Development of an Open-Source Model Rocket Flight Stability Computer and Filter

Conrad Motis

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Electrical and Computer Engineering

University of Washington
2026

Committee:
Jie Sheng
Morteza Nabavinejad
Hillary Stephens

Program Authorized to Offer Degree:
School of Engineering and Technology

University of Washington

Abstract

Development of an Open-Source Model Rocket Flight Stability Computer and Filter

Conrad Motis

Chair of the Supervisory Committee:
Jie Sheng
School of Engineering and Technology

The modern rocketry era has made advanced flight-control concepts increasingly visible, from the growth of orbital launch activity to the routine recovery of rockets through upright propulsive landing. However, educational tools available to students and hobbyists often remain separated from these modern control concepts. Open-source tools such as OpenRocket have shown the value of accessible, education-focused platforms for model rocket design and simulation, but fewer resources provide a complete, published, and experimentally tested approach for active stabilization in hobbyist-scale rockets.

This thesis presents the development of an open-source actively stabilized model rocket platform, focusing on the avionics, sensing, flight data collection, and steering methods needed for gravity-referenced stabilization. The system uses onboard inertial sensing to estimate the local gravity vector and commands the control surfaces to reduce lateral tilt relative to the local gravity vector during ascent. This approach is distinct from guidance, as the rocket does not navigate toward a target location or commanded trajectory. Instead, the objective is to provide active stabilization using a physically intuitive reference suitable for educational-scale vehicles.

The platform integrates low-cost embedded hardware, inertial measurement sensors, high-G acceleration sensing, onboard data logging, and active fin control to support repeatable flight testing and post-flight analysis. Flight data are used to evaluate the behavior of the stabilization system and examine practical limitations introduced by sensor noise, accelerometer saturation, vibration, and non-gravitational acceleration during powered flight.

The goal of this work is to provide an approachable and customizable open-source platform for students, educators, and hobbyists interested in aerospace control systems. Potential applications include classroom demonstrations, student competitions, altitude-targeting challenges, and experimental model rocket projects where safe, efficient, and repeatable flight behavior is desired.

1 Introduction

The modern rocketry era has made advanced flight-control concepts increasingly visible, from increased orbital launch activity to upright propulsive rocket recovery. However, educational tools available to students and hobbyists often remain separated from these modern control concepts. This thesis presents an open-source, actively stabilized model rocket platform intended to make embedded sensing, data logging, and flight-control experimentation more accessible at hobbyist and educational scales.

Prior educational and hobbyist model rocket work has generally emphasized passive stability, open-loop simulation, hardware construction, onboard data logging, or post-flight analysis. OpenRocket provides an accessible environment for model rocket design, stability analysis, and flight simulation, while Altus Metrum and the Portland State Aerospace Society demonstrate the value of open-source avionics for telemetry, data logging, and recovery systems [Nis13, Alt26, Por14]. Hardware-oriented open-source rocketry projects also show the value of publishing practical build information. Half Cat Rocketry's Mojave Sphinx project, for example, provides an openly documented amateur liquid-bipropellant rocket design with guidebook material, design files, drawings, simulation files, and launch procedures [Hal26]. Although Mojave Sphinx is primarily a propulsion and vehicle-hardware project rather than an active-stabilization system, it illustrates the educational value of documented, buildable, and repeatable amateur rocketry hardware.

In contrast, fewer open-source projects provide a complete platform for active stabilization during flight. Publicly documented active-control projects, such as BPS.space thrust-vector-control model rockets, demonstrate the feasibility of closed-loop stabilization at small scales, but differ from this work in actuator choice, vehicle architecture, publication focus, or availability of source code [BPS26b, BPS26a]. This project contributes a gravity-referenced, fin-actuated, open-source stabilization platform intended specifically for educational and hobbyist-scale model rockets [MBS25].

The remainder of this thesis is organized as follows. Section 2 reviews related work in open-source rocket simulation, passive aerodynamic stability, open-source avionics, and complementary filtering for inertial sensing. Section 3 defines the project scope, distinguishes active stabilization from guidance, and summarizes the development history of the flight computer across multiple airframes and flight tests. Section 4 presents the current V2 implementation, including the Apogee Mako airframe, firmware architecture, modular avionics stack, telemetry system, gravity-vector steering method, and custom complementary filter. Section 5 summarizes the main results and identifies future work needed to improve phase-dependent gravity-vector estimation during boost, coast, freefall, and recovery.

2 Related Work

Open-source model rocket simulation tools provide an important foundation for this work because they make aerodynamic design and flight-performance estimation accessible to students, educators, and hobbyists. OpenRocket is a prominent example of this approach. It provides a free and open-source environment for designing model rockets, estimating center of gravity, estimating center of pressure, evaluating static stability, and simulating expected flight performance before construction and launch. In this project, OpenRocket was used as a design-support tool for evaluating candidate airframes, motor selections, stability margins, and expected flight behavior. However, OpenRocket is primarily a simulation and design environment, while this thesis focuses on the physical implementation of an actively stabilized flight-test platform.

The aerodynamic stability methods used by OpenRocket are closely related to the Barrowman method for predicting the center of pressure of slender finned rockets. Barrowman developed practical methods for calculating aerodynamic characteristics of slender finned vehicles, including normal-force coefficient derivatives and center-of-pressure location [Bar67]. These methods are significant in model rocketry because passive static stability depends on the relationship between the center of gravity and center of pressure. In a conventionally stable rocket, the center of pressure is located aft of the center of gravity so that aerodynamic normal force produces a restoring moment when the vehicle is disturbed from its flight direction. This thesis uses the same basic stability concepts, but applies them in the context of active fin control rather than relying solely on fixed passive fins.

Attitude estimation is another relevant area of prior work because the steering method depends on estimating the gravity vector in the rocket body frame. Gyroscopes provide short-term angular-rate information, but attitude estimates obtained by integrating gyroscope measurements are susceptible to drift. Accelerometers can provide a gravity reference under low-acceleration conditions, but they are affected by vibration, sensor noise, saturation, and non-gravitational acceleration during powered flight. Mahony, Hamel, and Pflimlin developed nonlinear complementary filters on the special orthogonal group, $SO(3)$, for attitude estimation using measurements from low-cost inertial sensors [MHP08]. Their work provides a useful theoretical basis for combining gyroscope propagation with vector-based correction measurements.

The complementary filtering approach used in this thesis is narrower and more application-specific than a full attitude observer on $SO(3)$. Rather than estimating complete vehicle attitude for navigation or target guidance, the implemented filter estimates a gravity vector used for active stabilization. The filter propagates the gravity-vector estimate using gyroscope measurements and applies an accelerometer-derived correction when the measured acceleration is likely to contain useful gravity-direction information. During high-acceleration boost conditions, the accelerometer correction is reduced so that thrust-induced acceleration is not interpreted as a change in gravity direction. This allows the project to retain the educational simplicity of gravity-referenced steering while addressing practical sensor limitations observed during model rocket flight.

The first version of this project was presented in a previous UEMCON paper, which described the construction and launch testing of a servo-based model rocket using a Raspberry Pi Pico, BNO055 inertial measurement unit, microSD data logging, and movable control fins [MBS25]. That work demonstrated that a low-cost model rocket platform could survive launch, record useful inertial data, and command active stabilization corrections during ascent. The present thesis extends that work by restructuring the hardware and firmware into a more modular platform, improving data-logging performance, adding additional sensors and telemetry hardware, and developing a custom gravity-vector filter for high-acceleration flight conditions.

3 Project Scope and Proof of Concept

3.1 Providing a Well-Developed Platform

The engineering and experimentation conducted for this project have resulted in 25 launches of the flight computer across multiple circuit board revisions and four host-airframe configurations. These launches provided practical validation of the avionics architecture, active fin-control hardware, data-logging system, telemetry system, and post-flight analysis methods. The project is therefore presented not only as a single vehicle design, but as a developing open-source platform for educational active-stabilization research.

The scope of this work is centered on creating a system that can be built, modified, tested, and studied at hobbyist and educational scales. Emphasis is placed on accessible hardware, modular construction, understandable control methods, and repeatable flight-data collection. This allows the platform to serve both as a working flight computer and as a practical teaching tool for embedded systems, sensing, control, and aerospace experimentation.

The launch history should be interpreted as a sequence of flight-computer development tests rather than as a count of fully closed-loop active-stabilization flights. Some launches were performed with steering disabled, with the fins locked, with electronics installed but powered off, or with the flight computer configured only for data recording. These flights were still useful because they evaluated airframe integration, launch behavior, data logging, recovery behavior, hardware durability, and sensor performance under launch conditions.

The first major configuration was the V1 Estes Green Eggs prototype, which completed 11 flights using BNO055 gravity-vector steering and onboard data logging. These flights demonstrated that the proof-of-concept flight computer could survive repeated launches, record usable flight data, and command active stabilization corrections. Testing ended when the cardboard airframe structure could no longer support continued flights.

A redesigned Green Eggs configuration completed three additional development flights. Two of these flights used the BNO055-based steering approach but were underpowered for the additional vehicle mass. A third flight used an early custom complementary-filter implementation, but experienced a point-to-point wiring failure. These flights informed the later design process but were not used as primary evidence for filter-performance evaluation.

The V2 Apogee Mako configuration completed ten flights. Mako flight 1 carried the flight computer with the electronics powered off and served as a passive comparison flight. Mako flights 2 through 5 used the BNO055-provided gravity vector for active steering. Flight 5 was completed as a successful Tripoli Level 1 certification flight with active-stabilization electronics enabled. Mako flights 6 through 8 and 10 used versions of the custom complementary filter with steering enabled, while Mako flight 9 was flown with electronics enabled but the fins locked in place. Mako flights 6, 7, 8 and 10 provided useful logged data for comparing gravity-vector behavior between a mostly upright launch and a visibly angled launch.

The flight computer was also carried inside a LOC 5.5-inch Patriot rocket during a Tripoli Level 2 certification flight. That airframe did not include active steering components, so the flight was not an active-stabilization test. However, it provided an additional record-only avionics flight in a larger high-power rocket environment.

3.2 Stability Versus Guidance

In the context of rocket steering, it is important to clarify the scope and intent of this project. The system presented in this work is designed for active stabilization rather than guidance. It uses the estimated gravity vector as a reference and commands the control surfaces to reduce lateral tilt relative to the local gravity vector during ascent. The vehicle does not contain a navigation objective, target location, or commanded trajectory.

This distinction is significant because guidance requires steering toward a defined target or path, while stabilization seeks to regulate vehicle attitude relative to a reference. The model rockets used in this research are therefore not guided vehicles; they are actively stabilized educational platforms. The control objective is to improve ascent stability using local onboard sensor measurements, rather than to direct the rocket toward a predetermined destination.

The linear steering methods used in this project are intended for hobbyist and educational-scale model rockets. The goal is to provide an open-source control architecture that students, educators, and hobbyists can understand, build, modify, and adapt. Potential applications include student competitions, altitude-targeting challenges, and classroom demonstrations where the objective is to reach specified flight conditions safely, efficiently, and repeatably.

3.3 Platform Design Goals

The project uses readily available components and modular circuit boards to improve accessibility and simplify modification. Each major subsystem is implemented on a dedicated board or board section, allowing the avionics package to be expanded or adapted as project requirements change. Simple sensor drivers for this platform have been written and are provided with the firmware, and the hardware design is intended to remain compatible with alternative sensors, communication modules, and control configurations.

The flight computer is built around a Raspberry Pi Pico, which uses the RP2040 microcontroller operating at 125 MHz. The baseline Pico was selected as the development platform because it provides the required processing, communication, and timing resources without relying on additional wireless hardware. Other Pico variants provide additional hardware features, including wireless communication on Pico W boards and the faster RP2350 microcontroller, additional memory, optional RISC-V cores, and upgraded interfacing features on Pico 2 boards. Because the baseline Pico was used for development, the open-source firmware is expected to be portable to other Pico-family boards with only minor hardware and configuration changes.

The current avionics design uses a Bosch BNO055 inertial measurement unit, a Kionix KX134 high-G accelerometer, and a Bosch BMP390 pressure sensor. Since the Raspberry Pi Pico operates at 3.3 V, a passive microSD card adapter can be connected directly to the flight computer for onboard data logging. Hobby-grade servomotors, commonly available from online electronics suppliers and remote-control hobby stores, are used both for direct fin actuation and for linkage-based steering mechanisms.

A PA1616D GPS module and an Adafruit RFM95W LoRa radio module are used together for location reporting. The GPS data are not used for flight-control computations. Instead, GPS position data are broadcast to a handheld receiver, referred to as the ground station, to assist with locating rockets that may travel out of sight during ascent or recovery.

The microSD data-logging system and LoRa communication system use open-source libraries external to this project. The microSD card interface is based on the `no-OS-FatFS-SD-SDIO-SPI-RPi-Pico` library provided by carlk3, which supports SD card access for the Raspberry Pi Pico using SPI and SDIO interfaces [car24]. The LoRa radio interface uses the `pico-lora` library provided by akshayabali, which supports RP2040-based boards and Semtech SX1276/77/78/79 LoRa radio modules [aks21]. Both libraries are publicly available on GitHub and are integrated into the project firmware.

3.4 Proof-of-Concept Objective

The proof-of-concept stage was used to determine whether a hobbyist-scale model rocket could be actively stabilized using low-cost embedded hardware, onboard inertial sensing, and movable control fins. The initial objective was not to produce a fully optimized flight computer, but to confirm that the architecture could survive launch, record useful flight data, and command meaningful stabilization corrections during ascent. The results from this stage informed the later upgraded Green Eggs and V2 Mako designs.

3.5 Prototype Version and Results

The first prototype version of the flight computer was built around an Estes Green Eggs model rocket kit, as shown in Figure 1, which was modified to include a 3D-printed servo housing and four steering fins. In this initial configuration, the fins were mounted directly to the servomotor horns to minimize mechanical complexity and conserve space within the narrow airframe.

The first flight computer used a Raspberry Pi Pico, BNO055 inertial measurement unit, microSD card interface, status LEDs, and four servomotors. These components were assembled on a hand-soldered protoboard using point-to-point wiring. The completed electronics assembly was inserted

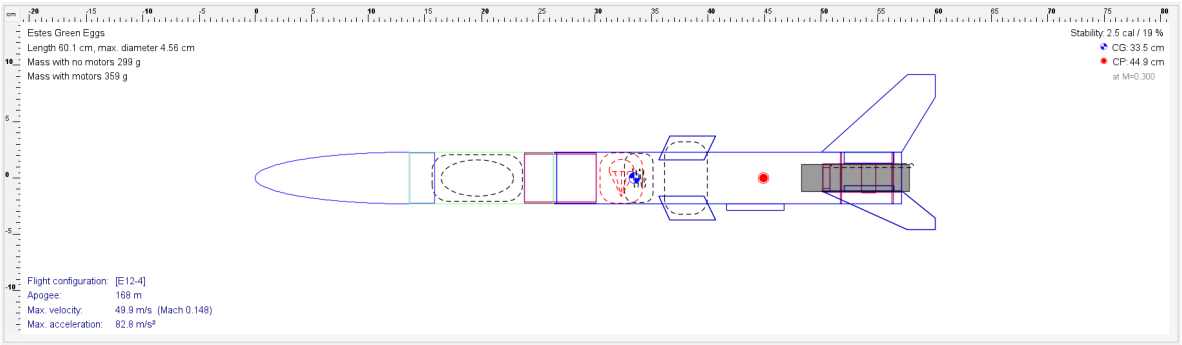

Figure 1: Original V1 Green Eggs prototype modeled in OpenRocket.

into the rocket through the forward payload section. A separate 3D-printed sensor puck was used to constrain the IMU orientation relative to the airframe, support user-accessible switches, and provide space for a GPS antenna beneath the plastic nose cone.

The early firmware was intentionally minimal, but functional. It implemented sensor polling, data logging, servo pulse generation, and basic gravity-referenced steering. The RP2040 programmable I/O peripherals were used to generate stable servo control pulses independently of the main program loop, while the C firmware computed steering updates from the IMU gravity-vector output. During this phase, flight data were logged to the microSD card for post-flight analysis, allowing the control behavior and sensor measurements to be evaluated after launch.

Several peripherals were physically included or planned during the first-version design, but not all were used in closed-loop flight control. In particular, GPS parsing and some slower peripheral operations were deferred because they could reduce the main loop rate and introduce visible stuttering in fin motion. This limitation helped define later design requirements, including interrupt-driven steering updates, faster data logging, and a more modular software structure.

Multiple launch tests confirmed that the first-version system could survive flight, record usable data, and command active stabilization corrections during ascent. This early design became the subject of a UEMCON paper focused on the construction, firmware, data logging, and flight behavior of the prototype. After the feasibility of the architecture was demonstrated through practical launch testing, the project was expanded into later hardware and firmware revisions with additional sensors, improved data collection, GPS location reporting, LoRa telemetry, and a more robust airframe integration.

4 Current V2 Implementation and Results

4.1 Apogee Mako Airframe

After the proof-of-concept prototype demonstrated successful active stabilization, a larger airframe was selected for the second version of the system. The V2 vehicle is based on a 3-inch-diameter Apogee Mako model rocket kit, modified to include an actively controlled fin canister. This canister serves as both the steering mechanism housing and the avionics bay, integrating the control fins, servo linkage system, flight computer, and supporting structure into a single forward-mounted assembly.

The larger airframe provided additional internal volume for sensors, data logging, telemetry hardware, and a more mechanically robust steering system. It also created a more suitable platform for higher-power flight testing, including flights beyond the small motors used for the first prototype. OpenRocket was used during the airframe design process to evaluate basic vehicle characteristics such as center of gravity, center of pressure, and expected flight performance with candidate motors, as shown in Figure 2.

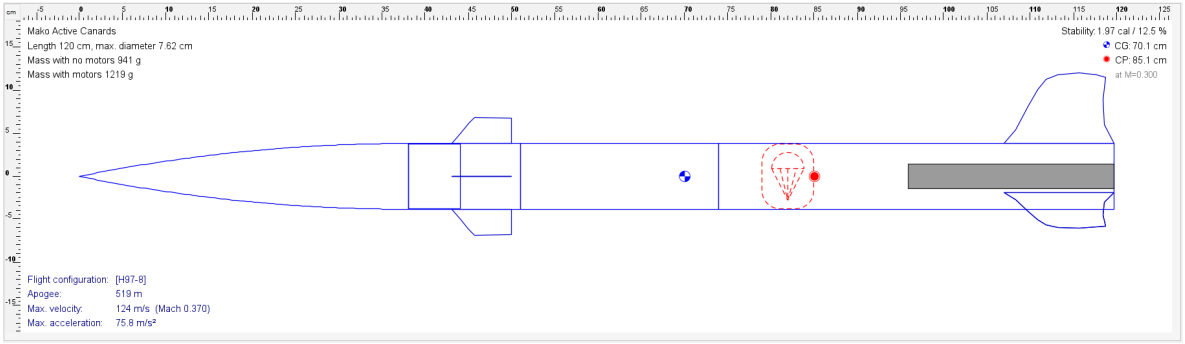

Figure 2: V2 modified Apogee Mako modeled in OpenRocket.

4.2 Firmware Improvements

The original prototype firmware was written primarily as a monolithic program, with most of the sensor, control, and logging logic contained in a single source file. For the V2 design, the firmware was refactored into a more modular structure. Individual device drivers and support libraries were created for the major sensors and peripherals, while shared state and configuration data were organized using C structs and pointers. This structure improves readability, simplifies debugging, and allows individual hardware features to be modified or replaced more easily.

The V2 firmware also separates time-critical steering operations from slower peripheral tasks. Servo control is handled using RP2040 programmable I/O state machines, which generate stable servo pulses independently of the main program loop. Steering updates are performed at approximately 100 Hz using a partial retrieval of sensor data and an interrupt timer, while other functions including complete sensor data retrieval, microSD card logging, GPS parsing, and LoRa transmission are managed through scheduled tasks.

Data logging was substantially improved by using SDIO communication with the microSD card through a PIO state machine and the open-source library provided by carlk3. This allows the flight computer to record data at nearly 100 Hz, with most samples spaced approximately 10 ms apart. Each data row is timestamped and written directly to a CSV file on the microSD card during flight.

The logged data include multiple outputs from the BNO055 inertial measurement unit, including linear acceleration, gyroscope readings, magnetometer readings, Euler angles, quaternions, gravity-vector components, and temperature. The BMP390 provides atmospheric pressure and temperature measurements for altitude analysis, while the KX134 high-G accelerometer records acceleration beyond the range of the BNO055. The flight computer also records its internally calculated gravity-vector estimate, allowing the onboard filter behavior to be compared against raw and sensor-fused measurements during post-flight analysis.

4.3 Hardware Improvements

The Mako host airframe required substantial mechanical and electrical redesign due to its increased size relative to the original prototype. In the first prototype, the control fins were mounted directly to the servomotor horns. For the larger Mako airframe, Connor Reed designed dedicated servo-housing internals with mechanical linkages between the servos and control fins, as shown in Figure 4. This arrangement provided a more robust steering mechanism suitable for the larger vehicle geometry and allowed the flight computer, servo housing, linkage system, and control fins to be integrated into a removable forward fin-canister assembly shown installed on the Mako rocket in Figure 5.

The servo housing was installed inside a cardboard rocket body tube and clamped in place using wooden retention plates. A new modular avionics stack was then designed around CNC isolation-routed circuit-board plates that mechanically support the sensors, flight-computer electronics, and inter-board connections. These circuit boards were designed by creating KiCad schematics and footprints and then fabricated using CNC isolation routing, as shown in Figure 6. The boards were then populated with their respective components and assembled with screw-together brass standoffs to form a rigid but reconfigurable flight-computer structure, as shown in Figure 3 and Figure 7. Pin headers and header extensions create electrical buses between plates, allowing the assembly to be unscrewed, disassembled, reassembled in different configurations, and expanded with additional plates as development progressed. This modularity was used later in development when a dedicated high-g accelerometer plate was added to the stack. Both 1 cm and 2 cm standoffs were used to accommodate different component heights and plate configurations, with header extensions bridging the larger gaps when 2 cm standoffs were installed. Screw terminals on the stack connect to a wiring harness that supplies servo power and pulse-control signals to the steering mechanism. The schematic remained consistent enough between the V1 point-to-point wired computer and the V2 CNC-machined board stack that the same avionics firmware remains compatible with both systems. An Adafruit lithium-battery charging module was also added to a dedicated plate in the computer stack to support recharging the replaceable battery.

The completed avionics stack was fastened to the same wooden retention plates that secure the servo housing, integrating the steering mechanism and flight computer into a single removable assembly. The plastic nose cone slides over the forward side of this fin canister, providing additional physical protection for the electronics during handling, launch preparation, and flight. Removing the nose cone provides direct access to the flight computer USB ports, power switch, and recording switch.

A separate ground station device was created to receive LoRa telemetry from the rocket and display the rocket's current flight phase, GPS fix status, and last known GPS coordinates on an LCD. This information is intended to aid recovery if the rocket travels out of sight during ascent or parachute descent. The coordinate display does not update after every received transmission. Instead, the firmware requires multiple consecutive GPS coordinates to be received within close proximity before updating the displayed position. This filtering step reduces the number of erroneous GPS coordinates shown on the LCD. The ground station can also be connected to a serial terminal over USB to view and log every received transmission. Figure 3 shows a demonstration of the ground station functionality.

Figure 3: An assembled V2 flight computer and ground station.

Figure 4: Steering-fin canister development hardware from multiple views, with background edited away. Left: 3-inch hollow tube servo housing. Center: 66 mm housing with servo linkage and fin. Right: V2 flight computer attached to the 3-inch servo housing..

Figure 5: V2 modified Apogee Mako rocket with flight computer installed, nose cone removed.

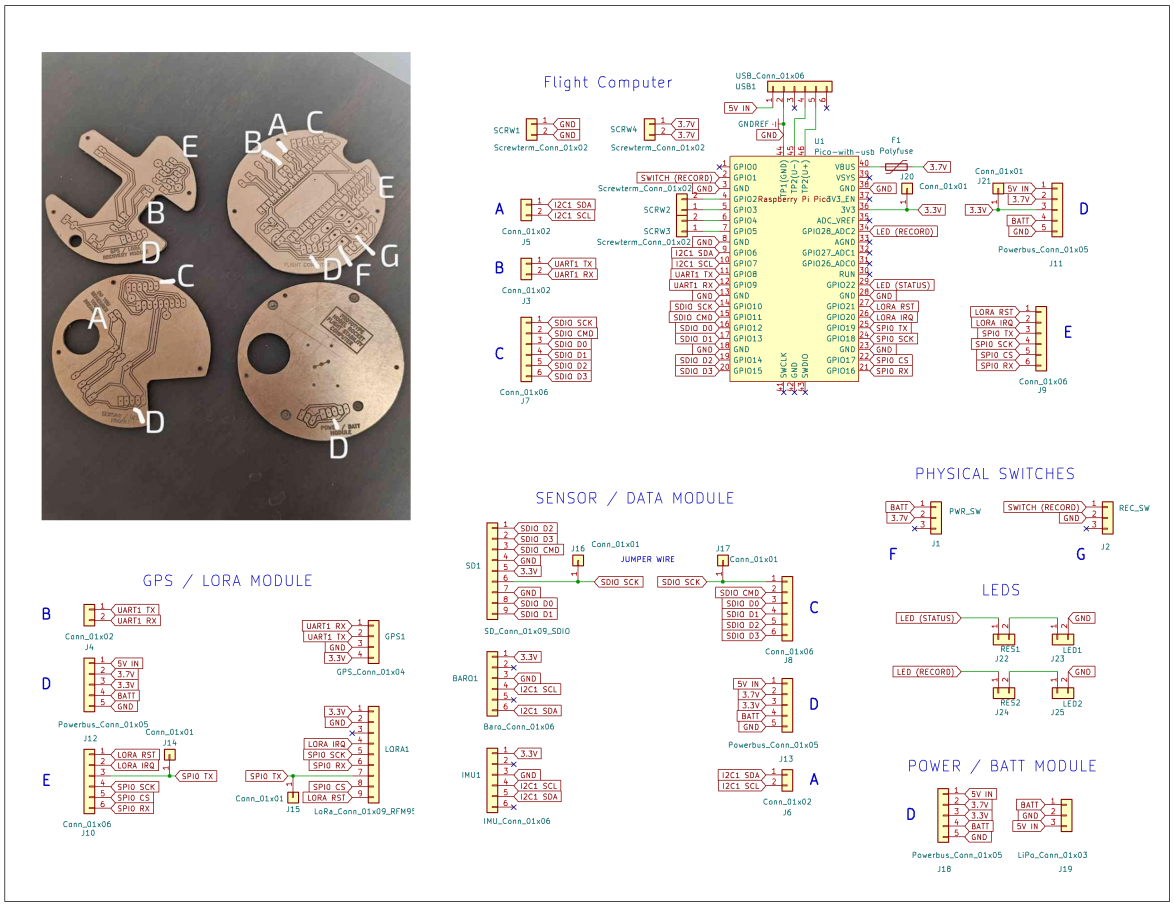

Figure 6: CNC isolation-routed circuit boards made using KiCad and Autodesk Fusion 360, with bus labels for a KiCad schematic.

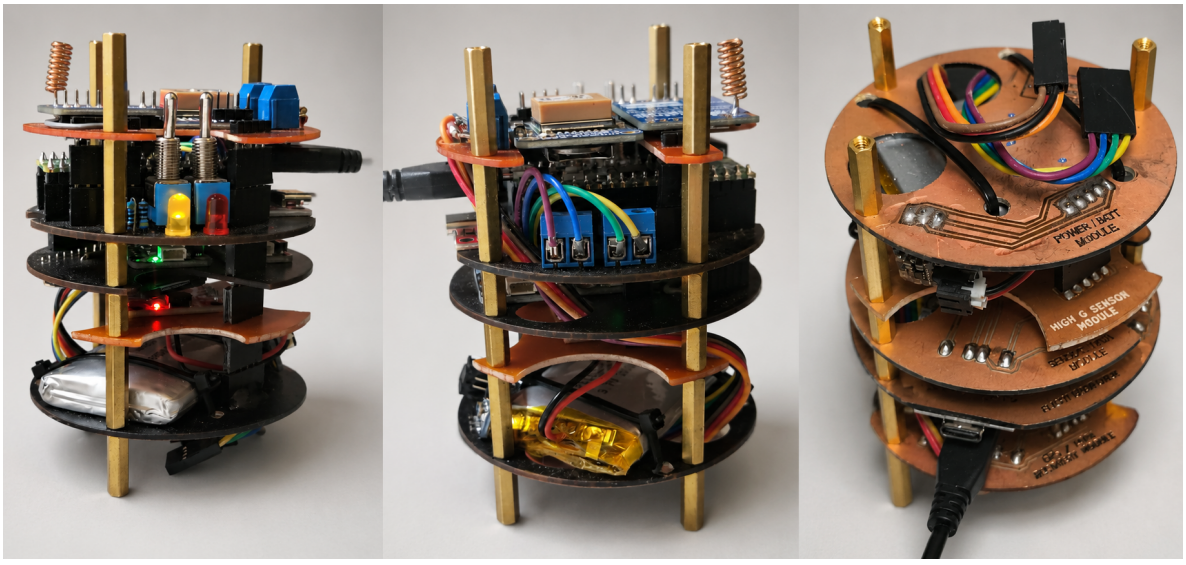

Figure 7: V2 flight computer from multiple views, with background edited away.

4.4 Gravity-Vector Steering and Filter Implementation

4.4.1 Linear Steering Using a Gravity Vector

The steering method used in this project relies on a gravity-vector estimate as the primary attitude reference. This vector indicates the direction of gravity in the rocket body frame and is used to determine how far the vehicle has tilted away from a vertical ascent orientation. The gravity vector is not measured directly. Instead, it is estimated through sensor fusion using gyroscope and accelerometer measurements, and may also include magnetometer data depending on the filter implementation.

Most prototype flights conducted for this research used the proprietary gravity-vector output provided by the onboard Bosch BNO055 inertial measurement unit for steering. This gravity vector is reported in $\mathrm{m/s^2}$. When the rocket is held vertically, the sensor ideally reports approximately $9.81\,\mathrm{m/s^2}$ on the body-frame z-axis and approximately zero on the x- and y-axes. As the rocket tilts, the z-component decreases while the x- and y-components increase according to the direction and magnitude of the tilt.

The initial steering implementation used this gravity-vector estimate directly. The x- and y-axis gravity components were multiplied by fixed steering gains and converted into servo pulse-width commands. Opposing control fins received equal and opposite commands so that the fin pair generated a restoring moment intended to reduce the measured tilt. This approach produced a simple linear steering law without requiring a full state-space controller or trajectory planner.

The linear steering implementation was intentionally paired with a short control lever arm in the first prototype to reduce the likelihood of overcorrection and oscillation. The lever arm is the distance between the rocket center of gravity and the aerodynamic center of the control fin. For a given fin-generated aerodynamic force, increasing this distance increases the resulting control torque. The first prototype used a canard-based steering arrangement rather than an aft-tail control arrangement, and OpenRocket was used during the design process to evaluate the vehicle geometry and stability. The resulting prototype had an approximate control lever arm of 5 cm and an overall length of 60.1 cm. The second implementation of steering on the Mako airframe had an approximate control lever arm of 22 cm and an overall length of 120 cm. Flight testing showed that both configurations were capable of producing active stabilization corrections during ascent and, under larger control inputs, broad turning behavior.

4.4.2 Implementation of a Gravity Vector Using a Complementary Filter

A limitation of relying on the BNO055 gravity-vector output is the restricted measurement range of its internal accelerometer. In the configuration used for these flights, the BNO055 accelerometer output saturated at approximately $\pm 39\,\mathrm{m/s^2}$, or about $\pm 4g$. This range is insufficient for many model rocket boost phases, where axial acceleration can exceed the sensor limit. When saturation occurs, the sensor data used by the internal fusion algorithm may no longer represent the true acceleration state of the vehicle, producing errors in both logged data and any steering command derived from the reported vector.

Another issue is especially important during powered flight is that an accelerometer does not measure gravity alone. It measures specific force, which includes non-gravitational acceleration caused by thrust, drag, vibration, and vehicle motion. During boost, the measured acceleration vector may therefore be dominated by thrust rather than gravity. If this vector is treated as a gravity reference without qualification, the attitude estimate can be driven toward an incorrect direction.

The rocket application does, however, provide a useful constraint when the vehicle is already initialized near an upright launch attitude. The primary thrust direction is approximately aligned with the rocket body-frame z-axis, while the stabilization objective is to reduce the lateral x- and y-axis components of the gravity-vector estimate. During boost, the accelerometer can no longer be treated as a direct gravity sensor. Although gravity continues to act on the vehicle, the accelerometer measures specific force rather than gravity alone, and the measured vector may be dominated by thrust, drag, vibration, and other non-gravitational effects. If the rocket is already initialized near an upright launch attitude and most of the measured acceleration remains aligned with the body-frame z-axis, the accelerometer measurement can provide only a weak consistency check on the assumed boost geometry, not an independent gravity reference. For this reason, the implemented complementary filter reduces accelerometer correction confidence during high-acceleration phases and relies more heavily

on gyroscope propagation until the measured acceleration becomes more representative of gravity direction.

Gyroscopes are susceptible to integration drift, while accelerometers are susceptible to vibration, sensor noise, saturation, and contamination from non-gravitational accelerations during powered flight. A complementary filter can combine these measurements by using the gyroscope for short-term attitude propagation and the accelerometer as a long-term correction reference. In this project, the gravity-vector estimate is propagated by gyroscope integration and corrected using a confidence-weighted gravity candidate derived from the high-G accelerometer. During high axial or lateral acceleration, the accelerometer correction is reduced to prevent thrust-induced acceleration from being interpreted as a change in gravity direction.

The bias-corrected gyroscope and high-G accelerometer measurements are defined as

$$\boldsymbol{\omega}_k = \boldsymbol{\omega}_{\mathrm{raw},k} - \mathbf{b}_\omega, \tag{1}$$

$$\mathbf{a}_{H,k} = \mathbf{a}_{H,\mathrm{raw},k} - \mathbf{b}_H. \tag{2}$$

The accelerometer-derived gravity candidate is computed from the normalized negative acceleration vector,

$$\mathbf{z}_{g,k} = -\frac{\mathbf{a}_{H,k}}{\|\mathbf{a}_{H,k}\|}. \tag{3}$$

The stored gravity-vector estimate is maintained in units of $\mathrm{m/s^2}$, but the filter operates on the corresponding unit vector,

$$\hat{\mathbf{u}}_{g,k} = \frac{\hat{\mathbf{g}}_{b,k}}{\|\hat{\mathbf{g}}_{b,k}\|}. \tag{4}$$

The correction error is computed using the cross product between the current gravity estimate and the accelerometer-derived gravity candidate,

$$\mathbf{e}_k = \hat{\mathbf{u}}_{g,k} \times \mathbf{z}_{g,k} \tag{5}$$

The corrected angular rate is then

$$\boldsymbol{\omega}_{c,k} = \boldsymbol{\omega}_k + K_p c_{a,k} \mathbf{e}_k. \tag{6}$$

The gravity direction is propagated using the corrected angular rate,

$$\dot{\hat{\mathbf{u}}}_{g,k} = -\boldsymbol{\omega}_{c,k} \times \hat{\mathbf{u}}_{g,k}. \tag{7}$$

$$\mathbf{v}_{k+1} = \hat{\mathbf{u}}_{g,k} + \dot{\hat{\mathbf{u}}}_{g,k} \Delta t. \tag{8}$$

$$\hat{\mathbf{u}}_{g,k+1} = \frac{\mathbf{v}_{k+1}}{\|\mathbf{v}_{k+1}\|}. \tag{9}$$

Finally, the unit-vector estimate is converted back into physical units,

$$\hat{\mathbf{g}}_{b,k+1} = g \hat{\mathbf{u}}_{g,k+1}. \tag{10}$$

The accelerometer correction is scaled by a confidence value $c_{a,k}$, which is computed from the measured axial and lateral acceleration components. This confidence term reduces the influence of the accelerometer when the measured acceleration is likely to be dominated by thrust, vibration, or lateral vehicle motion rather than gravity. The body-frame thrust axis is assumed to be aligned with the z-axis. Therefore, the axial and lateral acceleration magnitudes are defined as

$$a_{\mathrm{ax},k} = |a_{H,z,k}|, \tag{11}$$

$$a_{\mathrm{lat},k} = \sqrt{a_{H,x,k}^2 + a_{H,y,k}^2}. \tag{12}$$

The clamp function used by the confidence model is

$$\mathrm{clamp}_{[0,1]}(x) = \min\left(\max(x, 0), 1\right). \tag{13}$$

The axial confidence term decreases linearly as the measured axial acceleration increases above a selected threshold. The normalized axial acceleration term is

$$u_{\mathrm{ax},k} = \mathrm{clamp}_{[0,1]}\left(\frac{a_{\mathrm{ax},k} - a_{\mathrm{ax},0}}{a_{\mathrm{ax},1} - a_{\mathrm{ax},0}}\right). \tag{14}$$

The axial confidence is then computed as

$$c_{\mathrm{ax},k} = 1 - (1 - c_{\min})\, u_{\mathrm{ax},k}. \tag{15}$$

The value $c_{\min}$ prevents the accelerometer confidence from being driven to zero solely because of high axial acceleration. This is useful for rocket flight because high acceleration along the body z-axis may correspond to normal upright thrust rather than an invalid lateral gravity estimate.

A separate tilt gate is used to determine when lateral acceleration should strongly reduce accelerometer confidence. The tilt gate is defined as

$$\gamma_k = \mathrm{clamp}_{[0,1]}\left(\frac{a_{\mathrm{ax},k} - a_{\mathrm{gate},0}}{a_{\mathrm{gate},1} - a_{\mathrm{gate},0}}\right). \tag{16}$$

When axial acceleration is low, γ_k is near zero and lateral acceleration is not strongly penalized. When axial acceleration is high, γ_k approaches one, causing large lateral acceleration to reduce confidence more aggressively.

The raw lateral confidence term is defined as

$$c_{\mathrm{lat,raw},k} = \begin{cases} 1, & a_{\mathrm{lat},k} \le a_{\mathrm{lat,th}}, \\ \exp\left[-\frac{1}{2}\left(\frac{a_{\mathrm{lat},k} - a_{\mathrm{lat,th}}}{\sigma_{\mathrm{lat}}}\right)^2\right], & a_{\mathrm{lat},k} > a_{\mathrm{lat,th}}. \end{cases} \tag{17}$$

The final lateral confidence term blends between full confidence and the raw lateral penalty according to the axial tilt gate,

$$c_{\mathrm{lat},k} = (1 - \gamma_k) + \gamma_k c_{\mathrm{lat,raw},k}. \tag{18}$$

The total accelerometer confidence is then

$$c_{a,k} = c_{\mathrm{ax},k} c_{\mathrm{lat},k}. \tag{19}$$

The resulting confidence value $c_{a,k}$ is the scalar term used in the corrected angular-rate expression in Eq. 6.

$$\boldsymbol{\omega}_{c,k} = \boldsymbol{\omega}_k + K_p c_{a,k} \mathbf{e}_k. \tag{20}$$

The filter parameters were selected empirically through bench shake tests and revised after observing flight behavior. The values in Table 1 should therefore be interpreted as representative tuning values used during the later Mako flight-test period, rather than as a single fixed parameter set used for every flight. In particular, the upper axial-confidence threshold $a_{\mathrm{ax},1}$ was adjusted during testing; flight 8 used a reduced value, while later code revisions used a larger value closer to $32g$. Because the exact flight-specific firmware configuration was not fully fixed across these launches, the table reports the relevant tuning range where appropriate.

As a result, the filter behaves more like an accelerometer-corrected complementary filter during low-acceleration conditions and more like a gyroscope-propagated attitude estimate during high-acceleration boost conditions.

Table 1: Implemented complementary-filter tuning and confidence parameters used for the Mako flight analysis.

Parameter	Implemented value	Description
K_p	2.0	Proportional gain applied to the accelerometer correction error.
$a_{\mathrm{ax},0}$	$12.0\,\mathrm{m/s^2}$	Axial acceleration threshold where axial confidence reduction begins.
$a_{\mathrm{ax},1}$	Flight-dependent; approximately $6g$ in reduced-threshold testing and $32g$ in later code revisions	Axial acceleration value where the axial confidence reaches its minimum value. This value was adjusted during flight-test tuning and should not be interpreted as fixed across all Mako flights.
$c_{\min}$	0.20	Minimum axial confidence retained under high axial acceleration.
$a_{\mathrm{gate},0}$	$12.0\,\mathrm{m/s^2}$	Axial acceleration threshold where lateral-acceleration gating begins.
$a_{\mathrm{gate},1}$	$20.0\,\mathrm{m/s^2}$	Axial acceleration value where the lateral-acceleration gate reaches full strength.
$a_{\mathrm{lat,th}}$	$3.0\,\mathrm{m/s^2}$	Lateral acceleration threshold below which no lateral confidence penalty is applied.
σ_{lat}	$1.0\,\mathrm{m/s^2}$	Width parameter controlling how sharply lateral confidence decreases above the threshold.

4.4.3 Comparing Gravity Vectors

The complementary filter described above was implemented in C and executed as part of the real-time steering update. The steering routine runs approximately every 10 ms, allowing the gravity-vector estimate and servo commands to be updated at approximately 100 Hz. At the current maximum expected speed of the Mako airframe, approximately Mach 0.4, this update interval corresponds to about 1.37 m of vehicle travel between steering calculations. Since the modified Mako airframe is approximately 1.20 m long, the flight computer updates the steering solution on the order of once per vehicle length traveled at maximum expected speed. This provides a useful practical basis for the selected 100 Hz steering-update rate. To evaluate the filter before launch, the rocket was manually rotated about multiple body axes while recording both the BNO055-provided gravity vector and the gravity vector computed internally by the flight computer firmware. A representative comparison is shown in Figure 8. The BNO055 gravity-vector output is not treated as a perfect reference; however, bench testing showed that it provided a useful baseline for evaluating whether the custom filter produced a comparable body-frame gravity estimate during low-acceleration motion.

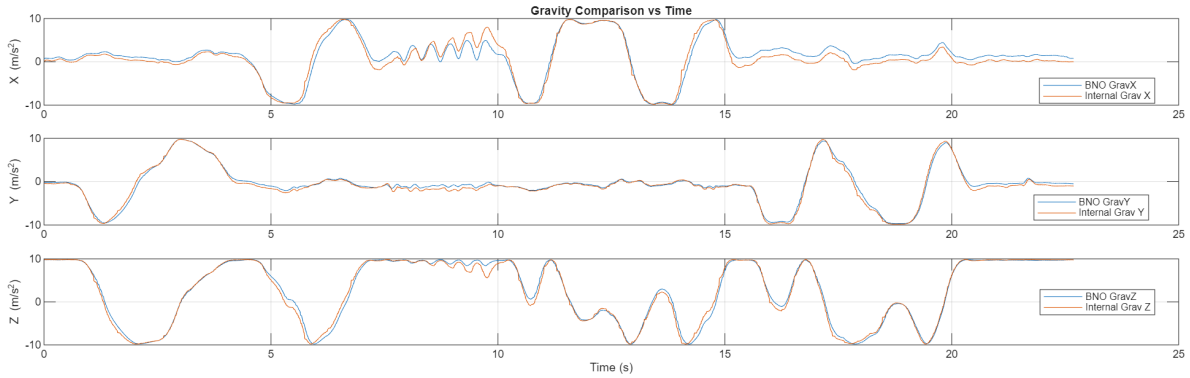

Figure 8: Gravity-vector comparison between the BNO055 output and the real-time flight computer firmware estimate during manual test movements.

4.4.4 Gravity-Vector Behavior During Flight

During launches using the complementary filter, accelerometer confidence was reduced during high-acceleration phases. This was done because vehicle acceleration can cause the accelerometer measurement to be dominated by thrust, drag, vibration, and other non-gravitational specific forces rather than gravity. Under these conditions, the accelerometer is not a reliable direct gravity reference. Reducing accelerometer confidence during boost causes the filter to rely more heavily on gyroscope propagation until the measured acceleration returns to a range where accelerometer-derived correction is more likely to represent the gravity direction.

The Mako flight series provided several useful comparisons between passive flight behavior, BNO055-based steering, custom-filter steering, and fin-locked control testing. Mako flight 1 carried the flight computer with the electronics powered off and visibly veered shortly after takeoff while still under boost. Mako flight 2 used BNO055-based active steering from the same launch pad and appeared straighter in launch video than flight 1. Mako flight 3 was launched at an angle using BNO055-based steering and appeared to make an S-shaped correction toward a more vertical trajectory, with a slight initial overshoot. Mako flight 4 was launched upright using BNO055-based steering and appeared to fly very straight, although the recovery system tangled and the rocket experienced a hard landing. Mako flight 5 used BNO055-based steering on an H97 motor and was completed as a successful Tripoli Level 1 certification flight.

Flights 6 through 10 used versions of the custom complementary filter in the flight computer. Mako flight 6 was launched upright with steering enabled and appeared to fly very straight. Mako flight 7 was launched at an estimated angle of approximately 10° to 15° with steering enabled and appeared to continue relatively straight along that angled trajectory. Mako flight 8 was launched mostly vertical with steering enabled and appeared to fly relatively straight. Mako flight 9 was flown with electronics enabled but the fins locked in place; it veered shortly after takeoff while still under boost. Mako flight 10 was launched at an angle with steering enabled and appeared to fly relatively straight. These visual observations are based on launch-video review rather than calibrated video tracking or photogrammetric reconstruction.

Data collection was limited before Mako flight 6, but a useful quantitative analysis of the custom filter comes from Mako flights 6, 7, 8, and 10. These flights provide a useful comparison because flights 6 and 8 were launched mostly upright, while flights 7 and 10 were launched at a visible angle. Figures 9 through 14 compare the BNO055 gravity-vector output, the internally calculated gravity-vector estimate, and the KX134-derived acceleration candidate for flights 8 and 10. In these comparisons, the KX134-derived acceleration candidate refers to the normalized high-G accelerometer vector used as the accelerometer correction direction in the filter, not to an independently validated gravity measurement.

During the labeled launch intervals, the internally calculated gravity vector and the BNO055 gravity-vector output remained broadly aligned in both flights. This suggests that, during the initial powered portion of the analyzed flights, the custom filter produced a gravity-vector estimate broadly similar to the BNO055 estimate. However, the KX134-derived acceleration candidate diverged from both gravity-vector estimates during launch. This behavior is expected because the high-G accelerometer measurement during boost is dominated by thrust, vibration, drag, and other non-gravitational specific forces rather than gravity alone.

The comparison between flights 8 and 10 suggests that launch attitude affected the acceleration-candidate behavior. In the mostly upright flight 8, the acceleration candidate separated from the gravity-vector estimates by a smaller amount during the launch interval. In the more angled flight 10, the acceleration candidate separated more strongly, reaching a larger angular difference from both the BNO055 and internal estimates. This is consistent with the physical expectation that, when the rocket is launched at an angle, the thrust-dominated body-axis acceleration is less aligned with the gravity direction. Under those conditions, accelerometer correction must be treated carefully because it can bias the filter toward the body-axis acceleration direction rather than the true gravity direction.

The largest divergence between the BNO055 gravity-vector output and the internal estimate occurred primarily after the powered launch interval, during low-acceleration coast, freefall, and vehicle reorientation under a parachute-assisted descent. This indicates that the filter limitation is not restricted to thrust-dominated boost. The accelerometer is also a weak gravity-reference correction source during near-ballistic flight, where the measured specific force may be small or dominated by rotation, aerodynamic loading, vibration, or recovery events. The flight data therefore support a phase-dependent interpretation: accelerometer correction is most useful during preflight and other

stable low-dynamic conditions, but should be reduced during both high-acceleration boost and low-acceleration freefall or coast.

The most likely cause is not a single gain value, but the validity of the accelerometer correction across different flight phases. During boost, the high-G accelerometer measurement is strongly affected by thrust, drag, vibration, and other non-gravitational specific forces. During coast and freefall, the measured specific force may become small or dominated by rotation, aerodynamic loading, or recovery-related motion. In both cases, the accelerometer-derived candidate may no longer provide a reliable gravity-reference correction. The filter must therefore determine not only how strongly to weight the accelerometer, but also when the accelerometer measurement should be considered physically meaningful for gravity-vector correction.

Additional comparison of Mako flights 6 and 7 supports the phase-dependent interpretation of the filter behavior. These flights used higher-power H97 motors, while flights 8 and 10 used lower-power F67 motors. As with flights 8 and 10, the BNO055 gravity-vector output and internal gravity-vector estimate remained comparatively close during the initial launch interval, while larger disagreement developed later during low-acceleration coast, apogee, and vehicle reorientation. The H97 flights showed more severe estimator disagreement than the F67 flights, suggesting that motor class, vibration, angular motion, and post-burn dynamics affected the filter behavior in addition to launch attitude. This reinforces the conclusion that the accelerometer correction model must be scheduled by flight phase rather than tuned only for a single boost condition.

Several improvements are possible for future versions of the filter. First, additional flight testing can be used to tune the axial and lateral confidence thresholds, the minimum axial confidence value, and the proportional correction gain. Second, magnetometer or heading information could be incorporated to reduce long-term rotation ambiguity about the body z-axis, although this would require careful treatment of magnetic interference and launch-site calibration. Third, the accelerometer model could be revised to account more explicitly for the expected boost condition and other phases, in which large axial acceleration is normal while lateral acceleration is more indicative of tilt, side loading, vibration, or off-axis motion. These changes would allow the filter to preserve the educational simplicity of gravity-referenced steering while improving stability and reliability during powered flight.

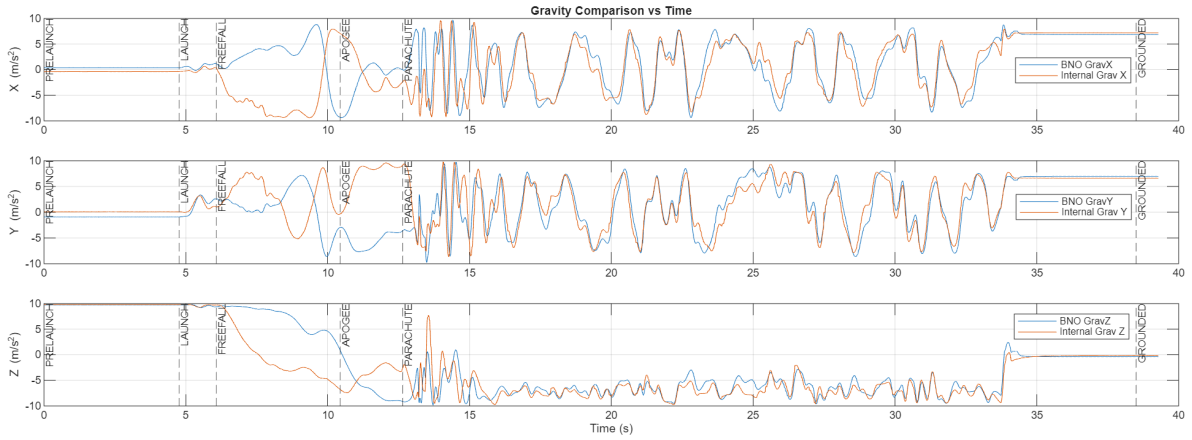

Figure 9: Gravity-vector comparison between the BNO055 output and the real-time flight computer firmware estimate during Mako flight 8.

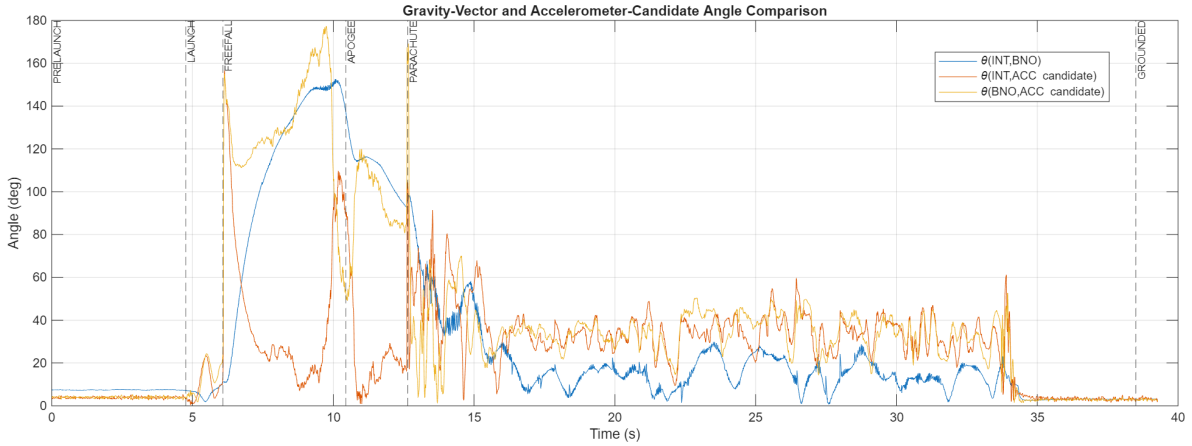

Figure 10: Angle comparison among the BNO055 gravity vector, internal gravity-vector estimate, and normalized KX134-derived acceleration candidate during Mako flight 8.

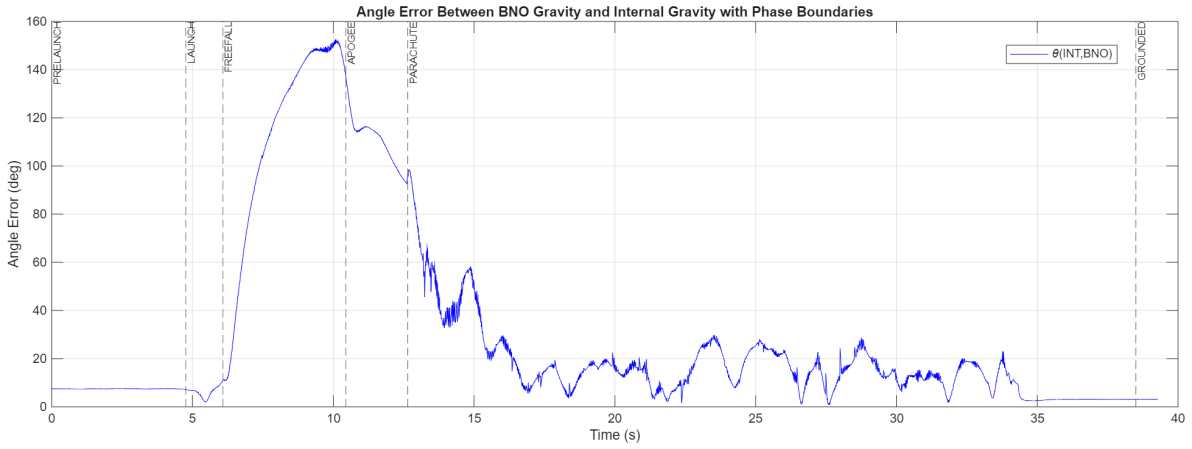

Figure 11: Vector-angle error comparison between the BNO055 gravity-vector output and the real-time flight computer firmware estimate during Mako flight 8.

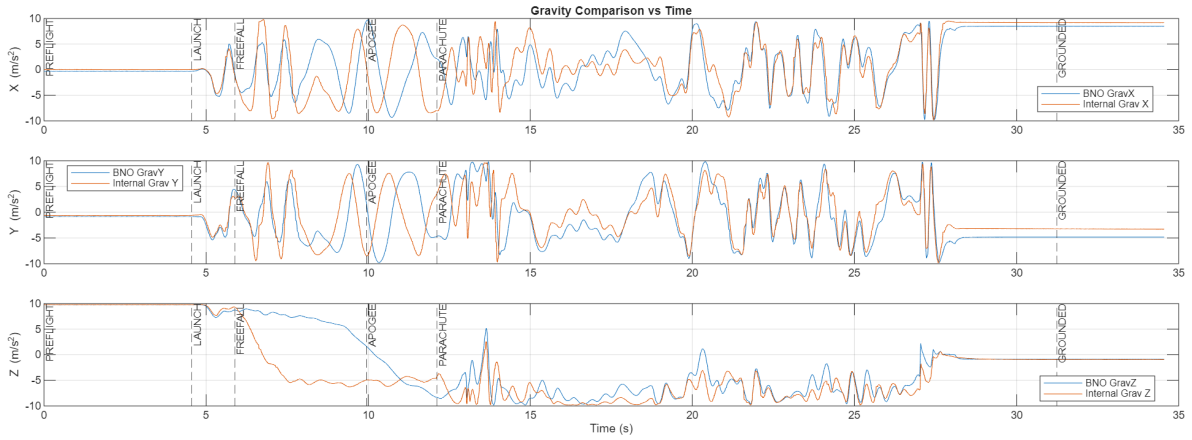

Figure 12: Gravity-vector comparison between the BNO055 output and the real-time flight computer firmware estimate during Mako flight 10.

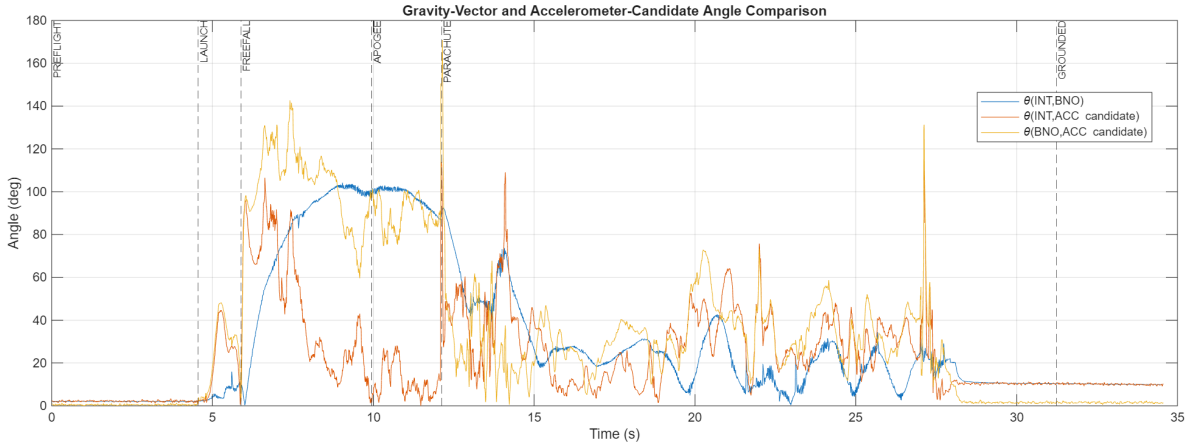

Figure 13: Angle comparison among the BNO055 gravity vector, internal gravity-vector estimate, and normalized KX134-derived acceleration candidate during Mako flight 10.

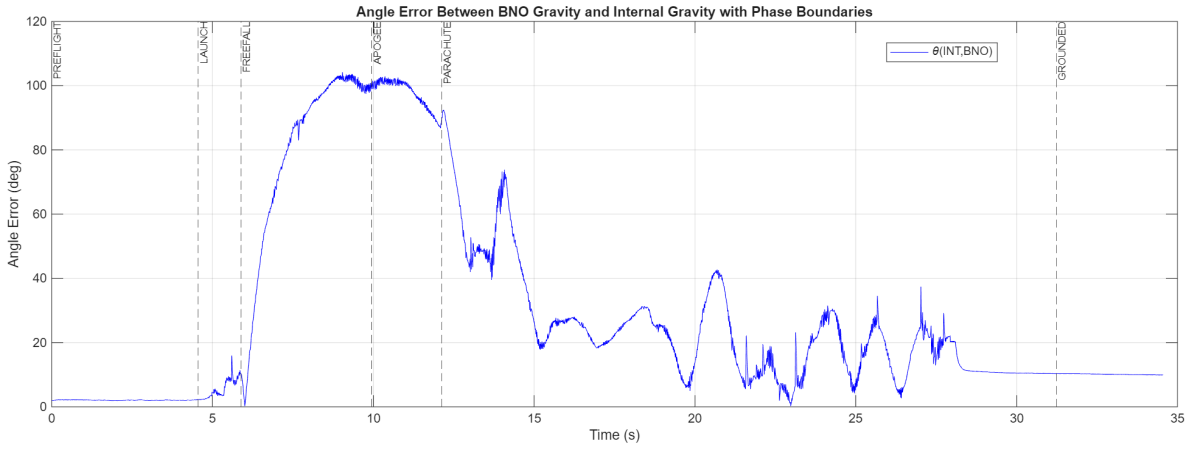

Figure 14: Vector-angle error comparison between the BNO055 gravity-vector output and the real-time flight computer firmware estimate during Mako flight 10.

5 Conclusions

This thesis presented the development of an open-source, hobbyist-scale active-stabilization platform for model rockets. The project was intended to make embedded sensing, flight data logging, and active control experimentation more accessible to students, educators, and hobbyists. Unlike guidance systems that steer toward a target or commanded trajectory, the system developed in this work uses gravity-referenced stabilization to command movable control surfaces to reduce lateral tilt relative to the local gravity-vector estimate during ascent.

The proof-of-concept version demonstrated that a low-cost model rocket could carry an embedded flight computer, survive launch, record usable inertial data, and command active stabilization corrections during flight. This first version used a Raspberry Pi Pico, BNO055 inertial measurement unit, microSD data logging, and servo-actuated fins installed in a modified Estes Green Eggs airframe. Although the early hardware and firmware were intentionally simple, the prototype established the feasibility of the architecture and identified several requirements for later development, including improved data logging, more modular firmware, interrupt-driven steering updates, and more robust airframe integration.

The current V2 implementation extended this work into a larger Apogee Mako airframe with a more capable flight-computer architecture. The revised system incorporated modular CNC isolation-routed circuit boards, a mechanically integrated fin canister, linkage-driven control fins, high-G acceleration sensing, barometric pressure measurement, GPS position reporting, LoRa telemetry, and SDIO-based microSD logging. The firmware was refactored from a monolithic prototype program into a more modular structure using separate peripheral drivers and scheduled tasks. Steering updates were performed at approximately 100 Hz, while flight data were recorded at nearly the same rate for post-flight analysis.

A major focus of the current V2 implementation was the development of a custom gravity-vector filter for use during high-acceleration model rocket flight. The filter combined gyroscope propagation with a confidence-weighted accelerometer correction derived from the high-G accelerometer. This approach was motivated by the limited acceleration range of the BNO055 and by the difficulty of using accelerometer data as a direct gravity reference during boost, when thrust, drag, vibration, and other non-gravitational specific forces can dominate the measurement.

Flight testing showed that the custom complementary filter could run in real time and produce a continuous onboard gravity-vector estimate. The Mako flight data also showed that the filter estimate remained broadly consistent with the BNO055 gravity-vector output during the initial launch interval of the analyzed flights. However, the KX134-derived acceleration candidate diverged from both estimates during boost, especially in the more visibly angled Mako flight 10. This supports the conclusion that the high-G accelerometer measurement was not a reliable direct gravity reference during powered flight, even though the internal filter estimate did not immediately diverge strongly from the BNO055 output. Larger divergence between the BNO055 gravity-vector output and the internal estimate occurred later during coast, freefall, and vehicle reorientation, indicating that accelerometer correction must also be treated carefully during low-acceleration flight. Future versions should therefore use a phase-dependent confidence model, initialize the gravity vector before launch, rely primarily on gyroscope propagation during boost and coast/freefall, and restore accelerometer correction only when the measured acceleration environment is consistent with a stable gravity reference.

Overall, the project demonstrates a working and extensible platform for educational active-stabilization research, but it also shows that reliable gravity-vector estimation across dynamic flight phases remains the central technical challenge. Future work should focus on phase-dependent confidence scheduling, improved filter tuning, better modeling of boost and low-acceleration coast/freefall conditions, and possible incorporation of magnetometer or heading information to reduce long-term rotation ambiguity. Additional work may include model-based boost compensation, in which expected thrust-axis specific force is explicitly accounted for before accelerometer measurements are used for gravity-vector correction. Further mechanical refinement of the fin canister, linkage system, and avionics packaging would also improve reliability and repeatability. These developments would move the platform closer to a robust open-source system for classroom demonstrations, student competitions, and hobbyist experimentation in active model rocket stabilization.

References

[aks21] akshayabali. pico-lora. GitHub repository, 2021. Accessed: 2026-06-06.

[Alt26] Altus Metrum. Altus Metrum: Open Hardware and Software Designs for High Powered Model Rocketry. Project website, 2026. Accessed: 2026-06-06.

[Bar67] James S. Barrowman. The practical calculation of the aerodynamic characteristics of slender finned vehicles. Master's thesis, The Catholic University of America, 1967.

[BPS26a] BPS.space. BPS.space. Project website, 2026. Accessed: 2026-06-06.

[BPS26b] BPS.space. Thrust Vector Control 3D Files. Project/product documentation, 2026. Accessed: 2026-06-06.

[car24] carlk3. no-OS-FatFS-SD-SDIO-SPI-RPi-Pico. GitHub repository, 2024. Accessed: 2026-06-06.

[Hal26] Half Cat Rocketry. Mojave Sphinx. Project website, 2026. Accessed: 2026-06-11.

[MBS25] Conrad Motis, Safina Baiocchi, and Jie Sheng. Development of a servo-based model rocket and flight behavior analysis using imu data. In *2025 IEEE 16th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2025.

[MHP08] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, 2008.

[Nis13] Sampo Niskanen. Openrocket technical documentation. Technical report, OpenRocket, 2013.

[Por14] Portland State Aerospace Society. PSAS Flight Computer: av3-fc. GitHub repository, 2014. Accessed: 2026-06-06.