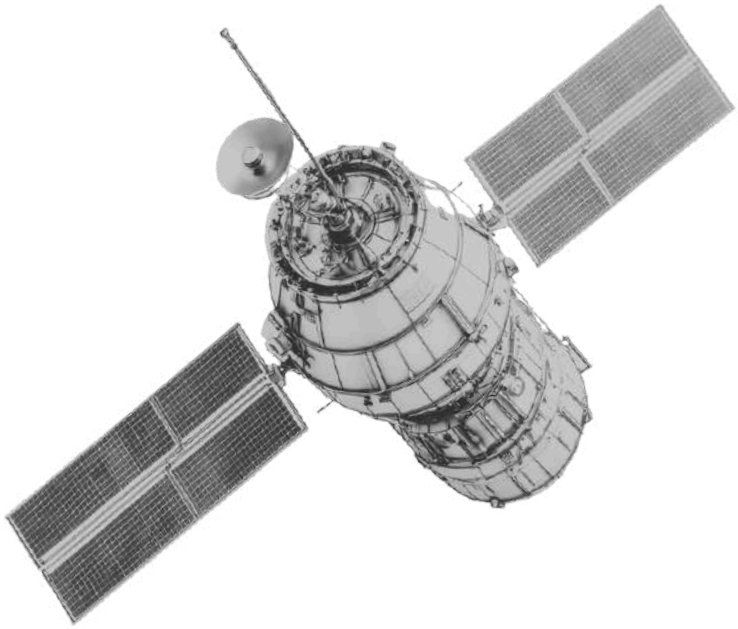
Nikhil Patel

Student Engineer & Designer

Project Portfolio

Author: Nikhil Patel
2023-2026



LinkedIn
My Website
GitHub

Team Project

University of Saskatchewan Space Design Team

Project UP

Project Overview:

During my first year on the University of Saskatchewan Space Design Team (USST), I worked on the mechanical design of the aft-end propulsion interface of our high power rocket. This included the boat tail, thrust plate, and external motor retention system (threaded crown). The aft-end is a high-leverage subsystem: it is where thrust enters the airframe, fins are structurally secured, and where the rocket's base drag is largely defined. Small design changes in this region can produce disproportionately large effects on performance, integration, and structural behavior.

As my first university-level aerospace project, I approached the work with an emphasis on clear requirements, subsystem integration, manufacturability, and iterative design, rather than isolated component optimization.

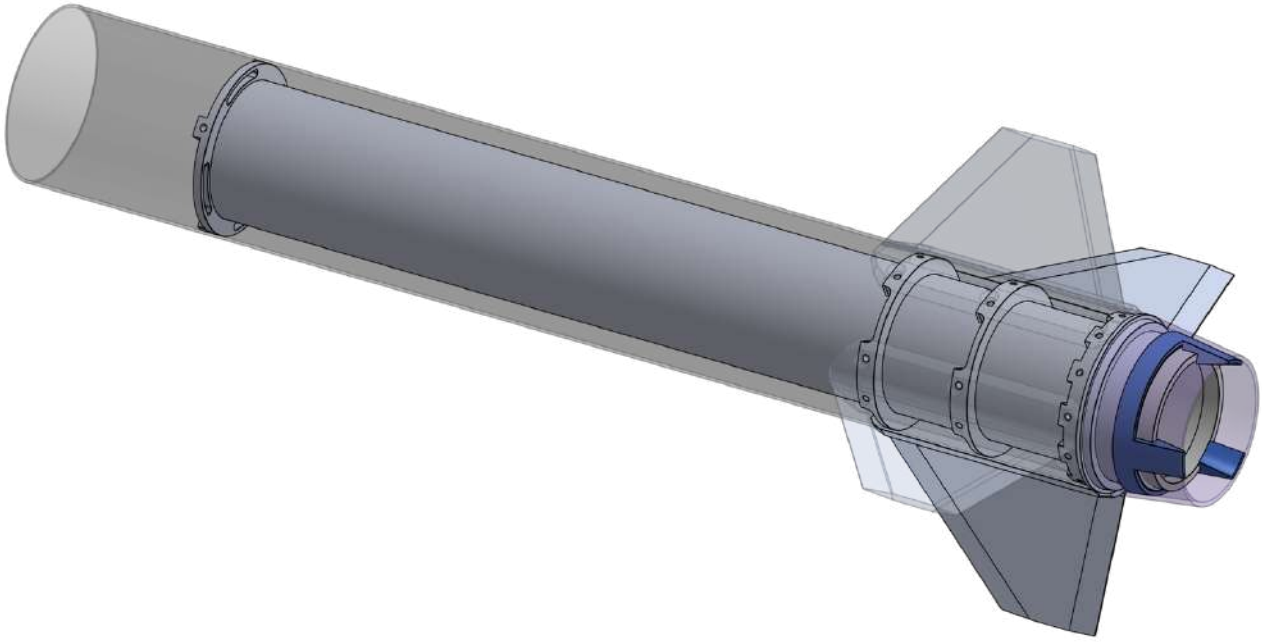


Figure 1A: Aft-end propulsion interface

The Engineering Problem:

The propulsion interface needed to satisfy multiple competing objectives:

- Reduce aerodynamic base drag at the aft end
- Provide a structurally sound thrust load path into the fuselage
- Enable safe, repeatable motor installation and servicing
- Preserve rocket stability through controlled motor positioning

Because this region directly affects both aerodynamics and structures, it was treated as a system-level design problem rather than a collection of independent parts.

Problem 1 - Base Drag Reduction:

Blunt rocket bases can contribute a significant fraction of total drag due to low-pressure wake formation. One effective mitigation strategy is the use of a boat tail, which smoothly transitions from the body diameter to the motor exit diameter. This helps reduce turbulent airflow exiting the rocket, streamlining aerodynamics and decreasing drag.

Key considerations included:

- A shallow boat tail angle to avoid flow separation
- Minimizing added mass while achieving meaningful drag reduction
- Compatibility with structural and retention hardware
- Steep boat tail angles risk boundary-layer separation, increasing turbulence and drag rather than reducing it.

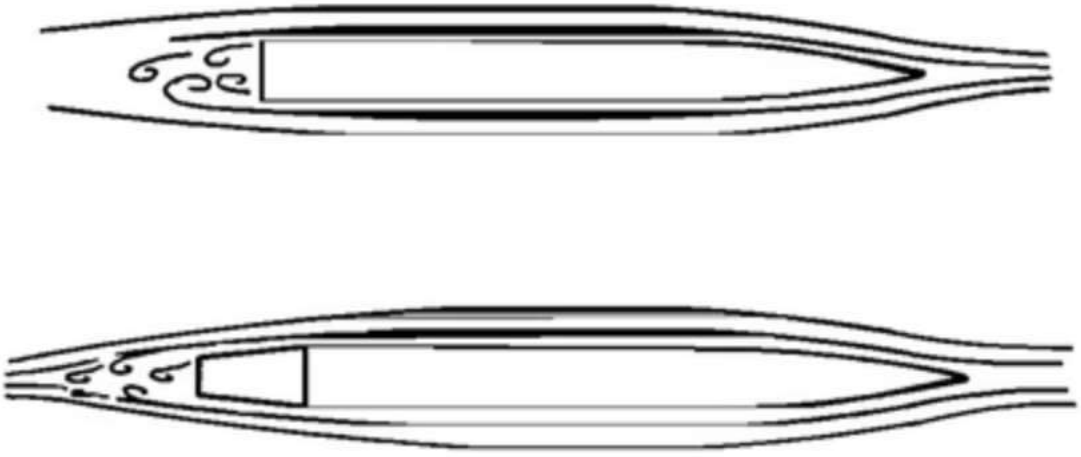


Figure 1B: Airflow comparison - non-boat tail vs. boat tail

Problem 2 - Motor Retention & Serviceability:

The motor retention system needed to be vibration-resistant, structurally reliable, and easily accessible during assembly and servicing.

Past internal motor retention approaches used by the team were difficult to access, increasing integration time and safety risk during motor installation and removal. Improving serviceability was therefore a core requirement, not a secondary constraint.

Problem 3 - Motor Position vs. Stability:

Rocket stability depends on the relationship between the center of gravity (CG) and center of pressure (CP). Axial motor position has a direct impact on CG location.

Design requirements included:

1. Flexibility to adjust motor position to meet stability objectives.
2. Preservation of a clean and direct thrust load path.
3. Avoidance of unintended load transfer into non-structural subsystems.

My Solutions:

I designed and implemented a composite boat tail shell system, using a shell-and-skeleton concept that would directly interface with the thrust plate. This approach reduced base drag while maintaining a low mass penalty and structural robustness.

To justify the design effort quantitatively, I used the Rayleigh drag relation to estimate the impact of base area reduction on overall drag:

$$D \propto \mathrm{A}_{base}$$

This analysis demonstrated that the boat tail could produce a meaningful performance gain, providing a performance-driven rationale for the added manufacturing complexity and cost.

I was personally in charge of designing, manufacturing, and testing the three components involved that were then mounted to our M class rocket for Launch Canada 2025. The final configuration contributed to an approximately 34% increase (12300 ft predicted to 16500 ft actual) in achieved maximum apogee compared to prior configurations.

Breaking down the three components:

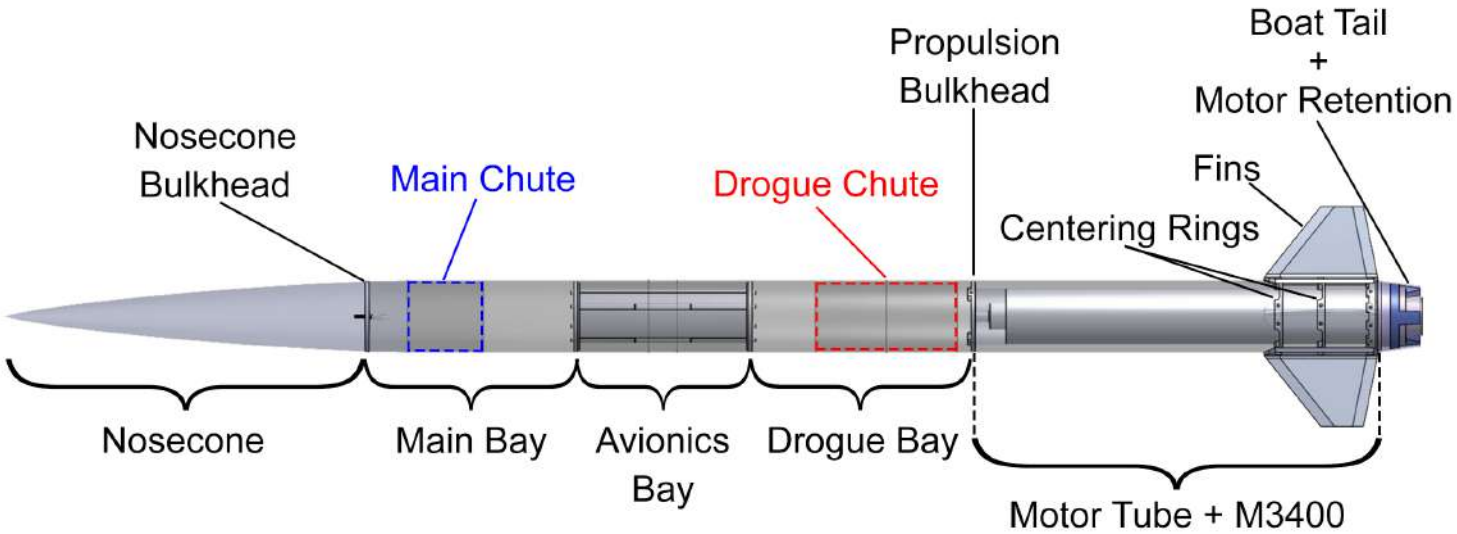


Figure 1C: Labeled rocket overview

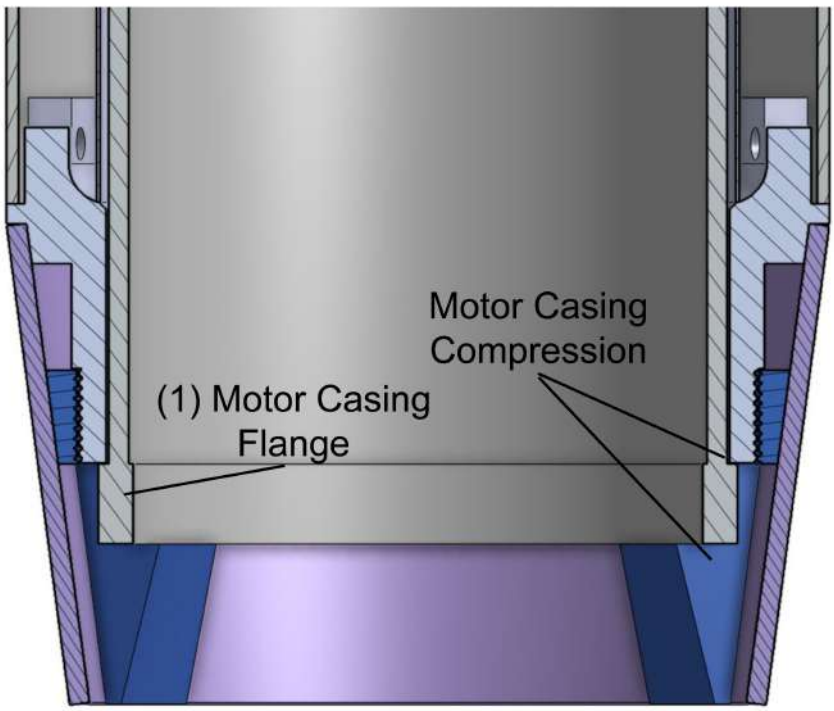


Figure 1D: Propulsion structure breakdown

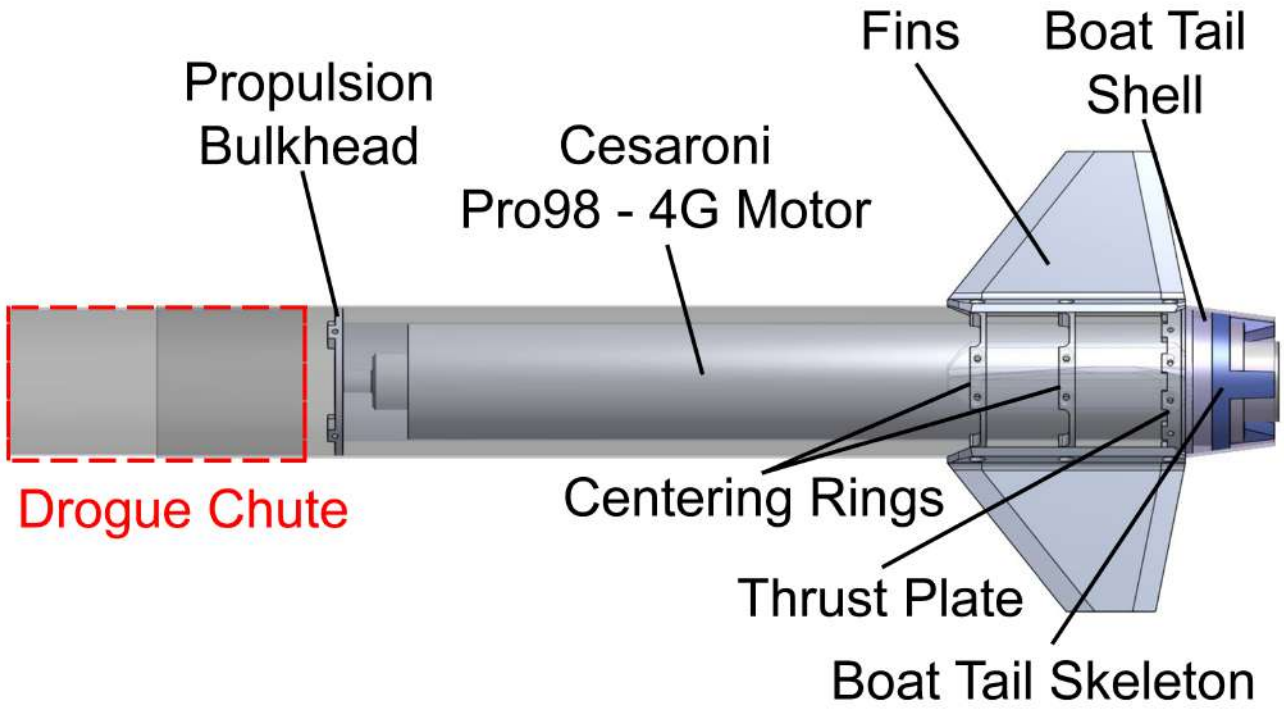


Figure 1E: Labeled aft end structure breakdown

1) Composite Aerodynamic Boat Tail Shell

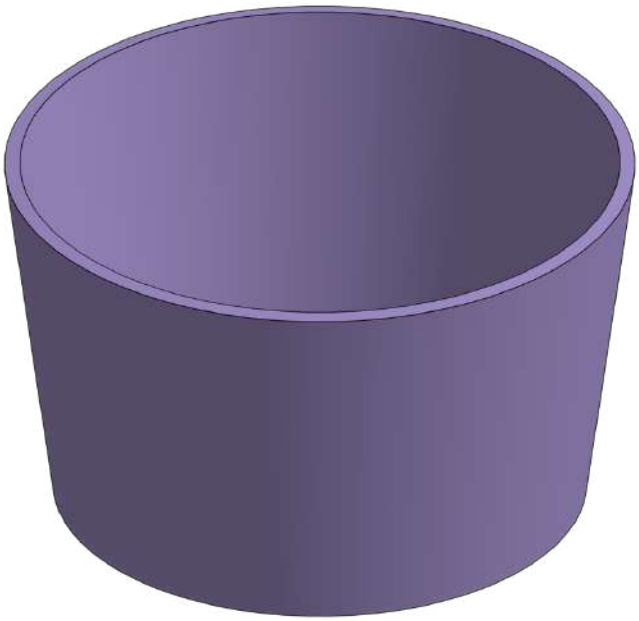


Figure 1F: Boat tail shell CAD

The first component consisted of a fiberglass molded “shell” that was on the exterior of the system. It acted as a surface that was responsible for transitioning down from fuselage diameter to motor exhaust diameter.
The shell was a manual hand layup of a fiberglass cone, cut and shaped around a 3D printed mold. It utilized a knitted fiberglass weave with 105 low-viscosity liquid epoxy resin. On recovery of the rocket, the fiberglass cone had been lodged into a log with chunks of rotted wood being stuck in it.

Figure 1G: On recovery boat tail destruction.

2) Threaded “Crown” (External Motor Retention)
The second part I designed was a threaded crown that interfaces with the thrust plate using large-diameter threads. This component enables external motor retention while simultaneously providing a structural interface for the boat tail shell.
Design benefits included:

- Improved accessibility during assembly and servicing
- Robust axial load transfer through well-distributed threads
- Simplified integration of aerodynamic and structural components

Figure 1H: Crown CAD (Left) and Crown Manufactured (Right)

3) Thrust Plate
In parallel, I developed multiple thrust plate concepts, including variants that were both compatible and incompatible with a boat tail. This explicit comparison reduced program risk in case the aerodynamic gains did not justify the added complexity.

The final thrust plate incorporates a projecting tube to control motor position and interface geometry. A major integration risk addressed during this phase was avoiding unintended thrust transfer into the fin cage through vertical structural connections.

The design effort focused on ensuring that all axial thrust loads were transferred cleanly into the fuselage.

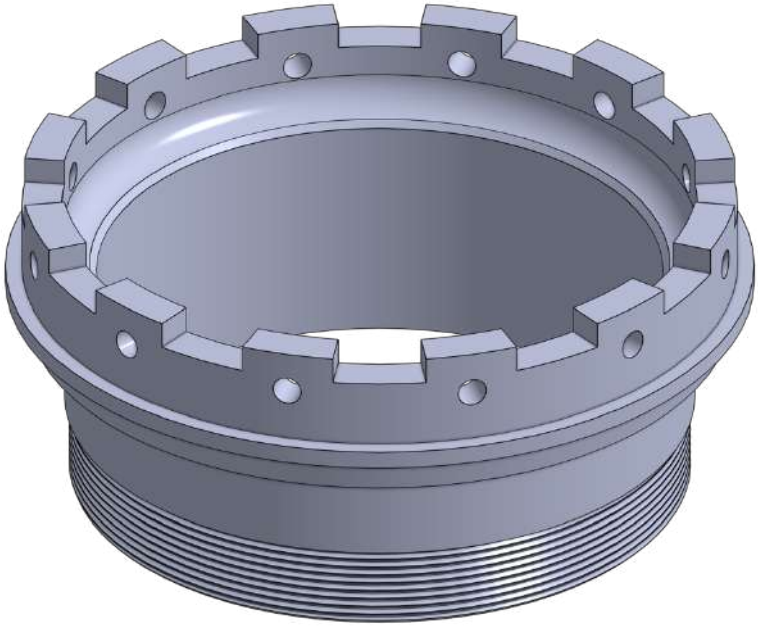


Figure 1I: Thrust plate CAD

Figure 1J: Partially manufactured thrust plate.

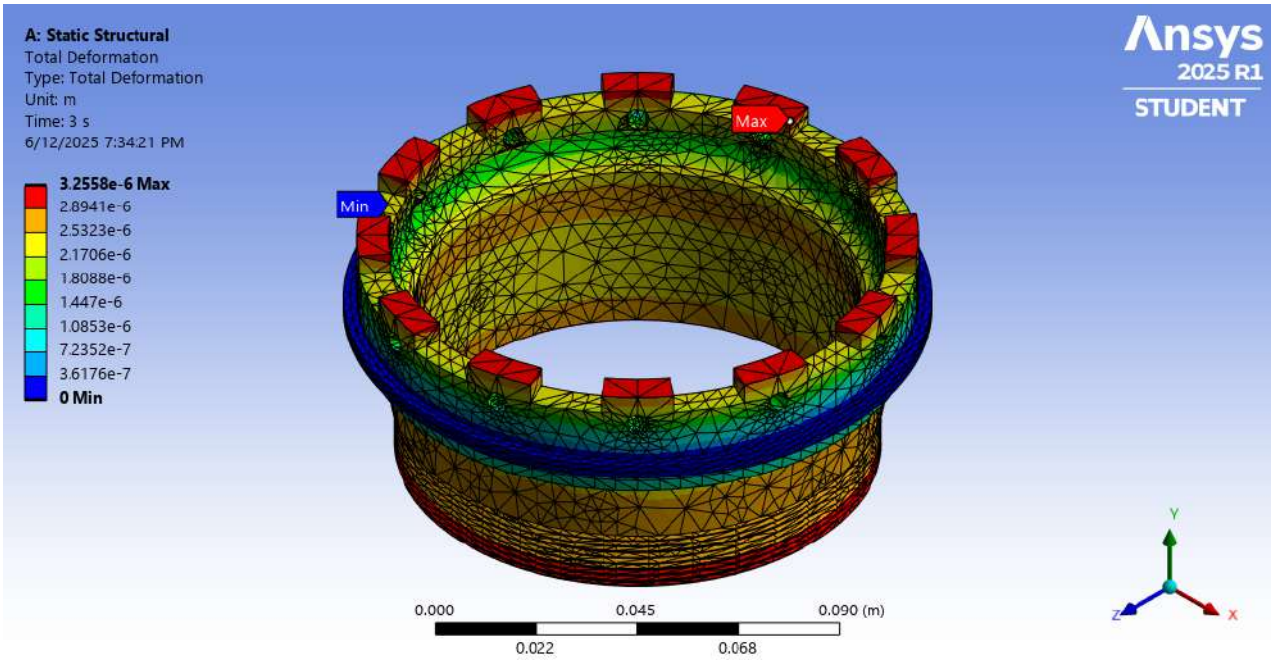


Figure 1K: Simulated total Deformation of applied forces (8kN) on thrust plate

Final Build - Materials & Manufacturing:

Threaded crown and thrust plate: Machined 6061 aluminum

Boat tail shell: Fiberglass epoxy composite layup

Structural intent: the thrust plate is bonded directly into the composite fuselage, creating a continuous and well-defined thrust load path into the airframe.

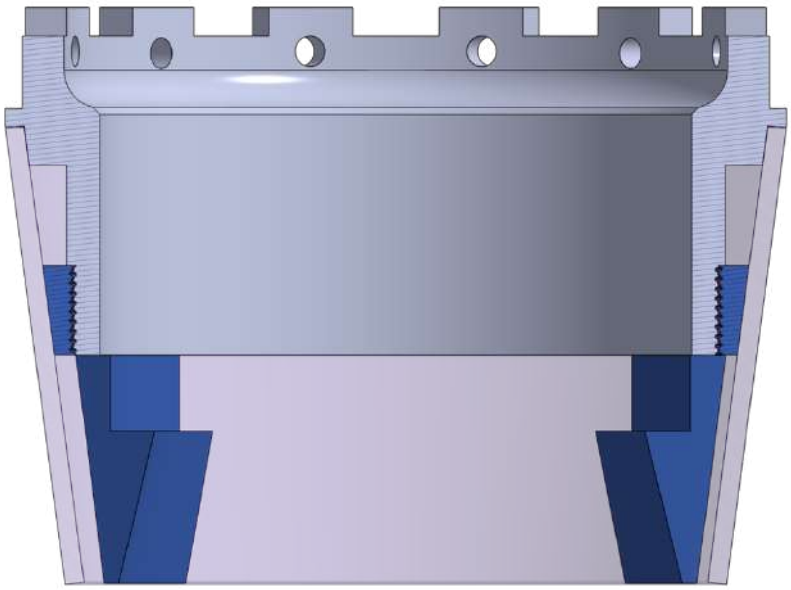


Figure 1L: Lower motor assembly CAD

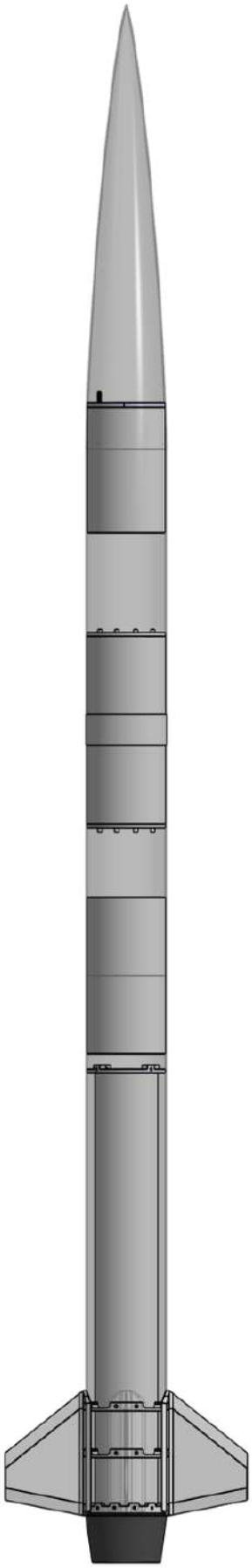


Figure 1M: Rocket overview

Personal Project

Sunrise Simulight

An Open-Source IoT Sunrise Alarm Lamp

Purpose and motivation:

I built Sunrise Simu light because standard alarm clocks felt harsh, and I wanted a wake-up system that worked with sleep cycles rather than against them. The core idea was simple: instead of a sudden sound trigger, the plan was to gradually increase room brightness, so the body transitions toward wakefulness more naturally. When I looked at commercial sunrise lamps, most options failed at least one of my constraints: they were not open source, lacked the scheduling/customization features I wanted, or were priced high enough that experimentation and iteration felt wasteful. So I designed and built my own solution end-to-end which included the hardware, firmware, enclosure, and documentation—and released it as an open-source project intended to be modifiable by others.

Figure 2A: Sunrise Simulight logo.

Schematics & development:

The design centers on a Wi-Fi capable microcontroller (ie. Esp32, ect) controlling a high-current MOSFET to drive a 12V lighting load. I used a DS3231 RTC (real-time clock module) as a fallback time base so schedules remain reliable even if the network is unavailable. The power system was designed to keep wiring simple using a single external power supply, while still supporting safe switching, soft-start behavior, and room for thermal/current protection as the load increases.

On the firmware side, I treated the sunrise effect as a control curve problem rather than a simple linear ramp. Brightness transitions were implemented with smooth easing (Bezier-style shaping) over configurable windows (roughly 30–90 minutes, depending on configured preference). Scheduling and overrides were handled through a lightweight structured data layer (JSON-style control objects), so adding features like a sunset routine or “weekend schedule” doesn’t require rewriting the architecture. To make tuning practical, I built a small desktop utility (Python) to quickly adjust ramp profiles and scheduler presets without manual reflashing each time.

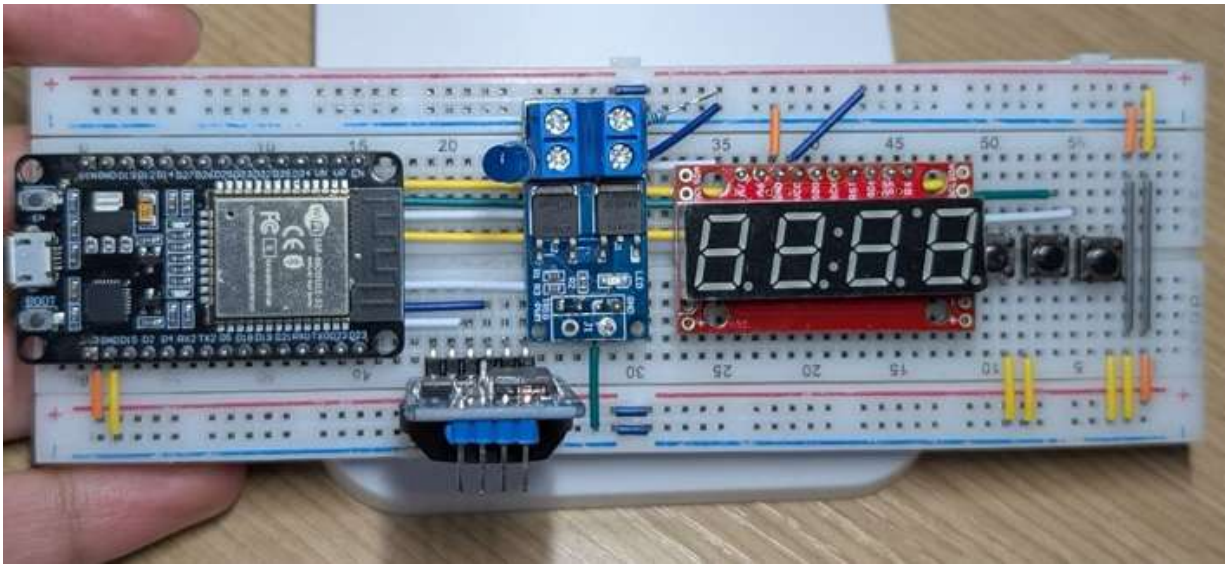


Figure 2B: Prototyped SAC with 5V Test LED

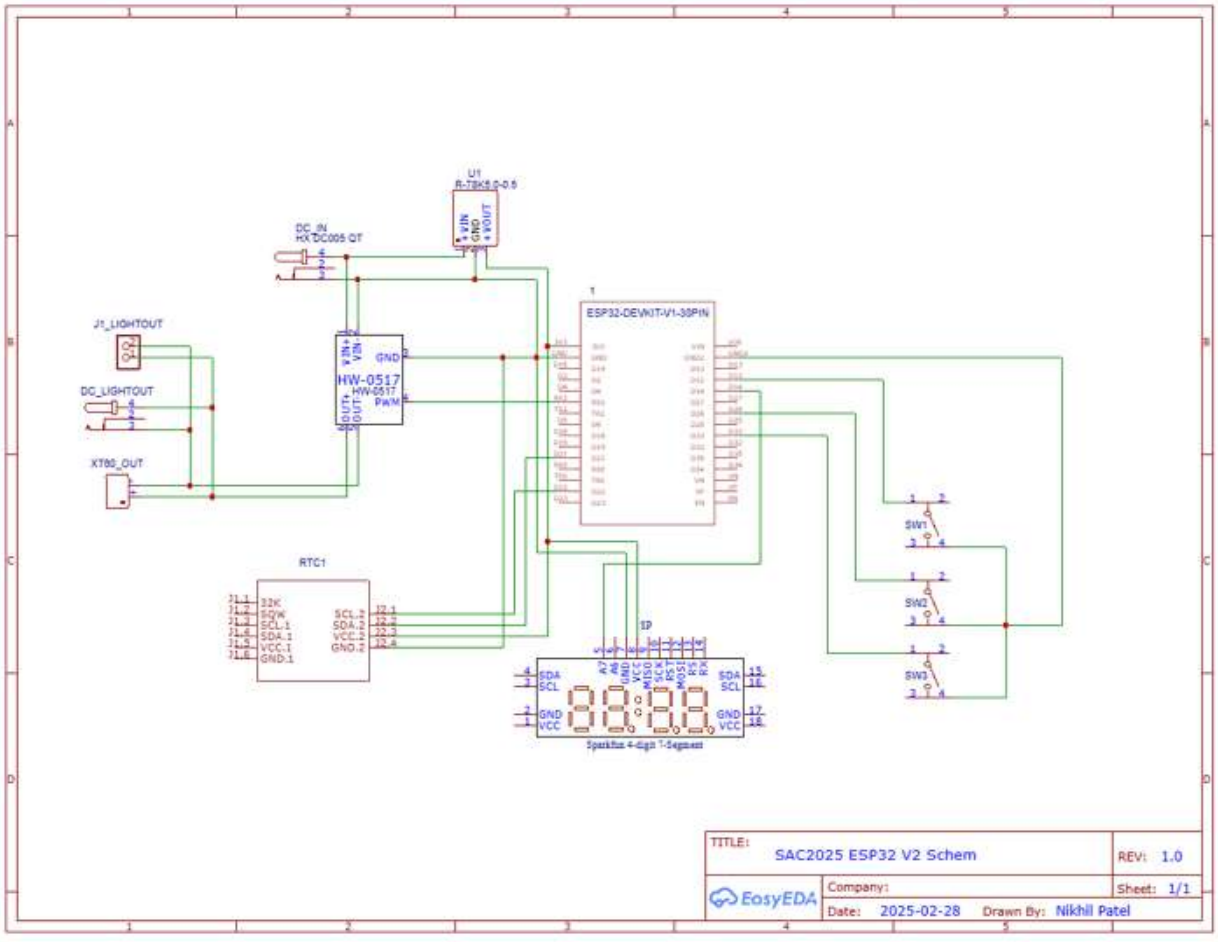


Figure 2C: Wiring Diagram Showing Simple COTS Components

Documentation:

Because the goal was true open-source reproducibility, I wrote documentation as if another student had to rebuild the project without asking me questions. The repo includes a circuit and BOM package (parts, ratings, and wiring intent), firmware notes describing the scheduler logic and command structure, and a quick start guide for integrating with common smart-home ecosystems. I also added a calibration and troubleshooting section so users can set practical targets (brightness levels, ramp length, and timing) and diagnose issues like flicker, unstable Wi-Fi provisioning, or incorrect schedule behavior.

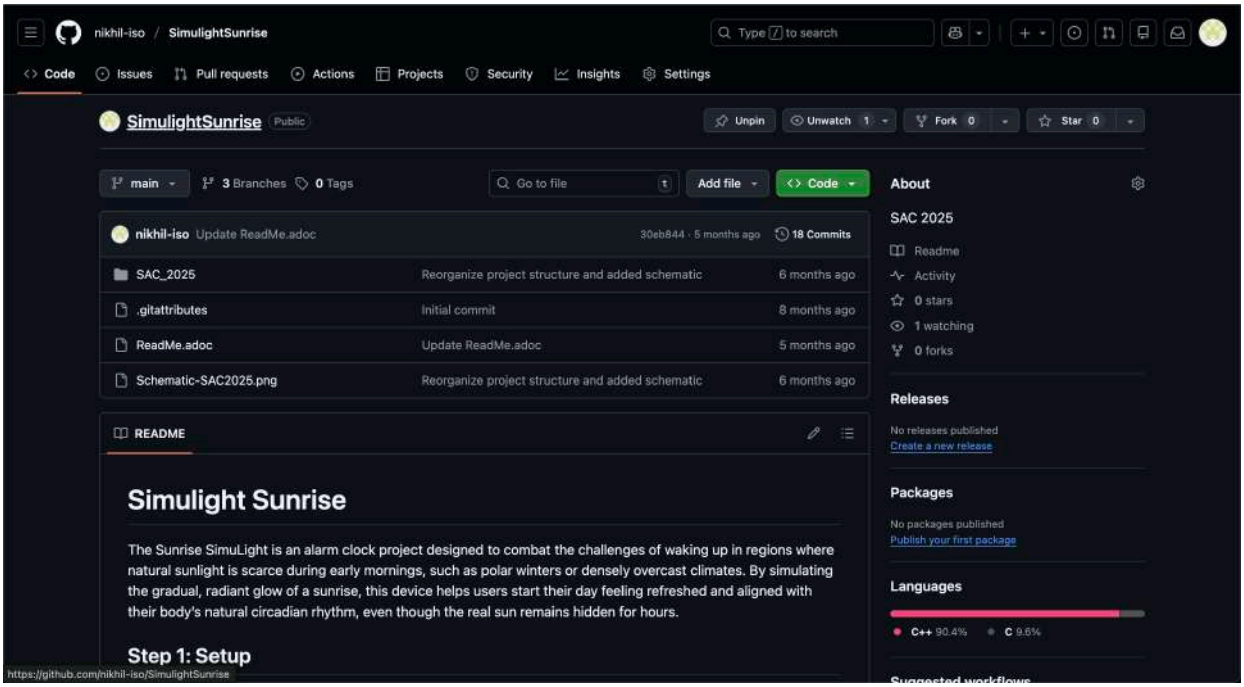


Figure 2D: GitHub Repository for open-source sharing

Tests & results:

Testing focused on three areas: (1) wake-up experience, (2) electrical performance, and (3) integration reliability. Functionally, the lamp ramps from off to full brightness with a smooth transition that avoids perceptible stepping, and the schedule execution remains consistent even when network time is unavailable (RTC fallback). Electrically, I iterated PWM and switching behavior to minimize visible flicker and tuned the power options so the system operates efficiently in typical brightness ranges (final revisions landed in the mid-80% efficiency range, depending on load and settings). For usability feedback, I ran a small user trial (roughly six) and collected qualitative responses that generally pointed to reduced grogginess compared to traditional alarms, which guided me towards the next set of defaults and ramp curve options.

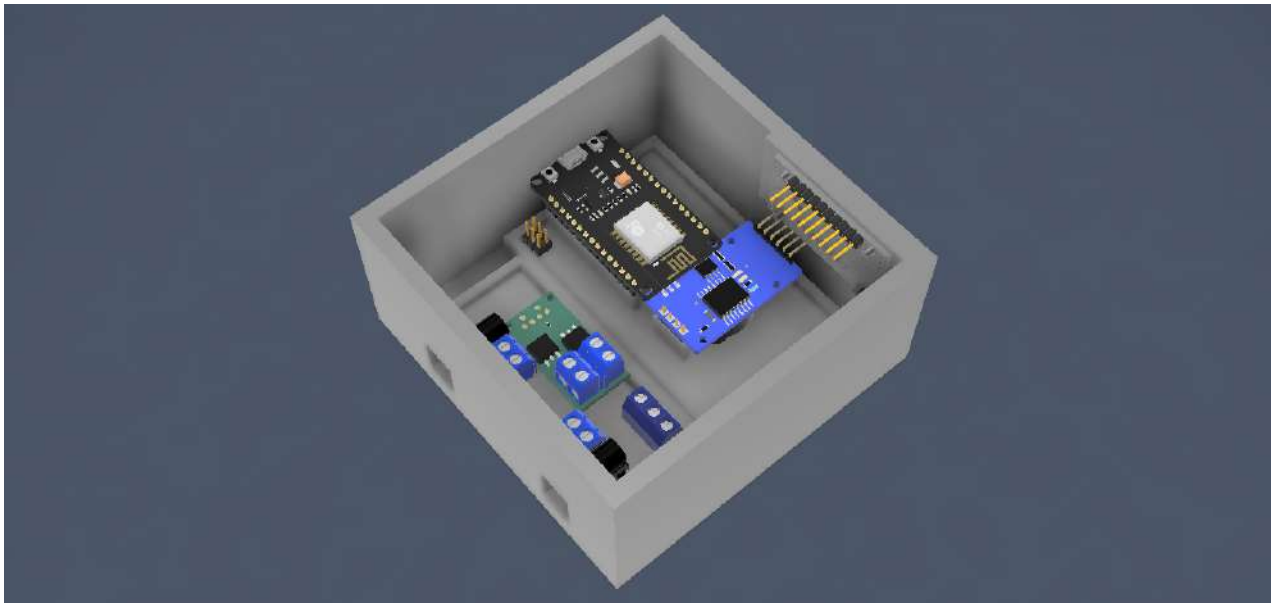


Figure 2E: Enclosure + Interior setup render

Future plans:

The next steps are mostly software-driven, since the hardware platform was intentionally designed to be extendable. The main planned upgrades include: a native mobile app that supports offline scheduling and guided routines, a local-first smart-home bridge (MQTT/Home Assistant style) for users who want less cloud dependence, and adaptive ramp shaping based on user preferences and historical behavior. On the hardware side, I want to add optional sensors (ambient light and basic biometric inputs) so the device can automatically adjust brightness targets, and I'm interested in transitioning to a PCB based design for small-run manufacturing while keeping it serviceable and mod-friendly.

Personal Project

Arduino Based Macropad

An open source wired keyboard

Purpose and Motivation

The objective of this project was to design and build a low-cost, highly customizable macro keypad that could serve both accessibility users and power users, while avoiding the cost and limitations of commercial macro boards. Many existing solutions are either closed-source, expensive, or inflexible in hardware and firmware customization. This project was motivated by the need for a modifiable input device that could be adapted to different user requirements, particularly for individuals with accessibility needs or specialized workflows.

A key design goal was to ensure the hardware remained serviceable and modular, allowing switches, encoders, and firmware behavior to be changed without replacing the entire device. By centering the design around an Arduino-based HID platform, the keypad remains fully cross-platform and does not rely on proprietary drivers. The project also served as a sandbox for experimenting with macro automation, lighting feedback, and enclosure ergonomics.

Schematics and Development

The keypad is built around an Arduino Micro, selected for its native USB HID support and ease of firmware development. The hardware consists of a 4 x 2 key matrix using Cherry MX mechanical switches, providing flexibility in switch selection based on user preference or accessibility requirements. The design supports rapid iteration and replacement of switches without desoldering.

Custom Arduino firmware was developed to implement multi-profile macro storage, per-key remapping, and adjustable debounce timing. Macros are handled entirely in firmware, allowing deterministic execution without host-side software dependencies. The enclosure was designed in CAD and 3D printed, incorporating a slight typing angle to improve ergonomics and reduce wrist strain. Custom keycaps were also designed to support clearer labeling and alternative geometries.

Throughout development, usability considerations informed design decisions, particularly around key spacing, actuation force, and feedback. The modular design allows the keypad to be repurposed for different tasks such as media control, productivity shortcuts, or accessibility-focused input layouts.

Documentation

A major component of this project was producing documentation sufficient for third-party replication and modification. Firmware documentation outlines HID report handling, macro execution logic, and internal state management. Hardware documentation includes a complete bill of materials, wiring diagrams, and enclosure assembly instructions.

Electrical schematics clearly describe the key matrix, USB interface, and power considerations. An assembly guide was written to ensure the keypad can be built using basic tools, with clear instructions for swapping switches or modifying the enclosure. Accessibility notes and contribution guidelines were included to encourage community-driven extensions and improvements.

Tests and Results

The final prototype achieved all primary functional requirements. The keypad operates as a plug-and-play USB HID device on Windows, macOS, and Linux without requiring drivers. Firmware testing showed consistent macro execution with sub-5 ms debounce variance, ensuring responsive and reliable input behavior.

From a manufacturability standpoint, the design successfully reduced cost by relying on commonly available components and a reusable PCB/enclosure combination. Assembly time was kept under 30 minutes for a complete unit, meeting the goal of accessibility and ease of modification. The open-source release resulted in early community engagement, validating the project's relevance and extensibility.

Future Plans

Future development will focus on improving usability and expanding feature depth without increasing hardware complexity. Planned upgrades include a graphical macro configuration tool with drag-and-drop layout editing and shareable presets. Wireless variants using BLE or low-latency 2.4 GHz communication are also planned to broaden deployment options.

On the firmware side, the macro engine will be extended to support conditional logic, application-aware layers, and enhanced feedback mechanisms such as haptic or audio cues. Longer-term goals include producing small-batch kits in collaboration with makerspaces and developing instructional material for use in accessibility and engineering education contexts.

More projects can be viewed on my website here.