CS CAPSTONE FINAL PROJECT ARCHIVE

MAY 30, 2020

HIGH ALTITUDE ROCKETRY PROJECT

PREPARED FOR

OSU AMERICAN INSTITUTE OF AERONAUTICS AND ASTRONAUTICS (AIAA)

DR. NANCY SQUIRES

Signature Date

PREPARED BY

GROUP 77

HART CS CAPSTONE

GRAYLAND LUNN

Signature Date

CAMERON CULP

Signature Date

Abstract

This document is an a summary of the work done for the HART team by the CS capstone members. This includes most of the documentation that the CS capstone team created, along with other summary documents and subsections. All work that the team would like to have considered for final grading is included within.

CONTENTS

1 Project Introduction

The 2019-2020 HART CS capstone team was assigned to support the engineers on the HART team in any way that they could. The initial problem was broadly defined and the team had much room to decide what the direction of the project would be. Work from previous year had been directed at developing an in house solution for an altimeter and user interface that would replace the expensive existing electronics. Due to issues with the architecture of the previously existing project and discussions with the team members in other engineering disciplines, the team made the hard decision that a new avionics chip would not be necessary. This decision was not made lightly, as there had been a lot of hard work put into the design and construction of the new chip.

A survey of the engineers on the HART team showed the CS group that there was a gap in understanding of the state of the rocket during launch. The data that was presented to the team was hard to use as a tool to recognize any type of issue with the progress of the flight. The provided analysis tool worked well for a post mortem of the flight, but during the flight, the team would have little understanding of any issues or successes that were occurring. To solve this usability issue, the team decided that it would provide a solution to this usability issue: a GUI. A GUI would provide a lens for the team to recognize the success of the launch in real time, and would also be accessible for anyone to view the launch status. The problem of understanding the data provided would be solved by information rich graphics instead of numerical values and would also be configurable for each launch.

1.1 Project Goals

- Identify a gap in the understanding of the team that could be solved within the time period of 6 months
- Decrease the time required to find the rocket after it has launched and landed safely
- Decrease the gulf of evaluation that came along with the launch of the rocket, improve the understanding of the launch in real time, not just after the flight has finished
- Design a maintainable and usable system that will work on any architecture and operating system
- Design a system that supports the naive and the experienced user equally in understanding the success of the launch

1.2 Additional Definitions

- **Avionics** The controlling software and hardware that is used on the HART rocket. The avionics system transmits telemetry data to the ground station in order for the team to better understand the flight status and to retrieve the rocket after a flight.
- **HART** High Altitude Rocketry Team
- **Karman Line** An altitude of 100 km above sea level; commonly understood as the boundary between the Earth's atmosphere and outer space.
- **Groundstation** The receiving system located on the ground that the telemetry data from the HART rocket gets transmitted to and processed on.

2 Requirements Document

2.1 Introduction

2.1.1 Purpose

The purpose of this document is to provide a detailed description of the requirements for the 2019-2020 HART CS capstone team and will be used as a future planning tool and a definition of what the client can expect from the team. These requirements define the purpose and scope of the project as it moves forward. The document will also serve as a contract of what kinds of support the other sub teams on the OSU HART club will be able to receive from our team.

2.1.2 Scope

The items covered in this document will be used as a metric for success or failure of the project and will also define how the project's designed systems will operate. These items cover the software design for interfacing systems both on a ground station and on an in flight embedded system. The requirements will also cover the interface needs between the two systems as well. Interface definitions such as packet definitions, communication protocols, and further implementation details are omitted to keep the document brief, readable, and testable. Each requirement will pertain to a specific component of the systems designed, and the software components will need to be tested to the specifications of these requirements.

2.1.3 Overview

The requirements listed are separated into system, functional, interface, groundstation, and performance sub-categories. The separation of categories is based on the larger goals of the project and the major areas of focus that the capstone group 77 plans to attack. Each category of requirement can be tested in a different manner, but each individual requirement in a category shall be tested using the same interface. This document is to be used as a road map for the future, however the nature of the project may require that change occurs in the future.

2.1.4 Additional Definitions

- **Avionics** The controlling software and hardware that is used on the HART rocket. The avionics system transmits telemetry data to the ground station in order for the team to better understand the flight status and to retrieve the rocket after a flight.
- **HART** High Altitude Rocketry Team
- **Karman Line** An altitude of 100 km above sea level; commonly understood as the boundary between the Earth's atmosphere and outer space.
- **Groundstation** The receiving system located on the ground that the telemetry data from the HART rocket gets transmitted to and processed on.

2.2 Requirements

2.2.1 System Requirements

The system requirements will define what the stakeholder can expect for the systems integration of the project. These requirements will be verified via the successful launch of the rocket and during development. The requirements in this section cover the interface needs for the systems and what role each system will play in the operation of the rocket.

ID: SR1

TITLE: Embedded Interfaces

DESC: The hardware on the rocket shall be capable of sending data to the ground station.

RAT: An interface for sending data will be provided so that the rocket's flight data will be visible to the team. This is needed for rocket recovery and safety of the launch. The rocket will be much easier to recover when it can communicate its GPS coordinates to the ground, and will be safer as teams will be able to tell if something is going wrong during the flight.

DEPEND: None

ID: SR2

TITLE: Embedded Interfaces 2

DESC: The embedded hardware shall provide one way communication only.

RAT: The rocket is not allowed to receive any data from the base station. This is a product of the fact that the rocket not be controllable from the ground. During launch, none of the parameters can be changed outside of the intial approved design. This is for safety and so that rockets cant be turned into missiles or bombs.

DEPEND: None

ID: SR3

TITLE: Embedded Telemetry

DESC: The hardware on the rocket shall provide telemetry data to the ground station at a rate of 4Hz at 15 miles.

RAT: To display informative data to the users, the interface must update multiple times per second. This update rate is based on existing constraints of the current rocket's embedded telemetry equipment.

DEPEND: None

ID: SR4

TITLE: Present Sensors

DESC: The rocket shall have sensors for altitude, air speed, GPS position, acceleration, and orientation.

RAT: This is the required information that will make the launch visible to the team. There must be backup systems for understanding altitude and airspeed, as these will be the sensors required for determining when the rocket is going to achieve stage separation. The data will be provided to the base station so that displays that have already been developed will be able to be used in the future.

DEPEND: None

ID: SR5

TITLE: Embedded Upload Procedure

DESC: The programs provided by the capstone team shall be able to be compiled and uploaded to the hardware via a single script.

RAT: This ensures that the verified programs will be accessible to the future CS capstone groups. This is important because it reduces the startup time and workload on the future software engineers that will have to work on this project. Since the turnover rate is very high (1/yr) this will provide a good starting place for the future capstone teams.

DEPEND: None

ID: SR6

TITLE: Embedded Criticality

DESC: Systems that the success of the flight depends on shall be separated in implementation from systems that are not flight critical.
RAT: This is so that the validaiton of data and transmission of data have a separate pipline from interpreting the parameters of the flight for determining stage separation and parachute launch. The data transmission and logging may be cpu intensive for a small microprocessor, while data interrupts processing may be lightweight. This will ensure the system is able to have maximum performance in conditions where it is most needed.
DEPEND: SR1, SR2

2.2.2 Functional Requirements

This section defines the operation characteristics of the HART rocket's software. These are the requirements that the systems described above must fulfil in order to be considered operational. These requirements will be tested and verified by the OSU HART CS capstone team. Verification and testing will be prioritized for systems that are more critical than others. This is so that any systems that control launch parameters can be shown to function correctly prior to launch. These level of the criticality of a component shall be provided within the functional requirement. *LEVEL 1* is high criticality and *LEVEL 2* is low criticality.

ID: FR1
TITLE: Data Logs
DESC: The embedded software shall log flight data for acceleration, altitude, and spatial orientation.
RAT: The project is going to need to set up support for future generations of HART team members. Part of this is going to be the use of historical data to generate simulations, define the parameters of data processing, and decide how to better anticipate potential issues.
CRIT: *LEVEL 2*
DEPEND: SR4

ID: FR2
TITLE: Stage Separation
DESC: The embedded software shall be able to determine when free-fall has been achieved based on 0g acceleration and GPS information and launch the second stage of the rocket.
RAT: The staging of the rocket has had issues in the past and this is perhaps the most critical event in data processing. The rocket will need to know with relative certainty when it is coasting so that we can fire the second engine and separate stages. If this is done incorrectly, the entire mission for the year 2019-2020 will be a failure.
CRIT: *LEVEL 2*
DEPEND: None

ID: FR3
TITLE: Position Interpolation
DESC: The embedded software shall be capable of interpolating position and velocity based on averages of acceleration.
RAT: The software is going to need to calculate the location of the rocket based on incoming data. This will be to ensure that the expected flight path is achieved, and so that the on-board computer's data can be compared with the ground station's data.
CRIT: *LEVEL 2*
DEPEND: FR1, SR4

ID: FR4

TITLE: Negative Velocity Detection

DESC: The embedded software shall be capable of determining when a negative velocity has been reached.

RAT: This is for determining when to deploy the parachute for the rocket. In last year's launch, the rocket's parachute was deployed when the device was travelling at more than 2 times the speed of sound. This feature will need to be robustly tested and determined based on a number of parameters so that the launch can be successful. This will be the biggest area of improvement for the software from last year to this year.

CRIT: *LEVEL 1*

DEPEND: SR4

2.2.3 Interface Requirements

This section defines the requirements for the groundstation graphical user interface. All data logged by the embedded systems on the HART rocket will be transmitted and received by the groundstation. All telemetry data received needs to be displayed to an interfacing application that allows the ground team to meaningfully visualize the data (acceleration, altitude, barometric pressure, thrust, GPS coordinates, and directional velocity). Verification and testing will be done with simulated data and during any test launches.

ID: IR1

TITLE: 3 Dimensional Flight Path

DESC: The software shall be capable of displaying the rockets location and flight path for both stages of the rocket.

RAT: Being able to log the flight path will allow the ground team to easily recover the rocket stages. Knowing the flight path will also allow the ground team to estimate trajectory in case of a dangerous path that threaten the safety of the rocket, the ground team, or surrounding environment.

DEPEND: None

ID: IR2

TITLE: Altitude Visualization

DESC: Interface shall display the altitude of the rocket.

RAT: Necessary to gauge the success of the overall mission in real time. The goal is to surpass the Karman line, and reach an altitude above the current university record of 144,000 feet high. Also, knowing the altitude of the rocket will help allow the ground team to monitor for potential conditions that threaten the safety of the rocket, the ground team, or surrounding environment.

DEPEND: None

ID: IR3

TITLE: Velocity Visualization

DESC: The interface shall display the current horizontal and vertical acceleration of the rocket.

RAT: The software will interpret the acceleration and altitude telemetry into velocity data. Being able to visualize the horizontal and vertical acceleration of the rocket will allow the ground team to monitor for potential conditions that threaten the safety of the rocket, the ground team, or surrounding environment.

DEPEND: None

ID: IR4

TITLE: Barometric Pressure Visualization

DESC: The interface shall display the current barometric pressure of the rocket.

RAT: Displaying the barometric pressure experienced by the rocket will allow the ground team to monitor for potential conditions that threaten the safety of the rocket, the ground team, or surrounding environment.

DEPEND: None

ID: IR5

TITLE: Thrust Visualization

DESC: The interface shall display the current thrust of the rocket.

RAT: The software will interpret the acceleration into the engines thrust. Being able to visualize the thrust of the rocket will allow the ground team to monitor stage progression. It will also allow the team to watch for potential conditions that threaten the safety of the rocket, the ground team, or surrounding environment.

DEPEND: None

ID: IR6

TITLE: Acceleration Visualization

DESC: The interface shall display the current acceleration of the rocket.

RAT: Displaying the acceleration of the rocket will better help the ground team watch for potential conditions that threaten the safety of the rocket, the ground team, or surrounding environment.

DEPEND: None

ID: IR7

TITLE: Stage Indicator

DESC: The interface shall display an indicator showing that the staging conditions have been met, and have completed successfully.

RAT: Knowing that the rockets first stage has completed successfully and the second stage has fired successfully will help the ground team continue to monitor the success of the project. It will also help the team monitor for potential conditions that threaten the safety of the rocket, the ground team, or surrounding environment.

DEPEND: FR2

2.2.4 Groundstation Requirements

This section defines the requirements for the groundstation hardware. All data logged by the embedded systems on the HART rocket will be transmitted and received by the groundstation. All telemetry data received needs to be processed so it can be sent to an interfacing system that allows the ground team to meaningfully visualize the data. Verification and testing will be done with simulated data and during any test launches.

ID: GR1

TITLE: Groundstation Pi

DESC: A raspberry pi at ground level, responsible for processing and storing the rocket telemetry data.

RAT: Even if the embedded systems on the rocket are generating data, it is meaningless without the ability to utilize said data. The groundstation is responsible for storing all the telemetry data it receives, processing it, and packing it to send off to the GUI interface.

DEPEND: None

ID: GR2

TITLE: Groundstation Receiver

DESC: A receiver at ground level, responsible for collecting real time telemetry data being generated from the rockets integrated systems and transmitted.

RAT: The rocket has a transmitter that is responsible for sending the telemetry data. The only was to receive and utilize the data is with a receiver that is connected to the groundstation raspberry pi.

DEPEND: None

ID: GR3

TITLE: Groundstation Display

DESC: A laptop with a display, connected to the groundstation raspberry pi.

RAT: The display is used to interface with the telemetry data. The display will show the interfacing software. This will allow telemetry data to be shown in a more coherent manner that is more easily understood and interpreted by the ground team. DEPEND: None

2.2.5 Performance Requirements

This section provides a specification metrics that will be used as benchmarks for the future implementation.

ID: PR1

TITLE: Interactive Visualization

DESC: The 3D trajectory and location visuals of the rocket should be interactive to allow the user to zoom in and/or rotate the view to get the best viewing angle.

RAT: Because the flight path can proceed in any possible direction, it will be necessary to have the ability to adjust the eye-position and/or viewing angle to prevent any occlusion of the displayed trajectory. Also, to and give the team a meaningful representation of the flight path.

DEPEND: None

ID: PR2 TITLE: Data Characterization

DESC: The data logged from flights shall be characterized using noise, stability, accuracy, and precision.

RAT: These metrics will be able to be calculated based on the data that is logged. These will provide future information to base tests off of.

DEPEND: None

ID: PR3

TITLE: Data Accessibility

DESC: Telemetry data shall be stored in a structured format and in a manner that allows for easy accessibility via standard text editors. Telemetry data has the potential to be used in the future for simulations and testing purposes via scripts or compiled data processors.

RAT: The sky's the limit. The HART team has made it their goal to launch a rocket past the Karman Line and surpass the current university record of 144,000 feet in altitude. As such, it is critical to make all data gathered readily available to future teams to help influence and perfect future designs.

DEPEND: None

2.3 Verification

Verification for each sub-category of requirements shall be done using the same method. Some tests may require manual verification, while others can be automated. This document provides the specification for how each group of tests will be performed. Implementation shall be separate and verification must be performed in order to assert that a requirement has been satisfied.

2.3.1 System Verification

Verification of system requirements will require manual testing once system integration has been achieved. Verification cases can be written by the programmers in the form of print statements to data log files, giving expected and actual outcomes for the tested components. Test cases must include the functionality that the programmer expects and and interface to assess that functionality.

2.3.2 Functional Verification

Verification of functional requirements will be performed using simulated interfacing device data and analyzing the output compared to the expected baseline. Simulated data shall be provided and tested for each component that is described within the functional requirements. These tests should be automated in nature with the output from testing scripts being a diff file of the expected output versus the actual output. *Implementation Note: The test cases and serial data shall output to a standard test file using a testing macro defined in the main project file.*

2.3.3 Interface Verification

Interface verification will be performed with simulated data and during live testing of the rocket. Information will be sent from the ground station to the graphical user interface after it gets processed and will be verified for expected output and proper visuals. The 3D flight path and location map will be tested for correct visuals, calibrated for proper direction, and tested for interaction functionality.

2.3.4 Groundstation Verification

Groundstation verification will be performed manually, and through simulations. The receiver on the groundstation and the transmitter on the rocket will need to be tested simultaneously to confirm that the expected telemetry data is being sent from the transmitter and accepted by the receiver correctly. The groundstation will also need to be tested for proper interpolation of telemetry data, proper packing of data, and that it makes it to the display interface correctly. There will also be live tests with the rocket.

2.3.5 Performance Verification

Performance verification shall be done by the end users: members of the HART team. The end users will verify that the system fulfils their needs over the course of the project. Needed changes to the requirements will be made as addressed by the end user.

2.4 Gantt Chart

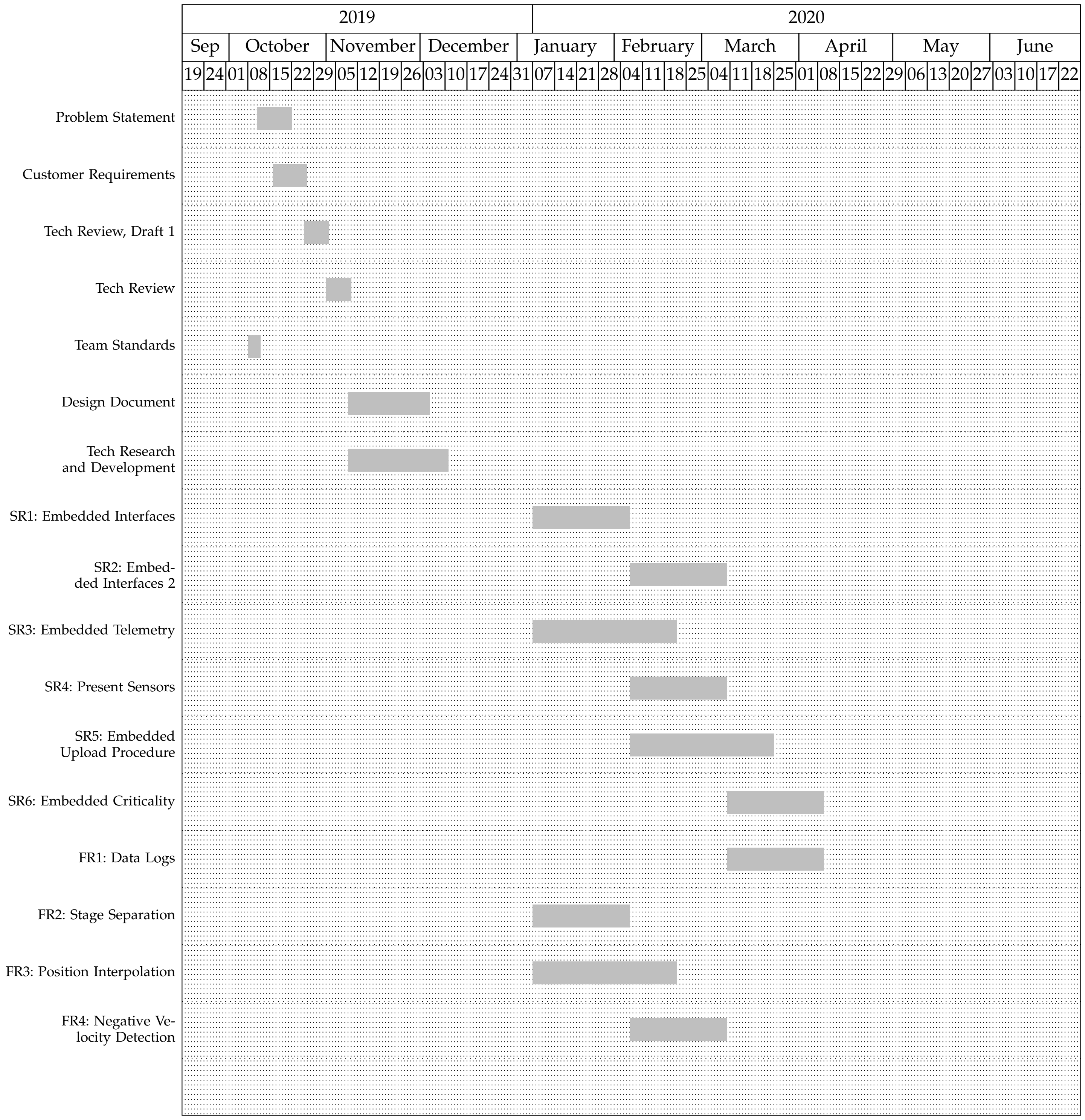


Fig. 1: Gantt Chart 1

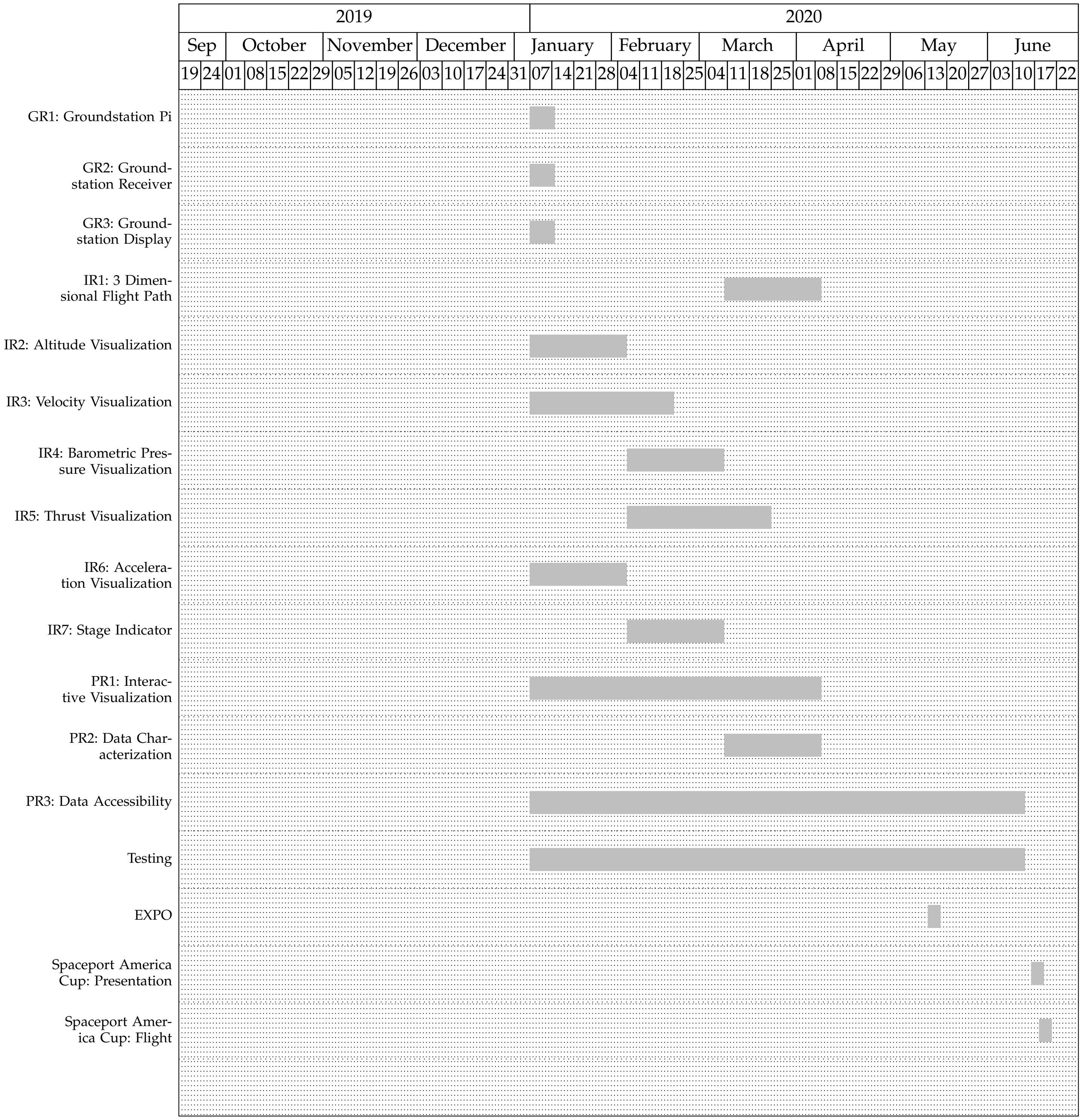


Fig. 2: Gantt Chart 2

3 Design Document

3.1 Introduction

3.1.1 Scope

This document is an outline for the software development that will occur over the course of the senior capstone project for the High Altitude Rocket Team. The future of the project will be in flux, but the general road map will be put in place. The document covers design plans GUI frontend, GUI backend, and the hardware-software interface for the devices that will be used within the rocket and on the ground station. The basis for these design decisions is research performed within technology reviews done by the authors and meeting with he HART group.

3.1.2 Purpose

The intent for the design document is that the authors will have a reference on the direction of the project. On top of that, the document will serve as a reference for the main stakeholders to check in on the progress of the project. The elements of this design will be the portions of the project that the HART computer science capstone team is committing to be responsible for. The team will be feature owners for software that is used on the base station and as interfaces. This document can be consulted when the team is at a loss for what to work on or when the team doesn't know what tools to use to accomplish the task of displaying the rocket's launch status.

3.1.3 Audience

The intended audience is composed of three groups. The other members of the HART capstone team will use this document to know how to interface with the CS portion of the project. The main stakeholder, Nancy Squires, can use this document to understand the general scope and target of the project's development. Finally, the CS HART capstone team will be able to use and review this document as a list of the project commitments and goals for development.

3.1.4 Definitions and Acronyms

- **CS:** Computer Science
- **Groundstation:** The system on the ground in charge of launching the rocket, and receiving telemetry data. It combines receivers for taking in telemetry data, and a Raspberry Pi computer, for processing the telemetry data, and sending it out over a WiFi network for the team to connect to and view.

- **GUI:** Graphical User Interface, the graphical representation of data that is received by the ground station and used as a reference to monitor the flight.
- **HART:** High-Altitude Rocketry Team

- **RF:** Radio Frequency, a method of data transfer using a radio modulation.

3.2 System Overview

The systems that the CS capstone team is responsible for on the HART project are the ground station's GUI, the ground station network services, and the interface that is used to get telemetry data from the rocket's embedded sensors. The GUI will be a website or local tool that receives information from a the Altus Metrum software. The GUI will display relative flight data like altitude, velocity, GPS location, temperature readings, and other pertinent information. This information will be displayed in a user friendly and interactive manner. The receiver in the ground station will get data from the rocket via a RF data link that is set up to receive data from the rocket. The data gathered on the rocket will be gathered either by sensors built into the on-board altimeter or by a sensor package that is built by the electrical engineering (ECE) capstone team. There are some trade-offs based on the source of the telemetry: using a package built by the ECE team allows for more degrees of freedom in terms of data rate and sensors information types, but it also means that the interface of hardware and software will need to be well defined and there will be a higher level of inter-dependency in the project. These trade-offs will be taken into account within the design.

3.3 Avionics Ground System Perspective

3.3.1 Design Stakeholders and Concerns

The ground station will be used as a monitor for the duration of the flight, and so the main stakeholders with be the members of the team present for the launch. The ground station is not a flight critical system, however it can have a large impact on how the team can use information from the flight. The members present should be able to use the ground station to retrieve the rocket, to understand the launch, and to notify other teams of locations where the rocket is going to touch down as a safety measure.

3.3.2 Design Viewpoints

There are three main viewpoints that the team will be taking into perspective: the viewpoint of the users during launch, the viewpoint of the other engineering students who will be working to interface with the user software, and the needs that are expressed by the members of the HART team. The ground station should be able to be used as a reference for now, but one of the future design goals is to have it as the only reference needed during the launch. Currently, there is a GUI that is used by the Altus Metrum devices that can be used to determine the flight critical launch parameters. The problem with that interface is the accessibility of the data. Data is displayed as raw numbers, and so its meaning is likely to be obscured to the naïve user, or even a user who is well versed in rocketry, but simply not familiar with the existing UI.

3.3.2.1 Context Viewpoint: Some features that the ground station should have will likely need to provide instructions, either explicit or implicit. These instructions will improve the usability for anyone and make the system more likely to be used during the launch.

3.3.2.2 Interface Viewpoint: The RF interface will have validation on inputs and a well defined interface, luckily packetization is already defined and handled by the devices. The incoming altimeter data will be filtered and validated before it is used, outliers will not be displayed in the same space as the filtered data. There is existing infrastructure for filtering the data, however the raw packet data that is used has a simple protocol, and so packaging is not validated. Future implementations of the interface will compare data received on the ground station from the rocket with expected formats and also provide routines for validating data transmission.

3.3.2.3 Structure Viewpoint: The CS capstone team has no plans for updating the structure of the groundstation. This is because the station has a pre-existing design that is compatible with a number of components that are flight critical, and the team is choosing to avoid messing with elements that will likely be left to other subteams.

3.3.2.4 Interaction Viewpoint: A minor goal of the team is to provide a product that is easy to use and makes sense when viewed from the perspective of a user who is unfamiliar with the product. This means that the groundstation should function when the power is turned on and the rocket is armed. The ground station should need no other user input, however there may be other user actions possible, like resetting the receiver's microcontroller, clearing cached data, or resetting the client in the case of a faulty connection.

3.4 Back-End Component Design

3.4.1 Telemetry

Telemetry data will be received on a RF 435 MHz data link from the rocket, including information that the mechanical engineering avionics sub team members would like to have saved from each flight. This information can be critical or extraneous, and varying levels of criticality will determine the frequency of packet transmission and the level of validation which that data will undergo. Status of the telemetry will be displayed within the GUI, to be used for debugging the status of the rocket.

3.4.1.1 Data Storage: The user client will include memory to log data to. This data will include all received telemetry, all received serial data, and all system status indicatiors. The stored data will mainly be used as a debugging resource by any engineering students working on the project. It should include any validation results as well.

3.4.1.2 Telemetry to Processing System: Telemetry will be coming in real-time to the groundstation three times a second. Because of this, it is best to process the telemetry data on a separate process then the receiver subsystem. This way the process in charge of interpreting the data can receive it at whatever rate it is given by the receiver to process as fast as we need it to. Desirably, as close to real time as possible to be in exact sync with what is happening with the rocket.

3.4.1.3 Telemetry to Front-end System: Just as the telemetry was coming from the receiver to be processed by a separate subsystem, the processed data will be sent off to another subsystem to be taken and presented to the front end graphical user interface. This is just as time constrained as all previous systems in the pipeline because we want to receive telemetry and present it as close to real-time as possible.

3.4.2 Data Processing

3.4.2.1 Velocity Derivation:

The rocket does not have an integrated speedometer so we will need to interpolate the speed of the rocket based on other incoming telemetry. The position data from the GPS and the difference in altitude from the altimeter will be use to calculate velocity independently, then averaged together to get the most accurate results. After we pass the limits of the GPS and altimeter, the velocity will need to be calculated from the acceleration of the rocket which will not be prone to limitations, but will be less accurate. GPS is the most accurate however COCOM restrictions limit us from using the GPS coordinates beyond 59000 feet in altitude or after we hit 1,200 miles per hour.

3.4.2.2 Kalman Filter: During the flight, the environmental conditions have the potential to interfere with the telemetry devices on the avionics bays. This can and will make the telemetry data we receive less accurate. For example, velocity and altitude will will determine the GPS's ability to function. COCOM regulations restrict GPS tracking faster than 1200 miles per hour or above 59,000 feet. Barometric altitude sensors will also become unreliable after the rocket goes super sonic because of the formation of a shock-wave over the rocket, or if the altitude surpasses 100,000 feet where barometric pressure drops too low to be accurately read by the sensors. Because of these uncertainties, we will use a Kalman filter algorithm to measure and generate an average of the measurements to get as accurate as possible data.

3.5 Front-End Component Design

This section gives an rundown of each piece of the graphical user interface. Every subsection will detail the visual aspect, how it is implemented, and our reason for the design of each component of the GUI.

3.5.1 Acceleration

3.5.1.1 Visual: The graphical user interface will display the acceleration of the rocket in two separate elements. One will show the acceleration of the rockets booster stage, and the other will show the acceleration of the rockets sustainer. The elements will be displayed in a form similar to a car gauge that reads out the acceleration for each.

3.5.1.2 Implementation: The rocket will receive acceleration telemetry from the rocket for each of the two stages of the rocket. The gauges for both the sustainer and booster stages will start at 0 with a needle starting at 0 when under no acceleration. When the rocket launches, the needles on the accelerometers will move clockwise like those on a car. Toward the far right of the accelerometer will be unacceptable acceleration level section indicated by a red zone.

3.5.1.3 Design: The reason behind this design choice is to show the real-time acceleration of the rocket on the familiar design of a car gauge. This way the team can get a better feel for what is going on. If the acceleration begins to reach the red zone, the HART team will understand that the rocket is approaching the maximum safe acceleration limit of the rocket quickly and efficiently.

3.5.2 Altitude

3.5.2.1 Visual: The maximum achieved altitude of the rocket will be displayed as a numerical value in the center of the screen. The current altitudes of each stage of the rocket will be displayed below the maximum. The altitude of the rocket will define the overall success of the launch for the competition. The altitude displays should show the real-time altitude points of the rocket and the maximum achieved altitude also know as its apogee. Continuing from last years team, if there is time, we will try to implement audio cues to indicate specified checkpoint altitudes that the sponsor indicated interest in last year.

3.5.2.2 Implementation: The rocket will receive altitude telemetry from barometric altimeters on the rocket. We will have to derive the max altitude of the rocket after a point because the data from the altimeters we have available become unusable after around 100,000 feet in altitude because they can no longer properly gauge pressure. The altitude values will be displayed on the screen with the apogee value in green so it stands out, and the other two altitude values in the same color as all the other numerical displays.

3.5.2.3 Design: The HART rocket teams goal is to design and launch a rocket to the highest possible altitude we can manage. The minimum desired goal is the current university record of 144,000 feet in altitude. Because of this, it is apparent that displaying of the rockets max altitude is necessary for determining the success of our rocket.

3.5.3 Pathing map

3.5.3.1 Visual: The path map with display the rockets flight path real time. There will be a 3D display area which will show planes of the target altitudes and the actual path of both stages of the rocket. An image of the spaceport terrain will be imported from google earth or similar software and set as the launch area for the for the ground plane. We will show the previous teams altitude as a separate plane, and the goal altitude as another plane. It will also be able to rotate and zoom in to get any required viewing angles or positions that the team may need.

3.5.3.2 Implementation: The rocket stages will send back telemetry data from the GPS's on the rocket. We will also need to derive portions of the rocket position from other data coming in from the rocket because after reaching a certain altitude, the GPU telemetry will stop by design as a countermeasure for potential missiles or rocket propelled weapon systems. We will use a Bezier curve to derive a smooth curve from any pertinent available data.

3.5.3.3 Design: The reason behind adding a flight path is so we have a rocket tracking system while it is flying. It is also designed to have the local terrain as the ground texture to help with recovery operation by having the ability to use terrain association on top of latitude and longitude coordinates making recovery more timely. Faster than a previous teams supposed, 8 hour recovery time. The 3D representation will have the capabilities to manipulate rotation and zoom in close to the path. This is to counter the potential for the rockets path to be on the same plane as the viewers eye direction, a situation which would make the flight path look like a line straight up and back down from perspective.

3.5.4 Position

3.5.4.1 Visual: Like the altitude displays, the graphical user interface of the rocket will show numerical values for latitude and longitude for both stages of the rocket. This is essential for recovery operations. It is a requirement that we are able to locate and recover the teams rocket after it has landed.

3.5.4.2 Implementation: The latitude and longitude displays will be separated by booster latitude and longitude and sustainer latitude and longitude. This will make it easy to distinguish between the two's positions. It would be a waste of time if there was confusion and we mixed up the latitude and longitude values of the two stages.

3.5.4.3 Design: The reason behind this design choice is to show the position of the two rocket stages. The latitude and longitude values are simple numerical values that have no necessary indications so we don't need something like a red zone for position. Because of this, we are using the same numerical display style as the altitude values. The reason for this implementation is because of the necessity for success of recovering the rocket.

3.5.5 Temperature

3.5.5.1 Visual: Similar to the acceleration gauges, the graphical user interface will include a temperature gauge for the currently active stage of the rocket. With the same visual design of the other gauge designed elements.

3.5.5.2 Implementation: Like all other gauges, the rocket temperature will start at 0, and the needle will move clockwise on the gauge as the temperature increases. There will be a red zone on the temperature gauge to indicate unacceptable temperature levels.

3.5.5.3 Design: The reason behind this design choice is to show the real-time temperature of the rocket on the familiar design of a car gauge. This way the team can easily visualize the rockets temperature and understand the value better. If the temperature begins to reach the red zone, the HART team will understand that the rocket is approaching the maximum safe temperature limit of the rocket versus displaying it as a numerical element.

3.5.6 Velocity

3.5.6.1 Visual: Similar to the acceleration gauges, the graphical user interface will include a velocity gauge for both stages of the rocket. With the same visual design.

3.5.6.2 Implementation: The rocket does not have a speed sensor. Because of this, we will have to derive the rocket velocity from GPS coordinates over time and the rocket acceleration. The gauges for both the sustainer and booster stages will start at 0 with a needle starting at 0 when the rocket is not moving. When the rocket launches, the needles on the speedometer will move clockwise like that of a car speedometer.

3.5.6.3 Design: The reason behind this design choice is to show the real-time velocity of the rocket on the familiar design of a speedometer gauge. This way the team can easily visualize the rockets speed.

4 Tech Reviews

4.1 Tech Review - Cameron Culp

1 Introduction

This document gives solutions for three features we will be implementing during the development of the HART rocket groundstation.

- Groundstation programming language.
- Groundstation telemetry storage.
- GroundStation telemetry noise reduction.

This document will detail the advantages and disadvantages to each potential solution to each of the three sections. This document is meant to act as an guide when deciding on which solution would be the most useful making our project and for accommodating the clients expectations for the HART rocket.

1.1 Glossary

- **HART** High Altitude Rocketry Team, the overall name of the team in charge of the design and creation of the HART rocket for the capstone project.
- **Groundstation** The groundstation is the device set up to launch and monitor the state of the rocket. It will receive telemetry data broadcast by the rockets transmitter. Because this is a high altitude rocket, we will need to be able to monitor the rockets position and telemetry data for recovery and safety purposes as well as determining the success of the mission.

2 Groundstation Programming Language

The groundstation telemetry will be processed by software on a raspberry pi running CentOS, a distribution of the Linux operating system. The software and server in charge of processing and displaying the the telemetry data will be written in a language that is capable of running on the hardware and operating system. The chosen language will also influence the potential libraries that are available for use and so should be chosen carefully.

2.1 C++

C++ [1] is a programming language created 34 years ago as an extension of the C programming language. C++ expanded C with features such as object oriented design, generics, functional features, and low level memory manipulation. The language is highly documented and widely used with a huge variety of available libraries to choose from. C++ is a compiled language that can be used for development on almost any operating system. It was designed with a bias toward embedded programming, performance, efficiency, and resource constraints. The language has also been extensively used in desktop applications, performance critical applications, servers, and operating systems. The current C++ language standard is C++17. C++ requires memory management to a lesser extent than C because of functions such as new which allocates memory, and delete which frees memory. Most C code can be compiled with C++ code and with a C++ compiler, but not the other way around.

2.2 C

C [2] is a general purpose procedural imperative programming language created 47 years ago. C features structured programming, lexical variable scope, recursion, and a static type system. Like C++, C is one of the most widely used languages today, well documented, and has numerous libraries. C is a compiled language that can be used for developing on most operating systems. C is commonly used to program operating systems, desktop applications, and embedded systems. It was designed to be compiled using a straight forward compiler for low level access to memory and to map efficiently to machine instructions with minimal run-time support. The current standard of C is C18. C requires more memory management than C++ and only uses primitive data types. C is the most efficient programming language on this list.

2.3 C#

C# [3] is a general purpose multi-paradigm programming language developed by Microsoft and created 19 years ago. The language was influenced by C++. C# features object oriented design, generic, and functional programming disciplines like C++. It is less efficient than C++ or C are, and features automatic memory management as well as garbage collection. C# is intended for creating applications for hosted and embedded systems, large and small, that use large robust operating systems, or for small dedicated functions. The current version of C# is version 8.0.

2.4 Python

Python [4] is a high level general purpose interpreted programming language created 29 years ago. The language has dynamically typed, object oriented, procedural, and functional design. Syntax emphasizes code readability with a significant use of white-space defined syntax. Python automatically manages memory and garbage collection. It has a huge number of libraries available and is very popular for interfacing. Current stable release is version 3.8.0.

2.5 Java

Java [5] is a general purpose programming language created 24 years ago that has similar syntax to C and C++ but less low level. It is an object oriented, generic, and imperative designed language. It is run in its own system, intended to be written once and ran anywhere without the need to recompile between environments. It is one of the most popular programming language, especially for client-server web applications. It manages all memory allocation and garbage collection on its own. Java is less efficient than C or C++. The current stable release is Java SE 13.

2.6 Language Recommendation

The rocket will be transmitting telemetry back from the rocket to the groundstation and getting processed real-time. The processed data will be displayed real-time as well, and stored for future use to analyze for simulations and for potential need to modify the design of the rocket. All telemetry will be sent back at a rate of 3 times per second. Because of this, efficiency is going to be a key factor. For this, C and C++ are the preferred solution for efficiency. This project is more focused on minor mathematical calculations, data storage, and displaying telemetry data. For this, object oriented design isn't particularly useful. All the languages we chose to look at as potential languages have a vast amount of potential libraries to choose from. Because of this, C or C++ are the top choices, and since efficiency is the biggest factor, and we do not need object oriented design, I recommend C for implementing the groundstation.

3 Telemetry Storage

As a CS capstone the telemetry data isn't so terribly important. However, for the rest of the HART team, this data serves an important role of being used for analysis. The future teams as well as our current team need this data to analyze and use for simulations and for trying to perfect the rocket in every way possible. The data will help the mechanical engineers and other structural teams make decisions on adjustments to the rocket's design, and help find any problems that may have arisen during tests. As such, it is important that we make this telemetry data readily available and store it in an easily read format.

3.1 XML

XML [6] stands for Extensible Markup Language, and is a markup language that defines rules for encoding documents in a way that is both human readable and machine readable. It has support through Unicode for many different human languages. The goal of XML is to be simple, general, and have usability across the whole internet. The language first started in 1996. XML can be used for a variety of different types of applications but is frequently used for we based ones.

3.2 JSON

JSON [7] which stands for JavaScript Object Notation, is an open standard file format used for sending and storing data. It is an extremely common data format which is used for many different things. The language was derived from JavaScript but nowadays it can be used by many other programming languages. JSON originally came to be in the early 2000's but was standardized in 2013. JSON is easy to understand compared to other similar languages. JSON is most often used for transmitting data from a server to a webpage.

3.3 CSV

CSV [8] which stands for comma-separated values, is a common data exchange format that is widely supported. Its initial creation is unknown but an informational RFC was dated back to October 2005. CSV is a delimited text file format that uses commas to separate values. Each line in the format is a different data record and each record has one or more fields separated by commas. CSV isn't completely standardized and the language becomes complicated when you start needing to use commas in the fields as part of the value.

3.4 Storage Recommendation

The data that we are storing and using will be displayed on client web based software. JSON was designed to work with JavaScript well and as we will be using a web based system, JSON would be the best choice for this. JSON and CVS have similar formatting but CVS becomes complex if you need to use commas in your data, CVS is also not standardized. XML and JSON are the more commonly used formats but JSON is much faster at parsing and serving data than XML is and JSON stores in a smaller file size. Since we are getting telemetry data for barometric pressure, velocity, GPS coordinates, altitude, and several others, 3 times a second for the duration of the rockets flight, and even after it lands, there will be a lot of data being stored and it will need to be done relatively fast and stored lightly. As such, I recommend JSON for speed, small file size, and good use for web based software.

4 Noise Reduction

Telemetry data being sent from the the rocket like any other signal data can and will have interference and noise. To be as accurate as possible, we will want to handle any data noise or errors of any kind. To combat this, we may want to implement some kind of filtering algorithm to improve the quality of the data.

4.1 No Noise-reducing Algorithm

Generally a little noise in the data is expected and it wouldn't be a major issue. We could just transmit the data back and store it as is without using any kind of filtering. Normally, it will come back with some noise and maybe missing portions but would be there overall. However, because we are working with a high altitude rocket that breaks the speed of sound, and is also intended to exceed a certain altitude, we run into some issues. Firstly, due to regulations, if the rocket surpasses 1200 miles per hour in speed or if it surpasses an altitude of 59,000 feet, then limits automatically get placed on GPS tracking devices to combat the potential for ballistic missiles or other potential security risks. Secondly, when the rocket breaks the sound barrier, there will be a supersonic shock-wave that will form causing the rockets barometric pressure sensor to read incorrectly.

4.2 Kalman Filter

Kalman [9] filtering also known as linear quadratic estimation is an algorithm that uses measurements observed that contain statistical noise and other errors and makes estimates for unknown variables. It has many uses but Kalman filtering is frequently used for guidance, controls, and navigation of aircraft and spaceships. Its also used for signal processing and motion planning and trajectory optimization. This will be helpful for getting an accurate trajectory for the rocket. The algorithm works in a two step process and produces estimates of the current state variables and their errors.

4.3 Particle Filter

Particle [10] filters are are used to solve issues in signal processing and statistical inference. It consists of estimating internal states when particle observations are made in sensors. Particle filtering uses a set of samples to represent posterior distributions of stochastic processes when given noisy or partial observations. It has been applied to tracking airborne objects, air to air and air to ground.

4.4 Reduction Algorithm Recommendation

For the filtering method, I recommend using the Kalman filter method. Previous teams have used this method and we have 2 members of last years team in the club currently. Kalman filter are also frequently used for aircraft and spaceships and we will be using it for rocket telemetry and trajectory optimization, which is probably why previous teams also chose to implement the same filtering algorithm. Rather than starting from scratch to use an algorithm that will work for the same thing, we should just use a Kalman filter since we have access to previous documentation. Kalman filtering will also be able to help us with the issue of surpassing the COCOM max altitude limit by making estimates for the state of current variables.

REFERENCES

[1] Albatross, "A brief description," 2019. [Online]. Available: http://www.cplusplus.com/info/description/

[2] "What is c programming language? basics, introduction and history." [Online]. Available: https://www.guru99.com/c-programming-language.html

[3] M. W. Y. V. A. A. T. D. G. W. N. T. P. K. t.-A. T. S.-S. P. C. M. H. j. J. j. Bill Wagner, Luke Latham and B. N. Al-Hashmi, "C# guide," 2018. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp

[4] "What is python? executive summary," 2019. [Online]. Available: https://www.python.org/doc/essays/blurb/

[5] L. Gupta, "What is java programming language?" [Online]. Available: https://howtodoinjava.com/java/basics/what-is-java-programming-language/

[6] M. Rouse, "Xml (extensible markup language)," 2014. [Online]. Available: https://whatis.techtarget.com/definition/XML-Extensible-Markup-Language

[7] "Introducing json." [Online]. Available: https://www.json.org/

[8] A. Rivera, "What is a .csv file?" 2019. [Online]. Available: https://www.efilecabinet.com/what-is-a-csv/

[9] S. Srini, "The kalman filter: An algorithm for making sense of fused sensor insight," 2018. [Online]. Available: https://towardsdatascience.com/kalman-filter-an-algorithm-for-making-sense-from-the-insights-of-various-sensors-fused-together-ddf67597f35e

[10] ——, "Particle filter : A hero in the world of non-linearity and non-gaussian," 2019. [Online]. Available: https://towardsdatascience.com/particle-filter-a-hero-in-the-world-of-non-linearity-and-non-gaussian-6d8947f4a3dc

4.2 Tech Review - Grayland Lunn

1 Introduction

This document is a review of all the systems that the author plans to work on when operating the rocket. These systems require varying degrees of work to be done, and the main focus of the project is going to be validation of the existing systems and making them work for the launch. While there is significant development to be done for the user interfaces and professionalization of the entire setup, we must remain focused on the group's main goal of launching the rocket as high as possible. This means that we would like to incorporate as much of the previous years capstones' work into our work so that we don't need to reinvent the wheel. This document goes over the main systems that the author plans on improving, fixing, and supporting.

1.1 Author's Statement

The author of this technology review has prior experience with embedded systems and low power approaches to gathering real time sensor data. This experience includes work on an in situ landslide sensor with restrictive power constraints and high precision requirements, and work on avionics systems for Garmin AT. The author is looking to work on interfacing the devices included in the project and focus on designing systems that would sync well together. Seeing as there is a good base of code that already exists, our primary goal is going to be making the system preform as best as possible and developing methods of verification that can be applied to future year's work.

1.2 Problem Statement

This project has two main forks of development. The first, looking through a more integration focused lens, handles integrating the software of the base station, the software on board the rocket, and the web services used for loading the user interface. This integration based approach will focus on sewing together all of the functional elements and making the system work seamlessly. The second fork looks at the software on the ground, which interprets the information gained during the flight. This is sensor data and results loaded by the base station. The software that is going to be used on this portion of the project will assume that the data gathered is accessible at all times. Existing problems include support for the Graphical User Interface (GUI) and embedded systems design using the integrated sensors and communication interfaces. Previous years of the project have had issues with determining when to fire the second stage of the rocket. As a result, one time the parachute deployed at mach 2, causing the flight to fail. This is an interesting systems design problem. The updates to the GUI are needed to provide real time data at low latency to the ground crew in order to better understand what is happening to the rocket. As it stands, there is little support for testing interfaces for hardware or the GUI. Writing maintainable tests that provide verification of desired outcomes will be a high priority.

2 Personal Responsibilities

2.1 Launch Systems

Included in the sensor package are two chips used for launching the rocket, activating the second stage, and deploying the powder charges that are used to activate the parachutes used when recovering the rocket. Currently the chips that are used to control the launch are programmed on the ground, based on simulations that are performed using the known geometry and weights provided by the structures team. These simulations and launch parameters will be assumed to be provided by other subteams within the project [1]. One consideration that will be taken into account is the use of wifi chips that are on board the second stage of the rocket. These chips are used to remotely enable the rocket's second stage when it is on the launch pad. This is a safety measure so that when the first stage is armed, the person who does the final safety checks will not be endangered by the launch system in the second stage of the rocket. The wifi chips that are used are currently password locked, posing a problem for the scheduled November 10th launch.

2.2 RF Link

The author's next focus for development is the radio transceiver that will be used to communicate sensor data from the rocket to the ground station. This telemetry link will need to be robust enough to communicate at a distance of 150,000 feet, as that is the elevation that the team is planning on launching the rocket [2]. There have been problems using this link in the past. Those problems include radio fidelity at long distance, power consumption, and incompatibility between different interfaces that are used to read incoming packets from the rocket. The incompatibility issue is the highest priority issue. This is because the incompatibility resulted in the rocket not being able to communicate with the ground station during a real launch, in turn negating an entire year of work for the capstone teams that worked on the HART rocket.

2.3 Raspberry Pi

The final area of focus for the author will be work on validating that the Raspberry Pi will be providing the appropriate data to the web interface and ensuring the integrity of that data. This will be mostly a validation task. Validation will be done on incoming packets from the rocket, in order to ensure that the RF link is being used properly and that the compiled code is running as expected. Validation at every step is a practice that is used commonly in networking to ensure that invalid data is being represented as valid.

3 Hardware Review

3.1 Launch Systems

The hardware that is on the rocket is developed by a small company named Atlas Metrum. They are known for supporting hobbyists making rockets that are used for lower level launches and amateur rocketry. They support open source development, but are a lightweight solution for the rocketry business. They employ the concept of redundancy to control all aspects of the launch critical elements. The *Telemega* is used to transmit current launch conditions and telemetry data. It is used for recovering the rocket using GPS information that is transmitted back to the ground station. The Telemega has a sensor package that includes an accelerometer for orientation detection, barometric altimeter for measuring altitude and a GPS chip for measuring location data [1]. All of these sensors are flight critical, so it is necessary that they have redundancy. This is where the *Easy Mega* comes in. The main difference is that the easy mega does not include a RF transmitter. This is so that power conssumption is kept to a minimum and the batteries can last as long as possible. Another reason is that two transmitters on the radio frequency using the same data rate can interfere with one another and cause the entire system to stop working [2]. Both the easy mega and the tele mega have the ability to fire the second stage of the rocket and to activate the powder charge for the parachute which is used to recover each stage.

While the rocket only uses the Atlas Metrum chips for launch control, the rocket also has an entire suite of sensors that are used for measuring launch telemetry. These are used to send the launch data to the ground station for display. Theoretically, the *Adafruit Feather M0* microprocessor that is currently used for the capstone designed transmitter could also be used for controlling the launch parameters. This would eliminate the need for extra batteries and for extra chipsets.

3.2 RF Link

The RF link is going to be subject to a decent level of inspection, as it was one of the previously implemented features that failed. The first way that the author plans on improving the RF link is implementing packetization within the RF transmissions. A simple UDP protocol will be used so that the receiver can keep track of the data that is transmitted. Also, a document will be kept for use in verifying the packets and the packet contents. One of the important features of using an UDP transmission scheme is that each packet comes with a checksum [3]. This can be used in verifying the data received will be in tact. Additionally, packetization provides start and end characters that can be used when listening for incoming packets. Without there start and end sequences, the data will be ignored and assumed invalid.

3.3 Raspberry Pi

The final section for inspection is the use of the raspberry pi as a main server for the project. The Pi will serve data from the RF link to the web server to be displayed on a user's computer. Raspberry Pi is a very versatile platform that is used in many prototyping and development applications. They provide the computation power and suite of a desktop computer that can fit into the palm of your hand. While the Pi is very versatile, the serving of information will be moderately intensive because of the problems that the RF transmitter has [4].

References

[1] Altus metrum. [Online]. Available: https://altusmetrum.org/

[2] N. R. Council, *The Evolution of Untethered Communications*. The National Academies Press. [Online]. Available: https://www.nap.edu/catalog/5968/the-evolution-of-untethered-communications

[3] M. Rouse. What is UDP (user datagram protocol)? - definition from WhatIs.com. [Online]. Available: https://searchnetworking.techtarget.com/definition/UDP-User-Datagram-Protocol

[4] Buy a raspberry pi – raspberry pi. [Online]. Available: https://www.raspberrypi.org

5 Weekly Blog Posts

5.1 Cameron Culp

10-18-2019

Progress So far, my individual progress is going well. I have created my resume and been assigned to the HART project. I have met in person with my sponsor and my team member and have received the project requirements. I have attended two project meetings so far. I joined the AIAA Rocketry club at Oregon State University as well as purchased student membership for AIAA to be an official member of the team. I have been briefed on the entire scope of the multidisciplinary project, and assigned a sub team for the project. I am part of the Avionics: CS subgroup along with Grayland Lunn. My capstone group has devised our set of team standards so we know what to expect of each other. I have written my problem statement and need to get together with Grayland to compile the final problem statement by Sunday, the 20th. I have been tasked with working on the graphical user interface for the rocket project. The goal is to coherently display telemetry data being received from the rocket in real time, and display it for the ground team to observe. The second goal of the GUI is to show the rockets flight trajectory and current position for both stages of the rocket in real time. The group was also tasked with reworking and testing the hardware for the ground station to make sure it works and is optimized as much as possible. We are also needed for working with the Avionics: ECE team to help with the embedded software on the rocket. We must also be flexible in the case of any other problems or needs arising that pertain to computer science. Spaceport America Culp is scheduled for June 16-20, 2020, and our goal is to hit an altitude of 144,000 feet high, the current university record.

Problems So far there are no major problems. Grayland and myself are relatively busy this term, I am taking 18 credits, 5 400 level classes that are project intense if I include the senior project. On top of that, I have drill with the Army once a month, which takes up the whole weekend, 2-3 days. I have a drill this weekend, 19-20th, and cannot work on any homework at my unit since there is no internet. This shouldn't be a major problem overall though as long as I schedule my time well. No project specific issues yet, we have not received any hardware to begin working with yet.

Plans Well first we need to receive the groundstation hardware, or its whereabouts if we aren't supposed to move it. There is a project meeting on Monday at 6pm where we will receive new information from all the teams. Grayland and myself also need to set up a subgroup meeting for the Avionics: CS subgroup. Because this project is part of an OSU club, there are students who are not part of capstone who may attend these meetings and even work with us, so myself and Grayland essentially represent the CS major for any interested CS students who are interested in the HART project. I also need to pay for my AIAA shirt and the club shirt once I figure out how to do that. Final group problem statement needs to be worked on.

10-25-2019

Progress I was given the access code for the HART rocket lab room. I received access to the groundstation for the rocket and tested the hardware aspect of it. Finished the second requirements document and got it turned in.

Problems No particular problems yet. I didn't have a monitor to test the groundstations software on so I'll have to bring one or relocate the groundstation for development and testing on.

Plans I intend to work on it this weekend if I get my other homework done. I have to make the CS Avionics meeting slide for next weeks monday meeting. We have the monday metting, and I intend to attend all 3 subteam avionics meetings just to keep up to speed

11-1-2019

Progress We have gotten access to the raspberry pi for the ground station. We talked to last years CS members about how the groundstation setup works. We had more meetines about the design and progress of all teams for the project as a whole.

Problems The raspberry pi is password blocked and we can't find documentation of any old passwords. I have a lot of time conflicts every week with meetings for the project and other classes or activities I have.

Plans I need to find someone who know what the old password to the raspberry pi is. I will work more on the Tech document. I also need to test the raspberry pi once I do gain access to the password to login to it.

11-8-2019

Problems So far we have gotten the password sheet from last years team, it was translated to a document and saved into the HART shared google drive for the whole tea, to look back on. We got access to the software on the raspberry pi. Telemega and Easymega chips have been coded with launch parameters for this weekend.

Progress We were unable to program the Egg timer chips so because they were password locked and no one could figure out the password from last year so we had to use a manual switch. Lots of research is needed at the moment on what the team last year did and how to use it.

Plans There is a test launch this weekend in Brothers Oregon. We will need to still get a new raspberry pi since it is Matt Forslands from last years team and he will want it back after this term when he graduates. We are going to need to copy the image onto a new raspberry pi for our team this year. The ground station also needs some cleanup but its not a priority.

11-15-2019

Problems We had a 2 stage launch last weekend. Launch didnt go to plan but not CS fault, improper parachute deployment. We have the full groundworks for our project now. Making a graphical user interface that takes the telemetry data and displays it in a meaningful way for the whole team. But we are going to move from last years method because it is inefficient for this years setup.

Progress Originally they HART team was wanting us CS to do integrated programming stuff and launch parameters on the telemega despite us not knowing the desired launch parameters. But the Avionics group as a whole decided to put ECE on that. Last minute changes to a different form of interface from web server and a web browser to having a c++ GUI interface program doing it. More practical and efficient for this years rocket.

Plans Figure out what libraries we are going to need for the project, and figure out the graphical design for the GUI before implementing it. Then we need to determing how we want to access data on a file with a program written by our team that is actively being logged to from another process.

11-22-2019

Problems So far we came up with a general layout for the GUI. And we have the full scope of what our project will be unless things change down the road. We will have graphs with telemetry data, and a second menu with the 3D rocket trajectory screen.

Progress We still need to figure out how we are going to do read and write on the same file at the same time using different processes on the computer. One method is to copy the file being written to and analyzing the copy, then making a new copy, and analyzing the difference, repeatedly. But we don't know about runtime consequences of this.
Plans We got asked to add another feature. The avionics team want us to find a way to obtain and display the prelaunch angle of the rocket if it is possible. We will also be attempting to make a unpolished version with just data being shown before the end of the term for the second test launch if possible.

11-29-2019
Problems We determined we will use OpenGL for the graphics API and we have the general layout design of the UI and how the 3D simulation portion will look and its functions. We talked to other teams from the ME and ECE groups and come up with the stuff they think will be useful for the overall project.
Progress Still need to determine how we will do read and write on a file at the same time. The writing appliction is for the Telemega chip that we dont have API access too.
Plans We planned to use guages like in a car to display certain telemetry data and then numerical outputs like an odometer to show info like altitude and such. The gauges will have safe and unsafe ranges.

2-7-2020
Problems Integrated the code from the telemetry reading with the gui. Added a Max speed readout as requested by the rest of the rocket team.
Progress Need physical hardware to test telemetry stream and software which must remain in the Graff hall capstone room. I dont have a windows laptop that works at the moment. Battery pack died.
Plans Once we have completed the test flight with the current state of the GUI, we will adjust or fix anything that may need fixing, and then we will begin work on the 3D flight pathing portion which was deemed less important than raw telemetry readouts and gauges.

2-14-2020
Problems Integrated the backend with the front end and did some testing and worked on optimization stuff. Found a big thing that could help optimization. Got team feedback.
Progress The GUI was taking a decent amount of seconds to load which isnt a major issue but its definitely noticeable. I figured out what part of it is I believe, 3D objects get loaded from a file for each instance of an object. Since many of them get reused many times like the Gauges and text objects, they should just be initially loaded and then copied instead of having to parse from a file each time. Should cut a lot of load time.
Plans Implement the optimization. Do an integration test with the rest of the team and see their take on the GUI. If they okay with it how it is, we will move onto the 3D flight pathing portion which was deemed secondary to gauge and readout data.

2-21-2020
Problems Continued optimization, not much can be done at the moment, got a key to gain access to the HART team capstone storage box so we can use the teledongle and AV bay with the telemega for testing purposes. Once the team

does a test flight with what is needed and everything is confirmed working as expected, we will proceed to the 3D portion of the project that they said was less important.

Progress Still the issue of not knowing what certain data is. We contacted the company who developed the telemega software and they told us that it is difficult to interpret, so we will have to figure it out somehow.

Plans Continue to optimize the software, we have a rocket launch in a week and we will be doing an avionics test with the GUI and hardware this weekend or Monday before the next HART team meeting.

2-28-2020

Problems Did a bit of optimizations as stated needed to be done before, with the model loading. Added the Mach Speed, max speed, and red color indicator stuff. The red is to show hey this gauge is hitting max capacity for this variable. Danger zone. It turns red once a gauge hits 80 percent of its max.

Progress Having an issue with the telemetry parser and c++ ¡limits¿ in the standard library. It was developed on linux with a stdint library for gcc. While the GUI was done on windows visual studio and uses MSVC stuff. So we need to convert it

Plans Working on the middle portion of the screen GUI. That was deemed lower priority and that it would be best for us focus on the numbers and gauges first because they are what the team needs to know. The center part is just ”Fluff” that looks cool. But since the rest is basically done, we should do the other part now.

3-6-2020

Problems Mostly just optimization. We figured out the issue with the telemetry data showing numbers that weren’t expected. I like the way the angle display looks, made it better than before.

Progress Trying to get viewporting to work with the 3D trajectory portion of the project. The gauges and readouts are displayed in orthographic view, and the 3D portion obviously needs to be in a perspective view. Plus both portions need different operations done on them. Just decided to put both portions as different viewports, but having difficulty making it happen.

Plans Optimization, want to test with the AV bay, and then live flights. Get opinions from the whole team after AV bay test and live flights, and modify as needed.

3-13-2020

Problems Lots of optimization, most of the 3D trajectory part is completed and just needs merged to the main project. Made a video and made PP slides for the rocket team project part.

Progress The backend and the 3D trajectory part are not merged with the main GUI atm. We are still having issues getting the backend to not have compilation errors in visual studio when it is moved from Graylands implementation to the main GUI. Something wrong with the stdint library he used and visual studio.

Plans I plan to work on the code over spring break to try and get the 3D trajectory part merged with the main GUI. We also need to get the backend parser issue fixed and merged with the main GUI also.

5.2 Grayland Lunn

10-18-2019

Progress This week the members of group 77 attended the OSU HART team meeting for the first time and started to understand the scope of the problem. Avionics will require that all sub systems are operational by April for tests and static fires of the rocket so that we can be fully prepared by the June 13 launch date.

Problems Some of the issues that I was confronted with included determining how we would fit into the team and also scheduling. I'd like to pursue my personal interest in developing software for the embedded systems within the rocket, but looking at historical information on the project, the only work previous capstones have done has been on the base station. Scheduling has been a challenge because I have 25 hours of work weekly and I also have some senior classes that are very demanding. There just isn't a whole lot of time for me to meet with my groupmates and the club, but I have been trying my best to make time.

Plans As a team we need to set up a sub team meeting for Avionics CS. This will be so that we can disseminate information to the other club members. We intend to meet with the ECE capstone team members and figure out the scopes of the issues that they would like to work one and carve out a nice that works for both teams.

10-25-2019

Progress This week we obtained the base station for the rocket setup and worked towards getting our L1 amateur rocketry certifications. We had the first avionics meeting and established who is going to be on each team and some collaborative efforts on the development of the rocket.

Problems Current problems that need addressing are the base station needs to be interfaced with a computer and sample data from the rocket needs to be retrieved. Serial communication from the radio needs to be compared with serial data written from a Raspberry Pi in the base station so that we can determine where the communication mismatch is.

Plans Plans for this next week involve preparing the AV bay for a tentative launch on November 2 in Brothers, OR. For this, the serial communication must be working by the launch date. We must also guarantee that data logging is available for the day of the launch. There is a lot to do this week.

11-1-2019

Progress This week we had the first all avionics meeting and established the need for the test launch, and moved the test launch back 1 weekend. I was able to program the primary flight computers, but need to find passwords for the timers and arming systems.

Problems There is little to no documentation on previous passwords used for the project. With the number of networking cards that we need. This will be a future goal to have them centralized.

Plans Need to get passwords ASAP. Start planning some additional work for the on board flight computer. Figure out a way to involve the ECE portion of the project and remove the electricals from inside the rocket. My main takeaway from this week is that we introduced a lot of problems to the launch by using a separate system for the internals of the rocket and we can use more of our delta v by reducing the electronics within the vehicle itself. This can also simplify the execution using the GPS retrieval systems.

11-8-2019

Problems No Problems this week, things went smoothly and we are ready for launch over the weekend.
Progress Got the rocket launch systems ready to go and prepped the on board altimiters for launch. Final tests are under way and the test rocket should be able to fly this Sunday.
Plans Taking a breather after the launch, we did a lot of research and prepping for the launch and are now going to take some time for not thinking about capstone. In the future we need to consider what flight systems are needed on the rocket itself. Many of the features that were developed by previous years are not only redundant to the chips included, but they are also not flight critical (i.e. they could be left off of the rocket and the launch would go just fine). Consider options going forward for making a ground station that integrates with the Altus Metrum chips and doesn't need any extra hardware (or weight) in the system. Have ECE capstones overhaul the ground station?

11-22-2019
Problems This week was relatively relaxed, we didn't do a whole lot with the project as there was a lot on our plates with midterms and other classes. Our main problem right now is a lack of understanding of how the logging is happening from the Altos UI. That will be looked into in the future and we plan on building on the UI and making our own graphical representations.
Progress As I said, there was little overall progress. We had our standard group meetings with the HART team and the Avionics sub team. This was about all that occurred from my end.
Plans We are planning for most of our actual software development to occur over the course of winter term, as our main system design is complete.

11-27-2019
Problems This week we ran into the issue of reading data as it is being written to an open file. This needs to be resolved so that we can get the data real time from the rocket's interface.
Progress We had our standard group meetings with the HART team and the Avionics sub team. This was about all that occurred from my end.
Plans We are gearing up for the start of development beginning over the course of winter break. We have a number of test launches coming up and we have committed to having the first iteration of the GUI ready by mid-February.

1-10-2020
Problems Over the break, the rocket's chips were configured so that they wouldn't power on when connected, and that was a big issue for me. I was able to get into contact with one of the Mechanical engineering capstone students and we figured out what was going on, so now I have a good spot to work from. I am planning on having a functional back end ready by Monday.
Progress Got the rocket's chip to a place where I can use it to develop with.
Plans I'm going to write a lot of software over the weekend, that is all! Also we are planning on getting the poster layout and rough finished by the end of January.

1-17-2020
Problems So far there have been no problems this week. We forgot to do a portion of our Gantt chart for team planning

this week, but that is all.

Progress This week I got the rocket's AV bay to start reading some of the data from a script. That script will now output to a larger module that displays the GUI.

Plans Next week we will demo a portion of the working GUI and have our Gantt chart prepped for the next meeting.

1-24-2020

Problems None this week!

Progress Lots to speak of, got most of the backend functionality.

Plans Full demo of the GUI with the device this Monday.

2-6-2020

Progress This week we got to work on integrating the GUI application and the back end parser. Hopefully it shouldn't be too hard. Also, we contacted the creator of the Altus Metrum products so that we could figure out how some of the data members are being put together, we have been stumped on this for a while. **Problems**

As mentioned, we've had issues with the data type for the raw measurement unit data. We're trying to resolve some of this, but it may just require the help of Keith Packard (mentioned above). **Plans**

We have a launch that is next weekend, and we really ought to have the GUI working by this monday with live data. We'll see how that goes!

2-13-2020

Problems Had an issue this week with not being able to contact Keith Packard, the designer of the Altus Metrum line of products. Unfortunately, one of the other team members did not forward our email as the had requested and so the message did not make it to Keith. Hopefully this week will be different. **Progress** We got a good looking draft of the poster together and also integrated the code bases of the rocket. Looking forward to presenting a functioning GUI to the team on Monday and Thursday at the design review. **Plans** Design review and rocket test launch next week!

2-20-2020

Progress Little progress this week, had other projects and midterms to take care of.

Problems Waiting on Cameron to finish up integration of the front and back ends, waiting on that before we can demo for the launch.

Plans Need to have demo prepped for design review and design review materials ready for March 3 review. Worried about having a demo ready in time, wondering what materials should be provided to the other teams for prep.

2-28-2020

Progress Progress for the week was work on redoing our poster and the design review. Email will be sent out this evening for invite to the review.

Problems This week Cameron and I got together to finish integration but we were not able to because the hardware was not present in the basement of Graff. This was the result of a miscommunication with the rest of the team. I need to be better about staying on top of emails.

Plans Design review next week!

3-6-2020

Progress This week we made a functioning demo for the team and for the design review. We had good feedback on tools that may help us resolve our outstanding issues and how to move forward with the project. The other teams that we listened to present were very interesting and had very cool solutions to problems that we hadn't seen before.

Problems Still looking at integration issues. Considering using docker as a solution to integration problems. May need to look at other alternatives too!

Plans We have a demo for the team this Monday and also need to have a video in for the beta implementation of the project. These are first and foremost in our near future plans.

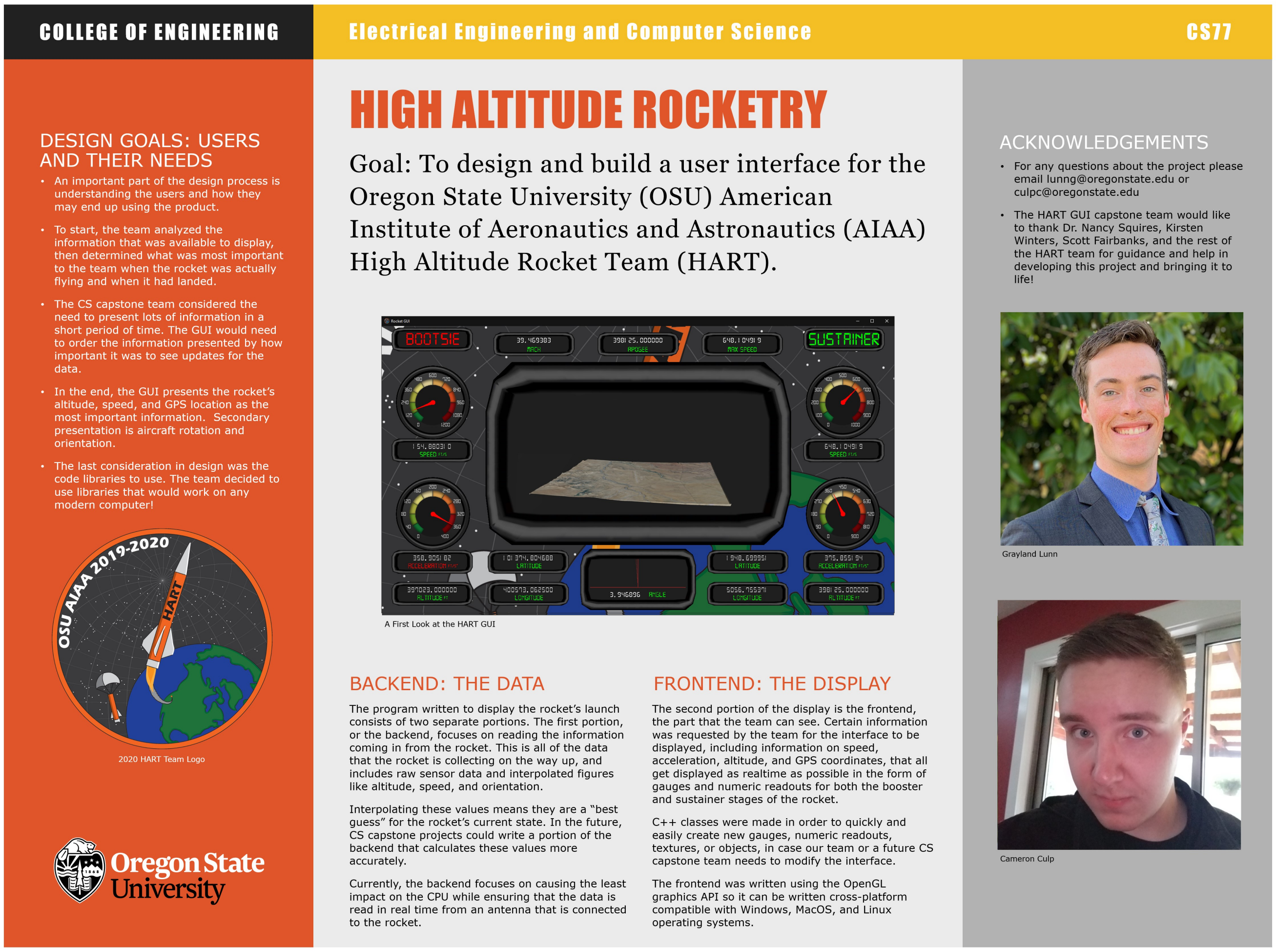
3-13-2020

Progress This week we were able to get to work on the demo video of our GUI for the class and for the capstone group.

Problems The teams progress has been stymied by the Coronavirus countermeasures that the world is taking so we'll have to see where that goes before we can have access to our materials again.

Plans We are waiting to see if we can get access to where the equipment is stored, waiting on that for further development.

6 Final Showcase Poster



COLLEGE OF ENGINEERING | Electrical Engineering and Computer Science | CS77

HIGH ALTITUDE ROCKETRY

Goal: To design and build a user interface for the Oregon State University (OSU) American Institute of Aeronautics and Astronautics (AIAA) High Altitude Rocket Team (HART).

DESIGN GOALS: USERS AND THEIR NEEDS

- An important part of the design process is understanding the users and how they may end up using the product.
- To start, the team analyzed the information that was available to display, then determined what was most important to the team when the rocket was actually flying and when it had landed.
- The CS capstone team considered the need to present lots of information in a short period of time. The GUI would need to order the information presented by how important it was to see updates for the data.
- In the end, the GUI presents the rocket's altitude, speed, and GPS location as the most important information. Secondary presentation is aircraft rotation and orientation.
- The last consideration in design was the code libraries to use. The team decided to use libraries that would work on any modern computer!

2020 HART Team Logo

Oregon State University

A First Look at the HART GUI

BACKEND: THE DATA

The program written to display the rocket's launch consists of two separate portions. The first portion, or the backend, focuses on reading the information coming in from the rocket. This is all of the data that the rocket is collecting on the way up, and includes raw sensor data and interpolated figures like altitude, speed, and orientation.

Interpolating these values means they are a "best guess" for the rocket's current state. In the future, CS capstone projects could write a portion of the backend that calculates these values more accurately.

Currently, the backend focuses on causing the least impact on the CPU while ensuring that the data is read in real time from an antenna that is connected to the rocket.

FRONTEND: THE DISPLAY

The second portion of the display is the frontend, the part that the team can see. Certain information was requested by the team for the interface to be displayed, including information on speed, acceleration, altitude, and GPS coordinates, that all get displayed as realtime as possible in the form of gauges and numeric readouts for both the booster and sustainer stages of the rocket.

C++ classes were made in order to quickly and easily create new gauges, numeric readouts, textures, or objects, in case our team or a future CS capstone team needs to modify the interface.

The frontend was written using the OpenGL graphics API so it can be written cross-platform compatible with Windows, MacOS, and Linux operating systems.

ACKNOWLEDGEMENTS

- For any questions about the project please email lunng@oregonstate.edu or culpc@oregonstate.edu
- The HART GUI capstone team would like to thank Dr. Nancy Squires, Kirsten Winters, Scott Fairbanks, and the rest of the HART team for guidance and help in developing this project and bringing it to life!

Grayland Lunn

Cameron Culp

7 Project Documentation

7.1 Backend/Parser

Code base for the project backend can be found at https://github.com/lunng/HART_CS_2020

Backend is designed as a module that can be imported to a project and work out of the box for reading incoming data from the TeleMega altimeter.

7.1.1 *Building and Running*

1) First, clone the repository to your local machine. This portion of the project is designed to be built using cmake. Clion has decent integration with cmake, we recommend you use clion to clone and build the repository, you can also use visual studio with the cmake extension installed.
2) After the repository is cloned, open the AltosUI on your local machine.
3) Connect and set up the Teledongle to your computer, power on the telemega altimeter, and configure the altimeter within AltosUI.
4) Within configuration, set up logging for the telemetrum.
5) Initialize the parser with a string path to the logging file, this can be either an absolute or relative path.

6) Set up your toolchain to work with cmake on your local machine.
7) Build and run, this will be different based on your IDE and install.

8 Recommended Technical Resources for Learning More

8.1 Websites

1) Keith Packard's reference on the Altus Metrum line of devices. This is essential knowledge.
   https://altusmetrum.org/
2) Multithreading in c++, understand how the parser spawns a new thread on creation and how sensor data is updated in the background of any program that imports the parser.
   https://www.tutorialspoint.com/cplusplus/cpp_multithreading.htm
3) Best website I found for beginner and some advanced OpenGL. Coveres model loading, shaders, lighting, and PBR.
   https://learnopengl.com/

9 Conclusions and Reflections

9.1 Camerons's Conclusions

Throughout the course of this project, I learned what its like to work as a team to develop software. Our capstone team went beyond just working on software and worked with a whole team of students on a multidisciplinary project to create a rocket, launch it, and in our case, build a GUI to watch realtime the conditions of the rocket. The project was a step up from the previous classes I have had which were designed to teach us how to develop software as a team and took place over a full year. We learned that communication and documentation is critical. We chose to start this GUI from scratch vs working on prior teams code because it makes it easier for future teams to continue working on out code base. We would also come to a hault until the other teams working on the physical rocket inform us what they would like to see on the GUI.

10 Conclusions and Reflections

10.1 Grayland's Conclusions

The entire capstones process provided valuable insights into the software development process as a whole and the details of working within a large team coordinating to accomplish a common goal. Technically, the project's scope was within my prior experience using microelectronics and working within the aviation sector. I learned that technical goals are sometimes more abstract than they may seem and a good software engineer will turn the abstract goals into concrete steps to be taken to achieve these goals. One thing that I felt was missing from the process was iterative development. The goals of the project were ever changing, however the structure of the project's development was not able to accommodate for these changing goals. An important takeaway is that for a team to develop efficiently, their development process must reflect the goals that they are trying to accomplish. If rapid iteration is a part of the team's goals, then the process must also support that. For our team, the waterfall development model hindered progress and stymied innovation. The team experience helped me understand the value of trust and communication within a project. A project cannot be finished to the expectations of the team members without these crucial elements. For expectations to align, communication is critical. For the progress to meet the expectations, the team must be able to trust that the others will be able to do their part in the process. I was lucky to have a team member who was excited to work on his portion of the project, and as such, the team succeeded in accomplishing most of its goals.

Appendix

.1 Essential Code Listings

.1.1 Backend

.1.1.1 TelemParser.cpp: This is the class that controls the operation of the parser. This Class spawns a new thread upon creation and starts reading the input file that it was initialized with. After creation, the parser sleeps for 200ms and then wakes to read any new data. Polling the parser should be done on a 200ms interval within the calling function. An example of the initialization and running of the parser is included within main.cpp.

.1.1.2 TelemString.cpp: The TelemString class is an abstract data class. It implements the functionality for any of the datatypes that are icluded in the Altus Metrum telemetry.pdf file (see AltOS UI application installation files for telemetry.pdf file). TelemString can be extended to implement any other data types that you may need to use in your project if they are not already included within your project.

.1.1.3 TelemStringTypes: Copy the implementation seen here to extend the TelemString classs for your needs.

.1.2 Frontend

.1.2.1 GUI.hpp: This is the class initializes graphics libraries, and is in charge of calling render and update calls per frame. It maintains all aspects of the GUIs state.

.1.2.2 Mesh.hpp: The mesh class utilizes obj$_{l}oadertoimport.objfilesintomemory.Meshclassmanages3Dobjectsinclud$

.1.2.3 Shader.hpp: Manages code that creates shaders and manages a shaders parameters. This will be combined with the mesh class to give a mesh lighting effects.

.1.2.4 Readout.hpp: This class is in charge of initializing a text readout box for gauges or standalone to display numbers or text that it is passed as a parameter.

.1.2.5 Gauge.hpp: Manages code involved with initializing and managing a gauge object. Including updating its information and rotating its dial.

.1.2.6 Booster.hpp and Sustainer.hpp: These are just both classes that manage different portions of the GUI. Booster manages booster gauges and readouts, and Sustainer manages sustainer GUI objects. This was done to shrink the GUI class and make things easier to manage and read.

.2 Additional Materials

You can find code and other materials in the group Githubs:

Backend, Grayland - https://github.com/lunng/HART_CS_2020

Frontend, Cameron - https://github.com/Lieutenant-Lewd-Strawberry/CS77-High-Altitude-Rocket

.3 Code Review Criticism and Responses

.3.1 Backend, Grayland

Full body of criticisms can be found in the Github repository for the project.

Criticisms

- ”More than testing having more accessible documentation and support guides.”
- ”There is a lack commenting within the code and external documentation on what each of the methods and objects do.”
- ”There were a few places that would benefit from more comments, but in general I felt that the comments were sufficient.”
- ”I would like to see a full integration test developed where they mock the data source on the backend and test frontend/backend integration.”

Responses

- Added file headers and descriptions.
- Updated build guide in project readme.
- Added future development goals as per HART team suggestion.
- Added function comments for tricky bits of code.
- Was not able to add tests as development time was limited.

.3.2 Frontend, Cameron

Full body of criticisms can be found in the Github repository for the project.

Criticisms

- ”There were some section that had hard to read blocks of code.”
- ”There is a lack commenting within the code and there wasn’t any documentation on what the code does.”
- ”I would like to see a test set up for the 3D trajectory portion without the rocket being launched.”

Responses

- Added comments in user created source and headers.
- Added comments on functions and their parameters.
- Was not able to add tests due to limits in development time.
- Could not remove blocky code because all information was necessary and changing it would need a design overhaul.