Oregon State University
College of Engineering

CS Capstone Progress Report

March 18, 2019

High Altitude Rocketry Project

Prepared for

OSU American Institute of Aeronautics and Astronautics (AIAA)

Dr. Nancy Squires _____ Signature _____ Date

Prepared by

Group 11
HART CS Capstone

Rick Menzel _____ Signature _____ Date

Matthew Forsland _____ Signature _____ Date

Abstract

This document is intended to act as a summary of work completed on the HART Computer Science Capstone project during Winter term 2019. It begins with a high-level description of the project and our goals before proceeding into a detailed breakdown of progress made and work remaining. Finally, it concludes with some discussion of the various problems/challenges experienced and the team's solutions to these issues.

CONTENTS

1 Project Purpose & Goals 3

1.1 Purpose . . . 3

1.2 Goals . . . 3

2 Winter Term Progress 4

2.1 Backend . . . 4

2.1.1 Test Data Generation . . . 4

2.1.2 Telemetry Reception & Storage . . . 4

2.1.3 Kalman Filter . . . 4

2.1.4 Server . . . 4

2.2 Front End . . . 5

2.2.1 2D Visualization System . . . 5

2.2.2 3D Visualization System . . . 5

3 Remaining Work 6

3.1 Backend . . . 6

3.1.1 Data Generation . . . 6

3.1.2 Kalman Filter . . . 6

3.1.3 Thrust . . . 6

3.1.4 Server . . . 6

3.2 Front End . . . 6

3.2.1 2D Visualization System . . . 6

3.2.2 3D Visualization System . . . 7

3.2.3 Stretch Goals . . . 7

4 Problems & Solutions 8

4.1 Backend . . . 8

4.2 Front End . . . 8

5 User Studies 9

5.1 First Feedback 9

5.2 Results 9

5.3 Second Feedback 9

5.4 Results 9

5.5 Plans 9

1 Project Purpose & Goals

1.1 Purpose

The primary purpose of this project is to produce software for the processing, storage and visualization of avionics data during rocket launches. HART is a multidisciplinary team, and the CS capstone team is working closely with ME and ECE teams who have responsibility for the actual production, transmission and reception of rocket telemetry. Once received at ground level, we will perform processing on the data, including the derivation of values such as thrust and altitude. Processed data is then written to non-volatile storage for preservation, and is broadcast over Wi-Fi for display on wireless equipped devices. The storage is important both to ensure the rocket performs as expected during tests and to allow future teams to learn from this year's project. The visualization allows team members to monitor for unsafe conditions, track (and recover) the rocket stages, and, at least in a way, to fly with the rocket they have built over the course of 9+ months.

1.2 Goals

The primary goal of the HART CS Capstone team is to capture and display telemetry from a high-altitude, high-speed rocket. This in turn serves several of the overall team goals. Firstly and most importantly, the software produced by our team is the primary method by which the team will know if we have been successful in reaching our target altitude of 150,000 ft. This verification of achieved altitude is critical because our team's primary mission is to launch a rocket to literally new heights, and a failure to capture accurate data would cast serious doubt on our accomplishments. Secondly, our software is intended to aid in the rapid recovery of both rocket stages post flight via the display of stage positions and GPS coordinates. Thirdly, we are providing the team with a means by which to vicariously participate in the rocket flight. Not only should capturing and displaying this data serve as an aid in monitoring for and determining the causes of any potential issues, but it is our hope that it allows the team as a whole to feel a greater sense of satisfaction with the project into which they have put so much time and energy over the past school year. The two most important milestones for our subteam will be the "Boosted Dart" launch (a full airframe flight where only the booster stage is ignited) tentatively scheduled for April, and the final competition launch at Spaceport America in June.

2 WINTER TERM PROGRESS

2.1 Backend

2.1.1 Test Data Generation

Test data for the Kalman Filter is procedurally generated according to a schedule of acceleration durations. The accelerations are flat values approximated from data collected during motor tests performed by the Propulsion subteam. Drag is approximated according to current velocity and an arbitrary constant, the latter of which is updated during descent to simulate drogue and main parachute deployment. Orientation is represented as a set of three unit vectors describing the X, Y, and Z axes. Test data is passed to a function that introduces statistical noise with a normal distribution provided by the Box-Mueller method. Standard deviations are based upon the variance of instruments measuring each variable. Noisy test data is passed directly to the Kalman Filter for testing. Both clean and noisy test data are additionally output to JSON files for front end testing.

2.1.2 Telemetry Reception & Storage

The telemetry reception subsystem uses the LibUSB library to interface with the antenna via USB port on a Raspberry Pi. The Raspberry Pi is running Ubuntu [Version pending]. USB interface variables have been determined by the ECE team, who is responsible for the assembly of the antenna and Serial-Parallel Interface. Received data is stored in a JSON file for use by the front end.

2.1.3 Kalman Filter

The Kalman Filter uses an array of doubles to represent the vector containing all variables. Constant matrices are initially identity matrices, with deviations determined by guess-and-check testing to achieve a visually optimal result. Matrix multiplication and inversion algorithms have been found online. A matrix-manipulation library is not used due to concerns about memory size constraints on the flight computer and a lack of need. The Kalman Filter was implemented in two steps, and is being tuned in three phases. The Kalman Filter was initially implemented to solely function on vertical motion to simplify the initial writing of the algorithm. The first phase of tuning the constant matrices occurs specifically with this version of the algorithm, to determine the effects of minor variations. The remaining 13 variables were then introduced to the algorithm, expanding the matrices to 16x16. The second phase of tuning occurs with this version of the algorithm, to most closely approximate expected flight data. The third phase of tuning occurs after the first flight test, when real flight data becomes available. This schedule was designed to minimize the complexity of progress moving from one step to the next.

2.1.4 Server

Although we had initially intended to serve our website in Apache due to the relative ease of configuring that platform, we have since decided to use a NodeJS server. This decision was largely informed by our data processing needs. Specifically, there are several Node libraries intended to simplify tasks like file I/O and the efficient parsing of CSV and JSON data. Due to the model we have chosen for moving data between the front- and backends, this added functionality was helpful, bordering on essential (as vanilla JavaScript is less than idea for these purposes). As of the time of this writing, we have setup a Node server and have successfully configured it to serve both the desktop and mobile pages in a client-side paradigm. It is also worth noting that we have built and configured a low spec laptop as a devtest environment to more closely approximate the Raspberry Pi that will run the production version. Research into the best way to update the visualization in realtime has, however, indicated that a server-side paradigm is likely the route we need to go. As such, we are currently reconfiguring our D3.js and Three.js code to run serverside. In addition to the base server, we have written working code to read in and parse our data files.

2.2 Front End

Work on the front end of the system is progressing well and is close to nearing completion. In addition to the specific component details found below, we have implemented a CSS grid layout for our dashboard, which allows us to quickly and easily position UI elements as desired. It is worth noting that we have opted to proceed with a design which displays data from both stages concurrently on the same screen. While previous years' teams have taken varying approaches in this respect, we felt that it was desirable to avoid the need to toggle between screens/views to monitor all of the available data. Both mobile and desktop pages have been written and are complete (both markup and CSS) except for some minor potential cosmetic changes.

2.2.1 2D Visualization System

The 2D component of the system is composed of 2 main pieces: a gauge-style display used to show current velocity, acceleration and thrust, as well as a digital-style readout for both altitude and GPS coordinates. Both of these components are currently complete and functioning as expected. Gauge code is finished and is stylistically based on the gauges from the Google Charts API. Gauges are currently user-configurable and fully customizable for parameters ranging from color and size to gauge name and maximum value. They feature both a needle-style display and a textual readout. The digital-style display code is also finished and has been integrated into the page for the altimeters and GPS readouts. Appearance of these displays is also easily customizable via CSS.

2.2.2 3D Visualization System

Because the 3D visualization system is intended more as a visual enhancement than necessary feature and because it does not display any critical data (i.e. both altitude and position are displayed elsewhere), our work so far has prioritized the 2D system. That being said, the 3D system is progressing well. Currently, the three primary planes (ground, stage separation, and target) are being displayed, including the desired grid affect for the stage separation and target planes. Additionally, what we believe to be the most challenging component, the 3D rotation handler, is complete. This has been integrated into the system and successfully tested with both mouse and touch input for both rotation and zooming. We have also implemented the code for displaying the stage flight path histories dynamically, as for displaying representations of the stages themselves. Lastly, the 3D component has been successfully integrated into its final place in the dashboard view, alongside the 2D components.

3 Remaining Work

3.1 Backend

3.1.1 Data Generation

Corner case datasets need to be generated, such as second stage ignition failure or catastrophic failure (CATO). File naming is done in Date-Time format, but a sequence of anomalous characters are being appended to file names. Data generation should optionally output to CSV format for ease of import into third-party visualization programs. Time permitting, the thrust will be read from motor files to more accurately represent the progression of the motor burning. Time permitting, drag will account for atmospheric pressure. And attempt at accounting for atmospheric pressure was made previously, and resulted in a simulated altitude of approximately two astronomical units and a velocity in excess of 3 * 10$\hat{5}$5 times the speed of light within 4 seconds of motor ignition. Such large values, which were stored in feet and feet per second, respectively, caused the program to grind to a halt. The attempt was promptly abandoned.

3.1.2 Kalman Filter

Time permitting, we may begin the second phase of tuning before the end of the term, but expect to complete both the second and third phases during Spring Term. We also hope to sufficiently document the algorithm such that future teams have an easier time comprehending the depths of its madness, as previous teams have failed to sufficiently explain it.

3.1.3 Thrust

An outline of the procedure to determine thrust has been produced, and if implemented will run on the telemetry reception subsytem. The thrust is not essential to rocket staging performance, and represents another variable to add to the Kalman Filter, which represents another 33 values to potentially tune.

3.1.4 Server

The majority of the remaining work with the server involves re-factoring our D3.js and Three.js code so that it works with our new server-side approach. The reason this is necessary is that the data flow we have chosen to use makes our initial client-side approach for the site unrealistic. Following this we will deploy the application to the production environment and perform testing with respect to performance and connectivity. Once the visualization code is updated we will need to write a simple transfer function to properly adjust real-world position into our 3D coordinate system. This will include code to adjust for launch position offset from origin, as well as an altitude conversion scheme.

3.2 Front End

3.2.1 2D Visualization System

As of this writing, there are only relatively minor tasks left to do for the 2D visualization system. We are currently using rough estimate maximum values and warning/danger ranges for the gauge readouts. As the airframe design is finalized we plan on adjusting these values based on the expected values produced by the Mechanical Engineering team's simulations (as well as our own simulations). Additionally, a couple of minor tweaks to the CSS layout may be needed to clean up the layout (though these should be minimal). Once these changes are complete we will be ready to move into additional user testing leading up to our final competition launch.

3.2.2 3D Visualization System

The majority of the remaining work for the front end involves the 3D position display. Perhaps the biggest piece remaining is the texturing of the ground plane. We intend to use an image of the launch area for this, but have not yet decided whether we want this to be a static image (the likely option) or if we want to position the texture in such a way that the launch position lines up with the xz origin. The current plan for this is to begin with a static image and implement a more dynamic solution as a stretch goal if time permits. Part of the reason for this delay is that our first launch takes place at a different location, and as such the final map would be pointless. Icons and colors for the booster, sustainer and combined rocket are not yet fixed (though changes here, if needed, would be trivial). Following completion of these tasks we will move into further user testing. If time allows, we are considering adding buttons to the 3D display to do things like reset it or snap to a certain view position.

3.2.3 Stretch Goals

As we near the completion of our hard requirements, we may be able to continue forward into some stretch goals. As mentioned above, one of these will be to test a dynamic texturing system for the ground plane that allows us to ensure that the launch point of origin is placed at the xz origin of our coordinate system. Another stretch goal that we have discussed with our client is the implementation of a voice-prompt altitude system. This would consist of a serious of audible (likely pre-recorded) cues that play when the rocket reaches certain altitude steps. Additionally, if we have the time we plan on including a camera/view reset button and possibly a clear button on the 3D position display.

4 Problems & Solutions

4.1 Backend

The literature on the Kalman Filter that we were able to find explains the purpose of the constant matrices in a single-variable example, but does not sufficiently explain how to extend the constants to a multi-variate example using matrices. Therefore, we intend to tune these constant matrices from identity matrices by guess-and-check, which will ultimately represent a significant amount of time wasted trying to determine the effects of changing each and every value by +/- 0.01. Because the size of these matrices increases parabolically with the number of variables, we have also elected to start with a comparatively simple 3-variable implementation tracking vertical motion, and then expand it to the full suite of 16 variables tracking vertical motion, horizontal translation, orientation, and temperature. While the procedurally generated data does roughly approximate various patterns we expect to see in the flight data, data collected during flight necessarily represents the most accurate data we can get. We have elected to tune the Kalman filter matrices in three phases, as outlined above. This will minimize the complexity of each step, and allows us to spend the most time working with real data.

Finally, our initial plan for testing our Apache server was to deploy it on one of our development machines using the Windows Subsystem for Linux (WSL). Unfortunately, this became an issue, apparently because Apache relies on several Linux kernel features that are missing from WSL. As such, we now plan to do server testing on another laptop running Linux in a standalone setup.

As discussed above, our initial plan for serving the final site was to use Apache. As we progressed, however, we found that the data flow (specifically the file I/O and data parsing) made NodeJS a better solution for our project. In light of this, we have implemented our server in Node and are progressing with the necessary re-factoring of our visualization code.

4.2 Front End

The main issue during the implementation of the system's front end has been a lack of prior familiarity with the APIs in question, namely d3.js and (at least initially) WebGL. For d3.js, there is fortunately a robust development community with abundant tutorials, which made it easy to relatively quickly produce a series of basic prototypes for UI elements and then refine them as desired. For WebGL, however, this lack of familiarity was more of an issue. A first draft of the 3D position display was completed in WebGL but we were unsure as to how to implement the mouse and touch functionality and further felt that the code even at that point had already become verbose to the point of being unwieldy. Research into how to implement mouse and touch functionality ultimately led us to a different API altogether, namely Three.js. After changing to this platform we were able to very quickly (within a day) surpass the progress made up to that point in WebGL and quickly and easily implement several new features (including the mouse and touch functionality). On top of this, we feel that the Three.js code is much easier to read and maintain versus the WebGL code. The standard geometry methods offered by Three.js (used in our design for the median and target planes as well as the stage icons) both made the code easier to write and read, but makes changing these elements as needed much simpler.

5 User Studies

5.1 First Feedback

Given that the interface has not yet been finalized, we have not yet completed an in-depth formal user study. We have however been engaging with the entire team periodically through the development process, presenting current iterations of UI elements during weekly team meetings throughout the term. This has culminated in the presentation of a semi-complete dashboard during the meeting that took place February 11th. During this meeting, the dashboard was presented to all team members in attendance and feedback was sought with regard to general clarity of information and the layout of UI elements.

5.2 Results

The biggest piece of feedback we received thus far has been the desire to have information for both rocket stages displayed on a single screen, something that was noticeably absent from previous years' efforts. To accommodate this, we made significant changes to the UI to allow for a higher number of gauges than originally anticipated and to display apogee information alongside altitudes and GPS coordinates for both stages.

5.3 Second Feedback

For the second round of feedback we took a somewhat unconventional approach. This entailed coding the mobile view live during one of our full-team integration tests. This gave us access to the full team and allowed us to gain live feedback as the code progressed. As a result, the mobile view is something of a design-by-committee and directly reflects the desires and opinions of our team (i.e. our users). Immediately following the completion of the mobile view we presented it along with an updated desktop page to the team during a team meeting.

5.4 Results

Due to the unconventional approach to gathering feedback during this stage, the results were essentially the mobile page itself. The one thing that was pointed out after development was that the sizing of elements was buggy on mobile screens of a certain size. This is something we intend on clearing up as we move forward.

5.5 Plans

As we near both code freeze and competition, we plan on taking the dashboard through several iterations of the presentation/feedback cycle as we near a final design. During this time, we intend on asking team members to provide concrete feedback related to the aesthetics, functionality and usability of the design. It is also important to note that HART has a test launch called "Boosted Dart" scheduled for April, at which time the entire airframe is assembled and launched for the first time. This will likely serve as the final and most important test for our system, as this will be the primary opportunity to test the system in an non-simulated environment prior to competition in June. While Boosted Dart is not a full scale launch (due to altitude restrictions we can only ignite the booster stage), it will allow us a chance to test our ability to receive, process and display live data from our actual rocket while it is actually in-flight.