Post-Flight Report
Midwest Competition 2018

Rocket Team at the University of Minnesota - Twin Cities

Mentor: Gary Stroick, 952-201-3002, president@offwegorocketry.com

Faculty Advisor: Brian Nichols, 612-626-5496, jwn@umn.edu

Student Lead: Jacob Grunklee, 612-231-8773, grunk012@umn.edu

Members: Ryan Thomas, Max Jetzer, Alex Wege,
Wayne Kosak, Paul Gross, Will Busacker, Sophia Vedvik,
Emma Lenz, Anna Cramblitt, Hannah Reilly,
Kyle Roberts, Garret Ailts

May 30, 2018

Contents

1 Post-Flight Executive Summary

The 2017-2018 Minnesota Space Grant Consortium Midwest Competition (henceforth "Midwest Competition") challenged teams to build a high-power rocket with an active orientation-control system. The rocket was required to reach 3,000ft and land safely at least twice. On separate flights, the orientation-control system was required to hold a constant roll orientation and roll to a series of predetermined compass angles. As a bonus challenge, teams could implement Xbee-based radio telemetry.

To meet these objectives, the Rocket Team at the University of Minnesota Twin Cities constructed a 68" long, 4" diameter rocket, dubbed "Wolfram Green." At competition, Wolfram reached apogees of roughly 3,600ft and 3,440ft AGL, flying on an Aerotech J800-T both times. The propulsion and recovery systems operated smoothly during both flights, with all parachutes deploying as scheduled. The only notable damage was a crack sustained during the second flight due to both ejection charges in the upper section firing simultaneously.

To actively control roll, pivoting elliptical canards were to be actuated through a gear set by four independent servo motors. The Roll Control Module (henceforth "R/C") included a custom sensor suite. Unfortunately, electrical failures leading up to the day of the launch locked out the gyroscope at competition. This ultimately prevented the control system from actuating at all during the flights. Issues with an onboard SD card limited the amount of data that could be logged on the R/C.

Figure 1: The fully assembled rocket

As mentioned in the Preliminary Design Report[1], the R/C is being developed by Rocket Team jointly for Midwest and for the Spaceport America Cup Competition. The goals of the R/C Project include providing team members with hands-on controls experience and developing a working control module. A more detailed overview of the R/C project goals and requirements can be found in [1].

The flight computer was subject to fewer electrical failures. This module was a separate computer system intended to log backup data, handle the Xbee, actuate the indicator LED, and record video. Data logging was operational; however, due to a buggy serial line between the R/C and flight computer, minimal data was logged. The LED was not be actuated for the same reason. A radio link could not be established between the rocket and the ground at the competition field.

Many lessons were learned as a result of Rocket Team's experience at the competition this year, especially in regards to setting realistic goals. A majority of our struggles were the result of setting lofty goals with no fall back plan. In the future we will strive to implement a robust, simple solution before moving on to more advanced approaches.

2 Launch Vehicle Assessments

Wolfram Green's airframe, propulsion, and recovery systems excelled at the competition. Once the av bay was assembled, both launches proceeded in a timely manner and were recovered safely without significant damage. No anomalies regarding the propulsion or recovery system were observed during either flight. Pre-flight and Post-flight checklists assisted in making sure every component of the rocket was assembled and inspected properly before and after the launch.

2.1 Propulsion System

All the proper hardware pertaining to Wolfram's propulsion system was brought to the launch site, resulting in a smooth and painless motor assembly process. The motor ignited without trouble and burned as expected during both competition flights. The flight profile during ascent closely matched expectations for the J800 motor used (see §3 for quantitative flight data).

2.2 Recovery System

Based on the flight data (again, see §3), Wolfram Green's parachutes were sized appropriately. The competition descent rates confirmed those experienced at the test launch. The same parachutes from the test launch were used in competition, as they showed to be a reliable recovery system. Ejection charge sizes were the same as what was used for the test flight, and ground ejection testing before the competition proved these sizes to be appropriate for the rocket. Both parachutes ejected as expected during both competition flights.

Figure 2: Slight damage was sustained during the main event on the second competition flight.

No damage was detected on the rocket after the first flight, but some small burnt areas on the parachutes were found after the second flight. A difference between these two flights is that during the first flight, only one pair of ejection charges ignited, though both pairs ignited during the second flight. As a result of both pairs going off, burnt areas on the parachutes were found and the tube was slightly cracked at the top of the upper body tube, shown in figure 2. This is not the first time tubes have been cracked due to slightly oversized charges. For future rockets, canvas phenolic tubes might be reinforced with fiberglass to prevent cracking.

To assist the team in locating the rocket after touchdown, a radio beacon was installed in the rocket, set to 223.890 MHz. Due to excellent flight conditions, this system was not needed because the rocket remained visible from the flight line throughout both flights, even after landing.

2.3 Pre- and Post-Flight Procedures

As a team mostly of experienced rocketeers, many of the items normally present on a safety checklist have become second nature. However, it remains essential to have a thorough, well-thought plan to ensure a smooth and successful launch. This is especially true when the excitement/stress of a launch becomes distracting, or when newer members are helping out. For reference, the safety checklists listed in Rocket Team's PDR are repeated here. Slight modifications will be made during future launches to improve team safety and effectiveness.

Pre-Flight Checklist

1. Inspect all shock cord connection points for upper and lower sections.
2. Assemble and install motor.
3. Fold main parachute into upper body tube and tape tracker to shock cord.
4. Insert drogue into lower section.
5. Inspect all wiring connections and zip-tie attachments.
6. Power altimeters on to test proper wiring and deploy settings.
7. Power all avionics off.
8. Wire all charges to the terminals.
9. Screw upper section to coupler.
10. Insert nosecone and upper shear pins.
11. Insert coupler in lower section. Ensure camera is fixed and faces primary mirror.
12. Insert lower shear pins.
13. Place rocket on launch rail.
14. Power up Raspberry Pi's.
15. Arm upper Stratologger, waiting for proper beep signals (three sequential beeps)
16. Arm lower Stratologger, waiting for proper beep signals (three sequential beeps)
17. Visual inspection
18. Photograph(s) of rocket on pad.
19. Photograph(s) of team members with rocket.
20. Launch!

Post-Flight Checklist

1. Locate rocket.
2. Take a picture of the rocket.
3. Power off electronics.

4. Check that there are no remaining live charges.
5. Inspect for any damage.
6. Disconnect spent charges from electronics.
7. Retrieve relevant flight data for competition officials
8. Bring recovered rocket to post flight check in table.
9. Give Altimeter 2 to competition official.

In general, these safety checklists did not provide enough detail. During safety checks, the RSO suggested that our safety checklists be updated to include masses of black powder charges and specific beep patterns for altimiters. These changes were recorded prior to launch and will be included in checklists during future launches. Furthermore, the pre-flight checklist will be divided into pre-pad and on-pad sections. We will also add more specific details pertaining to av bay assembly and powering up custom electronics.

3 Predicted versus Actual Flight Profile

As mentioned before, no major anomalies were observed during either flight and results were similar to what was predicted from flight simulations and previous test flight data. Most of the flight data from the competition was retrieved from Wolfram's redundant Stralogger Altimiters, and here that data is compared to OpenRocket-simulated flight profiles.

Table 1: Predicted vs Actual Flight Characteristics (Flight 1).

Parameter	**Predicted**	**Altimiter 1**	**Altimiter 2**
Apogee (ft)	3714	3607	3623
Peak Velocity (ft/s)	605	565	563
Peak Acceleration (ft/s^2)	614	500	400
Velocity under Drogue (ft/s)	-67	n/a	-57
Velocity under Main (ft/s)	-24	n/a	-25
Coast Time (nearest s)	13	13	13

Figure 3 shows the predicted versus actual altitude profile from flight 1. The data shown is taken from both of the onboard Stratologger Altimiters. These readings differ by less than 4 feet on average, which is why the measured plots overlap on the graph. The figure shows how altimiter 1 lost power during deployment at apogee, while altimiter 2 persisted throughout the flight. The predicted curve was generated in OpenRocket under atmospheric conditions similar to those experienced at the competition. Specifically, the rocket was simulated to launch from the exact longitude, latitude, and ground level of the flight field, with winds up to 2 m/s.

As the figure 3 indicates, OpenRocket slightly underestimated both the drag forces on the rocket and the drag coefficients of the parachutes. These miscalculations resulted in the observed peak altitude to be slightly lower than the predicted value, and the observed descent rates to be slower than expected. The former difference can be attributed to several factors. In the OpenRocket model, both camera mounts are modeled as single, 1.5"-thick fins. OpenRocket warns that fins of this thickness "may not be modeled correctly." Furthermore, the 3D-printed surface of these mounts (which amounts to

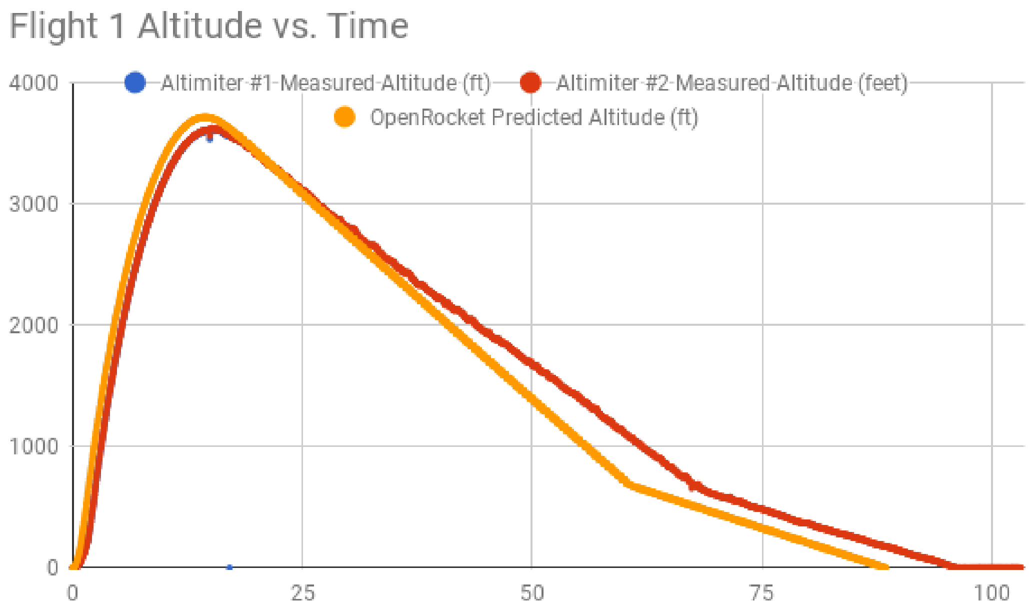

Figure 3: This data was captured by the onboard Stratologger Altimiters during flight 1, against the OpenRocket-simulated flight profile.

a substantial amount of surface area) are relatively rough, which texture is also not included in the model. The software therefore may have overestimated the peak apogee. In addition, the component masses in the model exclude the mass of the upper camera, as well as the paint.

During descent, OpenRocket predicted a much faster descent under drogue. This is most likely due to OpenRocket underestimating the drag coefficients of the parachutes and incorrectly modeling the drag induced by both camera mounts, as stated before.

Table 2: Predicted vs Actual Flight Characteristics (Flight 2).

Parameter	**Predicted**	**Altimiter 1**	**Altimiter 2**
Apogee (ft)	3714	3442	3448
Peak Velocity (ft/s)	605	564	562
Peak Acceleration (ft/s^2)	614	500	476
Velocity under Drogue (ft/s)	-67	-60	-60
Velocity under Main (ft/s)	-24	-22	-22
Coast Time (nearest s)	13	13	13

For reasons explained in §4, the custom sensor package was able to collect but not log data during the competition. As a result, our best data available is altitude from the Stratologger altimiters. As a result, acceleration measurements are subject to significant noise. The values shown in Tables 1 and 2 should be interpreted with uncertainty.

Similar to the first flight, the OpenRocket had a higher peak altitude and faster descent rate under drogue than what was actually observed with the Stratologger data for flight 2. No modifications were made to the rocket between the first and second flights, so we attributed these differences between the simulation data and observed data to the

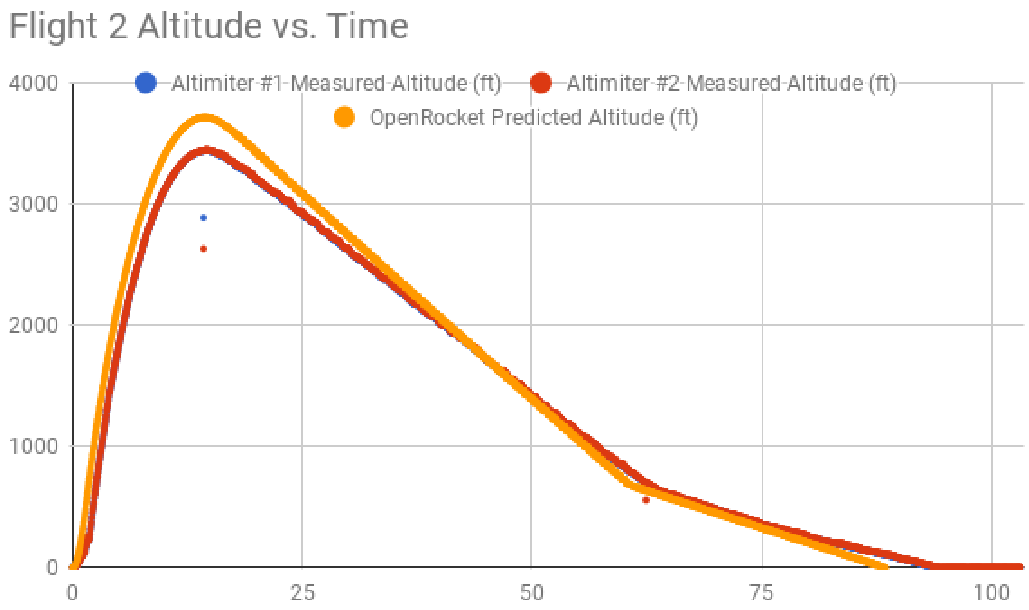

Figure 4: This data was captured by the onboard Stratologger Altimiters during flight 2, again plotted against the OpenRocket predictions.

same reasons as stated for flight 1. Figure 4 shows the altitude data received from the onboard Stratologgers compared to OpenRocket predicted altitudes, and table 2 shows more parameters calculated from the Stratologger data.

4 Electrical and Controls Systems

To control the roll of the rocket, Rocket Team designed pivoting elliptical canards actuated through a gear set by four independent servo motors. More details on the mechanical design and math behind the system is described in our PDR[1]. Unlike many of the teams at the competition who implemented PID control loops, Rocket Team developed a state-space controller, specifically a linear-quadratic, time-varying controller. This system implements an ideal controller, guaranteed to settle more quickly and accurately than a PID. Unfortunately, implementing this system drained an undue portion of student time over the course of the project. In order to fully utilize the state space controller, it was desirable to run it in a real-time environment. Adapting the real-time operating system to the Raspberry Pi Zero microcontroller also put a strain on student time.

Ultimately, it became clear by the competition that the electronics team members in particular had stretched themselves too thinly in an attempt to solve this challenge in an ideal way. While the rocket's performance suffered as a result, many team members would argue that the amount of learning that occurred as a result of this project could not have been so great if we had simply taken an easier way out. While this approach to the roll orientation challenge was intriguing and had a lot of potential to succeed, it was much too ambitious for undergraduate students without previous experience with such systems. For future challenges, our team will first try to develop a simple approach to the challenge before moving on to more elaborate solutions.

4.1 Roll Control Module

During previous test flights, the canards were actively actuating during flight, but electrical failures leading up to the day of the competition prevented the gyroscope from working. After trying several different physical sensors, the issue was isolated to the Raspberry Pi's SPI bus. There was not enough time at competition to further assess or resolve the issue. We were also unable to get data to log on the R/C module due to issues with writing and reading from the SD card.

Since the attitude determination system relied solely on gyroscope, there was no way to determine orientation during flight. Furthermore, since the controller was designed as a tracking problem (seeking a specific quaternion) rather than a stabilization problem (minimizing rotation), there was no ability to control. This was not a safety concern because the fin servos had a software block preventing them from rotating the fins more than 5 degrees past center. Furthermore, condition checks such as tipover and minimum controllable velocity were still being accurately informed by other sensors.

4.2 Flight Computer

On the other hand, the competition launches were relatively successful for flight computer. Many issues were resolved since the test launch as a result of adding an auxiliary Mobius action-camera and a new custom-printed pcb.

At the test launch, video was lost when the Raspberry Pi Camera cable wiggled loose during assembly. To avoid these cable failures, a custom pcb was installed to mount the flight computer. This new board drastically increased the available space in the lower half of the coupler. It also ensured that the camera connection would not move during flight. To further ensure video, an external camera was mounted to the side of the rocket. This camera had been avoided in earlier revisions of the rocket due to its large size and resulting aerodynamic costs. However, after having no data and no onboard footage at the test launch, these costs were reconsidered. Thankfully, both cameras recorded successfully at the competition launch. Both videos showed clear footage of the ground, canards and LED, though without the canards actuating, the rocket rolled naturally throughout the flights. Videos of the first flight are given by the sources [2] and [3], and videos of the second flight are given by [4] and [5].

Regarding future camera selection, the Raspberry Pi Camera will most likely not be used again for taking flight video. This camera was originally chosen for its small size, but it presented more design challenges than expected. The Mobius camera will probably be used in the future instead, as it has a similar aerodynamic profile, takes much better video, and is far simpler to integrate into a rocket design.

Other components onboard the flight computer included a pair of vibration sensors. These sensors were set up to log a time stamp whenever the rocket's acceleration exceeded a threshold value inherent to the sensor. The sensors were included in the design primarily out of research interest. As shown in figure 5, they were able to detect major flight events including liftoff, burnout, and parachute deployments.

Besides vibration, flight computer was designed to rely on the R/C module to obtain flight data. Unfortunately, the serial line between the two computing modules had a few bugs. In particular, there was likely some disagreement on the exact packet structure of the data being sent. All data was being passed as binary packed data (no character strings) for efficiency. As a result of this communications issue, flight computer could not actuate the indicator LED, log flight data, or send flight data to the ground. All of these

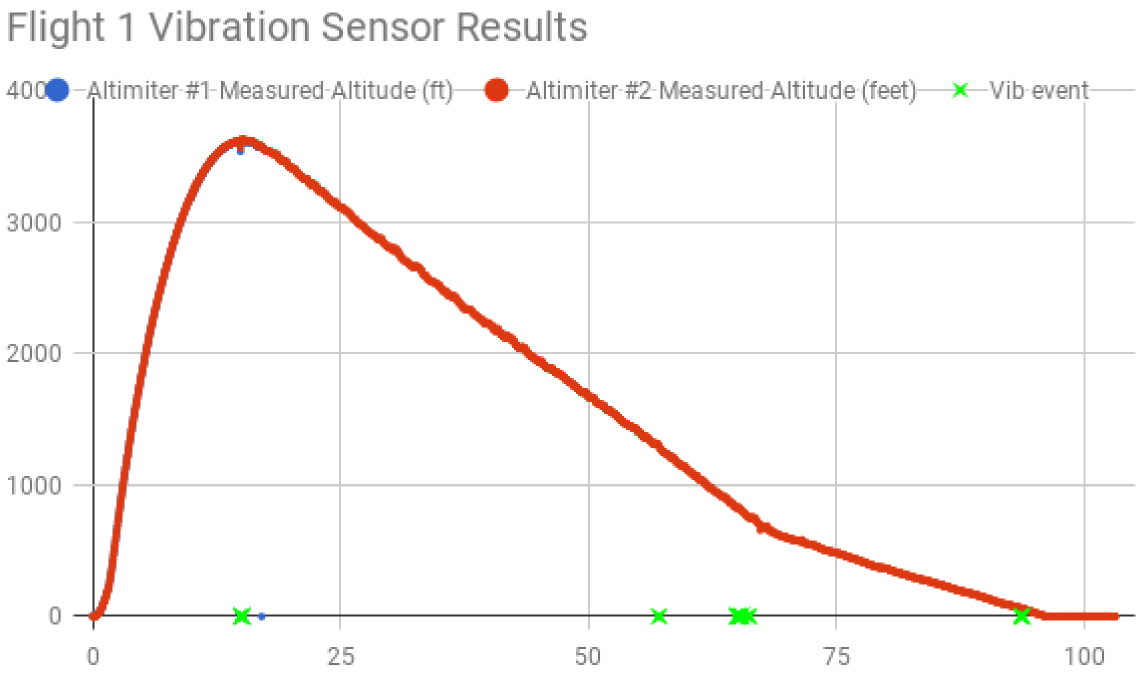

Figure 5: The vibration sensors used simply indicate the occurrence of acceleration events, not their magnitude. After aligning the events with the flight profile, one can see that the data collected represents major flight events. Recorded along the x-axis are drogue deployment, main deployment, and landing. Liftoff did not generate vibration events because the sensors were mounted vertically in the coupler and could only be triggered by acceleration perpendicular to the rocket body.

functions were working, but could not obtain the data. For example, the raw binary data files contained mostly zeros.

5 Bonus Challenges

Our team did not attempt bonus challenge part A.

5.1 Bonus Challenge B

Bonus challenge B asked teams to send slight data from the rocket to the ground in real time. We attempted this challenge using the required Xbee Pro 2.4GHz. Since the Raspberry Pi computer only has one serial port, a PIC24 microcontroller was used to multiplex UART lines. This system was operational in the lab. However, the PIC was programmed to only forward incoming packets of the correct structure, so if any data was missed or intercepted from other teams, the PIC would drop the entire packet. This system was intended to ensure the integrity of the data, but made it more difficult to debug the network. We intended to send data to the ground at 50Hz, with a 70-byte packet each transmission. These packets were intended to include a delimiter, a security code, flight computer timestamp, separation status, flight status, acceleration (all 3 axes), gyroscope (all 3 axes), magnetometer (all 3 axes), IMU (quaternion), roll rate, and estimated altitude. These values were represented as packed binary data rather than as character strings. The data was to be unpacked by a separate program on the ground after flight.

At the competition, the Xbees simply did not connect between the rocket and ground station. This could be due to the Xbee's relatively weak signal strength, the low noise tolerance of the 2.4GHz band, or interference from other teams. Rocket Team generally prefers the 900MHz band or lower for in-flight telemetry. At any rate, exactly zero packets were received by the ground station.

5.2 Bonus Challenge C

Bonus challenge C asked teams to transmit math expressions and receive the correct solutions from the rocket during flight. The list of expressions cycled through all possible operations. To prepare the data for transmission, we first generated a text file listing all 50 expressions in character format. A Python program was then run to compress each 5-byte (5 characters) expression into a 2-byte binary representation using the algorithm described in the PDR [1]. The resulting binary file was therefore only 100 bytes long. Our ground station read through the binary data two bytes at at time, wrapped each pair in a delimiter byte and the security code, and sent them over an Xbee radio. The flight computer was prepared to receive, unpack, and calculate the correct answer (neglecting order of operations, as requested). Since the result was guaranteed to be 2 digits, it was also guaranteed to fit in a 1-byte integer (technically a char type). The result was therefore going to be transmitted to the ground as a single byte integer. As mentioned, we did not have a working radio link between the ground and the rocket, so exactly zero math expressions were received or evaluated.

References

[1] R. T. at the University of Minnesota, "Preliminary design report." "Preliminary Design Report for the 2017-18 Midwest Project, outlines design and development process.".

[2] R. T. at the University of Minnesota, "Flight 1 raspberry picamera footage." Video footage from first flight of competition using the Raspberry Pi Camera. https://youtu.be/lW6Rd7w_ais.

[3] R. T. at the University of Minnesota, "Flight 1 mobius camera footage." Video footage from first flight of competition using the Mobius Camera. https://youtu.be/4Zer-Lb6M60.

[4] R. T. at the University of Minnesota, "Flight 2 raspberry picamera footage." Video footage from second flight of competition using the Raspberry Pi Camera. https://youtu.be/__B8Df3C9yE.

[5] R. T. at the University of Minnesota, "Flight 2 mobius camera footage." Video footage from second flight of competition using the Mobius Camera. https://youtu.be/QbzVzrIYdkI.