# Implementation and Uses of Telemetry Communications in a Sounding Rocket

*Andrew Darby*[1]
*University of Georgia, Athens, Georgia, 30602*

*Kyle Kim*[2]
*University of Georgia, Athens, Georgia, 30602*

*And*

*Parsa Novin*[3]
*University of Georgia, Athens, Georgia, 30602*

**The University of Georgia's International Rocket Engineering Competition (IREC) Team is developing a live video and data stream telemetry system for use on the 2025 competition rocket. As part of the competition, the "Live Rocket Video Challenge" has two possible categories to gain awards for, including the "Best Live Onboard Video," and "Best Live Telemetry Visualization." This research paper covers the system architecture and hardware used to appropriately design and implement the team's telemetry system, along with PCB manufacturing for electronics, and considerations for the unique challenges provided by limitations due to the nature of the competition. System design using an ESP32 with peripherals utilizing UART and I2C communication, as well as power electronics will be covered. The input of all data is packaged and sent over an RF communication system, then unpackaged and displayed on a custom-developed ground station.**

## I. Nomenclature

*COTS = commercial-off-the-shelf*
*Flight Computer = Missileworks RRC3*
*ESRA = experimental sounding rocket association*
*EGSE = electrical ground support equipment*

## II. Introduction

The University of Georgia's Rocketry Team designs and manufactures high-power rockets for the International Rocket Engineering Competition (IREC), an annual competition with multiple divisions for students. With the upcoming 2025 competition being the second time our Rocketry Team will compete, it has become apparent there is a need for data and performance metrics in order to validate the team's design process for future rockets. In order to meet this need and provide valuable data that can be used to iteratively design and improve upon future rockets, the team has incorporated live telemetry and data acquisition for a limited number of flight performance metrics.

Additional factors into the team incorporating telemetry include the ESRA "Live Rocket Video Challenge," which presents teams with the opportunity to win an award for either or both live video broadcasted from aboard the rocket, and rocket telemetry.

---

[1] Undergraduate Student, College of Engineering, AIAA Student Member (ID: 1810179)
[2] Undergraduate Student, College of Engineering, AIAA Student Member (ID: 1810191)
[3] Undergraduate Student, College of Engineering, AIAA Student Member (ID: 1810339)

The team's telemetry system collects data from an Inertial Measuring Unit, one of the COTS flight computers (Missileworks RRC3), and BTU.656 protocol data from a COTS Runcam Nano 2 whose analog NTSC video data has been digitized and serialized for microcontroller readability. Data is packaged and broadcasted to a radio modem with a linearly polarized dipole antenna setup, which is then received by the ground station and displayed on a custom-made UI, of which the screen it is displayed on is shared with ESRA judges for the competition with a standard HDMI connection.

## III. Design

### IV. System Design

The overall design process followed a functional decomposition type of organization, where broad ideas were gathered for what data points would be both valuable and possible to collect, then broken down into the most essential components that would be needed for acquisition and interfacing with the system. Team members identified the most basic data points of attitude, velocity, acceleration, and most namely altitude to be necessary for flight analysis, and as a complementary data set for the payload experiment.

The majority of sensors were not decided out of best fit for the task, but instead out of what components were already present within the team's inventory to save costs due to the project's inherent risk with the uncertainty of the probability of success due to the lack of experience the team had with such an undertaking. The team started to develop concepts of what the system would look like, taking into account communication between the devices in our system, with the first iteration shown below as a block diagram.

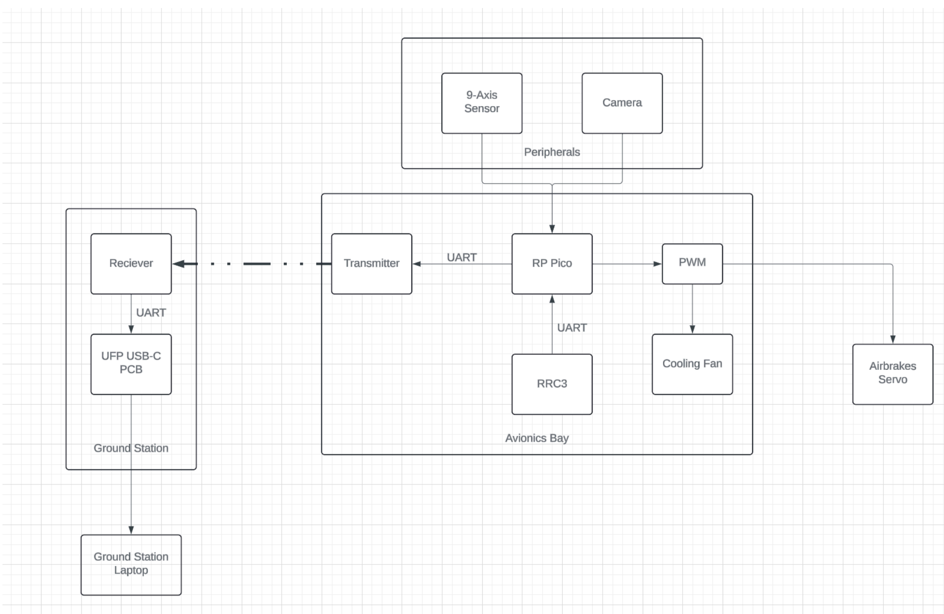


*Figure 1: Initial System Overview*

The first system incorporated a Raspberry Pi Pico (2040), which was once again chosen due to the excess inventory the team had from previous endeavors, but once testing started, the team had difficulties implementing the microcontroller as it became quickly apparent the dedicated UART ports on the pico were not numerous enough to facilitate efficient testing with serial data streams from the EGSE, RRC3, and RFD900X transmitter, therefore the decision was made to switch to an ESP32 based microcontroller. Factors taken into account during the switch included processing speed and processing cores, both of which are highly important for the air brakes control system which is sharing the same microcontroller as the telemetry system, and power draw requirements.

The pico and ESP32 both have a dual core ARM processor, but differ in clocking speed, with the ESP32 capable of faster calculation. The significantly higher power draw of the ESP32 compared to the pico did not pose an issue to the team, as the avionics bay of the rocket was designed with a large internal volume that could easily hold enough power storage for the system. A useful previously unforeseen advantage, however, was the inclusion of flash memory that ESP32 based microcontrollers have, which the team decided would be used for on-board data storage in the event that the live-telemetry cuts out at any point during the flight.

A new system was designed with more information from initial phases, which centered around the ESP32 microcontroller, and new considerations were taken into account for the requirements and capabilities of the new processor.

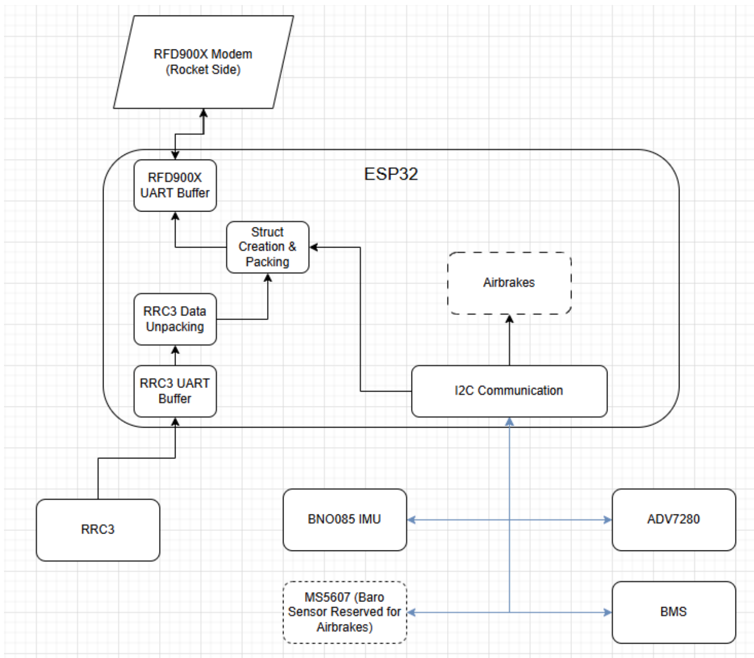


*Figure 2: Final System Overview*

The scope of this system overview has been reduced to further emphasize the communication and data being transferred between components directly related to telemetrics (and airbrakes components sharing the same system).

In the case of the RRC3, which the team has previously used as a flight computer, to our team's knowledge no other team has used as a source of flight information despite it being capable of broadcasting data over serial communication, thus the majority of the trouble in system integration has come from this peripheral. Depending on the configuration of the data stream selected, the RRC3 is capable of broadcasting timestamp, altitude, velocity, temperature, and charge deployment status, shown in the table below:

| # | Description | Example | Char Representation |
|---|---|---|---|
| I | Timestamp | 30 2E 35 | 0.5 |
| II | Separator | 2C | , |
| III | Altitude | 35 34 35 32 | 5452 |
| … | | | |
| X | Carriage Return (EOL) | 0D | N/A |

*Figure 3: Indicized Values of RRC3's Data Stream*

Note in this configuration it is not possible to isolate values without physically breaking the stream apart and processing each value individually while taking the whole system into consideration, with the carriage return at the end of each broadcast signifying the point at which a single broadcast ends.

The rest of the team's integration for sensors follows standard procedure, however, with I2C being used for calling data types from each peripheral needed to populate the struct, which is packed, serialized, then sent over the UART connection to the radio modem, which is broadcasted to the ground station for unpacking.

The final consideration in the system was the Runcam Nano 2, in the interlaced analog NTSC outputting format. Because the ESP32 does not have any ADC pins, (although if it did, would not be able to handle the computational requirements for the NTSC analog to digital conversion), an external ADC was required. The team chose the ADV7280, capable of outputting a parallel stream of 8-bits in the BTU.656 communication protocol in a YCrCb 4:2:2 format.

Due to the parallel nature of this output, the team also included an 8-bit asynchronous 256Kbit FIFO-buffer IC which would be able to store frames of video output from the ADC while the ESP32 processed the data to be stored in the struct sent to the ground station.

On the ground station is a reverse-frame compiler which takes the interlaced data from the broadcast and converts the YCrCb data of the pixels into regular RGB values, and draws these lines of pixels to create a frame. The frame is stored within a file, and then displayed onto the UI of the ground station (mockup in figure 6).

As of the time of writing this paper, the team is waiting on components that have been ordered to test the described system, but it is likely frames will need to be thrown out often due to the high speed at which our buffer will fill up at, and to avoid random data loss, the team will control the data that is lost.

## V. Hardware Design

A large amount of effort was emphasized in following the DTEG rules from ESRA. In almost all cases applicable, this meant switching from connectors such as JST-XH to JST-XA for the positively locking housing they have over the friction fit of the XH series. This also meant using Lithium-Ion cylindrical shaped batteries over any other form of batteries, but the team had planned this from the beginning. Due to the quickly rising complexity of the system, the team immediately knew a manufactured PCB would be necessary to organize and house all the small components necessary for use.

Initial designs explored the idea of housing all components on one board, including the RRC3 via 4-40 standoffs that could be surface mounted onto the board, but given size constraints there would either need to be a PC 104 stack or larger components that can interface without the need of surface mounting such as the radio modem and batteries would need to be moved off the board. In the case of batteries, this was a positive change as the 2s1p setup previously used could expand to a 2s3p setup allowing for a much longer battery life negating any worries about the system losing power while waiting on the pad.

The battery system was designed to be chargeable through a microUSB port placed in the camera shroud of the rocket, drawing power and also enabling UART communication with the ESP32 once it has been sealed within the av bay to allow for testing and any diagnostic troubleshooting that may need to be done without taking apart the structure of the rocket. Due to the 2sXp setup of the batteries, a BMS IC (BQ25887) was incorporated to balance cells and provide insight to the system about the status of the batteries, giving us another data point that could be used to isolate system failures.

The modem, both fortunately and unfortunately, had to be taken off the board due to size as well. As shown in the CAD rendering of the avionics sled below, it stands very tall, and when soldered onto the board made it extremely difficult to fit both on the footprint of the PCB and vertically in 3D space. The difficulty of implementation of the modem did however point the team to look towards the previously unconsidered propagation of electromagnetic signals that it operates with, and in its current state would not be able to communicate with the ground station in any way, as it sits within the aluminum transition of the rocket, additionally shown with the avionics sled below.

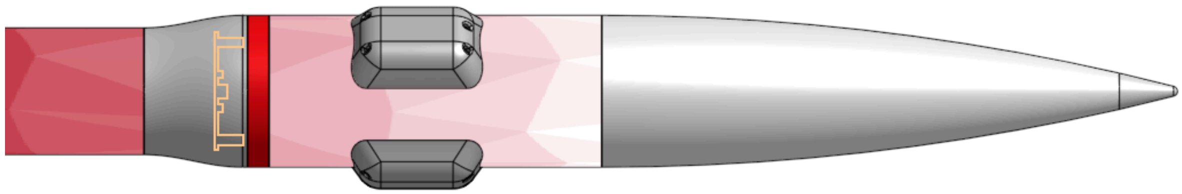

*Figure 4: Forward side of the rocket with the PCB support structure highlighted*

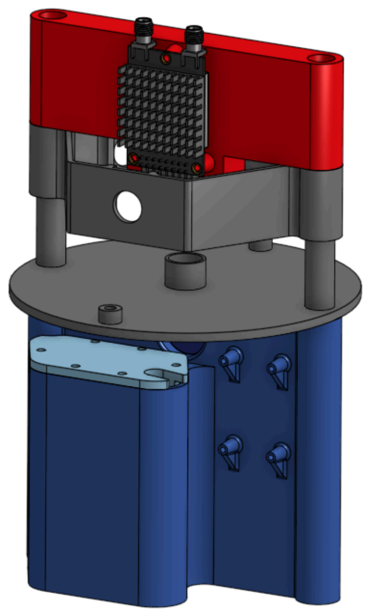

*Figure 5: CAD representation of avionics sled inside avionics bay*

This issue was remedied through the inclusion of shrouds/shark fins for the antennas, allowing them to sit outside the body of the rocket in their own assembly. Because the camera already was designed with its own assembly, this change had the surprising upside of increasing aerodynamic symmetry when incorporated with the camera shroud. The final shrouds can be seen in figure 4. SMA splitters and extenders were bought to be ran from the output of the modem to the ¼ wavelength dipole antennas, oriented in a horizontal/vertical orientation to have a linear polarization that has both a complex and real component at all times which matches the polarization of the ground station, thus giving a component of the wave which can be interpreted by either modem at all times. Also convenient with this setup is the inclusion of the payload housing directly behind the shark fins, which is made of 6061 aluminum and can be grounded to the system to promote reflections off of this body, further increasing the signal strength in the direction pointing away from the rocket.

## VI. Integration

### A. System Overview

The previously described systems all focus on data acquisition on the ESP32 before being broadcast to the RFD900x radio modem, but from the perspective of transmission the system will appear quite differently. Due to the very large difference in data acquisition speeds of various data points, with the RRC3 data updating at a max speed of 2Hz, and a new line of pixel data for video refreshing at a rate of about 15.7kHz, memory must be managed carefully to avoid overflow, data loss, or incorrect transmission.

The RFD900x comes with software that makes configuration easy, specifically for transmission. Among changeable parameters, airframe length is the most significant for our application, as it determines the max amount of data that can be sent at once. At values greater than 255, data starts being split among individual waves, so to keep the system predictable the team maxed out the frame length at 250.

This means that an entire frame of footage cannot be sent at once without compression, but instead due to the downtime in waiting for an update from the RRC3, row data of the frame can be sent, with necessary IMU updates attached to the body's remaining free data.

Once received by the ground station, data will be compiled and displayed to the UI, which uses openGL to display the attitude and rotation of the rocket. Below is a graphical mockup of the UI that will be displayed on the ground station, including all telemetry metrics gathered by the system.


*Figure 6: Mockup of custom telemetry UI*

## VII. Conclusion

The telemetry system developed for the University of Georgia's 2025 IREC rocket will provide valuable insight into flight parameters live from the rocket, allowing for a better understanding of rocket behavior, and allowing the team to pinpoint parts of the rocket flight that may affect performance, and can be better improved upon in the future. With a creative design process that grew out of this need and cultivates freedom in choices to promote a more robust design.

With a test flight that has proved efficacy of the basis of the system, further design iteration has spurred finalization of the design with testing of all systems in series with one another following isolated testing to prevent unknown points of failure.