HyRo

Team 28

Jason Klindtworth | Josh Asher | Layne Nolli

CS461

Fall 2016

Requirements Document

February 15, 2017

Contents

1 Introduction

1.1 Purpose

This document contains the software requirements that describe components of a system for a hybrid rocket that will communicate instructions, retrieve data, log commands, visualize data, and visualize controls for such a rocket. This document will provide scope, definitions, an overall description, and specific requirements for this system. This system will be used by the OSU Hybrid Rocket team to launch and visualize data from their rocket. It will also provided a way to view logged data. Persons involved include Nancy Squires, ME Senior Students, ECE senior students, and other rocket club members that have chosen to be part of this team. Our audience members have an inclination to participate in rocketry. The system if successful will be used a base design for future hybrid rocket teams. The goal of the team is to eventually enter regular competition.

1.2 Scope

There are 2 separate components to this avionics system.

1.2.1 HyRo OS

HyRo OS is a program residing on a Beagle Bone Black embedded Linux computer that will be onboard the hybrid rocket. This software component will collect sensor and possibly GPS data from onboard ECE subsystem, which provides sensor data and command execution, and send this data to the ground team. HyRo OS is also responsible for receiving rocket commands from the ground team and responding to those commands by sending the command to the ECE subsystem. The purpose of this component is to provide reliable communication to the ground team for commanding the rocket and gathering its data.

1.2.2 HyRo VS

HyRo VS is software residing on a traditional computer that will send and receive data and commands to the rocket's onboard HyRo OS component. HyRo VS will be the ground teams graphical interface for data visualization, logging, and issuing commands to the system. It will present the ground team with input buttons to issue appropriate commands and data visualization windows to monitor data transferred from the hybrid rockets on board HyRo OS. This software will benefit the ground team by providing data visualization from the hybrid rocket that previously was not human readable. It also will provide command input into the system that benefits the ground team by ease of use and safety. Along with providing the ability to load data from previous operations.

1.3 Definitions, acronyms, and abbreviations

- **ESRA:** The Experimental Sounding Rocket Association is a non-profit organization founded in 2003 for the purpose of fostering and promoting engineering in rocketry.
- **AIAA:** The American Institute of Aeronautics and Astronautics is the world's largest technical society dedicated to the global aerospace profession.
- **Avionics System:** A system for controlling launch and other system function on a rocket, along with collecting data from onboard sensors.
- **Onboard:** Any component of the system that is housed in the rocket chassis.
- **Beagle Bone Black(BBB):** A micro controller running Debian Linux connected to onboard components of the rocket.
- **HyRo OS:** The HyRo rocket team's onboard operating system.
- **HyRo VS:** The HyRo rocket team's command and visualization software running on a traditional computer.
- **Embedded:** A software system that is running on microcontroller with no visual output.
- **Hybrid Rocket:** A rocket that has both solid fuel and liquid fuel. This creates the ability to throttle the rocket's motor.
- **GPS:** Global position satellite. In the aspect of this paper, a sensor that collects positional data in relation to the earth. This data is formatted for use in GPS applications to visualize the location of the sensor.
- **Ground Team:** The group of rocket team members controlling the rockets operation and viewing its data. Located a distance from the launch site on the ground.
- **Traditional Computer:**A computer, like a laptop, running an operating system. The operating system we will consider are Windows or Linux.
- **Remote Filling:** Filling the oxidizer from a safe distance via commands from the ground team.
- **Arming:** Preparing the rocket to ignite.
- **Disarming:** Reversing the rockets arming condition.
- **Launch:** Releasing the rocket into the skies!

- **Abort:** Completely stopping the rockets launching operations.
- **Ignition:** Starting the rocket motors combustion process.
- **Liquid Oxidizer:**Liquid part of the rockets fuel.
- **Oxidizer Tank:** Tank were the liquid oxidizer is held.
- **Chamber:** Where the two fuels mix.
- **PSI:**Pounds per square inch. A unit measuring pressure.
- **Pa:** Pascals. A unit measuring pressure.
- **PWM:** Pulse Width Modulation - An electronic control signal used to change state of electrical components.
- **GPIO:** General Purpose Input Output lines are used to communicate to sensors and electronic components.
- **SPI:** Serial Peripheral Interface - A protocol to communicate with micro controllers.
- **I2C:** Inter-Integrate Circuit - A protocol to communicate with micro controllers.
- **OS:** Operating System - the main system running on a micro controller or a traditional computer.
- **LSM9DS0:** The LSM9DS0 is a system-in-package featuring a 3D digital linear acceleration sensor, a 3D digital angular rate sensor, and a 3D digital magnetic sensor.
- **TGY6114MD:**A powerful high specification digital sail winch servo with metal gears that can be programmed to operate from 1 to 6 turns.
- **BM180:**Digital pressure sensor.
- **API:**An application program interface is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact.
- **RAM:**Random Access Memory
- **Gauge:**A graphical device to display the readout of a sensor. For example a speedometer.
- **Serial Communication:**Data sent back to back through a medium.

1.4 References

Beagle Bone Black hardware specifications from elinux.org [http://elinux.org/Beagleboard:BeagleBoneBlack]
BMP180 Digital Pressure Sensor data sheet [https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf]
LSM9DS0 3D accelerometer, 3D gyroscope, 3D magnetometer data sheet http://www.st.com/content/ccc/resource/technical/document/datasheet/ab/2a/3b/45/f0/92/41/73/DM00087365.pdf/files/DM00087365.pdf/jcr:content/translations/en.DM00087365.pdf]

1.5 Overview

The rest of this document contains over all descriptions of the different components of this system. Followed by specific requirements and metrics of these components.

2 Overall Description

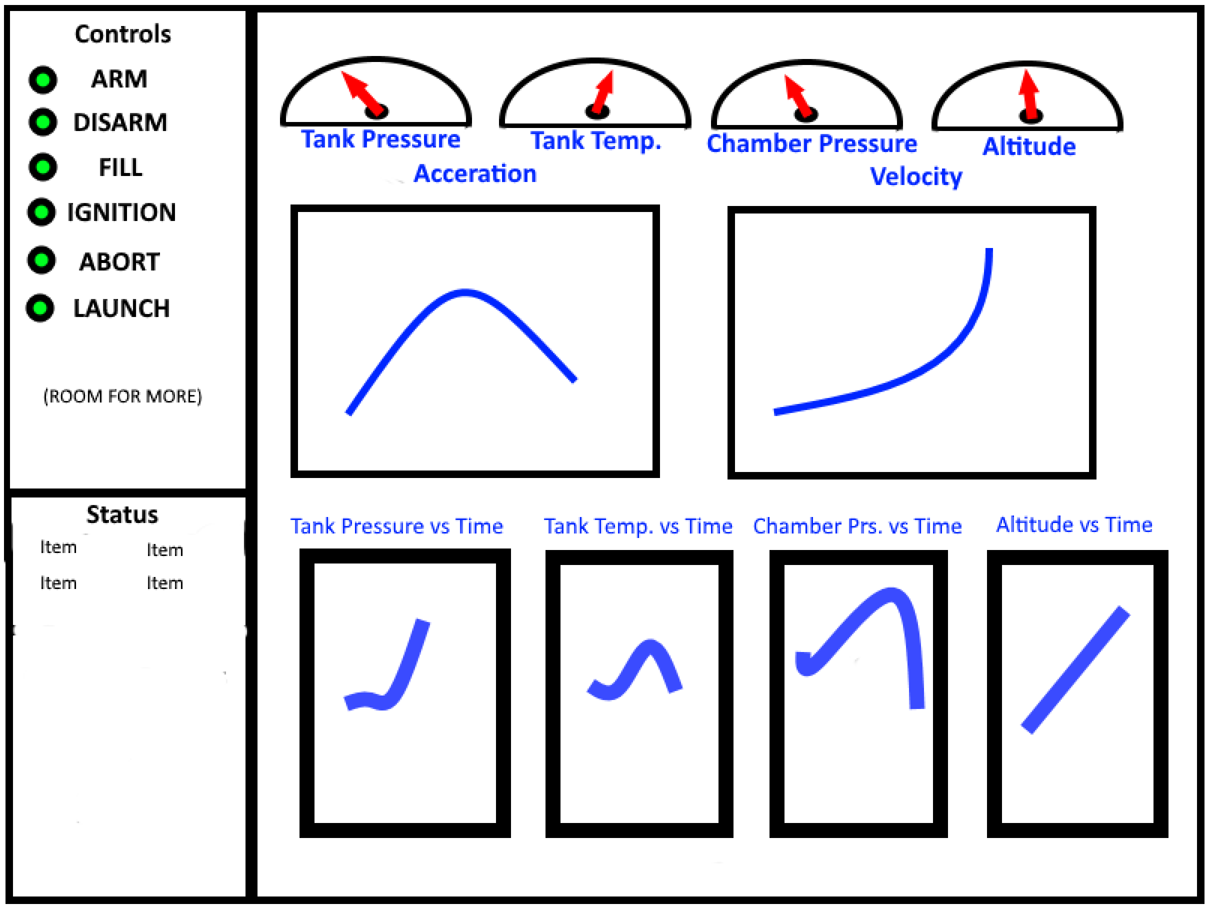
2.1 Product perspective

Our software components are part of the overall system of the Hybrid Rocket. These components encompass remotely filling, arming/disarming, aborting, ignition, recovery, gathering sensor data, and radio communication operations. On the traditional computer they encompass serial communication to the XBee radio transceiver, data loading and re-visualization, user input, and live user data visualization. Our software may need to calculate the rockets position through GPS. This at the moment is a stretch goal.

2.1.1 System Interfaces

The HyRo VS component of our system will house the user interface. This interface will be divided into a command menu, system diagnostics/status window, and different visualization widgets to display the data. The full graphical user interface will fill the entire screen. The command menu will house six buttons with the potential for more to be added if necessary. All commands have not been decided on because the rocket has yet to be made and requirements and

Figure 1: Basic Mock Up of User Interface for Hyro showing implementation of UI layout and positioning of objects like buttons, graphs, and dials for the user to process at a glance. This is a test mock up and may not be exact to the final product.



commands may be presented to our system as other teams design their parts of the hybrid rocket. The six buttons that are required at the moment are: Arm Rocket, Disarm Rocket, Fill Liquid Oxidizer, Launch, Abort Launch, and Ignition. The command menu will be located to the left of the screen and take up approximately 1/4 the width of the screen and approximately 1/2 the height of the screen. Buttons in the menu will be evenly spaced and sized.

Under the command menu on the left hand side of the screen will be the system diagnostics and status window. This window will take up 1/4 the width of the screen and 1/4 the height of the screen. This screen will report any message from the radio transceivers regarding completed commands or errors.

The visualization windows will be housed in a parent container that will encompass 3/4 of the screen width and the total height of the screen. Not every data item has been decided on by our the sub teams at the moment. If any more sensors are added to this system once it is developing these will be added to this document with and explanation as to how they will be visualized. The current group of data to be visualized includes: Chamber pressure, Chamber temperature, acceleration, barometric pressure, velocity. Stretch goal data includes GPS, tank pressure, and tank temperature. Each individual item might be represented as a gauge or graph.

2.1.2 Hardware Interfaces

Each of the three components to our software will require specific interfaces to different hardware aspects depending on which part of the system they are located in.

HyRo OS HyRo OS will communicate to the ECE subsystem through interprocess communication. These ECE subsystem collects data, moves servos that control various action internal to the rocket. HyRo OS includes radio frequency transmitter API for communication. The radio module used communicates to the BBB via serial communication. All sensor communication is to be designed be the ECE team and comprise the ECE subsystem.

HyRo VS HyRo VS will need to communicate to a serial port on the computer it is residing on. This will be done through either a Windows or Linux serial port access API. It will also interfaces with files on the traditional computers file system.

2.1.3 Software Interfaces

We will be interfacing various outside software products. These include the Debian Linux operating system running on the BBB, the Operating system running on the traditional computer which will be either Linux or Windows and possibly an external GPS program which is part of our stretch requirements.

2.1.4 Communication Interfaces

There are two interfaces for communication in our system and both require the use of a serial interface. The first is radio communication with a radio frequency transceiver. There will be one onboard the rocket connected to the BBB and one connected to the traditional computer via a USB interface. The serial communication is accessed through an API available for the XBee radio transceivers. We will design a protocol to transfer commands and data over these serial interfaces. Each command will have a name and parameters associated with it. Data will have a name associated with it followed by the actual data. All data will be passed in strings delineated by either colons or commas.

2.1.5 Memory Constraints

The HyRo VS aspect of our system will be constrained by the memory of the traditional computer it is located on. Available memory should be significant enough that we should not be constrained. This is because our program will have a small footprint and the data we log will not exceed 200 megabytes.

The HyRo OS will be constrained to 512 megabytes of RAM and 4 Gigabytes of onboard long term memory. Our onboard system will need to be able to run in the RAM available and not store over 4 gigabytes of data.

2.1.6 Operations

There are 4 modes of operation in our system: pre-launch, staging, in flight, and post flight. Pre-launch operations will be non-interactive and nothing will be displayed in the visualization windows.

Staging will be an interactive process where the user will provide input to various command buttons. The buttons will have to be executed in order, except for the abort button. Components will monitor user input and prime the ignition system via the ECE subsystem.This subsystem will not allow commadn to be issued in the wrong order. If the abort command is issued all other function will be overridden and the system will return to its pre-launch state. Once the sequence has been completed the software will switch to the in-flight mode. A support function that logs all commands to a log file will be implemented to record these actions.

The in-flight mode will disable the command menu and begin data collection from the onboard sensors. At this point all sensor data processing function will be activated. The data will be received from the serial port, parsed through the data processor, and sent to the appropriate data processing functions. These functions are responsible for visualizing the data on our graphs and gauges. These operations are to be monitored by the user for visual satisfaction. The data will also be logged by a logging function in a text format to be used by the post-flight mode. Post flight mode will be fully interactive. The command menu will still be disabled but data stored by the data logger will be able to be viewed in graphs that related the data over time of flight. These graphs will be viewable by selecting data to load from the data loading menu.

2.1.7 Site Adaption Requirements

When the hybrid rocket is ready to launch the software will be in pre-launch mode and a sequence of commands must be sent to the rocket in order for the rocket to be safely launched. This sequence is fill, arm, ignite, and launch. At any moment someone can cancel this sequence if the rocket is deemed not safe to launch by using the abort button.

2.2 Product functions

Our software as a whole will function as a way to communicate information from the hybrid rocket to a traditional computer on the ground. The ground team will interact with the ground computer to look at the information from the rocket and pass information back to the rocket. Both software components each have their own unique functions.

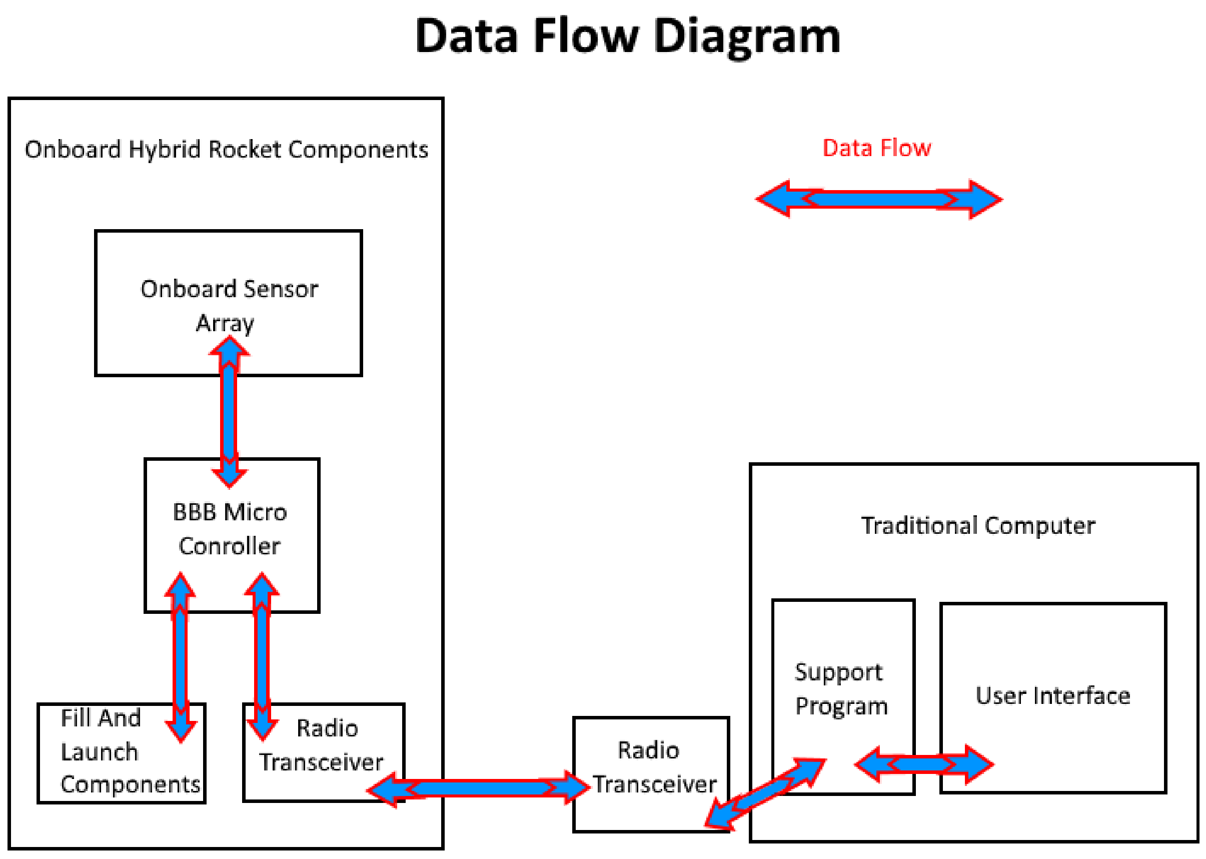
2.2.1 HyRo OS

- Send and Receive communications from a radio transceiver to communicate with the ground team.
- Collect sensor data from any onboard sensors when in flight.
- Send sensor data to the ground computer.
- Monitor fill and launch systems and send information to ground team.
- Receive commands from ground team.
- Send electrical signals or messages to launching components when appropriate command is received.
- Make sure commands are executed in correct order.

2.2.2 HyRo VS

- Send and Receive communication from a radio transceiver to communicate with the hybrid rocket.
- Provide response and error message window.
- Provide a menu of commands to stage, launch, or abort the rocket mission.
- Log all commands that have been sent.
- Collect data sent from the rocket and store it.
- Collect data from the rocket and visualize it on the screen in its appropriate gauge or graph while rocket is in flight.
- Provide a log of the data.
- Provide the ability to laod data into the program from previous flights.

Figure 2: Data Flow Between System Components.



2.3 User characteristics

Intended users of this software require a knowledge of rocketry and safe handling of launching a rocket. These users will be undergraduate students, graduate students, and professors included in the rocket team. They will have sufficient knowledge of the operation and safety requirements of launching a hybrid rocket according to the safety regulation of ESRA.

2.4 Constraints

2.4.1 Regulatory Policies

Operation of our software must stay in the bounds set by ESRA and the safe handling and operation of launching a hybrid rocket.

2.4.2 Hardware Limitations

HyRo OS will be onboard a BBB with only 512 megabytes of ram and 4 gigabytes of storage and must function under these restrictions. The BBB will also be inside of a volatile environment inside the rocket that could cause it to overheat.

Timing is also a major concern in communicating sensor data and commands. All communication between the rocket, its sensors, its launching components, and the ground team must be within 1 second or less.

2.4.3 Interfaces to other applications

If GPS is used we will have to output data to a GPS application in the correct format for that application.

2.4.4 Control Functions

Launching of the rocket must be monitored in the correct order or our software should not continue the sequence to launch the rocket.

2.4.5 Reliability Requirements

Data in our system must be transferred and interpreted in a complete and reliable fashion. Incomplete commands will be ignored. Incomplete sensor data will also be ignored. Data loss will not exceed 10 percent or our system will not be reliable.

2.4.6 Safety Considerations

Our software must not stage and launch the rocket without receiving the appropriate sequence of commands from the ground team. Failure to do so may result in harm to team members.

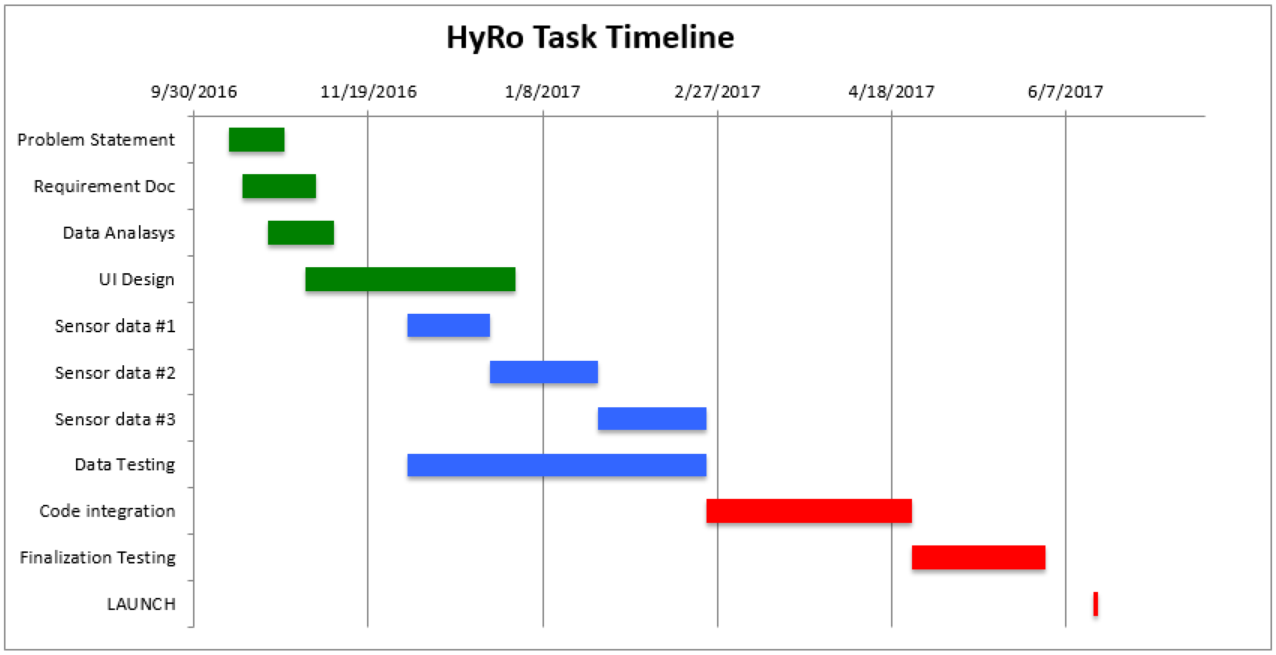
2.5 Assumptions and dependencies

Our onboard software system HyRo OS assumes that it will be installed on a Debian Linux OS that is running on a BBB. HyRo OS assumes it will be connect to a XBee radio transceiver, and the ECE sub system.

Our HyRo VS software running on a traditional computer assumes that it will be communicating with a XBee radio transceiver.

2.6 Apportioning of Requirements

Figure 3: Gantt chart for HyRo Workflow over the course of the project. This chart is highlighting overlapping responsibilities between team members that are broken into manageable chunks to ensure time management.



2.6.1 GPS - (Stretch Goal)

A GPS sensor might be installed on the rocket. This will require the HyRo OS to be able to communicate with this sensor and transmit its data. The HyRo VS then will need to be able to collect this data and either display it in a GPS window of our design or pass it onto to a GPS application that will interpret the data to find the rockets location.

3 Specific requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.1 Fill Command Button Input

The fill command input button will activate the fill servo to fill the chamber with oxidizer. This command is inputted from a button on the command menu. The command will be formatted in an English representation. This command must be sent within 1 second. This command must be sent before the arming command is given. Other commands will not function out of this sequence. It will be located in the command menu of the left hand side of the screen.

3.1.1.2 Arming Command Button Input

The arming command input button will send a command to the system that will arm the rocket. This command is inputted from a button on the command menu. The command will be formatted in an English representation. This command must be sent within 1 second. This command must be sent before the ignition command is given. Other commands will not function out of this sequence. It will be located in the command menu of the left hand side of the screen.

3.1.1.3 Ignition Command Button Input

The ignition command input button will activate the ignition system of the rocket. This command is inputted from a button on the command menu. The command will be formatted in an English representation. This command must be sent within 1 seconds. This command must be sent before the launch command is given. Other commands will not function out of this sequence. It will be located in the command menu of the left hand side of the screen.

3.1.1.4 Launch Command Button Input

The launch command input button will release the rocket when it is ignited. This command is inputted from a button on the command menu. The command will be formatted in an English representation. This command must be sent within 1 second. This command must be sent last in the sequence of commands. Other commands will not function out of this sequence. It will be located in the command menu of the left hand side of the screen.

3.1.1.5 Disarm Command Button Input

The disarm command input button will disable the arming of the rocket. This command is inputted from a button on the command menu. The command will be formatted in an English representation. This command must be sent within 1 second. This command can be issued anytime the rocket has been armed. It will be located in the command menu of the left hand side of the screen.

3.1.1.6 Abort Command Button Input

The abort command input button will return the system to its pre-launch state and stop the rocket from launching. This command is inputted from a button on the command menu. The command will be formatted in an English representation. This command must be sent within 1 second. This command can be issued at any time and is not effected by other command buttons. It will be located in the command menu of the left hand side of the screen.

3.1.1.7 Command Output to radio transceiver

Any time a command is received from the command menu the HyRo VS will send the command to the radio transceiver. The UI for the commands will all be formatted in plain English. Commands must be delivered in under 1 second.

3.1.1.8 System Status Input

The system status input receives error and command response data from the radio transceiver to be passed to the user interface. The status will be formatted with the item name appended by the status data from that part of the system. Status messages will be delivered in under 1 second.

3.1.1.9 Sensor Data Input

The sensor data input shall receive data from the radio transceiver to be output to the user interface. Each sensor will have its own respective units of measurement, that will be predetermined in the software, once sensors have been finalized. Data will be sent in raw form and need to be converted using predetermined conversion formulas for that particular sensor. Accuracy depends on the accuracy of the sensor that the data came from. Each input will be related to its respective graphical output in the user interface. The data will be formatted with a respective sensor name append by the data from that part of the sensor.

3.1.1.10 Sensor Data Output

The sensor data output will convert the data into a graph and display it on the user interface. It will have received this data from the sensor input. The data will either be graphed on a normal x-y-graph or adapted onto a gauge with a needle indicating current levels of the data within a range. The range will depend on the type of data received and be preprogrammed in the software. Units will depend on which sensor supplied the data. Each graph or gauge will take up a section of the right hand window of the user interface in its appropriate sub window.

3.1.2 Hardware interfaces

3.1.2.1 Radio Transceiver Input/Output

Both software components will have an interface to their respective XBee radio transceivers that will support data transfer between the rocket and the traditional computer. Any commands or data received from the various input sources will be placed onto this interface to be transmitted to the other radio transceiver. Data will be formatted with an English name representing what input it came from appended by the actual data separated by commas.

3.1.2.2 Sensor data Input

Sensor data input will be received from the various sensors on the system to be transmitted to the ground teams user interface. Accuracy of data depends on the accuracy of the sensors and the ability of the radio transceivers to communicate entire data package. When sensor data is inputted it is packaged and transmitted to the radio transceiver. Data format will be dictated by the format of the sensor output that has yet to be determined.

3.1.2.3 Launch Control Electronics Input/Output

This interface will monitor input from the electronic controls of the system and output their status to the radio transceiver. Data format of this part will be component name appended by the state of the component. Commands will be received to change the state of these components and appropriate electronic signal will be outputted to corresponding electronic components by the ECE subsystem. Timing is critical and input and output must be performed in under 1 second.

3.1.3 Software Interfaces

3.1.3.1 Linux OS Interface

This interface will need to connect to the operating systems UART lines to communicate with the XBee. It will also be running in the Linux environment and follow rules pertaining to that environment.

3.1.4 Communication Interfaces

3.1.4.1 Serial Communication Interface

All communication except for to the electronic launch components will be done via serial ports. Input and Output to these ports will be done through an API provided by the XBee's manufactures.

3.2 Functions

3.2.1 Command Input

3.2.1.1 Requirement 1

Inputs must be done in previously mentioned sequence or are otherwise considered invalid.

3.2.1.2 Requirement 2

If command is not delivered user will have to repeat the command input.

3.2.1.3 Requirement 3

If the rocket components are not ready for a particular input, the command will be ignored.

3.2.2 Command Output

3.2.2.1 Requirement 1

Sequence of the command will be checked before output.

3.2.2.2 Requirement 2

Command will be check against programmed commands to make sure that is valid.

3.2.2.3 Requirement 3

If a command is received out of sequence or is invalid the system will report back to the user interface that the command could not be completed.

3.2.2.4 Requirement 4

If a command is successfully output, status is returned to user interface to allow for next command to be inputted.

3.2.2.5 Requirement 1

Sensor input will be checked for completeness according to the data sheet of the sensor outputting the data.

3.2.2.6 Requirement 2

Sensor input that is found to be invalid will be dismissed. Data loss tolerance, within 10 percent, is acceptable in order to meet timing requirements.

3.2.2.7 Requirement 3

Sensor data will be gathered in a loop and packed into a packet to be sent to the radio transceiver.

3.2.3 Sensor Output

3.2.3.1 Requirement 1

Sensor output will be displayed on the user interface in graphical format.

3.2.3.2 Requirement 2

Sensor data will arrive in time and in order. This requirement will allow the data to be displayed in a time oriented graph.

3.2.3.3 Requirement 3

Sensor data will be converted to its output format depending on the type of data supplied by the sensor. The exact formulas are yet to be determined until further work is done on designing the hybrid rocket.

3.2.3.4 Requirement 4

Sensor data will not be outputted if it is found to contain errors or is out of bounds of the range of the sensor.

3.3 Performance Requirements

There will be only one user interface open at any given time. Supporting the interaction of only one user at a time. The system will need to be able to handle data in the amount of about 100 megabytes a minute. This measurement is from last year's data and may change. 95 percent of received data packets need to be processed in under 1 second. 100 percent of command data packets need to be processed in under 1 second.

3.4 Logical database requirements

We will not be interacting with a logical database in this project.

3.5 Design constraints

We will have limited system memory on the BBB. Our program must run in under 512 megabytes of RAM, and not log over 3 gigs of data on the onboard system. The BBB will be in a heat intense environment, but the Mechanical engineering team is responsible for providing heat shielding to our unit.

3.6 Design Metrics

Our project inherently has metrics involving correct data acquisition, the validity of data, the speed of transferring data, the accuracy of data being displayed on the screen, and the correctness of saving the data. Each of these we can place a unit of measure on.

- **Correct Data Acquisition -** If the data the sensors or actually producing matches the data that our program gathers to process we can call our data acquisition a success. To measure this we will have to take measurements with analog, or other known accurate digital devices of the same thing our sensors our. If we control the environment on the data matches up we know this part of our program is accurate.
- **Speed of Transferring Data -** The data must be collected and transfered within a reasonable amount of time. From rocket to ground unit data should be delivered within one second. There are a lot of factors involved and each one will have its own speed metric. These metrics will be listed here thoroughly when the complete rocket system is built.
- **Validity of the Data -** If we somehow introduce bit errors into our data or do not convert and process it correctly it will be invalid. To test for this will be the same as for data acquisition only this is at a different stage of our program.
- **Accuracy of the Data On Screen -** After data has been transferred and processed we need to accurately represent it on the screen. Each data element will be measured in its own units i.e pounds per square inch. These units need to be accurately represented on the screen to scale. When data arrives and is draw on the screen it needs to be within 1 percent of the actual values. To test this we will compare our on screen values to values processed by another pogram. Similar to above we need to verify we give the same answer as everybody else.
- **Correctness of Saved Data -** The logged data needs to match the screen data that was displayed.

3.7 Standards compliance

There are no explicit standards or regulations imposed on our project.

3.8 Software System Attributes

3.8.1 Reliability

Reliability will be measured with time constraints, correct data transfer, correct data visualization, and correct communication to launch the rocket. Timing is critical to many components and data needs to be communicated in less than 1 second for most components in order for the system to perform correctly. Data needs to be accurate or commands and visualization will not return expected results. Sequences must be followed to insure safety and proper launching of the hybrid rocket.

3.8.2 Availability

Our HyRo VS system must maintain communication with the onboard HyRo OS system in order to maintain availability of the user interface. If data communication is broken the interface will not be able to allow humans to issue commands or see visualized data. If communication is broken the program will attempt to reestablish connection until it either establishes communication or the user exits the interface.

3.8.3 Security

We have no security constraints imposed on our project.

3.8.4 Maintainability

The software should be well documented and modularized so next year's rocket team can build off of it with ease. Classes and inheritance will be used to support object oriented approaches.

3.8.5 Portability

HyRo OS will only be portable too other Debian Linux system since the APIs used to access system components are built for this type of OS. The HyRo VS is to be written in Python which is portable to many OS's including Windows, Linux and mac OS.

4 Index

Index

Students:

Jason Klindtworth

Signature

Date

Josh Asher

Signature

Date

Layne Nolli

Signature

Date

Client:

Nancy Squires

Signature

Date