CS Capstone Technology Review

November 8, 2018

High Altitude Rocketry Project

Prepared for

OSU American Institute of Aeronautics and Astronautics (AIAA)

Dr. Nancy Squires

Signature Date

Prepared by

Group 11

Matthew Forsland

Signature Date

Abstract

This document outlines three major design choices faced in the implementation of the 2018-19 HART groundstation. It explores possible solutions to each design choice, examining advantages and disadvantages presented by each solution. A recommendation for the design choice described in a section is made at the end of the relevant section, which explains the reasoning for that choice.

CONTENTS

1 Introduction

This document proposes solutions for three problems faced in the development of the project:

- The software language the groundstation will be implemented in.
- The algorithm used to eliminate noise in received telemetry.
- The format that the telemetry is stored in for later use.

This document will explain the underlying advantages and disadvantages to each proposed solution for these three problems. This document is meant to act as an aid in deciding which solutions would be most beneficial to building our project while accommodating client expectations for the end product.

1.1 Glossary

- **COCOM**[1] An abbreviation for "Coordinating Committee for Multilateral Export Controls." It ceased to function in 1994, but has a legacy of GPS restrictions preventing GPS-guided ballistic missiles. These restrictions will affect the telemetry available for a period of the rocket's flight when it exceeds 515 meters per second and 18 kilometers altitude above ground level.
- **Groundstation** The groundstation will receive telemetry data broadcast by the rocket during flight. Because the rocket will exceed 5,000 feet above ground level, the point at which it becomes invisible to the human eye, broadcasting telemetry is necessary for the rocket to be tracked and its state to be known.
- **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.

2 Ground Station Software Language

The ground station software will be run on a Raspberry Pi computer running a Linux Operating System. The software language the software is written in must be able to run on this hardware. The choice of language will also influence the available libraries and systems to implement the GUI used to display data.

2.1 C

C is an imperative programming language that was first released in 1973[2]. It has since become one of the most prominent languages currently in use. C lacks non-primitive data types, so any data structures used in the project must be manually defined. C can be compiled to run on almost any system. It also has a large volume of third-party libraries that can be used to implement or augment certain algorithms used in the project. It requires manual memory management, reducing the amount of memory necessary for the project to function. It's the most efficient language presented here, and so most suitable for realtime data collection.

2.2 C++

C++ is a child of C, which introduces object-oriented programming concepts to C. Like C, C++ can be compiled to run on a wide variety of systems. C++ also requires manual memory management, but with a lesser degree of freedom than C. Because C++ supports object-oriented programming however, the functions 'new' and 'delete' are used to automatically allocate and free memory for data structures. C++ has native libraries that give support for a number of non-primitive data structures, like dynamic-length strings and vectors.

2.3 C#

C#[3] is a multi-paradigm programming language that is syntactically similar to C, but was developed by Microsoft. C# offers Object Oriented Programming, like C++. Unlike the C family of languages, it offers automatic memory management and garbage collection. Native libraries offer support for functional programming. C# is less efficient than C and C++.

2.4 Java

Java is a fully Object Oriented language, and is generally similar to C#. Java runs in its own environment, which allows it to be run on most systems. Java automatically manages all memory, including allocation and garbage collection. Command line arguments can improve the efficiency of this management, but it is generally less efficient than the C family of languages.

2.5 Python

Python is an interpreted language featuring dynamically-typed variables. Python is interpreted by compiled C code. Because it is interpreted, it is significantly slower than all other solutions listed here, which are compiled languages. Python provides automatic memory management. There is a very large number[4] of native and third-party libraries that can be used to implement or augment certain algorithms used in the project.

2.6 Language Recommendation

Because we are receiving live telemetry broadcast from the rocket, program efficiency is a very high priority. The rocket will transmit data at a known rate, and whether on one or more threads, the groundstation must be ready to receive and process the new data, or at least store it for later reference. C and C++ are favorable solutions to this requirement, and Python is unfavorable. All languages proposed here offer a large number of native and third-party libraries that may contain useful functions for the project, as well as very active communities that can offer support on any issues we encounter. Implementation of mathematical operations on received telemetry is therefore not reliant upon selection of language from those proposed. The project does not stand to benefit greatly from object-oriented programming principles, such as inheritance and encapsulation. Classes, and the related program architecture, are more useful for organizing multiple groupings of functionality. Our project is more concerned with the processing of inter-related data and its storage in memory. As such, I recommend the use of C to implement the groundstation software.

3 Telemetry Noise Reduction

Any data measured by the rocket will have statistical noise and other discrepancies. We must use an algorithm to account for this noise to improve the visualization of the rocket's state.

3.1 Kalman Filter

Kalman Filtering[5], or Linear Quadratic Estimation, is an algorithm that models the state of a set of variables. It uses a joint probability distribution to reduce statistical noise in the sources of those variables. A Kalman filter makes a prediction for the next set of inputs and their uncertainties, then applies variable weights to the real values of the inputs based on the probabilities of those inputs occurring given the prediction. The algorithm requires only the present input measurements, the previously calculated state, and an uncertainty matrix.

3.2 Hidden Markov Model

A Hidden Markov Model is a stochastic model used to model randomly changing systems. It assumes that the real state is not directly visible, but can be measured. Each state has a probability distribution over possible measurements, so a sequence of measurements can produce a joint-probability distribution to determine the most likely sequence of real states. This algorithm is simpler to implement than a Kalman filter, but requires extensive flight testing to properly generate joint-probability distributions for measurements.

3.3 No Noise-reducing Algorithm

If no algorithm were used, the telemetry received from the rocket would be sent directly to the visualization software. This is the simplest possible approach, but will produce noisy and incomplete data. When the rocket passes Mach 1, a supersonic shockwave will form over the rocket, causing a large pressure spike that barometric altitude readings will measure as a momentary drop to at-or-below ground-level. Furthermore, when the rocket exceeds a 515 m/s or 18 kilometers in altitude[1], the GPS will not provide data due to COCOM restrictions put in place to prevent GPS-guided ballistic missiles.

3.4 Algorithm Recommendation

The teams developing the groundstation for the previous two years' HART teams chose to implement Kalman filters, and I recommend that we do the same. This provide access to people with knowledge of the design and implementation of the filter, and access to examples of the implemented algorithm. Kalman filters are also used extensively in other applications in the aerospace industry[5], and the method of noise reduction is much more calculated. Rather than training a joint-probability distribution with extensive tests, we can generate equations for the uncertainties of measurements from different devices based upon their technical documentation. For example, due to COCOM[1] restrictions, the GPS will not produce useful data above 515 meters per second and 18 kilometers altitude above ground level. The filter can mask out this missing data, or fake data produced manually to give to the algorithm, by presuming that GPS data is completely unreliable and should not be used at all when the modeled state is above this threshold.

4 Telemetry Logging

As this year's effort is only a stepping stone toward the ultimate goal of launching a student-built rocket to the Karman Line, it is critical to make all the data gathered available to future teams in order to inform their subsequent designs.

4.1 CSV

The .csv file format separates data values by commas. The order of data on each line must be known beforehand to make useful interpretation of the data, but requires minimal parsing.

4.2 JSON

The data can be stored as a JSON Object, which is intended for usage by web applications. Similar to .csv formatting, the structure of the data must be known beforehand. JSON objects require more parsing by non-web-based applications to extract data.

4.3 XML

Extensible Markup Language (XML) is a markup language that encodes documents in both a human-readable and machine-readable format. The structure is generally similar to HTML in format. This represents the most parsing of the offered options to extract data.

4.4 Format Recommendation

JSON and CSV present similar formatting and parsing requirements. Because the visualization software will be implemented in a web-interface, making the data accessible to client-side web-based software is a priority. JSON is designed for this application, and as such is the recommended solution to this problem. XML requires much more parsing to extract data, and is less reliant on the format of the data being known a priori, so it is not recommended.

References

[1] C. C. for Multilateral Export Controls, “Cocom gps tracking limits,” 2010. [Online]. Available: http://ravtrack.com/GPStracking/cocom-gps-tracking-limits/469/

[2] D. M. Ritchie, “The development of the c language*,” 2003. [Online]. Available: http://www.bell-labs.com/usr/dmr/www/chist.html

[3] Wagner, Latham, Wenzel, Dykstra, Kulikov, tompratt AQ, Solarin-Sodara, Carter, Hoffman, joaocns0, jaredsimpson, and Al-Hashmi, “C# guide,” 2018. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/

[4] P. Piotrowksi, “Build a rapid web development environment for python server pages and oracle,” 2006. [Online]. Available: https://www.oracle.com/technetwork/articles/piotrowski-pythoncore-084049.html

[5] P. Zarchan and H. Musoff, *Fundamentals of Kalman Filtering: A Practical Approach.* American Institute of Aeronautics and Astronautics, Incorporated, 2000.