Model Rocketry Telemetry and Launch Control ECE 568 Spring 2025 Research Project Report

Colin McKinney
Team Leader
mckinn88@purdue.edu

Chase Engle
engle29@purdue.edu

Chris Silman
csilman@purdue.edu

Collin Hoffman
cbhoffma@purdue.edu

Cole Thornton
thornt45@purdue.edu

Brandon Crudele
bcrudele@purdue.edu

William Li
li5005@purdue.edu

Abstract—This paper describes our work in developing a pair of embedded systems for use in amateur rocketry: a flight computer and a ground station. The flight computer featured several integrated sensors to measure flight characteristics, atmospheric conditions, and landing location. The ground station featured launch control mechanisms for safety and the ability to display data. We designed the two systems to be linked using the LoRa protocol, allowing for real-time data streaming and commands between both systems. We describe the overall system design and give both qualitative and quantitative evaluations. Our system can be the basis for future work of other rocket enthusiasts or used by fellow educators.

Index Terms—Model rocketry, LoRa, MCU, telemetry

I. Introduction and Motivation

Our project goals were to develop a pair of embedded systems for use in model rocketry. One system is a ground station capable of receiving data from the rocket and sending commands, such as to arm or disarm the ignition system or to launch the rocket. The system onboard the rocket featured several integrated sensors to measure flight characteristics, atmospheric conditions, and landing location. We designed the two systems to be linked using the LoRa protocol, allowing for real-time data streaming or commands between the two systems.

In designing our system, we researched existing commercial solutions. We focus on two: the JollyLogic AltimeterThree and the Multitronix TelemetryPro family of products. The JollyLogic unit is inexpensive (∼$100) but limited in the types of data it collects and displays. It is also only a flight unit and has no capability of controlling rocket ignition. Multitronix units are much more capable and robust, but are also dramatically more expensive (∼$3,000). Our goal was to design a system with capabilities similar to those of the Multitronix unit but at a cost closer to the JollyLogic unit.

The main application for our project is amateur rocketry and its potential uses in education. One team member (McKinney) is an educator and has used model rocketry as part of several courses. Model rocketry is also often used in youth organizations (such as Scouting) and education (such as high school and university physics). Our system can be the basis for future work of other rocket enthusiasts or used by fellow educators.

II. Methodology

A. Safety and Regulatory Requirements

Throughout the project, the team worked under existing regulations that govern the use and operation of model rocket devices and radio transceivers. These include FAA guidelines for Class 1 model rockets, which allow a total mass not to exceed 1500 g, a fuel mass not to exceed 125 g, and construction from frangible materials such as wood and plastic (FAA 7400.2 31-1). These regulations also dictate requirements when operating in the proximity of an airport; this was especially important for our project, since the Purdue airport and nearby airspace are widely used for training. We also were aware of the FCC and international regulations on radio transmissions. The LoRa radio system we used is part of the 33 cm band reserved for scientific use, which does not require specific FCC licensing. However, two members of the team, McKinney and Silman, are licensed amateur radio operators.

With respect to range and launch safety, the team followed common sense guidelines promoted by the National Association of Rocketry.

B. Hardware: Flight

1) Rocket and Motor: We chose an Estes “Big Daddy” rocket kit as our flight platform. This model has a very large fuselage with an inner diameter of 74 mm, giving us a large space for an avionics bay. We built two versions of the rocket, named Pete I and Pete II. Pete I was used for fit testing and completed one launch to test flight characteristics. Based on changes in the design of the avionics bay, we modified the design and construction of Pete II. It features laser-cut wood bulkheads (instead of the thick paper used in Pete I) and integrated wires running from the payload area to the rear bulkhead. These wires are connected internally to the ignition control system and externally to the rocket motor igniter.

For rocket engines, we used several models of solid propellant motors manufactured by Estes. The are encased in rugged cardboard and include a rudimentary thrust nozzle and ejection charge.

The ejection charge detonates after the main propellant load has been depleted, sometimes after a short delay. The pressure from the ejection charge is responsible for forcing the rocket nosecone off, allowing the recovery mechanism to open. Figure 1 shows the nozzle of the engine and the engine encasement. The naming convention used by Estes is of the form XN-D, where X is a letter, N a number, and D a delay time in seconds. The letter and number indicate the class of engine and approximate total impulse, with letters further in the alphabet being larger in size and capable of greater total impulse. During the project, we used A8-3 (for an ignition static fire test, Boiler 1 and 2), C11-3 (for dummy payload launch, Boiler 3), D12-3 (Boiler 4-6), D12-0 (Boiler 7 and 8), and E12-3 engines (Boiler 9).

Fig. 1. Estes D12 engine

The small hole shown in Figure 1 functions as the exhaust nozzle. To ignite the rocket, an igniter is inserted into this hole and held in place with a small plastic plug or with tape.

Due to the large size of the avionics bay, we needed to extend the fuselage to allow enough space for the recovery mechanism. We chose to design a nosecone extension instead of a fuselage extension. The nosecone extension is permanently bonded to the nosecone using a cyanoacrylate adhesive.

Figure 2 shows the finished rocket. The visible red plastic part is the nosecone extension.

2) Flight Computer and Sensors: The flight computer used the Adafruit RP2040 with integrated LoRa radio (see Figure 6). It was linked to several sensors using STEMMA QT connectors, such as a 9-axis degree of freedom sensor, a pressure/altitude sensor, a GPS sensor,

Fig. 2. Pete II configured for launch

and a temperature/humidity sensor. The MCU controlled the ignition relay using a single GPIO pin. Figure 3 shows some of the sensors mounted in a prototype of the avionics bay.

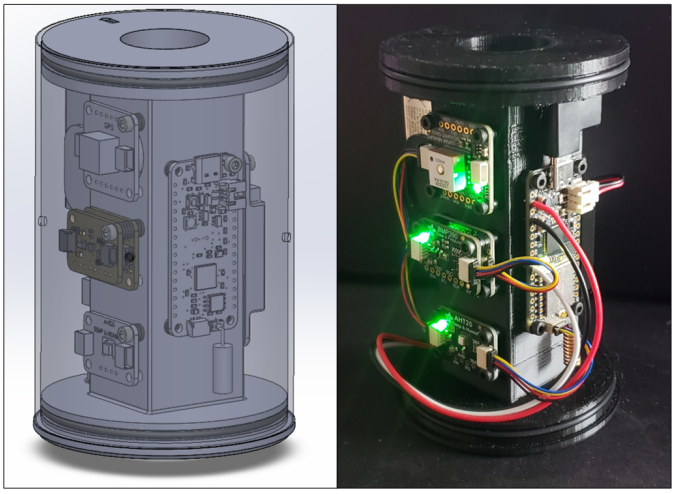

Fig. 3. Avionics bay shown in CAD and as a completed 3D-printed prototype

3) Avionics Bay: We designed a 3D-printable avionics bay assembly to protect the electronics and ensure reliable parachute deployment. The assembly consists of a main bay, two O-rings, sensors, batteries, wiring, a cardboard tube, and a protective sleeve.

As shown in Figure 3, the avionics bay is printed from PLA plastic and reinforced with threaded brass inserts. Sensors are mounted at threaded points along the interior walls, allowing for easy installation and handling during prototype development. To protect the electronics, O-rings are placed at both ends of the bay, forming a seal. A 3D-printed PLA sleeve surrounds the bay, shielding it from the high-temperature gases generated during flight.

To enable parachute deployment, a pass-through channel was integrated into the avionics bay to direct the hot ejection charge gases toward the nosecone. This design reduced the available payload volume, creating challenges for hardware clearance. We addressed this by performing tolerance analysis and iterating through several design versions to achieve a proper mechanical fit. Given the low glass transition temperature of PLA and the thin walls surrounding the channel, a removable cardboard tube was added as thermal insulation. This insert protects the bay from heat damage during deployment and can be easily replaced, allowing the avionics assembly to be reused across multiple flights. This design proved successful, as no thermal or mechanical damage to the avionics bay or flight computer were observed.

4) Ejection Charge and Recovery Mechanism: After engine burnout, the motor ejection charge is triggered to eject the nosecone and allow the recovery system to deploy. The ejection charge gas is very hot. Most model rocket enthusiasts use paper wadding to protect the parachute from this hot gas, and initially we did, too. In the Boiler 4 launch, using Pete I, the ejection charge was not sufficient to force the nosecone off and the parachute out of the fuselage, resulting in the paper wadding catching fire and the plastic parachute melting. For Pete II, we switched to a nylon parachute and a reusable heavy cloth parachute protector.

5) Ignition Control: Ignition of the rocket engine was facilitated by a relay. The relay is controlled by a GPIO pin on the MCU. When triggered, the relay closed a circuit between a 9 V ignition battery and the engine igniter. This caused the igniter to heat and ignite the propellant inside the engine.

After pressing the LAUNCH button on the ground station, an RF signal is sent to the rocket. This signal initiates the closing of the relay to ignite the engine. If the rocket fails to move after 10 seconds, the on-board system moves into the IGNITION FAILURE state. In this state, the relay will open, preventing the rocket from launching unpredictably and ensuring the ignition terminals on the base of the rocket are not energized. The rocket signals the ignition failure to the ground station, which commands both it and the rocket computers to reset. If the rocket does move, the state machine transitions to flight and then eventually recovery.

C. Hardware: Ground

Cost, capability, and performance are important factors in any project development. For the ground station, we chose to use commercial off-the-shelf products from Adafruit, which eased initial development through their I2C STEMMA QT connectors. For the MCU, we chose the Feather RP2040 with RFM95 LoRa Radio since the built-in radio module reduced total footprint. Additionally, the MCU provided enough processing power, speed, and RAM to accomplish our project objectives. Other MCUs, such as the Feather M0 variant with an ATSAMD21G18 chip, were lacking in speed, EEPROM, and a built-in I2C STEMMA QT connector.

1) Ground Computer Sensors: The ground station consists of three total data collection sensors. These sensors are the AHT20 Temperature and Humidity, BMP390 Temperature and Pressure, and the MiniGPS GNSS module. The objective of these sensors was to give the user enough information to determine the current launch site weather conditions. I2C was used for all communications between the data collection sensors and the MCU. The overlap between sensors for their data was useful for dual redundancy and provided an averaged, more accurate display of site information.

2) Ground Computer Interface: The ground station contains a 24-pulse rotary encoder, a key switch, and a button for user input. These can be seen in Figure 5. These components provided ways to control the ground station state. Feedback was given to the user via the two OLED screens attached to the MCU.

The rotary encoder was chosen as it provided an easy way to accept multiple different types of user inputs while still providing a small form factor. It utilized two hardware interrupts to determine direction changes and could be pressed to act as a button. These encoder actions directly controlled the main screen OLED. Other options considered were an attachable keyboard or button-controlled system, however, these would require more hardware for interfacing with the MCU or more GPIO pins than the MCU provided.

The key switch falls into our “safety is paramount” approach, as requiring physical key access helps provide positive control over the launching of the rocket. The button, which is the final user input before initializing a launch, can illuminate to alert the user that the rocket is ready for ignition.

3) Ground Computer PCB and Housing: The ground station printed circuit board (PCB) was designed in KiCAD and significantly reduced the amount of wiring required and provided a small form factor for fitting all the modules into an enclosure. The PCB was a large breakout board of the already existing pins on the Feather.

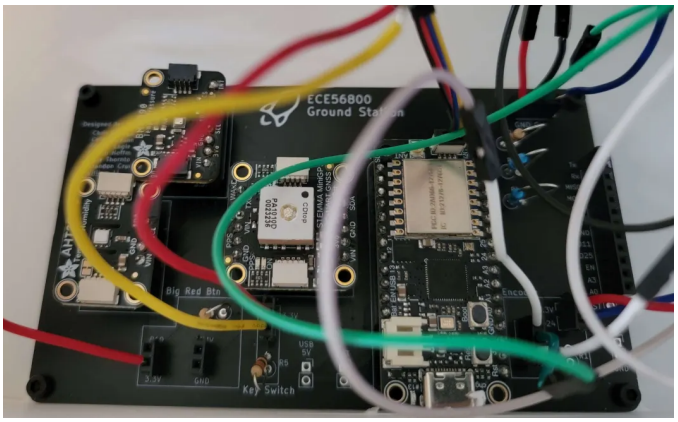

Fig. 4. Inside the Ground Station Enclosure

As shown in Figure 4, all modules (temp, GPS, MCU) were slotted into socket headers, and the components on the lid of the enclosure used spade-quick connectors that fed into the PCB's sockets. These allowed for the relatively easy removal of the lid, and the removal and connection of components.

A 3D-printed ground station enclosure was designed in SolidWorks to house the PCB and user interface components. Design for assembly principles and tolerance analysis were applied to ensure accurate fit and ease of assembly. The OLED screens, LED, rotary encoder, ignition switch, and launch button are mounted to the enclosure panel, where there is adequate access. The panel is then wired to the PCB and inserted into the enclosure base, where it is secured using a friction fit.

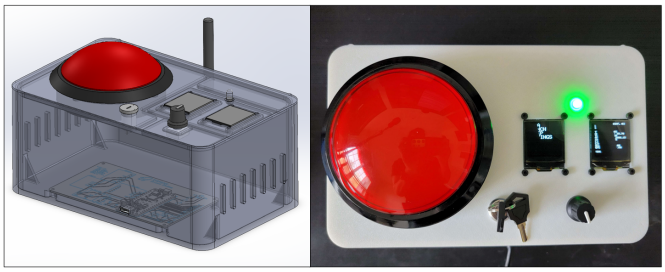

Fig. 5. Ground station shown in CAD and as a completed 3D-printed prototype

D. Radio Frequency (RF) Communication

RF communication for the project used a long-range (LoRa) radio link. LoRa communication uses low-power chirp spread spectrum (CSS) signal modulation with demonstrated effectiveness in a variety of long-range wide-area network (LoRaWAN) and Internet of Things (IoT) applications. The proven IoT applications described in [1] included space-to-ground communication and environmental monitoring. The 915 MHz (33 cm) frequency band was selected for its designation as an international Industrial Scientific and Medical (ISM) band, which does not require an operating license in North America. All project design, test, and evaluation was completed within the continental United States (CONUS) with live-fire rocket and RF link testing in West Lafayette, IN.

A project design requirement was to send and receive data between the rocket flight computer and the ground station for commands and telemetry feed. The project implemented this using twin MCUs (Adafruit Feather RP2040 with 915 MHz RFM95 LoRa Radio), shown in Figure 6, which were integrated into the flight computer and the ground station.

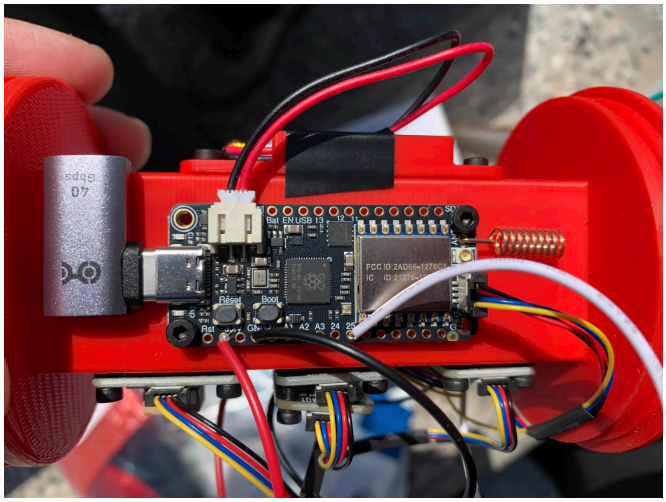

Fig. 6. Adafruit Feather RP2040 RFM95 with LoRa radio and spiral antenna

The flight computer and ground station were designed to use distinct compatible antenna form factors. The flight computer MCU used a simple spring antenna with nominal power and gain values of 5 W and 2.15 dBi. The ground station used a 33 cm wavelength monopole wire antenna.

Implementing [2], the project design paired the flight computer and ground using McCauley's existing RadioHead Arduino code library for the RFM95 radio driver class described in [3]. We also evaluated the RHReliableDatagram manager sub-class described in [4] for addressed, acknowledged, retransmitted datagrams. This solution seemed ideal for critical transmissions where confirmation is required, such as commands from the ground station to the rocket, but was less suited to the real-time requirements of data transmission. This is an area for future work.

E. Software and Control Flow

1) Integrated State Machine: For software, the idea was to use a synchronized state machine on both the ground station and the flight computer. To achieve this, all transitions that occurred on the flight computer were sent to the ground computer through RF to indicate its new state and the status of the sensors. If a state transition was initiated by the ground computer, the flight

computer would always look for that message in a wait state, and upon reception, it would notify the ground computer of the transition. The final transition case was from launch to flight and subsequently flight to recovery. These two transitions were determined by the 9-axis degree of freedom sensor (BNO055). The flight computer would transition to the flight state when the sensor saw a speed greater than noise for at least 3 readings. Similarly, the flight computer would transition to the recovery state when it saw a speed within "noise range" for more than 10 polls. This large, shared state machine made managing information much easier and increased the safety of the launch sequence. A subset of the state machine is given in 7, with the full machine in the appendix.

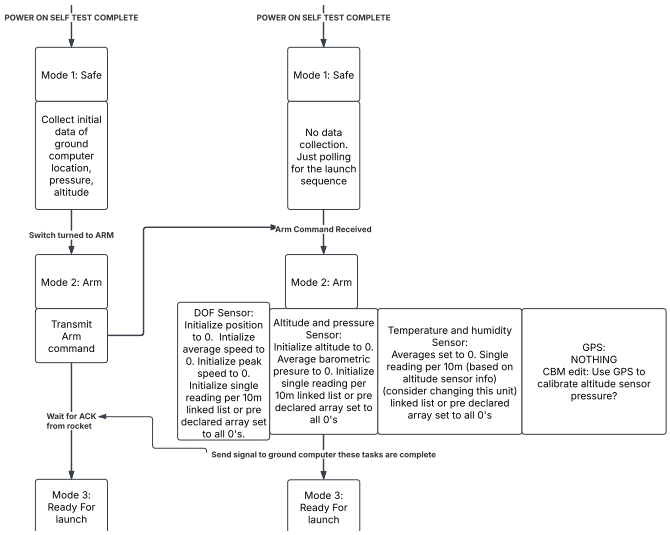

Fig. 7. Subset of Integrated State Machine

2) Sensor Architecture: Because each sensor was worked on one at a time, they each have their own class definition. With more time, the sensors could have used a common class that they all inherited and iterated on. The basic idea for the sensors was to have 3 common functions. The first function, initialize would be called during the bootup sequence and it would make sure that the sensor was connected and could be initialized. The next function, setInitialDataValues would set up the data structures that would be used for collecting information. The common data structures were a float that would contain the maximum or minimum value for that sensor, an array for maintaining a rolling average, and an array for maintaining a data point at each meter. The final common function was collectData, in this function, we would fill in the data structures listed above. The rolling average and peak were found using very typical means. To maintain the data point at each meter, the altitude sensor's current height would be passed to each of the sensor collect data functions and then would be used as the most current index in a data array. At the end of each loop, a class called the RF manager would look into all of these classes' data structures and see if there was anything new to report. If there was, it would attempt to send it to the ground computer so that it could be displayed to the user.

3) UI Architecture: The ground station user interface architecture is made up of two OLED screens and is controlled by a 24-pulse encoder. There were two screen variants in our design, defined by two separate classes, the MainScreen and AuxiliaryScreen. These were created to be as modular as possible; hence, there is no incompatibility or coupling of screen designs if more than one instance of each exists. If the user had more than two screens (easily made possible through the STEMMA QT connector), they could set up additional auxiliary screens to display more data if required for their use case, provided they configured each one to use a separate I2C address.

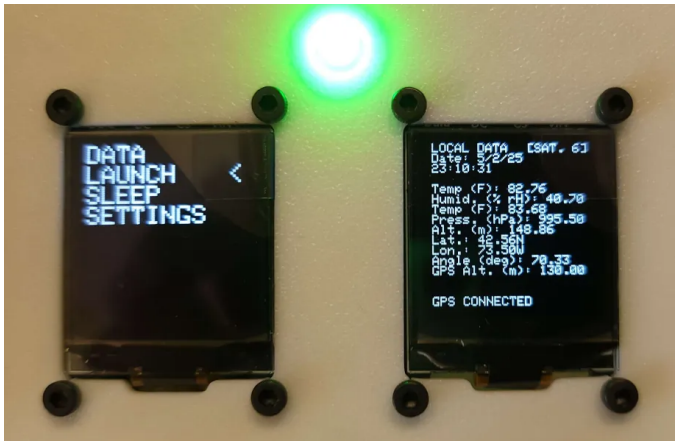

Fig. 8. Ground Station OLED Screens w/ Data

The internal architecture of the screen consists of easy-to-read methods with a simple pointer management system for keeping track of where everything is. Each screen page has a custom and uniquely defined ScreenNavInfo struct that was used to identify the indices of the menu options on the screen and specify the screen that the index should jump to.

All input to the screen was processed in one method, whose inputs leaned on custom enumerations, making it as easy to read as possible. This is all to reduce the amount of code debt in the future.

III. Evaluation

Project design of experiment included a build-up approach of iterative hardware and software design, test, and redesign or debug, presented in Figure 9. Verification and validation of specified or intended behavior was performed before proceeding to follow-up testing. Ground tests, including simulation and modeling, were completed before live-fire ground tests. The rocket air vehicle was obtained and initially tested as original equipment manufacturer (OEM), Estes nominal design, then modified for avionics bay and payload fit. Ground testing of discrete

embedded system components was first accomplished in a bench environment prior to avionics bay integration. Flight computer and ground station functionality were independently verified prior to integration and RF link tests.

Test Point	Description	Remarks	Category
FC.G.1	Flight Computer (FC) operational checkout - ground	• Verify basic functionality of all discrete components: Adafruit microprocessor (MCU), sensors, Stemma QT I2C connections	Hardware/ Software
FC.G.2	FC avionics bay fit check	• Iterative verification/validation of size/fit until redesign suitable for safety of flight test (SOFT) with *BOILER* rocket	Hardware
GS.G.1	Ground Station (GS) operational checkout - Ground	• Verify basic functionality of all discrete components: Adafruit microprocessor (MCU), sensors, Stemma QT I2C connections	Hardware/ Software
RF.G.1	Radio Frequency (RF) link operational checkout - Ground	• Verify basic two-way reliable communication between FC and GS	Hardware/ Software
RF.G.2	Telemetry and sensor polling check	• Evaluate reliable transmission and receipt of FC sensor telemetry data to GS	Software
RF.G.3	GS command and control for rocket ignition - Ground	• Evaluate reliable remote command and control of FC state via GS wireless link (non live-fire)	Software
GS.F.1	GS command and control for rocket ignition - Flight	• Evaluate reliable remote command and control of FC state via GS wireless link (live-fire with *BOILER* rocket) • May combine with FC.F.1 for test efficiency	Hardware/ Software
FC.F.1	FC avionics bay safety of flight test (SOFT)	• Evaluate FC avionics bay design suitability for flight in *BOILER* rocket with dummy load • Iterative verification/validation until redesign suitable for flight with actual payload	Hardware
FC.F.2	FC payload mission	• Evaluate whole-of-system functionality, performance, and safety during mission profile with live and recorded telemetry data from *BOILER* rocket, including launch, flight, and recovery phases	Hardware/ Software

Fig. 9. Tests and Test Conditions Matrix

Data was collected using bench serial monitor output for discrete testing, in-flight telemetry received by the ground station, and comparison ride-along data using a commercial off-the-shelf (COTS) Jolly Logic AltimeterThree unit, which accompanied the rocket flight computer payload. Simulation data was created using OpenRocket v23.09 modeling software and considered truth data for quantitative comparison.

A. Static Fire and Flight Tests

Static fire and flight tests were conducted outdoors in vicinity of West Lafayette, IN and consisted of a total of nine experimental missions. Mission results are summarized in Figure 10.

Date	Mission(s)	Result	Remarks
8 April 2025	BOILER 1 BOILER 2 BOILER 3	SUCCESS FAIL PARTIAL SUCCESS	• Good static fire of B engine using 9V source. • Multiple ignition failures using OEM (Estes) igniters with C engines; good ignition with Apogee ignitor. • Good C engine launch (dummy payload), parachute did not deploy, rocket redesign for body extension.
22 April 2025	BOILER 4 BOILER 5 BOILER 6	PARTIAL SUCCESS FAILURE FAILURE	• Good ignition, mass of rocket/payload impacted apogee, parachute ejection timing, no data recovered. • Ignition failure, cause unknown. • Ignition failure, cause unknown, updated SW for fault detection.
28 April 2025	BOILER 7 BOILER 8 BOILER 9	SUCCESS FAILURE PARTIAL SUCCESS	• Good D engine launch with SW update, on-board rocket data recovered but no telemetry received. • Ignition failure, cause unknown. • Good launch with E engine, no data recovered.

Fig. 10. Mission Log

Deficiencies encountered included multiple ignition failures due to faulty OEM igniters as well as unreliable RF radio link logic, which required troubleshooting and resulted in the addition of fault detection code. While flight computer functionality was verified from on-board post-flight data download, issues remained with ground station reliable capture of telemetry. Rocket vehicle weight due to payload remained a significant constraint on apogee altitude and time remaining after parachute deployment before ground impact.

Overall design objectives were only partially met, as depicted in Figure 11.

B. Physics and OpenRocket Simulations

Based on the first flight test, and after measuring final mass values of the avionics bay, nosecone extension, etc.,

Parameter	Objective(s)	Results	Compliance
Rocket Flight Computer (FC) Control and Telemetry	• Control rocket ignition for launch • Transmit various telemetry data • Minimize size, weight, power, and cost (SWAP-C)	• Processed arm and launch commands from GS using finite state machine (FSM) architecture • Tx GPS position (lat/long) @ 1 Hz (recovery only) • Tx temp (deg F) • Tx humidity (%RH) • Tx baro pressure (hPa) • Tx altitude (m) • Total FC payload weight was causal factor to severely limited apogee and time of flight. • Total FC payload cost of $250 (compare to Multitronix Kate-3 at $1,440)	Met Met Partially Met
Radio Frequency (RF) Data Link	• Establish and maintain a two-way wireless link between FC onboard rocket and remote GS • Ensure reliable data link	• Implemented two-way RF link using Adafruit RFM95W transceivers with 915 MHz band LoRa. • Reliable data link used packet error correction method to minimize telemetry loss (RadioHead Arduino library for reliable datagrams)	Met Partially Met
Ground Station (GS) Command and Control	• Safely command remote rocket launch sequence • Display received telemetry	• Implemented/tested GS with PCB and federated system elements using safety interlocks for command/initiation of remote rocket launch sequence within FSM architecture • Displayed formatted telemetry data on GS miniature screen	Met Partially Met

Fig. 11. Design Objectives Summary

we realized that the rocket mass was significantly higher than our initial estimates and expectations. For example, the dummy weight flight test used two 9 V batteries to increase the rocket mass to 250 g. Our total mass ended up being 490 g, so flight performance from the 250 g test was not going to be useful in predicting future flight values.

Instead of doing another launch with an increased dummy mass, we did a quick estimate of apogee and maximum velocity along with more detailed simulations using OpenRocket. We first used Tsiolkovsky's equation,

$$\Delta v = g I_{sp} \ln\left(\frac{m_{wet}}{m_{dry}}\right), \tag{1}$$

where m_{wet} is the mass of the rocket with fuel, m_{dry} is the mass of the rocket without fuel, g is surface gravity (9.81 m/s^2), and I_{sp} is the specific impulse of the engine. Solid fuel motors such as the D12-0 have an approximate I_{sp} of 80 s, a fuel mass of 23.8 g, and a total mass of 40.5 g. This gives an approximate Δv of 36 m/s, which, assuming an instantaneous impulse, would predict a 66 m apogee. The observed value from the Boiler 4 and 7 launches was on the order of a 22 m apogee and 20 m/s maximum velocity. The large difference between these values is due to drag.

OpenRocket uses a sophisticated numerical differential equations solver. It allows a user to make a schematic representation of both the rocket geometry and mass distribution. These together give a much better estimate of flight characteristics. We started with a schematic of the standard Big Daddy model by Robert Rummel, then modified it to include our nosecone extension and extra payload mass. Figure 12 shows a schematic representation of our rocket.

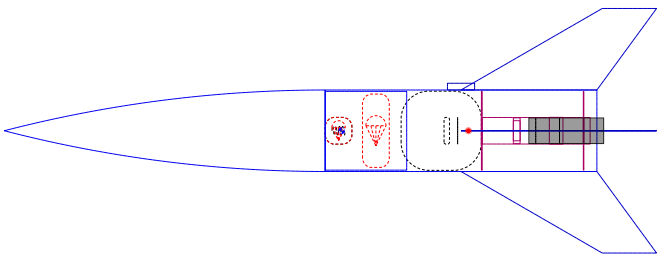

Fig. 12. Schematic representation of the rocket in OpenRocket

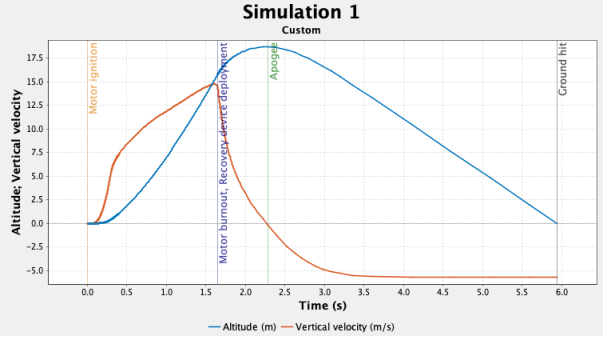

Fig. 13. OpenRocket simulation with a D12-0 engine

Using JollyLogic recorded data from the Boiler 4 launch (which used a D12-3 engine) as reference values, we adjusted OpenRocket simulation parameters such as the drag coefficient. We then ran simulations with a D12-0 engine to predict flight performance before the Boiler 7 launch. Figure 13 shows OpenRocket's simulation for a D12-0 engine.

Figure 14 shows the values predicted by the OpenRocket simulation along with the recorded values from our flight sensors and the JollyLogic unit for the Boiler 4 and Boiler 7 launches.

Mission	Result	Details
Boiler 4	**Partial Success**	Apogee ~25 m. Data collection failure. Parachute deployment occurred too late. Action: Changed motor, improved code and data retention methods.
Boiler 7	**Success**	**Apogee** 16.1 m (flight computer) 22.6 m (Jolly Logic) 18.8 m (OpenRocket) **Peak velocity** 7.51 m/s (flight computer) 19.4 m/s (Jolly Logic) 16.5 m/s (OpenRocket)

Fig. 14. Summary of Data from Boiler 4 and 7

C. Comparison to JollyLogic Altimeter

The JollyLogic is a small integrated sensor system that is easy to attach to any rocket payload. Some limitations to this device are the inability to access the data during the launch. The JollyLogic can be recovered after the launch, and only then can the data be viewed using a Bluetooth connection (AltimeterThree, now discontinued) or on an integrated screen (AltimeterOne and Two). Our embedded system uses RF to send flight data to the ground station. The RF implementation separates the two products and their processes.

Another difference is the quantity of sensors on board the JollyLogic. Their product includes velocity, acceleration, and altitude. As well as these sensors, our design includes temperature, humidity, and GPS. Since our design used multiple discrete components and required a custom avionics bay, the mass and size of our design is much greater than those of the JollyLogic. However, we hope that future work will allow us to decrease the form factor and mass of our solution.

IV. Discussion

A. Limitations of Our Approach

Our approach has some limitations, especially concerning size, weight, power, and cost (SWaP-C), the deployment of the system on the rocket, data reliability and storage, and sensor values.

Due to our system being deployed on small, commercial rockets, SWaP-C considerations are paramount. As discussed during our presentation, the performance of the rocket was hindered by the relatively high mass of the system payload, thus decreasing peak velocity and apogee. Part of this high mass was the need to include a battery onboard to power the system components. Size also played an important role, as an enclosure had to be specially designed to fit inside the model rocket. Thus, this enclosure may not work on other class rockets, which have smaller or larger body diameters. Finally, the cost of our system is significantly higher than that of the commercially available JollyLogic unit, at about 2.5x the cost.

Additionally, our system currently transmits data in real-time from the rocket to the ground station via an RF link, however, this link is not reliable. As a result, the collected data may be incomplete and, due to data not being logged locally, cannot be retrieved later. Thus, the data collected by our system is suitable for certain amateur use cases, but not for scenarios where full and accurate data reporting is critical.

B. Important Lessons Learned

Throughout this design experience, the team learned various practical lessons about the development of complex and connected systems. With our project, a significant amount of time was spent debugging various issues with the sensor and RF system; however, it was not always clear what the root cause of these issues was. The team learned that this is incredibly common in even medium-sized engineering projects such as our own, let alone large projects with dozens, if not hundreds, of engineers on staff. We will highlight two famous failures for comparison: the failure of a Patriot missile battery during the first Gulf War (1991) and the failure of the first Ariane 5 launch (1996).

- During the first Gulf War, a US Army Patriot missile system failed to intercept an Iraqi Scud missile due to a floating point rounding error that accumulated due to long operating times. This issue was known before the war and regular resetting of the system was part of

standard procedure. However, the stresses of combat deployment led one crew to overlook this procedure. As a result of the intercept failure, 28 U.S. soldiers were killed.

- The first launch of the Ariane 5 rocket occured at the Guiana launch site in Kourou, French Guiana. The Ariane 5 flight computer improperly reused code from the previous Ariane 4, which had a different launch profile. This resulted in an overflow error in a variable that tracked the vehicle's horizontal velocity. The launch vehicle began to break up from aerodynamic stress about 40 seconds into flight, and was then intentionally detonated by flight controllers.

Our project suffered from similar mistakes. The root cause of failure to acquire data from the Boiler 4 launch was that the rocket never transitioned states from LAUNCH to FLIGHT. This occurred because we forgot to uncomment out a small section of code designed to detect this transition. It needed to be commented out for ground-only testing of the sensor and RF systems (since the rocket would not be moving), but we failed properly configure the code for a live flight before launch. This and subsequent tests revealed other problems, such as the need for detecting ignition failures and the need to be able to recover data manually from the rocket using a serial connection if and when RF transmission failed.

Overall, the experience gave the team a new appreciation for the complexity of real rocketry and spaceflight systems, along with a more nuanced understanding of the circumstances of famous examples of failure.

C. Reflections on Peer Review

The team reviewed the peer review of our final presentation as provided by the teaching team. We definitely concur with the constructive feedback given, but would like to highlight one point as a limitation of our project, and hence a direction for future improvement.

Several of the reviewers commented on the small amount of data given in the presentation. In reality, this was essentially all the data we were able to collect in our testing, due to various combinations of programming and RF issues. We had hoped to be able to present more comprehensive data obtained from a series of launches. We attempted to repeat the Boiler 7 launch several times using the same type of engine, but had a number of ignition failures, possibly due to a depleted ignition battery. This lack of repeated data, and hence our inability to conduct any sort of statistical analysis comparing our data with reference values from our JollyLogic unit or OpenRocket, is very much an area for future improvement.

D. Potential Future Work

As mentioned in previous sections, the final system has several areas of improvement. These areas include SWaP-C, data reliability and logging, sensor accuracy, and safety features.

When designing a rocket telemetry system, especially for smaller model rockets, minimizing the impact of the telemetry system on the rocket is key. With this in mind, the size, weight, and power of the telemetry system will have the most effect on rocket performance and system integration. As previously mentioned, a special enclosure was developed to mount the system inside the rocket. This is not a very modular solution, and decreases the practical usability of our system. In the future, our team would like to develop a PCB with only the necessary components to decrease the size and weight of the system. This will also improve system modularity, as the system will consist of a single PCB as opposed to many components wired together.

The existing RF system does not reliably transmit data in real-time to the ground station. The team would like to fix this issue by both improving the reliability of RF data transfer and logging data to an auxiliary storage unit on-board the rocket. With this approach, even if data is not received at the ground station, there is a local copy available on the rocket.

However, this data is not very useful if it inaccurate. In the future, the team would like to calibrate the sensors to ensure that the collected data matches simulated values and data collected from other sensors, such as the JollyLogic. Ultimately, this leads to a higher quality system that is appropriate for actual usage on rocketry projects.

Finally, the team would like to improve the safety features of the launch system. Currently, only one human operator is required for the rocket to be launched. This poses the hazard of accidental launches, which is dangerous. The team would like to mitigate this issue by requiring that two human operators are involved in the launch sequence. This coordination will ensure that accidental launches do not occur.

V. Team Contributions

- McKinney (Team lead): Regulatory compliance, safety, rocket construction, flight testing, OpenRocket simulations, code editing, document and presentation editing, task assigning, liaison to teaching team.
- Engle: Flight software, sensor class architecture
- Silman (Ground lead): Ground software, UI architecture, custom PCB design
- Hoffman: Flight software, presentation design
- Thornton (Flight lead): avionics bay design, nosecone extension design, ground station enclosure design, CAD
- Crudele: Flight and ground software, flight testing
- Li (RF lead): RF architecture, hardware/software integration
- All team members: writing for presentations and reports, participation in final presentation

VI. Conclusion

Our project largely achieved its goals of developing two embedded systems for use in model rocketry. Multiple tests and careful analysis of the results highlighted programming or design deficiencies, which we addressed with further development and tests. Our latter tests demonstrated the robustness of our improvements, but also identified areas where further development or redesign is warranted. The experience of working on this project gave us a better appreciation and understanding of high-profile failures of rockets due to issues with their own embedded systems.

REFERENCES

[1] A. Maleki, H. H. Nguyen, E. Bedeer, and R. Barton, "A tutorial on chirp spread spectrum modulation for lorawan: Basics and key advances," IEEE Open Journal of the Communications Society, vol. 5, pp. 4578–4612, 2024. doi: 10.1109/OJCOMS.2024.3433502.

[2] "Adafruit Feather RP2040 RFM95." (May 2023), [Online]. Available: https://learn.adafruit.com/feather-rp2040-rfm95/using-the-rfm-9x-radio.

[3] "Radiohead: Rhrf95 class reference." (2025), [Online]. Available: https://www.airspayce.com/mikem/arduino/RadioHead/classRH___RF95.html.

[4] RadioHead: RHReliableDatagram Class Reference. [Online]. Available: https://www.airspayce.com/mikem/arduino/RadioHead/classRHReliableDatagram.html.

Appendix

The order of team members listed was the order they joined the project team. No other meaning from the order should be inferred.

The team is especially thankful to Professor Lon Porter and the Wabash College 3D Printing and Fabrication Lab for helping us fabricate parts.

Project related files are available on Github: https://github.com/cbpmckinney/ece568rockets. This includes design files for the PCB, printed components, OpenRocket simulation, and all code.

Figure 15 shows full version of the state diagram as discussed in the Methodology section.

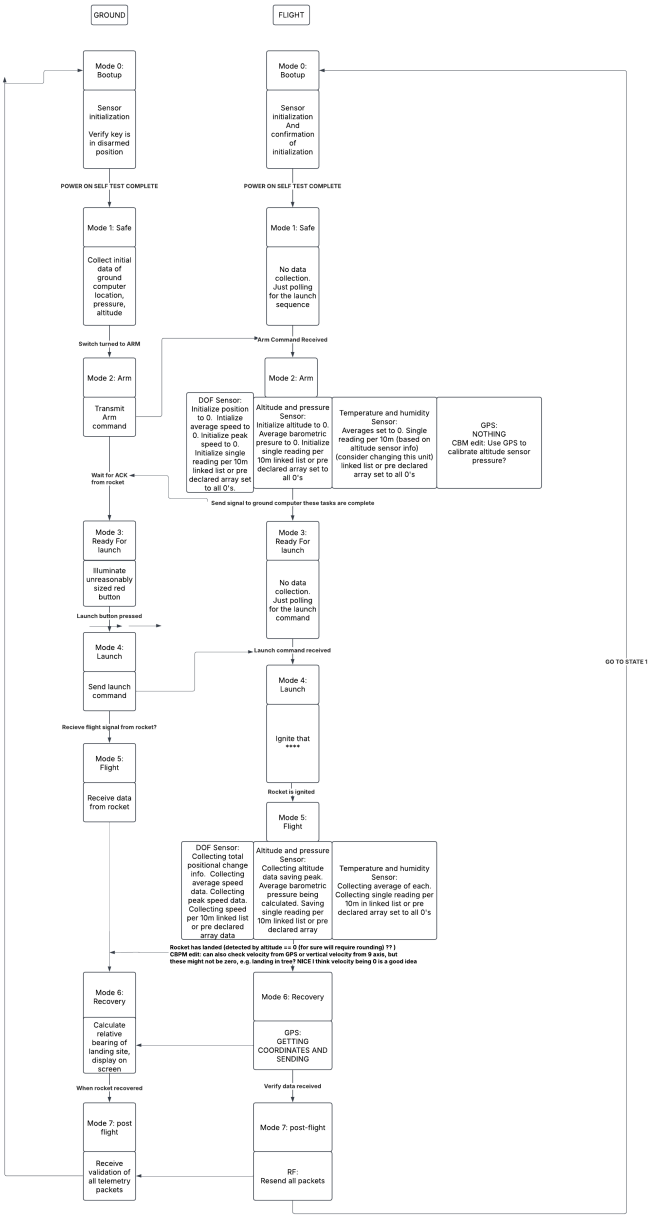

Fig. 15. Integrated state machine