocessor, up to 240 MHz 2. Upto 16 MB QuadSPI flash.	Processing power, memory and enables easy interfacing with peripherals.
RYLR993	Telemetry	1. Range upto 10km 2. Operates at 915 or 868 MHz frequency	Long range and easy to interface using UART and AT commands with a compact form factor.
SD Card Module	Data Logging	1. Houses 8GB microSD Card 2. SPI Iterfacing with SD Card	To logging test data

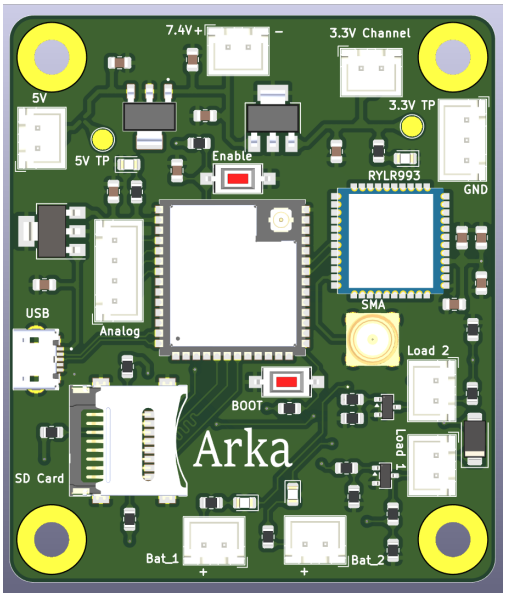

Figure 30. Ignition System PCB - Front View

I. Batteries for Ignition System

The Batteries used for the Ignition system are described in this section. The batteries were selected following the guidelines provided by DTEG in section 6.17, while ensuring components could operate for long hours. 4 X Li-ion 21700 cells with 3.6V 4000mAh 9C, where 2 are used for the board and the other two for igniters.

Table 10. Battery Specification

Component	Form Factor	Battery Specification	Battery Runtime (hours)
Arka PCB	2 X Li-ion 21700 cells	3.6V 4000mAh 9C	6.5+
Igniters	2 X Li-ion 21700 cells	3.6V 4000mAh 9C	4+

J. Real-time Telemetry Ground Station and Data Visualization

A custom ground station application was developed using a modern web-based stack (React.js and Next.js) to provide mission-critical telemetry visualization . The architecture is designed for high-frequency data throughput and low-latency rendering of flight dynamics and engine testbed parameters.

1. Telemetry Processing and Data Architecture

The ground station utilizes the Web Serial API to establish a connection with the modular PCB *Arka* at a baud rate of 115,200, which redirects the data packets through the serial port to the interface. Incoming packets are parsed such that it extracts 22 distinct parameters, including six-axis inertial measurement unit (IMU) data, environmental barometric pressure, and GPS coordinates.

2. Visualization and 3D Mission Monitoring

The dashboard operates in Flight, Motor, and Analyze modes, featuring:

- **Real-time Kinematics:** High-speed *uPlot* charts render altitude, velocity, and G-forces.
- **3D Reconstruction:** A Three.js engine visualizes live attitude and trajectory by projecting GPS coordinates into a local Cartesian workspace.
- **Safety Console for Motor mode:** Critical commands (ARM, LAUNCH) are secured by a numeric challenge-response system to prevent accidental actuation.

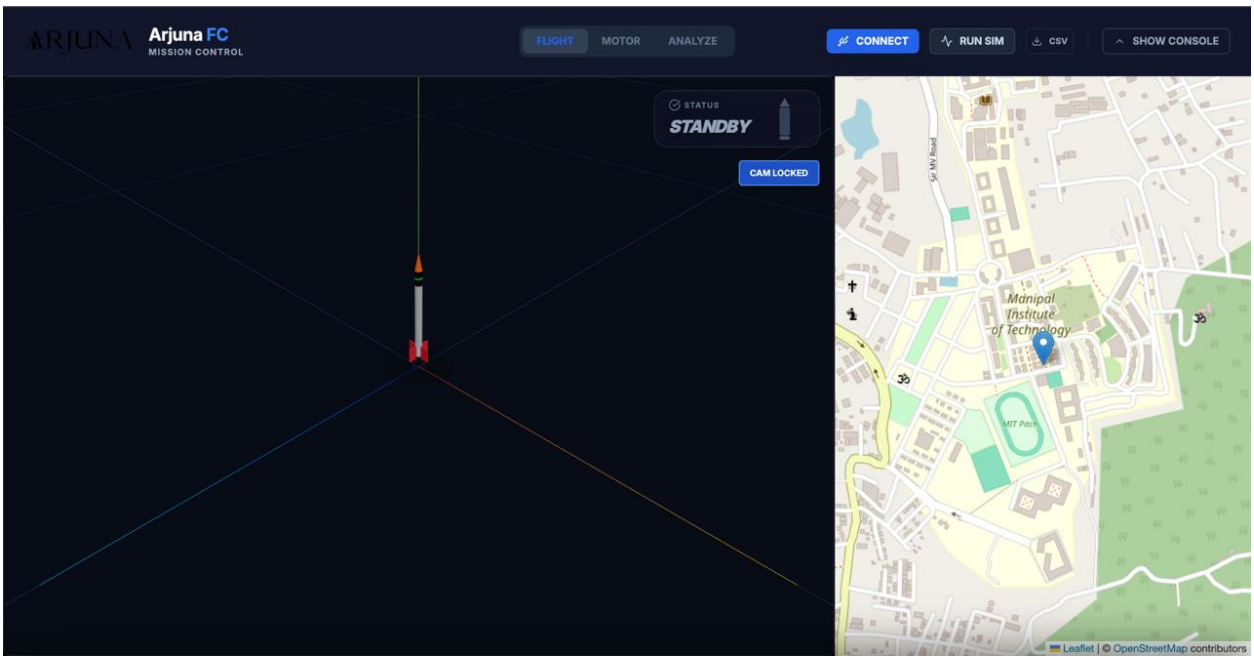

Figure 31. Ground Station Home Screen with 3D simulation space and GPS Map for recovery.

V. Payload

A. Aim:

The payload conforms to the shape of a 3U cubesat. It intends to demonstrate precise control of a ball over a movable platform and compare the visual coordinates from the Raspberry PI camera with the coordinates from the touchscreen to demonstrate the reliability of visual data under high vibrational environment.

B. Key Modeling Assumptions and Rationale

All choices are made to favor physical accuracy where it counts and numerical stability where necessary (integration step, interpolation).

- **Inertial acceleration:** The vehicle inertial acceleration is modeled as $a_{\text{vehicle}} = 10g$ upward. This is the inertial acceleration used to compute the non-inertial frame pseudo-acceleration. The reported effective acceleration used by the dynamics is

$$g_{\text{eff}} = g + a_{\text{vehicle}} \approx 11g \approx 107.9 \text{ m/s}^2,$$

  and this value is read and used consistently by the physics routines.

- **Rolling dynamics:** The ball is a solid sphere rolling without slip. The implemented per-axis acceleration relation is

$$\ddot{x} = \frac{5}{7} g_{\text{eff}} \sin(\theta_x) + a_{\text{dist},x},$$

  and equivalently for the y axis. The code evaluates $\sin(\theta)$ directly; linearized arguments are used only for local interpretation.

- **State representation and integrator:** Second-order mechanics are cast to first-order form with the state $\mathbf{x} = [x, \dot{x}, y, \dot{y}]^\top$. The integrator is classic RK4 executed at a fixed 240 Hz physics timestep. The 240 Hz choice is conservative (approximately 50× the servo mechanical bandwidth) and is enforced in the physics thread for reproducible timing.

- **Actuator model:** Servo mechanics are modeled as follows:

$$\tau\dot{\theta} + \theta = \theta_{\text{cmd}},$$

  with $\tau = 0.033$ s (derived from the vendor travel-time spec using the standard first-order approximation where settling time $\sim 3\tau$). The discrete update uses the simulation timestep to implement the exponential-like response.

- **Deadzone and integral handling:** A 3 mm radial deadzone is implemented. Inside the deadzone the controller output is set to zero and the integral state is frozen (not reset) to avoid step discontinuities on exit.

C. Disturbance Environment and Preprocessing

The disturbance pipeline in the code is designed to be robust and realistic:

1. Disturbances are time-indexed acceleration series (applied independently to X and Y).
2. The pipeline supports both recorded traces and band-limited synthetic disturbances. Example synthetic disturbances used in verification include low-frequency and higher-frequency vibrations.
3. Disturbance amplitudes in validation traces can reach multi-m/s^2 peaks briefly.

D. Performance Summary (Representative Run)

Performance metrics below are computed directly from the simulation traces.

Table 11. Representative steady-state metrics

Metric	X axis	Y axis
RMS position error (mm)	9.73	6.74
RMS servo tracking error (deg)	1.139	0.811
Position–tilt correlation	0.943	0.941

Narrative: After initial transients, the controller maintains the ball within a few millimeters of the nominal setpoint. Transient disturbance peaks can cause larger excursions; these are damped and the system returns to the deadzone without oscillation or repeated actuator saturation.

E. Figures

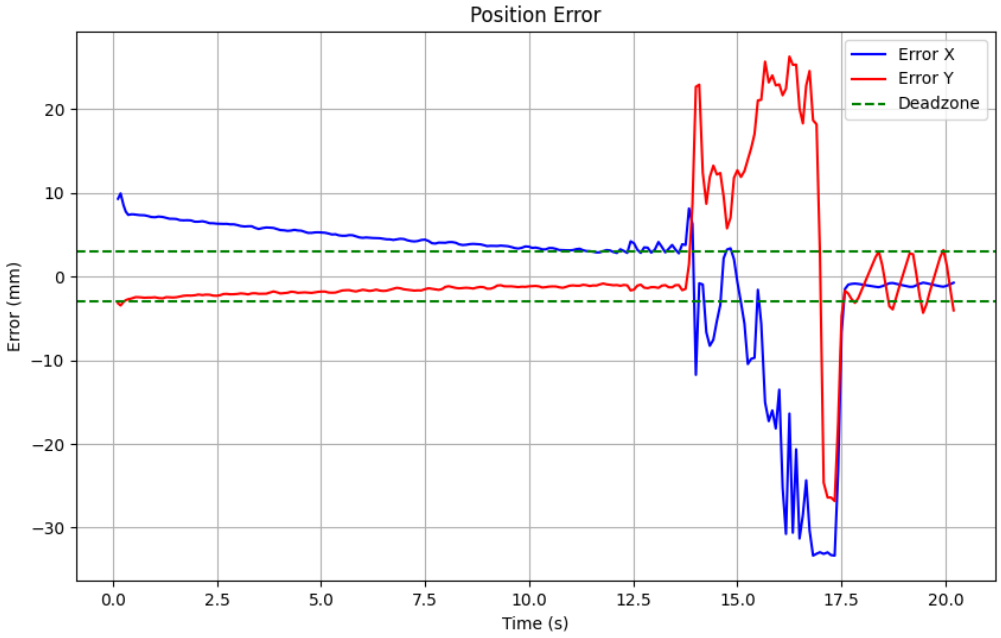

Figure 32. Ball position error in X and Y with deadzone bounds.

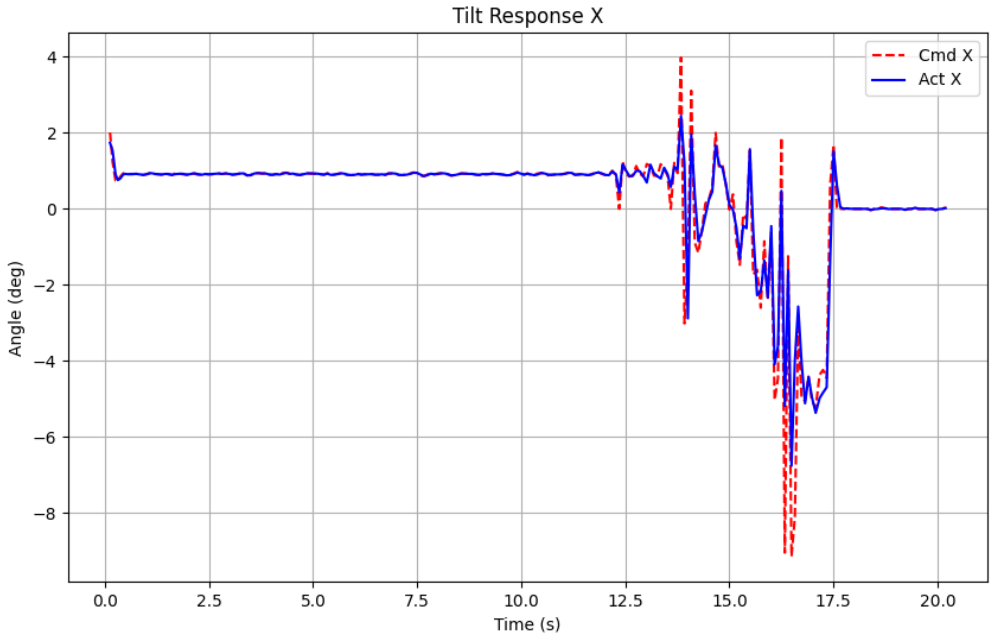

Figure 33. Servo tilt response for X axis: commanded vs actual with saturation limits.

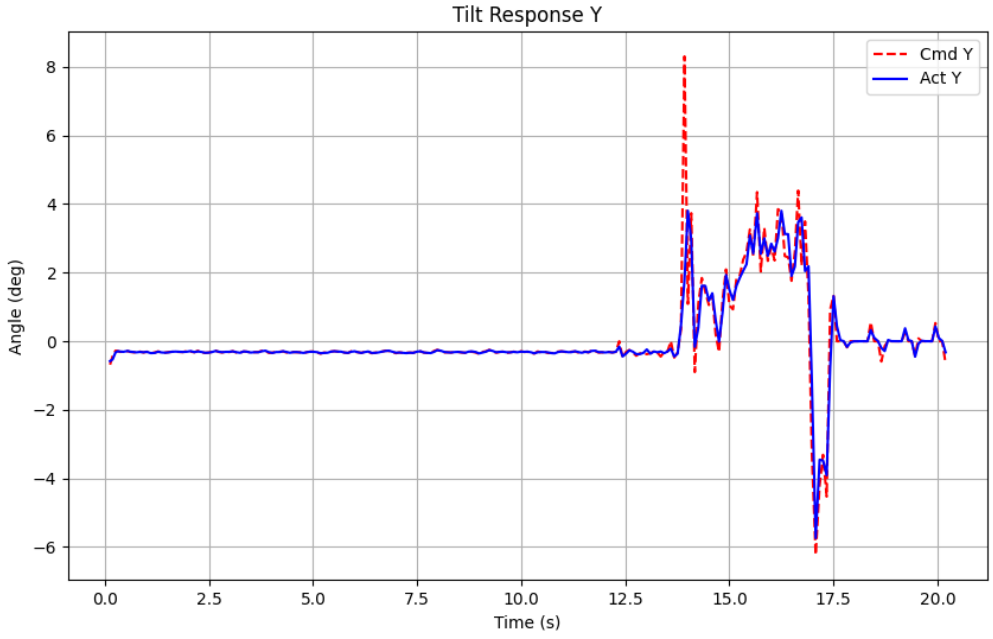

Figure 34. Servo tilt response for Y axis: commanded vs actual with saturation limits.

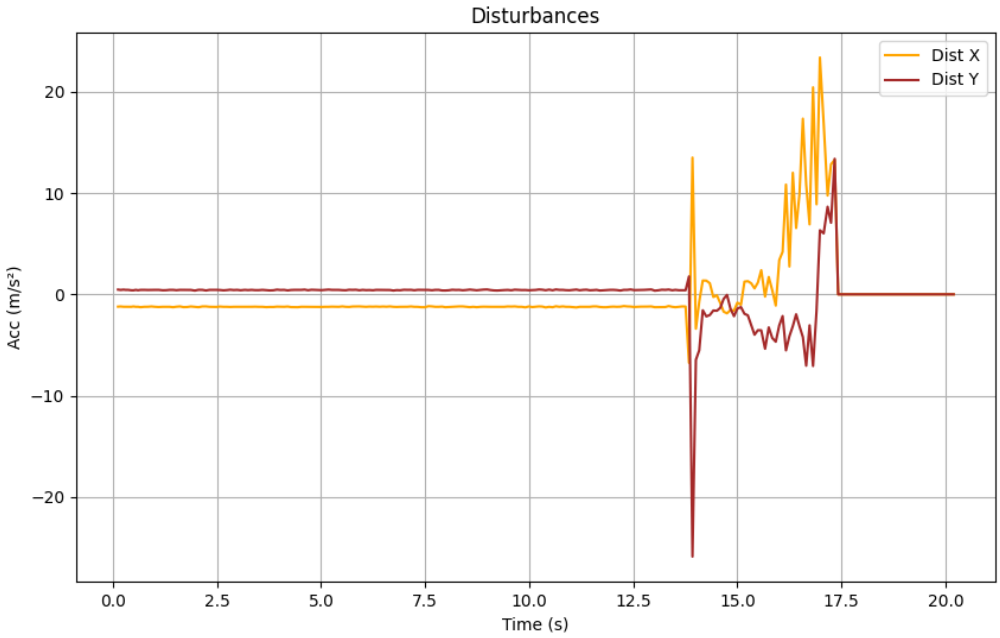

Figure 35. Injected lateral disturbance acceleration for X and Y axes.

F. Limitations and Traceability

This module is explicit about its simplifying assumptions and their traceability to the code:

- The actuator is an approximation; load-dependent nonlinearities and high-frequency motor dynamics are not modeled.
- Slip is not simulated dynamically; slip risk is assessed via analytic bounds and enforced saturation.
- All configuration parameters (gains, τ, μ, sampling rate, deadzone radius, saturation limits) are constants in the code and are logged for reproducibility.

These limitations are conservative choices made to keep the module focused, numerically robust, and reproducible.

G. System Architecture and Objectives

1. Mission Statement and Physical Constraints

The primary objective is to conduct rigorous performance comparison between contact-based (resistive touchscreen) and non-contact (computer vision) sensing modalities in a high-vibration, space-constrained environment. The platform measures 76.7 mm $\times$ 63.65 mm and balances a 10 mm diameter stainless steel ball (mass $\approx$ 4.1 g). Actuation is provided by two orthogonal MG90S micro-servos.

2. Dual Sensing Architecture

The touchscreen provides deterministic, low-latency position measurement via resistive voltage divider. The Raspberry Pi camera independently observes the ball using HSV color segmentation and blob detection. Critically, vision measurements do not feed into the control loop–this architectural separation prevents vision-induced instabilities while enabling clean comparative analysis of sensing accuracy, latency, and noise characteristics between modalities.

H. Touchscreen Sensing and Calibration

1. Calibration Procedure and Coordinate Transformation

Linear interpolation converts raw ADC counts to physical coordinates:

$$x_{mm} = \frac{(X_{raw} - X_{min})}{(X_{max} - X_{min})} \times \text{Width}$$

Calibration constants (established during pre-flight setup):

- X-axis: $X_{min} = 212$ (0 mm edge), $X_{max} = 3885$ (76.7 mm edge), yielding 47.9 counts/mm
- Y-axis: $Y_{min} = 255$ (0 mm edge), $Y_{max} = 3860$ (63.65 mm edge), yielding 56.6 counts/mm

The controller calculates centroid relative to plate center (38.35, 31.825 mm), which serves as the zero-error setpoint for PID. Non-uniform count density reflects resistive layer tolerances and is compensated during mapping.

2. Multi-Stage Filtering Pipeline

To reject electrical noise and servo-induced vibration while preserving phase margin:

Filter Design Justification: The median filter was chosen over averaging because it provides superior rejection of impulsive EMI spikes without smearing transient signals. The 3-sample window balances noise rejection against group delay—larger windows would exceed the 20–30 ms latency budget. The exponential filter's $\alpha = 0.7$ parameter was empirically tuned: higher values ($\alpha > 0.8$) allowed excessive noise into the derivative term, causing servo chatter; lower values ($\alpha < 0.6$) increased phase lag beyond the 60° stability margin, making the system oscillatory.

- 3-sample median filter: Rejects impulsive spikes from PWM electromagnetic interference. Introduces 1.5-sample group delay (15 ms at 100 Hz).
- First-order exponential filter: Attenuates high-frequency measurement noise with time constant $\tau = \frac{1-\alpha}{\alpha} \times T_s$. With $\alpha = 0.7$, $\tau = 33$ ms, providing -3 dB corner frequency at 4.8 Hz while maintaining 60° phase margin at the control crossover frequency.

$$x_f[k] = \alpha x[k] + (1 - \alpha) x_f[k - 1]$$

I. Discrete-Time PID Control Implementation

1. Control Law and Gain Selection

Operating at $T_s = 0.01$ s (100 Hz), the control output $u[k]$ (target plate angle in degrees) is computed as:

Sampling Rate Rationale: The 100 Hz rate was selected to provide $> 10\times$ oversampling relative to the expected ball motion bandwidth (∼5–8 Hz based on the 350 ms divergence time constant). This satisfies the Nyquist criterion with margin while remaining within the Teensy 4.1's computational budget—each control iteration consumes $< 1\%$ CPU, leaving headroom for telemetry and IMU processing.

$$u[k] = K_p e[k] + K_i \sum e[j] T_s + K_d \frac{e[k] - e[k-1]}{T_s}$$

PID component functions:

- Proportional: Provides loop stiffness—generates corrective angle proportional to position error. Higher K_p increases bandwidth but reduces phase margin.
- Integral: Eliminates steady-state bias caused by servo deadband and bearing friction. Accumulator clamped to ±50 deg·s to prevent windup during saturation.
- Derivative: Provides velocity-proportional damping to counteract the $5g/7$ acceleration term. Computed from filtered position to avoid noise amplification.

J. Actuation Kinematics and Constraints

1. Mechanical Transmission Ratios

The servo-to-platform linkage ratios are asymmetric due to spatial constraints:

- X-axis: 2.33:1 ratio (35° servo → 15° plate). Link geometry: 17.5 mm servo arm, 45 mm platform lever.
- Y-axis: 2.41:1 ratio (29° servo → 12° plate). Shorter lever arm compensates for narrower plate dimension.

2. Software Saturation and Safety Limits

Commanded angles are strictly clamped:

$$\theta_{limit,x} = \pm 12^\circ, \quad \theta_{limit,y} = \pm 10^\circ$$

These limits prevent mechanical binding and keep the ball within the touchscreen active area. Exceeding these angles risks servo stall (> 500 mA current draw) and potential loss of ball contact.

K. Computer Vision Pipeline Architecture

The Raspberry Pi 4 (1.5 GHz quad-core ARM) with IMX708 camera module captures 640×420 frames at 120 FPS. The pipeline prioritizes latency over resolution—smaller frames reduce processing time while maintaining adequate spatial resolution (8.3 pixels/mm).

Resolution and Frame Rate Selection: 640×420 was chosen as the minimum resolution providing reliable blob detection—the 10 mm ball occupies ∼83 pixels diameter, sufficient for accurate centroid calculation. Higher resolutions (e.g., 1920×1080) would quadruple processing time without proportional accuracy gains. The 120 FPS rate was selected to match the 100 Hz control rate with margin—lower frame rates (60 FPS) created 16.7 ms acquisition latency that degraded comparative analysis. The IMX708's rolling shutter was acceptable because the ball's maximum velocity (∼200 mm/s at 12° tilt) produces < 1.7 pixel motion blur during the 8.33 ms exposure.

1. Image Processing Stages

- Acquisition: Rolling shutter at 120 FPS provides 8.33 ms frame period. Automatic exposure locked to prevent brightness fluctuations from LED panel reflections.
- Color Segmentation: RGB → HSV conversion isolates red ball. Dual hue thresholds (0–10° and 170–180°) capture both orange-red and magenta-red spectral components. Saturation > 100 and Value > 50 reject white/gray false positives.
- Morphological Filtering: 5×5 kernel erosion (2 iterations) removes noise. Subsequent dilation (3 iterations) bridges small gaps in the ball blob caused by specular reflections.
- Contour Analysis: Blobs outside 200–3000 pixel area or circularity < 0.6 are rejected. Circularity = $4\pi(\text{Area}/\text{Perimeter}^2)$ discriminates the spherical ball from elongated reflections.
- Centroid Mapping: Moments M_{10}, M_{01} compute pixel centroid (x_{px}, y_{px}). Linear transformation maps to physical coordinates using plate dimensions.

L. Data Logging and Telemetry

1. UART Communication Protocol

A 115200 baud UART link (11.52 kB/s theoretical) transmits a 14-variable packet every 10 ms: raw/calibrated touchscreen coordinates, camera coordinates, error signals, PID components (P, I, D), commanded servo angles, and IMU-derived platform orientation (roll, pitch from MPU6050 accelerometer). Each packet is 42 bytes (ASCII CSV format), yielding 4.2 kB/s actual throughput at 100 Hz.

2. Asynchronous Logging Thread

The Raspberry Pi runs a Python thread that asynchronously writes telemetry to a 32 GB microSD card while managing the camera stream. A 500-entry circular buffer prevents data loss during the high-vibration launch phase (peak 12g, 60s duration). Additionally, raw frames are saved at 1 Hz for post-flight verification of the vision algorithm.

M. End-to-End Latency Analysis

Total system latency is critical for stability. Analysis accounts for all delay sources from sensor stimulus to physical platform response.

Latency Measurement Methodology: ADC acquisition time was measured using GPIO toggling on the Teensy 4.1 with an oscilloscope, capturing the interval from SPI chip-select assertion to data-ready flag. PID computation time was profiled using the ARM Cortex-M7 cycle counter (DWT_CYCCNT register). The median filter's 1.5-sample group delay is a theoretical value—median filters delay the output by half the window size. Servo response was characterized by commanding step angle changes and measuring platform tilt with the MPU6050 IMU at 1 kHz—the 35 ms value represents the average time from PWM edge to 90% of final position for 6° steps. OpenCV processing time was logged by timestamping frame capture and centroid output, averaged over 1000 frames during steady-state balancing.

Touchscreen Path (Control Loop):

- ADC acquisition: ∼0.1 ms (13 clock cycles at 60 MHz)
- PID computation: < 0.5 ms (floating-point arithmetic on 600 MHz Cortex-M7)
- Median filter group delay: 15.0 ms (1.5 samples at 100 Hz)
- Servo response: 35.0 ms (20 ms PWM period + 15 ms mechanical transit for 6° step)
- Total: ∼50.6 ms

Vision Path (Observation Only):

- Frame capture: 8.33 ms (120 FPS rolling shutter integration time)
- OpenCV processing: ∼12.0 ms (HSV conversion, morphology, contour analysis—profiled average)
- Servo response: 35.0 ms (hypothetical if used for control)
- Total: ∼55.3 ms

The touchscreen path achieves 4.7 ms faster response, primarily due to elimination of image processing overhead.

N. Component Descriptions

1. 4-Wire Resistive Touch Screen

A 4-wire resistive touch screen consists of two flexible conductive layers separated by tiny insulating spacer dots. When pressure is applied to the surface, the layers connect at that exact point, completing a voltage divider circuit. The XPT2046 measures the X position by applying 5V across the X and X- plates while reading the touch voltage on the Y plate, then switches to measure Y by applying voltage across Y plates. This provides the coordinates of the touch point.

2. XPT2046 Touch Controller

The XPT2046 is a 12-bit successive approximation ADC controller designed specifically for 4-wire resistive touch screens. It interfaces via a 4-wire SPI bus to the Teensy 4.1, providing precise X,Y coordinates through voltage measurement of the touch point's resistive voltage divider.

3. Teensy 4.1 Microcontroller

The Teensy 4.1, positioned on the middle PCB layer, serves as the controller for servos while receiving data from the IMU and resistive touch screen. It communicates with the Raspberry Pi via UART.

4. BNO055 IMU

Positioned on the middle PCB layer alongside the Teensy 4.1, the BNO055 serves as the payload's orientation sensor, delivering real-time detection of launch, ascent, apogee, and descent phases through integrated accelerometer, gyroscope, and magnetometer sensing. Unlike basic IMUs like the MPU9250 that output raw sensor data requiring onboard processing, the BNO055 features a built-in sensor fusion engine that automatically combines all nine axes into clean, drift-free orientation and acceleration values, eliminating the computational burden on the Teensy during flight.

5. MG90S Servos

The MG90S servos were selected for their compact size, metal gears, and balanced torque-to-weight ratio, ideal for the payload's space constraints while reliably countering ball displacement during rocket acceleration. Unlike plastic-gear servos that fail under vibration or larger high-torque models adding unnecessary weight and power constraints, the MG90S provides robust performance with low idle current draw, fitting within the power budget alongside camera and Teensy operation.

6. Arducam 12MP IMX708 Autofocus Camera Module 3

The Arducam 12MP IMX708 Autofocus Camera Module 3 uses a back-illuminated IMX708 RGB image sensor with a maximum still resolution of 4608 × 2592 (12 MP). The module supports high-resolution color video output. Video modes at 120 fps for smooth motion capture, making it suitable for real-time ball tracking and dynamic vision tasks on Raspberry Pi platforms via the CSI cable interface.

7. Raspberry Pi 4 Single Board Computer (8GB)

The Raspberry Pi 4 (8GB RAM) serves as the primary onboard processor for vision processing and high-level coordination. It interfaces with the camera via CSI and communicates with the Teensy over UART.

8. MicroSD Card 128GB (SanDisk Extreme Pro)

High-speed storage used for the operating system, logging, and image buffering. Selected for reliability under vibration and sustained write cycles.

9. CSI Camera Cable (15-pin FFC, 300mm)

Flexible flat cable connecting the IMX708 camera to the Raspberry Pi CSI interface, enabling low-latency image transmission.

O. System Power Architecture

1. Power Distribution Overview

1. Buck Converter powers Servos and LED Ring (Voltage: 6V) which passes through LM7805.
2. LM7805 regulates power to Teensy, IMU, XPT2046, Buzzers (Voltage: 5V). Filters are present to isolate noise.
3. UPS HAT power supplies Raspberry Pi 4B and camera exclusively (Voltage: 5V from dedicated 2 Li-ion cell battery). Ensures the RPi is properly powered for the required time.

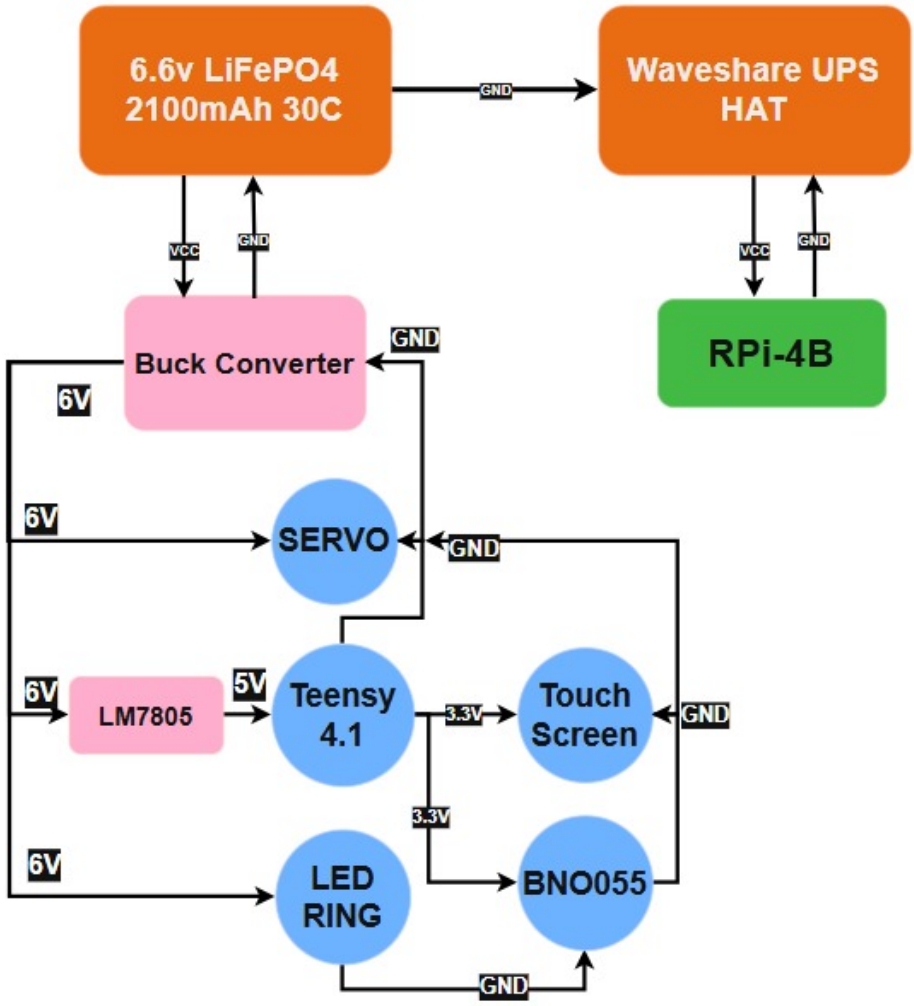

Figure 36. Power Distribution Flowchart (Fig1.1)

Battery Characteristics: 6.6V 2100mAh 30C LiFePO4 Battery. Max Current: 63A. Runtime for payload: 1.57 Hr. Rationale for LiFePO4: Stable chemistry and superior thermal runaway protection; Sufficient Maximum Current; Small Size.

Component	Voltage (V)	Idle (mA)	Avg (mA)	Peak (mA)	Avg Power (mW)
BNO055 IMU	3.3	0.08	6.04	12	19.932
MG90S Servo x2	6	20	660	1300	3960
Teensy 4.1	5	20	60	100	300
XPT2046	3.3	0.1	0.55	1	1.815
4-wire Touch	3.3	5	27.5	50	90.75
Small Buzzer	5	1	15.5	30	77.5
LED Ring 16x WS2812	6	1	480.5	960	2883
Subtotal	-	47.18	1250.09	2453	7332.997
Adjusted (80% Converter Efficiency)	-	56.616	1500.108	2943.6	8799.5964

Table 12. Power Distribution Table (T1.1)

P. Payload PCB Stackup

Layer 1 takes power from the battery, distributes power to buck converter, LED, and servos. LM7805 regulates voltage to 5V and sends it to the layer below. Connects to the middle layer via a 40-pin header array.

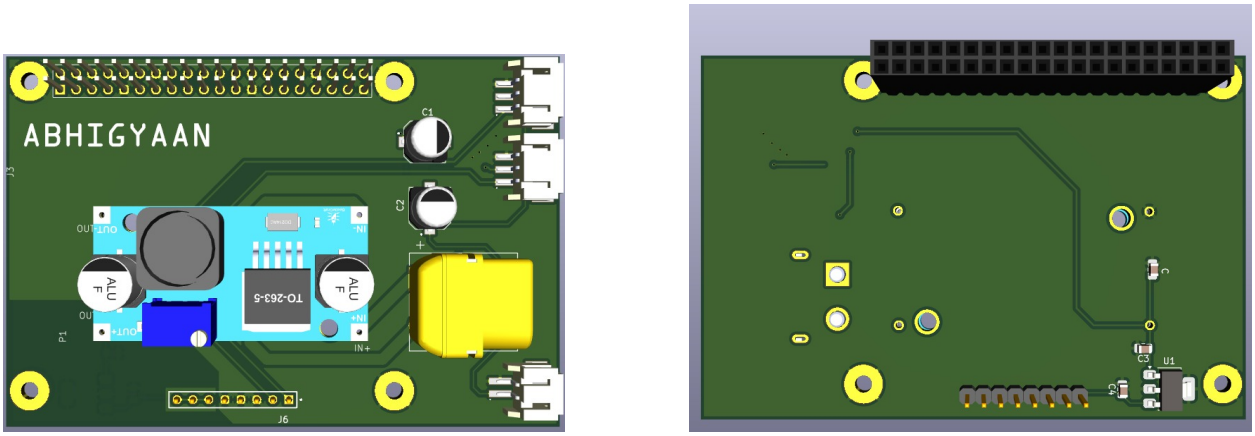

Figure 37. Layer 1

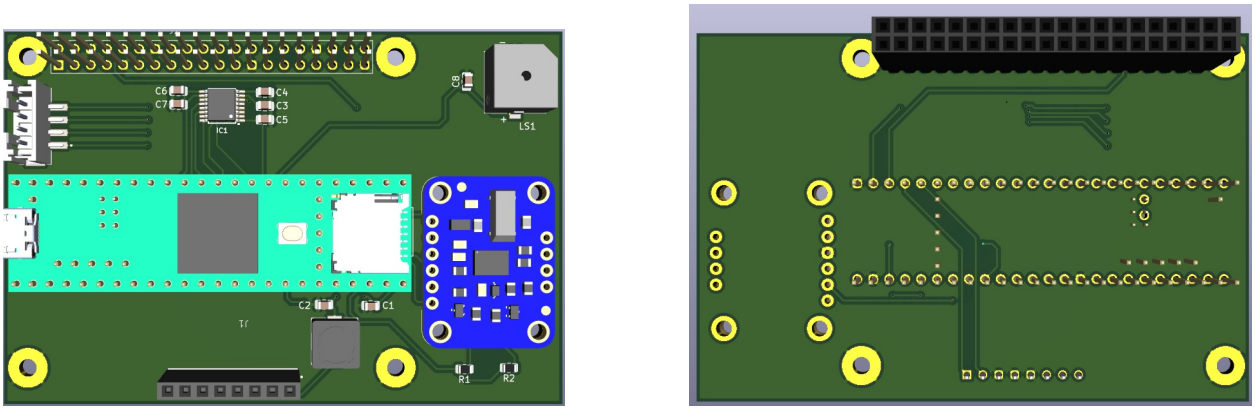

Figure 38. Layer 2

Layer 2 consists of Teensy 4.1, XPT2046, 4-Wire Resistive Touch Screen, BNO055 IMU. This layer takes regulated 5V from the upper PCB and powers its components. Adequate filters are placed to ensure noiseless supply to components.

Layer 3: The RPi 4 acts as the base layer of the stack; other layers are designed to be in-line with its hole and pin positions. The RPi and Teensy communicate with each other; the RPi gives coordinates from the camera and the Teensy gives coordinates from the touch screen. RPi is separately powered by a UPS HAT.

- **Grounding and Noise Management:** The PCB layers have two large ground planes with stitching vias to reduce impedance and improve thermal dissipation. LC filter near Teensy Vin supply to clear off noise from the switching of buck converter. Capacitors near noise-sensitive pins to filter off any high-frequency

Q. Mechanical Design Report

The payload consists of a two-axis ball balancing mechanism integrated inside a 3U CubeSat structure. The mechanism is mounted above the electronics bay, which contains the Raspberry Pi, Teensy, UPS hat, and supporting PCB. From a mechanical point of view, the electronics bay mainly acts as the structural base for mounting the payload and transferring loads to the main structure. The outer structure of the CubeSat is made from Aluminium 6061, which serves as the primary load-bearing frame.

Aluminium 6061 was selected because it provides sufficient strength while keeping the mass low. It also maintains structural rigidity during launch and ensures proper alignment of internal components. All internal parts, including the electronics bay and payload, are mounted to this structure.

The electronics bay and the balancing platform are made from ABS plastic. ABS was selected mainly to reduce weight and allow easy manufacturing while still providing enough strength for this application. The electronics bay supports the onboard components and also serves as the mounting base for the servo motors and ball balancing mechanism. The platform mass is approximately 60 grams.

The balancing mechanism consists of a rectangular platform supported at its center using a ball joint. This ball joint carries the weight of the platform and allows it to rotate freely about two perpendicular axes. Because of this, the servo motors are not required to support the platform weight directly and only provide the torque required to tilt the platform.

Two MG90S servo motors are used for actuation. Each servo is connected to the platform using a servo arm of length 17.5 mm and a pulling rod of length 45 mm. Each linkage assembly weighs approximately 20 grams. When the servo rotates, the linkage pushes or pulls the platform, causing it to tilt.

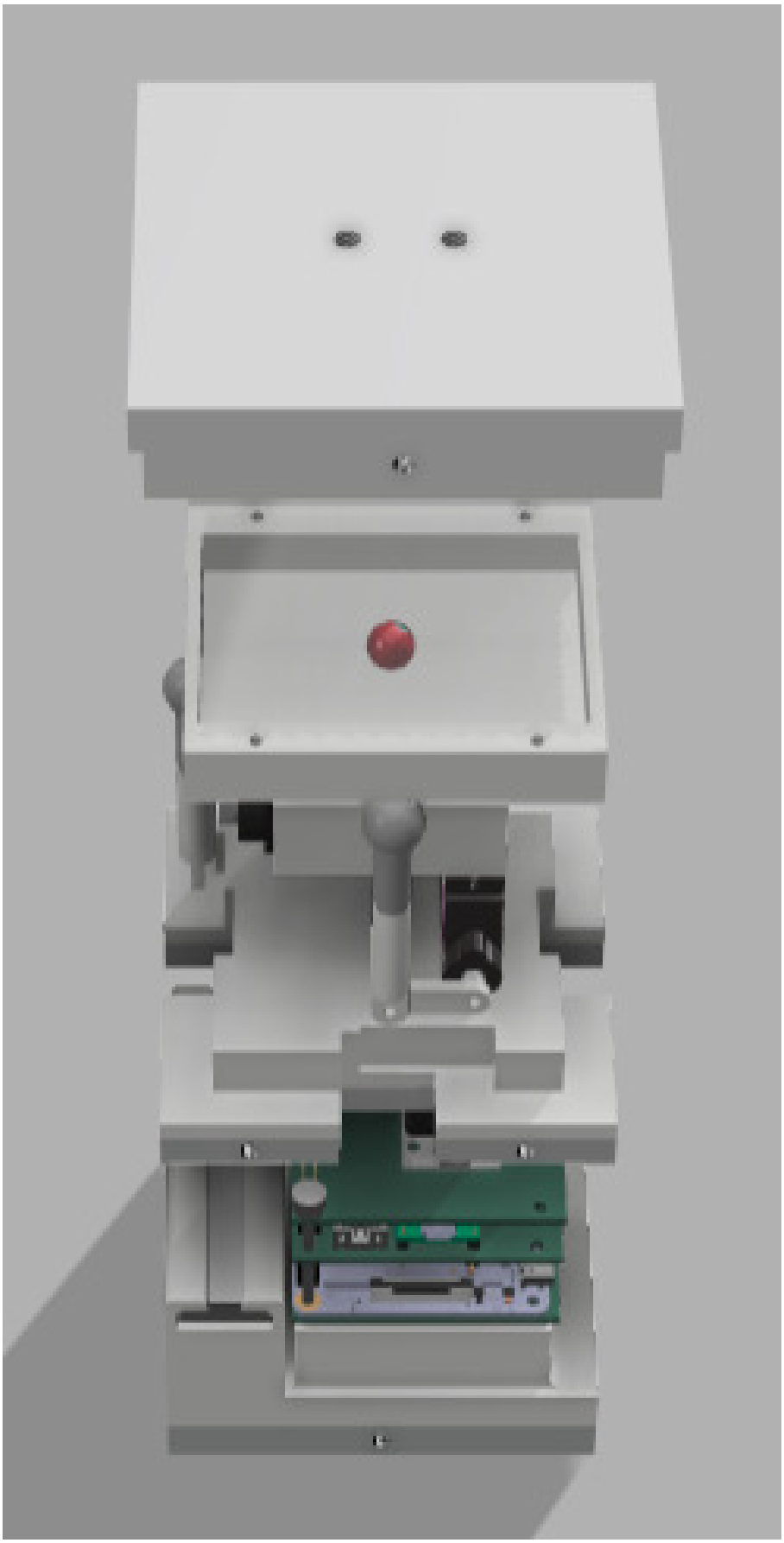

Figure 39. CAD

The overall platform dimensions are approximately 83 mm × 74 mm, with an active ball movement area of 70 mm × 54 mm. A steel ball of diameter 10 mm is used. Based on its size and material, its mass is approximately 4 grams.

An acrylic enclosure is mounted above the platform to prevent the ball from escaping during launch vibrations or operation. Acrylic was selected because it is lightweight and rigid and provides sufficient containment for the ball.

The payload is designed to withstand launch loads of up to 10 g. The aluminium structure carries the main load, while the electronics bay and mounting components transfer loads between the payload and the structure. All components are fastened securely to prevent movement during launch.

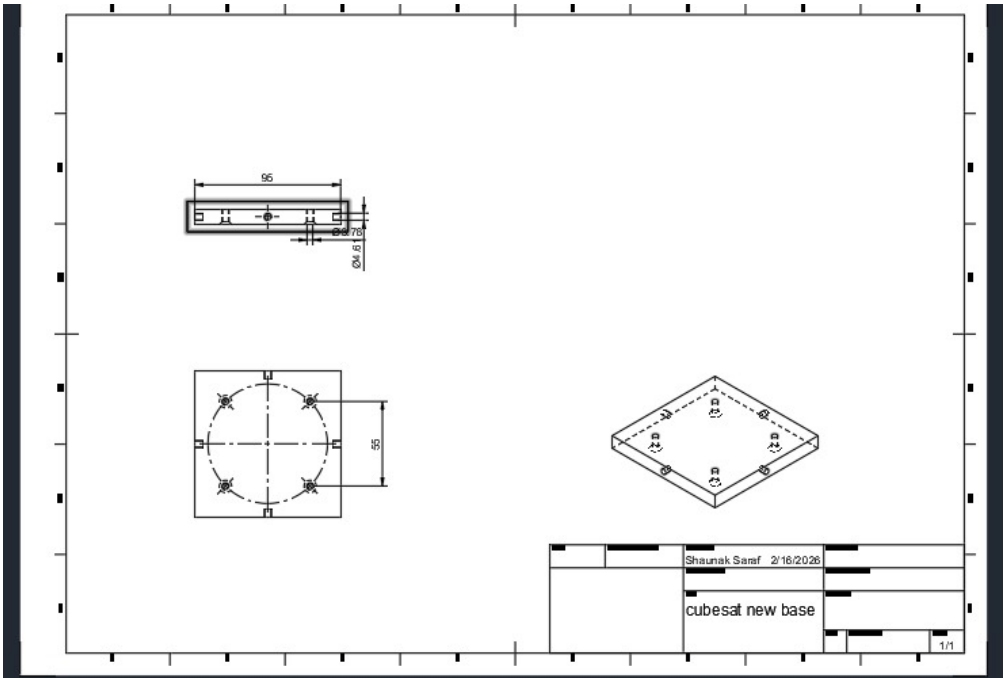

Figure 40. Engineering Drawing

mech_one.png

mech_two.png

Figure 41. Assembly

Threaded connections are used to mount the payload to the aluminium plate using two M5 threaded rods with a thread engagement length of 20 mm. The supported mass is approximately 3.5 kg. Under 10 g launch acceleration, the resulting force is approximately 343 N. Since this load is shared between two threaded rods and the engagement length is sufficient, the threads can safely withstand the applied load without risk of failure.

The torque required from each servo is due to the ball on the tilted platform and the linkage connected to the servo. The mass of the steel ball is 0.0041 kg and the maximum distance of the ball from the platform center is taken as 0.032 m. The torque acting on the platform due to the ball is

$$\tau_{\text{platform}} = m_b g r_b \sin\theta$$

$$\tau_{\text{platform}} = (0.0041)(9.81)(0.032)\sin\theta$$

$$\tau_{\text{platform}} = 0.00129\sin\theta\,\text{Nm}$$

This torque is transmitted to the servo through the linkage. The servo arm length is 0.0175 m and the pulling rod length is 0.045 m. Therefore,

$$\tau_{\text{servo,ball}} = \frac{0.0175}{0.045} \times 0.00129\sin\theta$$

$$\tau_{\text{servo,ball}} = 0.00050\sin\theta\,\text{Nm}$$

Each servo arm and pulling rod has a mass of 0.02 kg, and its center of mass is located at half the servo arm length, which is 0.00875 m. The torque due to the linkage is

$$\tau_{\text{linkage}} = m_l g r_l \sin\phi$$

$$\tau_{\text{linkage}} = (0.02)(9.81)(0.00875)\sin\phi$$

$$\tau_{\text{linkage}} = 0.00172\sin\phi\,\text{Nm}$$

The total torque required from each servo is

$$\tau_{\text{servo}} = 0.00050\sin\theta + 0.00172\sin\phi$$

Under launch acceleration of 10 g, the torque increases by a factor of 10,

$$\tau_{\text{servo}} = 0.0050\sin\theta + 0.0172\sin\phi$$

The maximum torque occurs when the sine terms are equal to 1,

$$\tau_{\text{servo,max}} = 0.0050 + 0.0172$$

$$\tau_{\text{servo,max}} = 0.0222\,\text{Nm}$$

This is the maximum torque required from each servo under launch loading conditions.

- **FOS for servos:** From the torque calculations under 10 g launch load, the maximum torque required from one servo was found to be:

  $$\tau_{\text{required}} = 0.0222\ \text{Nm}$$

  The rated torque of the MG90S servo is:

  $$\tau_{\text{servo}} = 0.215\ \text{Nm}$$

  So the factor of safety is:

  $$\text{FoS} = \frac{0.215}{0.0222} = 9.68$$

VI. Management

A. Rocket Logo – ARJUNA

The rocket is named **ARJUNA**, written in a gold gradient font symbolizing excellence and achievement. The letter "A" incorporates the structure of a bow, subtly integrating the identity of the legendary archer. Arjuna was known for his unmatched focus and ability to hit a target with absolute precision.

In rocketry, this translates directly to trajectory control, aerodynamic stability, and guidance accuracy. A rocket must maintain focus on its intended flight path and mission objectives, just as Arjuna focused solely on the eye of the bird. Naming the rocket Arjuna signifies precision, determination, and the ability to execute a mission without deviation.

B. Mission Patch

The circular mission patch for ARJUNA visually narrates the launch journey. It depicts a rocket ascending upward toward a radiant sun, with mountains and forests below representing Earth. Birds flying across the sky symbolize freedom, ambition, and breaking natural boundaries.

The bow-shaped silhouette integrated into the rocket's form connects the design to Arjuna's identity as an archer. The rising motion signifies technological advancement and progress, while the sun represents energy, vision, and aspiration. The circular border reflects mission unity and completeness, and the black outer ring conveys strength and professionalism.

Overall, the patch illustrates Arjuna's ascent from Earth toward higher goals, symbolizing innovation and forward momentum.

C. Payload Patch

The payload is named **GANDIVA**, after the divine bow of Arjuna. In mythology, Gandiva was the weapon that enabled Arjuna to achieve victory. In aerospace symbolism, the rocket (Arjuna) is the vehicle, while the payload (Gandiva) is the true functional objective of the mission.

Just as a bow gives purpose to an archer, the payload gives purpose to the rocket. The design shows a warrior drawing a golden bow against a radiant sun, symbolizing concentrated energy and controlled power release. The radiating light represents data transmission, impact, and the functional output of the mission.

This naming establishes a logical hierarchy: Drona represents knowledge and guidance, Arjuna represents execution and precision, and Gandiva represents mission impact and objective fulfillment.

D. Team T-Shirt

Figure 42. Front and Back Design of Team DRONA T-Shirt

Figure 43. Front and Back Design of Team DRONA T-Shirt

Table 13. Subsystem Cost Breakdown

Subsystem	Sum of Cost
Avionics	156198.11
Payload	30565.00
Mechanical	243836.22
Management	14523.20
TOTAL	**445122.53**

Table 14. Shipping Cost Breakdown

Subsystem	Cost
Avionics	4282.66
Mechanical	21430.24
Payload	223.39
Management	–
TOTAL	**25936.29**

TOTAL MONEY SPENT : 481908.82

VII. Appendix

A. Appendix: Simulations

A. Aerocover

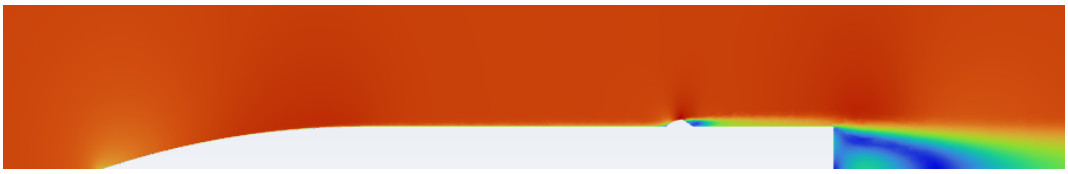

Figure 44. Camera cover at Mach 0.5

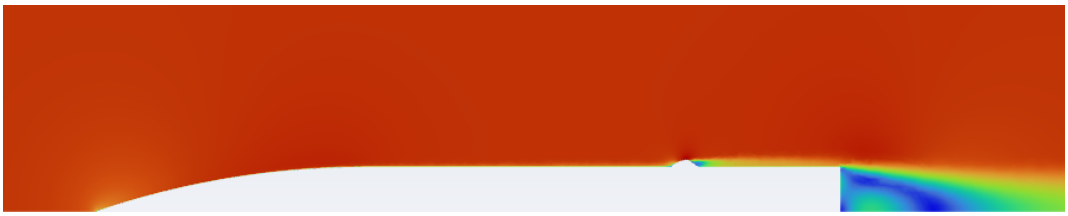

Figure 45. Camera cover at Mach 0.6

B. Nosecone

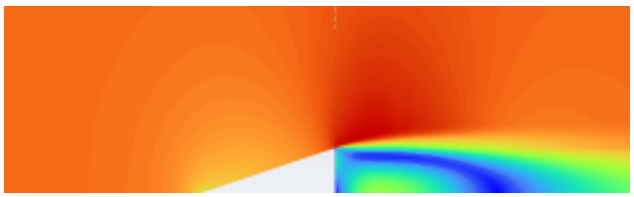

Figure 46. Conical nosecone at fineness ratio of 3.0 at Mach 0.85

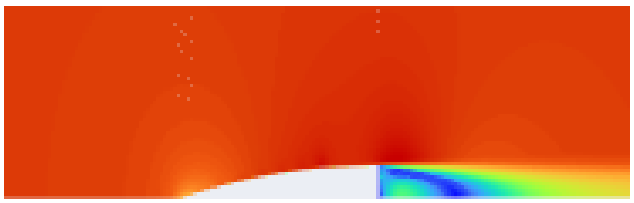

Figure 47. Parabolic nosecone at fineness ratio of 3.0 at Mach 0.85

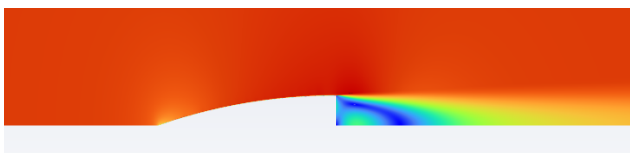

Figure 48. Ogival nosecone at fineness ratio of 3.0 at Mach 0.85

Table of Design Points

	A	B	C	D	E	F	G
1	Name	P1 - rho	P2 - fuselage	P3 - report-def-0-op	Ret...	Retained Data	Note
2	Units	m	m				
3	DP 0 (Current)	1.3875	0.21818	0.31633	☑	✓	
4	DP 1	1.2135	0.224	0.32318	☐		
5	DP 2	1.5735	0.2127	0.31371	☐		
6	DP 3	1.3875	1E-05	0.37249	☐		
7	DP 4	1.299	0.2209	0.32592	☐		
8	DP 5	1.7715	0.2072	0.33808	☐		
*					☐		

Figure 49. Parametric optimisation for fineness ratio of Tangent Ogive Nosecone at Mach 0.85

C. Airbrakes

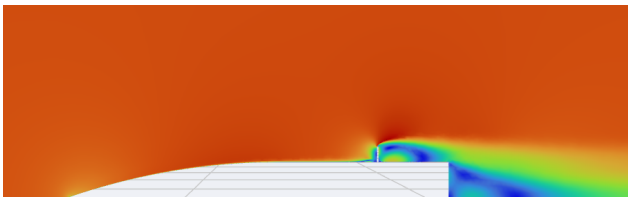

Figure 50. Airbrakes at 50% extension, Mach 0.3

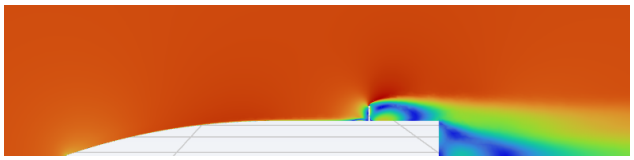

Figure 51. Airbrakes at 50% extension, Mach 0.4

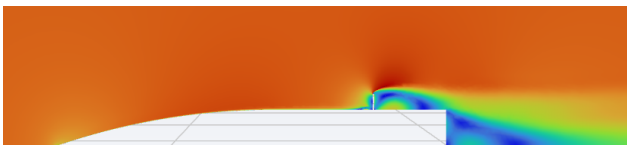

Figure 52. Airbrakes at 50% extension, Mach 0.5

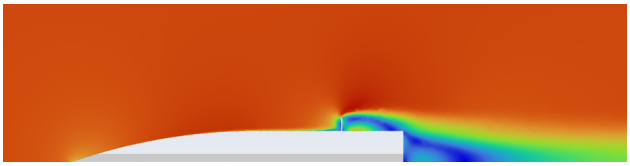

Figure 53. Airbrakes at 50% extension, Mach 0.6

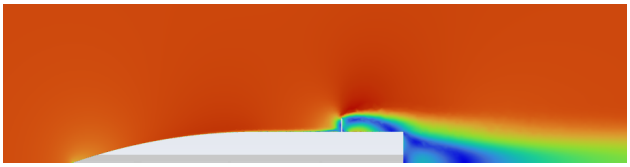

Figure 54. Airbrakes at 50% extension, Mach 0.7

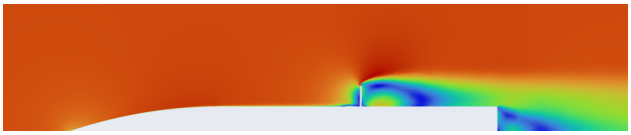

Figure 55. Airbrakes at 100% extension, Mach 0.3

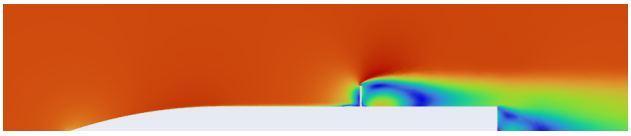

Figure 56. Airbrakes at 100% extension, Mach 0.4

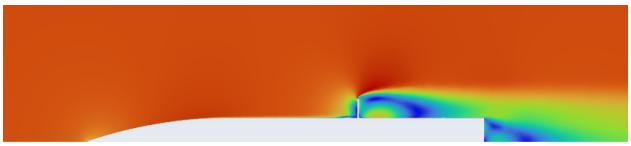

Figure 57. Airbrakes at 100% extension, Mach 0.5

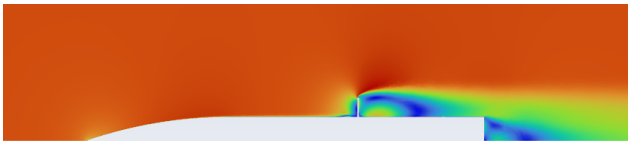

Figure 58. Airbrakes at 100% extension, Mach 0.6

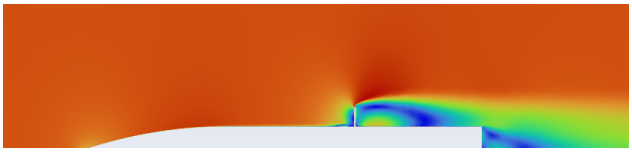

Figure 59. Airbrakes at 100% extension, Mach 0.7

D. Stagnation Port

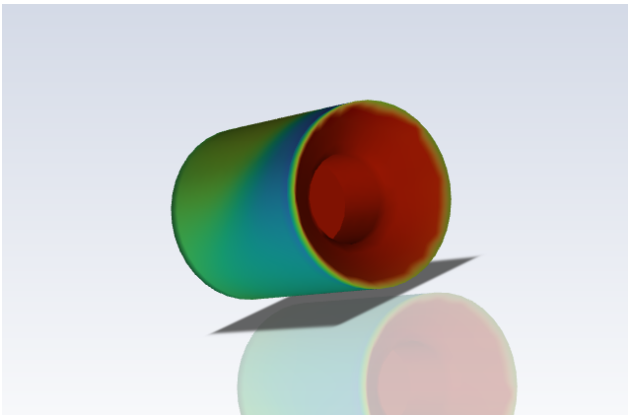

Figure 60. Stagnation Port at 15 AOA, Mach 0.1

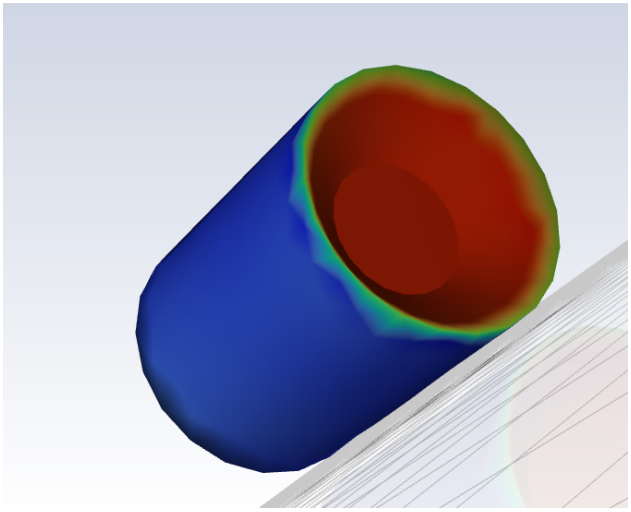

Figure 61. Stagnation Port at 15 AOA, Mach 0.85

B. Appendix: Transient flow in Stagnation Tube

```
import csv
import numpy as np
import matplotlib.pyplot as plt

from scipy.sparse import diags, coo_matrix
from scipy.sparse.linalg import spsolve

t_flight = []
T_outer = []
P_outer = []
rho_outer = []
M_outer = []

# Import the data from csv file obtained from ORK
with open("atm.csv", newline="") as f:
    f = ["t,T,P,rho,M"] + [line for line in f if '#' not in line]
    reader = csv.DictReader(f)

    for row in reader:
        t_flight.append(float(row['t']))
        T_outer.append(float(row['T']))
        P_outer.append(float(row['P']) * 100)
        rho_outer.append(float(row['rho']))
        M_outer.append(float(row['M']))

# Dimensions and properties of the stagnation tube. Everything is in SI units.
L = 1
D = 2e-2

# Because of the bending of the tube in 3 positions, there will be some head loss. To
    account for that,
# we are assuming a 25% loss of dynamic pressure at each bend.
bend_positions = [0.25, 0.50, 0.75]
K_bend = 0.75

# Properties of atmosphere.
gamma = 1.4
R_gas = 287
T_ref = 300
mu = 1.81e-5
P_base = P_outer[0]
c_sound = np.sqrt(gamma * R_gas * T_ref)
rho_ref = P_base / (R_gas * T_ref)

# Discretization parameters.
N = 500
dt = 1e-3
t_final = 60
dx = L / (N - 1)
bend_indices = [int(loc/dx) for loc in bend_positions]
steps = int(t_final / dt)

# Initialising the state variables spatially.
P = np.full(N, P_base)
u = np.zeros(N)

# History for plotting errors.
time_hist = []
P_wall_hist = []
P_inlet_hist = []

# Time marching loop for the implicit solution.
for step in range(steps):
    t = step * dt

    print(f"Calculating time step: {t:.3f}s")

```

```
    # We use u from previous step to calculate friction.
    # A small value is added to avoid division by 0. Using np.maximum will take too much time if computation is large.
    u_abs = np.abs(u) + 1e-6

    # Friction factor obtained as a variation of Reynolds number in the flow.
    # The flow will be pretty much laminar or moderately turbulent so we can use Blasius correction for the friction factor
    # ignoring the effects of roughness of pipe.
    #
    # Source: https://en.wikipedia.org/wiki/Darcy_friction_factor_formulae#Blasius_correlations
    Re = (rho_ref * u_abs * D) / mu
    f = np.where(Re < 2300, 64/Re, 0.316 * Re**(-0.25))

    # Coefficient of resistance (R_coeff) is the coefficient of the velocity in the momentum equation which accounts for the frictional losses due to the wall and bends.
    # Momentum eqn: du/dt + ... + R_coeff * u = 0
    R_coeff = (f * u_abs) / (2 * D)

    # Adding the losses of bends at those indices.
    for idx in bend_indices:
        # Spreading the losses of bend over the entire cell.
        # Should it be spread over multiple cells around the region? This is a first order approximation for the time being.
        R_coeff[idx] += (K_bend * u_abs[idx]) / (2 * dx)

    # An implicit scheme was made use of to make the solution unconditionally stable even with large time steps.
    # We implicitly solve for P and u using a linear system Ax = B.
    #
    # Here, x is a 2N x 2N vector of unknowns ordered as [P0, u0, P1, u1, ..., PN, uN].
    # This could have also been done using two different spatial vectors for P and u but making a coupled system with that was a headache.
    #
    # Discretisation of the equations as follows:
    #
    # 1. Continuity Equation:
    # (P_i_new-P_i_old)/dt + rho*c^2*(u_i-u_i-1)/dx = 0  (Backward Difference u)
    # => P_i_new + (dt*rho*c^2/dx) * u_i_new - (dt*rho*c^2/dx) * u_i-1_new = P_i_old
    # => P_i_new + alpha * u_i_new - alpha * u_i-1_new = P_i_old
    # => P_i_new + alpha*(u_i_new-u_i-1_new) = P_i_old
    #
    # 2. Momentum Equation:
    # (u_i_new-u_i_old)/dt + 1/rho * (P_i+1-P_i)/dx + R_coeff*u_i_new = 0 (Forward Difference P)
    # => (1 + dt*R_coeff)*u_i_new + (dt/(rho*dx))*P_i+1_new - (dt/(rho*dx))*P_i_new = u_i_old
    # => (1 + dt*R_coeff)*u_i_new + beta * P_i+1_new - beta * P_i_new = u_i_old
    # => (1 + dt*R)*u_i_new + beta*(P_i+1_new - P_i_new) = u_i_old
    #
    # The above scheme is implicit because we are solving P_i_new from u_i_new simultaneously both being in LHS.
    #
    # Gradient schemes chosen for P and u are forward and backward respectively to get rid of checkerboard instability
    # by making sure the ith cell is dependent of both (i-1)th and (i+1)th cell.
    #
    # The equations used here are for 1D transient pipe flow.
    # Source: https://docs.aft.com/impulse10/TransientFlowEquations.html

    # Coefficients used in the above equations.
    alpha = (dt * rho_ref * c_sound**2) / dx
    beta  = dt / (rho_ref * dx)

    # The diagonals of the sparse matrix A are built using coo_matrix of scipy. The other way was just messy.
    data = []
    rows = []
    cols = []
```

```
125
126     # The right hand size vector B.
127     B = np.zeros(2*N)
128
129     for i in range(N):
130         # Indices of the P and u of the ith node in A.
131         idx_P = 2*i
132         idx_u = 2*i + 1
133
134         # Continuity equation solver.
135         if i == 0:
136             # Boundary condition: Pressure inlet which is the stagnation pressure obtained
    from flight data.
137             # Boundary Condition: P0 is Fixed (Inlet)
138             # Equation: 1 * P0 = P_inlet
139             rows.append(idx_P); cols.append(idx_P); data.append(1)
140
141             Patm = np.interp(t + dt, t_flight, P_outer)
142             Matm = np.interp(t + dt, t_flight, M_outer)
143
144             # Stagnation pressure obtained from isentropic relations.
145             B[idx_P] = Patm * (1 + 0.2 * Matm**2)**3.4
146         else:
147             # Equation: P_i_new + alpha*(u_i_new-u_i-1_new) = P_i_old
148             rows.append(idx_P); cols.append(idx_P);   data.append(1)      # P_i_new
149             rows.append(idx_P); cols.append(idx_u);   data.append(alpha)    # alpha *
    u_i_new
150             rows.append(idx_P); cols.append(idx_u-2); data.append(-alpha)   # alpha * u_i-1
    _new
151             B[idx_P] = P[i]
152
153         # Momentum equation solver.
154         if i == N-1:
155             # Boundary Condition: Wall where the velocity is 0
156             # Equation: 1 * uN = 0
157             rows.append(idx_u); cols.append(idx_u); data.append(1)
158             B[idx_u] = 0
159         else:
160             # Equation: (1 + dt*R)*u_i_new + beta*(P_i+1_new-P_i_new) = u_i_old
161             damping = 1 + dt * R_coeff[i]
162             rows.append(idx_u); cols.append(idx_u);   data.append(damping)  # (1 + dt*R) *
    u_i_new
163             rows.append(idx_u); cols.append(idx_P+2); data.append(beta)     # beta * P_i+1
    _new
164             rows.append(idx_u); cols.append(idx_P);   data.append(-beta)    # beta * P_i_new
165             B[idx_u] = u[i]
166
167     # A compressed sparse row matrix is made to create the matrix A from coo_matrix to
    reduce memory usage.
168     A = coo_matrix((data, (rows, cols)), shape=(2*N, 2*N)).tocsr()
169
170     # A sparse solver is used for this linear system.
171     X = spsolve(A, B)
172     P = X[0::2]
173     u = X[1::2]
174
175     time_hist.append(t)
176     P_inlet_hist.append(P[0])
177     P_wall_hist.append(P[-1])
178
179 time_hist = np.array(time_hist)
180 P_in = np.array(P_inlet_hist) / 1000
181 P_wall = np.array(P_wall_hist) / 1000
182 error = (np.array(P_wall_hist) - np.array(P_inlet_hist)) / np.array(P_inlet_hist) * 100
183
184 plt.figure(figsize=(10, 8))
185
186 plt.subplot(2, 1, 1)
187 plt.plot(time_hist, P_in, 'k-', label='Inlet Pressure')
188 plt.plot(time_hist, P_wall, 'r-', linewidth=1.5, label='Transducer Pressure')
```

```
189
190 plt.title('Stagnation Tube Pressure')
191 plt.ylabel('Pressure (kPa)')
192 plt.legend()
193 plt.grid(True)
194
195 plt.subplot(2, 1, 2)
196 plt.plot(time_hist, error, 'b-')
197 plt.title('Stagnation Error')
198 plt.ylabel('Error (Pa)')
199 plt.xlabel('Time (s)')
200 plt.grid(True)
201
202 plt.tight_layout()
203 plt.savefig('pitot_stagnation_lag.png', dpi=300)
204
205 print(f"Max Error: {np.max(np.abs(error)):.2f} Pa")
```

Listing 1. CFD Code for Transient flow in Stagnation Tube

References

[1]Raspberry Pi Ltd., *RP2350 Microcontroller Datasheet (RP2354B)*, Document No. RP-008373-DS-2. Available: https://pip-assets.raspberrypi.com/categories/1214-rp2350/documents/RP-008373-DS-2-rp2350-datasheet.pdf

[2]STMicroelectronics, *LSM9DS1 iNEMO Inertial Module Datasheet*. Available: https://www.st.com/resource/en/datasheet/lsm9ds1.pdf

[3]AKER Technology USA, *S2 Series Oscillator General Specification (S23305T)*. Available: https://aker-usa.com/files/content/products/oscillators/S2-General-Specification.pdf

[4]Reyax Technology Co., Ltd., *RYLR993 LoRa Transceiver Module Datasheet*. Available: https://reyax.com/upload/products_download/download_file/RYLR993_EN.pdf

[5]u-blox AG, *NEO-M9N GNSS Module Data Sheet*, Document No. UBX-19014285. Available: https://content.u-blox.com/sites/default/files/NEO-M9N-00B_DataSheet_UBX-19014285.pdf

[6]Infineon Technologies AG, *DPS368 Barometric Pressure Sensor Datasheet*. Available: https://www.infineon.com/assets/row/public/documents/24/49/infineon-dps368-datasheet-en.pdf

[7]Advanced Monolithic Systems, *AMS1117 Low Dropout Voltage Regulator Datasheet*. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/5011/AMS1117.pdf

[8]NXP Semiconductors, *MPX5100 Differential Pressure Sensor Datasheet*. Available: https://www.nxp.com/docs/en/data-sheet/MPX5100.pdf

[9]Alpha & Omega Semiconductor, *AO3400A N-Channel MOSFET Datasheet*. Available: https://www.aosmd.com/res/datasheets/AO3400A.pdf

[10]Espressif Systems, *ESP32-S3-WROOM-1 and ESP32-S3-WROOM-1U Datasheet*, Version 1.1, Shanghai, China, 2022. Available: https://documentation.espressif.com/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[11]Reyax Technology Corp., *RYLR993 AT Command Guide*. Available: https://reyax.com/upload/products_download/download_file/RYLR993_AT_Command.pdf

[12]Amazon (Generic Manufacturer), *Micro SD Card Module Datasheet/Schematic*. Available: https://images-na.ssl-images-amazon.com/images/I/91tTtUMDM3L.pdf

[13]LDZY Electronics Co., Ltd., "Common JST Connector Types and Their Applications." Available: https://ldzy.tw/common-jst-connector-types-their-applications/

[14]Microchip Technology Inc., *MIC5209: 500mA Low-Noise LDO Regulator Datasheet*. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/20005720A.pdf

[15]OpenRocket Technical Documentation https://openrocket.sourceforge.net/techdoc.pdf

[16]Rocket Body Design Considerations by Richard Nakka https://nakka-rocketry.net/RD_body.html

[17]Fluid Dynamic Drag by Hoerner https://ia600606.us.archive.org/17/items/FluidDynamicDragHoerner1965/Fluid-dynamic_drag__Hoerner__1965_text.pdf

[18]TESTS OF THE NACA 0010-1. 50 40/1.051 AIRFOIL SECTION AT HIGH SUBSONIC MACH NUMBERS https://ntrs.nasa.gov/api/citations/19930087593/downloads/19930087593.pdf

[19]Black Powder Estimation by Rocketry Calculator https://rocketrycalculator.com/rocketry-calculator/bp-estimator/

[20]Imperial Rocketry: Shock loading https://wiki.imperialrocketry.com/en/Airframe__Recovery/Estimating_loads_on_shock_cords/Estimating_loads_on_shock_cords

[21]Measurement of Aircraft Speed and Altitude https://ntrs.nasa.gov/api/citations/19800015804/downloads/19800015804.pdf

[22]Transient Pipe Flow Equations https://docs.aft.com/impulse10/TransientFlowEquations.html

[23]PredictWind Manipal https://www.predictwind.com/weather/india/karnataka/manipal/february

[24]Discrete Gust Model: 1-cosine https://in.mathworks.com/help/aeroblks/discretewindgustmodel.html