PSAS Capstone Project:
Reaction Control System Software Requirements Document

05/06/2016

Document Version #1

Reaction Control System Software

Table of Contents

1 Introduction

The Portland State Aerospace Society (PSAS) is a student aerospace engineering group working out of Portland State University - their goal is to design and produce sophisticated low-cost open source rockets. One of the PSAS's newest developments is the Reaction Control System (RCS). This system is comprised of a mechanical system that utilizes cold-gas jets to provide flight stabilization of the rocket in flight, as well as software used to control these jets.

Our group, the RCS Software Development Team (SDT), will be re-implementing the existing python software for the RCS using a new programming language called Rust. In addition to the control software for the rocket, the SDT shall also be developing a testing framework for the simulation of a rocket in flight. This testing framework shall be making use of the JSBSim API, which will be used to provide the simulated physics for our testing framework, as well as processing the physics data provided by our testing framework. One of the main features of the new RCS software shall be its ability to be used in both flight mode and testing mode. During compile time, the User shall be able to specify the software mode of the compiled binary. Features that are only necessary for the respective software modes will be included during compile time.

1.1 Purpose and Scope

This document will provide a high-level view of the software that our group will develop. In this document, you will find the basic expected functionalities of the software requested by PSAS, and agreed to by our software development group. Topics covered in this document are descriptions of the User and Stakeholders in this project, Use Cases, Functional Requirements, Non-Functional Requirements, Verification Methods, and

Milestones & Deliverables. Details of this document will be subject to change during the course of software development.

1.2 Team Members

Brian Breniser (breniser@pdx.edu)	Team Lead
Cort Alexander (Cort@pdx.edu)	Sponsor Liaison/ Backup Lead
Paul Lee (pclee@pdx.edu)	Requirements and Specifications Documenter
Tyler Alway (alway@pdx.edu)	Risk Management Documenter
Chris Liebert (liebert@pdx.edu)	Project Schedule Documenter
Sohail Nayani (snayani@pdx.edu)	Communication Documenter
Harrison Bailey (hgb2@pdx.edu)	IT/DevOps

1.3 Target Audience

This document is to inform our Sponsor, the Portland State Aerospace Society, the details surrounding the software to be developed.

1.4 Document History

Document version #1: Initial Document

1.5 Approval

Signatures made below by the Software Development Team Lead, and by key Stakeholders indicate an agreement by both parties of the software specifications detailed in this document. Details of this document are subject to change by either the Software Development Team and Stakeholders; in such an event, modifications shall be made to a new draft of this document. Each document version must undergo signed approval by the Software Development Team Lead, and by Key Stakeholders.

Brian Breniser (Team Lead):_____

Theo Hill (Key Stakeholder):_____

Jamey Sharp (Key Stakeholder):_____

1.6 Terms and Definitions

1.6.1 Reaction Control System (RCS)

A general term to describe a hardware and software based system that utilizes cold-gas jets to control a rocket in flight.

1.6.2 RCS Prototype

This describes the current hardware prototype of the Reaction Control System. The final deliverable software shall be able to control the RCS prototype hardware actuators.

1.6.3 RCS Software

This describes the software the SDT shall be providing at the conclusion of this project.

1.6.4 Core Components

This will be a general term used to describe “common components (see Figure #1)” of software used in both the flight mode and the testing mode (see Figure #1). This portion of the code will take in sensor data (simulated or real), determine any course corrections that are needed, and signal the controller interface (simulated or real).

1.6.5 Testing Harness

This encapsulates software components only used during test mode (see Figure #1). This portion of the code can be included or excluded during compile time.

1.6.6 Sensor Hardware

In the prototype RCS, the only sensor is an Inertial Measurement Unit (IMU). Sensors relevant to our project are the gyroscope, accelerometer and the magnetometer.

1.6.7 Sensor Interface

During testing mode, the Sensor Interface shall be able to take simulated physics from JSBSim, and produce realistic sensor outputs for the Core Components to utilize.

1.6.8 Controller Hardware

In the prototype RCS, the cold-gas jets take the form of solenoid actuators. These actuators or controlled indirectly by the Core Components software via GPIO pins.

1.6.9 Controller Interface

During testing mode, the Controller Interface shall be able to receive instructions from the Core Components.

1.6.10 Other Terms

GPIO	General Purpose I/O
I2C	Inter-Integrated Circuit
Users	PSAS members and the general public.
Sponsors	The PSAS group
SDT	The RCS Software Development Team

2 Product Overview

The RCS Software to be developed for this project shall be a Rust based re-implementation of the existing RCS software written in Python. The software developed by the SDT will also include a testing framework that will allow users of this program to simulate various launch conditions.

Refer to the “Reaction Control System (RCS) Software Architecture” document for a detailed description of the software to be developed.

2.1 Users and Stakeholders

Here we define the role of users and Stakeholders with this software. Who will be using, deploying, developing the software are defined here.

2.1.1 PSAS Members

PSAS members who use this code will be expected to have knowledge of how to compile software from source code. The RCS Software Development Team shall not provide any further support beyond descriptions of how to compile and run the program.

2.1.2 RCS Software Development Team

The software development group is comprised of the members listed in Section 1.2. The RCS Software Development Team shall be responsible for the design and development of the RUST implementation of the RCS, as well the Testing Harness.

2.1.3 General Public

The software we are producing shall be licensed under the GPL2 open source licence. Therefore other users of this software will be the general public. These users will be expected to have knowledge of how to compiling software from source code. RCS

Software Development Team shall not provide any further support beyond descriptions of how to compile our code run the program.

2.2 Use Cases

This section will describe scenarios of how the users will interact with our software.

2.2.1 Testing Mode

Use Case Description: During compile time, the User shall be able specify the build of the software in Testing mode.
Step-by-Step Description: 1. User will compile the RCS software at compile time to specify Testing Mode. 2. User will then initiate the compiled program. 3. After program termination, the User shall be able to retrieve telemetry data created during test mode. **Exit Condition:** Program will self-terminate, or will terminate upon User specified condition.

2.2.2 Flight Mode

Use Case Description: During compile time, the User shall be able specify the build of the software in Flight mode.
Step-by-Step Description: 1. User will compile the RCS software at compile time to specify Flight Mode. 2. User will load the Compiled program into the RCS Prototype. 3. Initiation of the RCS software in flight mode shall be handled during the Rocket launch protocol. **Exit Condition:** Program will self-terminate, or will terminate upon User specified condition.

3 Functional Requirements

Covered in this section are the functional requirements requested by the Sponsor and agreed upon by the RCS Software Development Team.

3.1 Compile Time Flight/Test Mode Determination

At compile time, the user shall be able include or exclude the Testing Harness from the compiled program. Inclusion of the Testing Harness shall be a property of the program in Test mode, while exclusion of the Testing Harness shall be a property of the program in Flight mode.

3.2 Test Mode and Flight Mode

In the following sub-paragraphs, we outline requirements for various modules shown in Figure #1.

3.2.1 JSBSim

During Test Mode, the Testing Harness shall include the use of the JSBSim API. The Sensor Interface shall be able to make use of simulated physics provided by JSBSim. The Controller Interface shall be able provide data that can be handled by JSBSim.

3.2.2 Sensor Interface

The Sensor Interface shall be a simulated IMU sensor. The Sensor Interface shall receive simulated physics data from JSBSim, and produce realistic sensor data that shall be sent to Core Components.

3.2.3 Controller Interface

The controller interface shall be a simulated solenoid actuator. The controller interface shall receive instructions from the Core Components, and produce realistic response normally made by the solenoid actuators that shall be sent to JSBSim.

3.2.4 Core Components

The Core Components shall be able to take in sensor data, determine the correct actions needed for rocket stabilization, and produce a signal for the solenoid actuators (in flight mode) and the controller interface (in test mode). During Testing mode telemetry data of the rocket in flight for various launch scenarios shall be saved onto a disk drive.

3.3 Flight mode

The RCS shall not be able to access the Testing Harness during Flight mode.

3.4 Telemetry Data

Telemetry data shall be in the format specified by the PSAS packet serializer.

4 Nonfunctional Requirements

Covered in this section are the non-functional requirements requested by the Sponsor and agreed upon by the RCS Software Development Team.

4.1 Platform

During Flight mode, the RCS software shall run in a Linux Debian system environment. During Test mode, the Software shall run on any Linux based system or a Windows based system.

4.2 Testability

In the event multithreading is used during software development, the output of a program shall be reproducible regardless of the order in which threads are scheduled/preempted.

In the event random numbers are used during software development, every seed value used with a pseudo-random number generator shall always produce the exact same output for multiple executions with the same seed value.

4.3 Maintainability

The software analogs for the Controllers and Sensors shall be of modular design to support the addition or removal of Controllers and Sensors in the future by PSAS.

4.4 License/Open Source

The RCS Software being developed in this project shall be licensed under the GPL2 license.

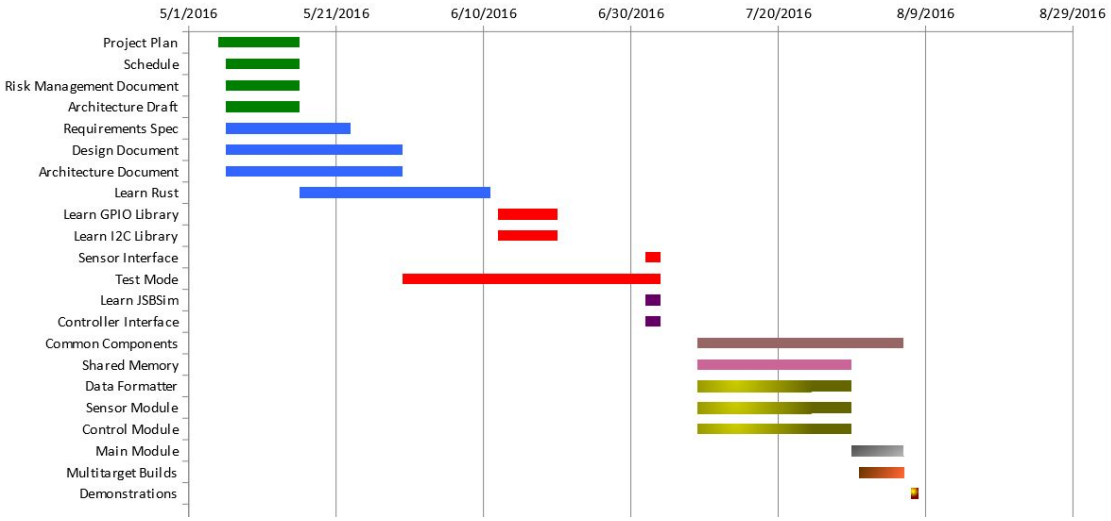
5 Verification

Here we outline verification methods for the Functional and Nonfunctional Requirements listed in Sections 3 and 4 of this document.

Table 1: Verification Methods

Paragraph	Requirement	Verification Method
3.1	User shall be able to compile the program for Flight or Test mode by respectively excluding or including the Testing Harness.	Inspection
3.2.1	Testing Harness shall be used in conjunction with the JSBSim during Test mode. Testing Harness shall be able to handle data provided by JSBSim. Data sent to JSBSim shall be of the proper format.	Testing
3.2.2	Sensor Interface shall be able to handle simulated physics from JSBSim and produce the appropriate simulated sensor output.	Testing
3.2.3	Controller Interface shall be able to receive instructions from Core Components.	Testing
3.2.4	Core Components shall be able to receive real/simulated sensor data, determine what needs to be done by the actuators, and produce instructions for the actuators/controller interface.	Testing
3.2.4	During test mode, Core Components shall write telemetry data onto a disk drive.	Testing
3.3	The RCS software shall not be able to access the Testing Harness accidentally or otherwise during Flight Mode.	Testing
3.4	Telemetry data produced by the RCS software shall be in a format consistent with the PSAS packet serializer	Testing
4.1	During flight mode, the RCS software shall be able to run on a Linux Debian system. During test mode, the RCS software shall be able to run on any Linux based system or a Windows 7/10 system	Testing
4.2	Results and responses from the RCS software shall be able to produce reproducible output in a multithread environment and if random variables are used.	Testing
4.3	The RCS software shall support the addition/removal of actuators and sensors.	Testing
4.4	The RCS software shall be licensed under the GL2 licence.	Inspection

6 Milestones and Deliverables



Date	Milestone
5/16/2016	Risk Management Document
5/16/2016	Architecture Draft
5/23/2016	Requirements Spec
5/30/2016	Design Document
5/30/2016	Architecture Document
6/11/2016	Learn Rust
6/20/2016	Learn GPIO Library
6/20/2016	Learn I2C Library
7/4/2016	Sensor Interface
7/4/2016	Test Mode

Date	Milestone
7/4/2016	Learn JSBSim
7/4/2016	Controller Interface
8/6/2016	Common Components
7/30/2016	Shared Memory
7/30/2016	Data Formatter
7/30/2016	Sensor Module
7/30/2016	Control Module
8/6/2016	Main Module
8/6/2016	Multitarget Builds
8/8/2016	Demonstrations

6.1 Project Design

The RCS design will be presented in document form consisting of an overview of the system components, both hardware and software.

6.1.1 Architecture Document

The architecture document will provided a basis for the implementation of core components which are identified and described at a high level.

6.1.2 Design Document

Pseudo-code will be provided as a basis for implementation of core components which are identified and described in the Architecture document.

6.2 Stretch Goals

Here we define extra software or features requested by the Sponsor.

6.2.1 Satellite Reaction Wheel

The satellite reaction wheel is a system similar to the RCS. While the hardware used to implement this system is very different from the RCS, the software used to control the satellite reaction wheel will be very similar to that of the RCS.

6.2.2 Data Visualization Software

Data visualization software will take the form of a visual display of the data that will be produced by the telemetry simulated or provided by our Testing Harness or from the rocket during flight.

6.3 Final Deliverables

A demonstration of our software working in conjunction with the RCS prototype shall be delivered on August 8, 2016. We shall also demonstrate that the software running in test mode shall perform all necessary functionalities specified in the functional and nonfunctional requirements section of this document.

7 Figures

Figure #1: