PSAS RCS Capstone Presentation

Team: Brian Breniser - Chris Liebert - Paul Lee - Harrison Bailey - Cort Alexander - Sohail Nayani - Tyler Alway

Sponsor: PSAS
Project Organization

Brian Breniser - Lead & Embedded library architect
Chris Liebert - Scheduler & Test library architect
Paul Lee - Requirements guru & Telemetry data architect
Harrison Bailey - DevOps / Backups engineer & Main logic architect
Cort Alexander - Backup lead & Control module architect
Sohail Nayani - JSBSim specialist & Test library architect
Tyler Alway - Risk Management & Sensor module architect
Project Overview

The sponsoring group is PSAS - Portland State Aeronautical Society
Our sponsor contacts are: Jamey Sharp, Theo Bailey

Reaction Control System aka Flight Controller:
- Software used to control and monitor the aircraft
- Keeps the rocket stable in flight and provide roll control
- Written in Rust

Why is this required?
- Transferring from fin based controller to cold gas jet based controller, allowing aircraft to be controlled in thin atmosphere
- Want to improve testability for existing design
Rust

What is Rust?
- New open-source language sponsored by Mozilla
- Strongly statically typed, compiled, cross-platform language

Advantages
- Memory safe WITHOUT garbage collection
- Comparable speeds to C, C++
- LLVM backend which allows for powerful optimization

Gotchas
- No one on the team has prior Rust experience
- New concepts to learn, such as ownership and borrowing of variables
Requirements Summary (1/2)

Software shall be able to receive sensor data, determine the correct response for rocket roll-stabilization, and send the appropriate control signal to cold-gas actuators.

Software shall be of modular design to allow for the addition/removal of hardware components.

User shall be able to include/exclude testing features during compile time:
- Flight mode shall not compile with testing features
- Test mode shall compile with testing features
Requirements Summary (2/2)

During Test Mode:
- JSBsim shall be utilized to provide and compute physics data.
- Sensor and actuator hardware shall be simulated and produce realistic responses.

Stretch Goals
- Satellite reaction wheel
- Telemetry visualization software
Project Plan & Schedule (1/2)

May 2nd - June 6th:
- Work Breakdown Structure
- Risk Management
- Requirements
- Design
- Architecture
- Project Plan

Iterative Development:
- Iteration every 3 weeks
- First iteration
  - Common components
  - Test Flight Interface on Prototype
  - Software Test on Hardware Components
- Second iteration
  - Optimization
  - Refinement
- Third iteration
  - Finalization
  - Stretch goals
Architecture

- Common components reduce variation between flight & test modes
- Flight mode uses embedded libraries to communicate with hardware
- Test mode uses JSBSim to model hardware responses
V&V Plan

Unit Testing in Rust:
- Mark test with `#[test]` & run with cargo test

Flight Mode:
- Test on individual prototype hardware
- System integration and operations

Test Mode:
- Use JSBSim to simulate the physics of real hardware
- Produce realistic and reproducible tests
- Verify consistent output between test mode and flight controller
1. Requirement inflation
   ● Mitigation factor: Break project into core requirements and stretch goals

2. Under estimation of time necessary to complete certain tasks
   ● Mitigation factor: Early feedback from members and allow for re-estimating of time needed for specific tasks. A couple weeks of padding at the end of the project

3. Data loss
   ● Mitigation factor: Multiple backups
Risk Management (2/2)

4. Misinterpreting client specifications/requirements
   - Mitigation factor: Maintain clear communication, multiple versions/iterations of project

5. Poor implementation (buggy)
   - Mitigation factor: Lots of testing and an iterative development process
Summary

We get to work with rockets!

Not many opportunities to build a flight controller!

We get to program in Rust, the most loved language on Stackoverflow 2016

Working with PSAS has already been awesome
Questions?