# Groundstation: Design

**Team #25**
**High-Altitude Rocketry Challenge**
Natasha Anisimova
Terrance Lee
Albert Morgan

**Abstract**

The *Groundstation* software will collect telemetry from a rocket while is in flight and graphically display the telemetry in real-time. Groundstation is made up several different components: collection of data, storage of data, interpolation of data, and display of data. This document will describe in detail the design of Groundstation.

CONTENTS

**I Introduction** 3
- I-A Identification . . . 3
- I-B Date of Issue . . . 3
- I-C Scope . . . 3
- I-D Stakeholders . . . 3

**II Structure View** 3

**III Interaction** 7

**IV Algorithm** 7

**V Glossary** 7

**References** 7

# I. Introduction

## A. Identification

This document will provide and in-depth software design description of the Groundstation software package.

## B. Date of Issue

This document is issued on December 2, 2016. At this time, the design of the software is complete, but no work has begun on the implementation.

## C. Scope

This document will provide details about the Groundstation software package. The concerns of the software will begin when the telemetry is received via the serial port on the Raspberry Pi. Collecting the data from the rocket and converting it into a protocol readable through the serial port is the concern of the Oregon State University (OSU) American Institute of Aeronautics and Astronautics (AIAA) High-Alititude Rocketry Team avionics section.

## D. Stakeholders

The stakeholders of the Groundstation software package are members of the OSU AIAA High-Altitude Rocketry Team. The OSU AIAA High-altitude rocketry team is a group of multidisciplinary engineering students who are working to together to build the rocket. The stakeholders require the ability to:

- View the telemetry in real time in order to track the altitude and aid in recovery
- View the data graphically so that it is easy to understand
- Log the data so that it may be analyzed at a later date

# II. Structure View

**PC**
Entity
*Author:* Your name here
*Type:* Component
*Purpose:* Why does it exists?
*Contents:* I'M MAKING CHANGES!

**Package Manager**
Entity
*Author:* Albert Morgan
*Type:* Subprogram
*Purpose:* The package manager will install, track, and update software dependencies on the server.
*Contents:* Because Groundstation will be using Node and JavaScript for both the frontend and backend, Node Package Manager (NPM) will be used [1]. NPM has a large repository of both server-side and client-side JavaScript packages.

**Frontend**
Entity
*Author:* Natasha Anisimova
*Type:* Component
*Purpose:* User interface
*Contents:* Stuff here.

**Backend**
Entity
*Author:* Albert Morgan
*Type:* Component
*Purpose:* The purpose of the backend is the facilitate communication between the rocket and the user.
*Contents:* The backend takes care of all data collection, transformation, logging, and serving that data to the user. A Node server will handle reading the data from the serial port, transforming the data into JSON, storing the data, and making the data available to the user.

**Node**
Entity
*Author:* Albert Morgan
*Type:* Subprogram
*Purpose:* Node is the software that runs the backend.
*Contents:* The backend will run on Node. Several choices were considered to run the backend, particularly Node, PHP, and Ruby. The backend was chosen two criteria:

- Speed. The backend will run on a Raspberry Pi, so speed is important to minimize the system resources used.
- Interoperability. The backend needs to work with multiple components, including the logging software, the data coming in from the serial port, serving the frontend to the user.

Node uses an event driven architecture, which is ideal for reading data from the serial port. PHP and Ruby on Rails are both HTML preprocessors, so they don't get activated until the web site is requested. Node, on the other hand, runs continuously in the background and uses an event driven architecture. The event driven architecture is ideal for reading the data from the serial port. Additionally, Node much faster than either ruby or PHP.

**Serialport**
Entity
*Author:* Your name here
*Type:* Library
*Purpose:* Node serialport library
*Contents:* Stuff here.

**Log**
Entity
*Author:* Albert Morgan
*Type:* Data store
*Purpose:* Store the data in non-volatile storage for later retrieval and update new connecting clients
*Contents:* Groundstation will store telemetry in JSON format. The log should limit the possibility of the corruption of the data due to a programming error and make the data easily accessible for newly connecting clients. Several choices were considered for the logging, including relational databases, JSON documents, and logging the telemetry. A relational database would be unnecessary because the data does not have any relations that need to be tracked; each telemetry packet is an independent piece of information. Logging the telemetry as it comes in from the rocket would limit the possibility of corruption. A programming error in the JSON parsing routines could cause all of the data to be unusable. However, storing the data in JSON format would make it very easy for new clients that connect to the server in the middle of the rocket flight to be updated with all of the past data. For this reason, the telemetry will be stored in a JSON format.

**jQuery**
Entity
*Author:* Your name here
*Type:* Libary
*Purpose:* UI stuff
*Contents:* Queries the J

**3.js**
Entity
*Author:* Natasha Anisimova
*Type:* Libary
*Purpose:* 3.js is a JavaScript 3D library used to create and display animated 3D computer graphics using WebGL.
*Contents:* Since the information about the rocket will be displayed by using a web browser through a local Wi-Fi network, making sure the information is displayed on time and correctly is crucial. 3.js allows for easy and rapid development of WebGL applications, which means we can make the most of the specialised graphics hardware on the users' PCs that it gives access to.

**Rocket**
Entity
*Author:* Your name here
*Type:* Component
*Purpose:* Gets high
*Contents:* ZOOM

**Web server**
Entity
*Author:* Albert Morgan
*Type:* Process
*Purpose:* The web server will serve three primary functions:

- Server web pages to the clients.
- Receive telemetry from the serial port and convert it into json.
- Make the JSON data available to the clients.

*Contents:* The web server needs to be stable so that there is no data loss during the flight. Additionally, the web server should be as lightweight as possible to minimize the resource consumption on the Raspberry Pi. Groundstation will use the Apache [2] web server. This web server was chosen over other options primary for it's stability. Apache uses a one-thread-per-connection model, so if one thread crashes, it will not affect the rest of the connections. NGINX [3] was also considered. On large-scale servers, it is much more lightweight because it uses a many-connections-per-thread model. However, the number of connections will be small and the reuse of threads could reduce stability. Lighttpd [4] was also considered. Although Lighttpd is much more lightweight than Apache, it is not nearly as mature. Because stability is a primary concern, Apache was selected.

**Web browser**
Entity
*Author:* Albert Morgan
*Type:* Process
*Purpose:* The web server
*Contents:* The client will use a web browser to connect to the Groundstation web server and access the content. The web browser may be any of:

- Chrome version 54 or higher
- Edge version 14 or higher
- Firefox version 49 or higher
- Safari version 10 or higher

**Web browser composition**
Relationship
*Author:* Albert Morgan
*Type:* Composition
*Contents:* The web browser runs on the PC.

**Frontend composition**
Relationship
*Author:* Natasha Anisimova
*Type:* Composition
*Contents:* The Frontend will take what the Backend of the software gives it and use 3.js to create and display information to the user.

**Backend composition**
Relationship
*Author:* Your name here
*Type:* Composition
*Contents:* Stuff

**jQuery composition**
Relationship
*Author:* Your name here
*Type:* Composition
*Contents:* Stuff

**3.js composition**
Relationship
*Author:* Natasha Anisimova
*Type:* Composition
*Contents:* The JavaScript Library 3.js will be used with WebGL to create the Frontend of Groundstation.

**Node composition**
Relationship
*Author:* Your name here
*Type:* Composition
*Contents:* Stuff

**Serialport use**
Relationship
*Author:* Your name here
*Type:* Use
*Contents:* Stuff

**Log composition**
Relationship
*Author:* Your name here
*Type:* Composition
*Contents:* Stuff

**Web browser use**
Relationship
*Author:* Your name here
*Type:* Use
*Contents:* Uses the web server

**Frontend / Backend relationship**
Relationship
*Author:* Your name here
*Type:* Composition
*Contents:* Backend servers frontend

**Backend / Rocket**
Relationship
*Author:* Your name here
*Type:* Use
*Contents:* Gets data from the rocket

**NPM / Frontend**
Relationship
*Author:* Your name here
*Type:* Use
*Contents:* Frontend uses NPM

**NPM / Backend**
Relationship
*Author:* Your name here
*Type:* Use
*Contents:* Backend uses NPM

## III. Interaction

Talk about how the system will get data from the serial port and how it will get sent to the web browser.

## IV. Algorithm

Stuff about the event-driven architecture maybe.

## V. Glossary

- **Accuracy:** The absence of errors in the telemetry GS receives, logs, and displays.
- **Binary data:** Telemetry that represents one of two states, for example, "stage 2 activated" and "stage 2 not activated."
- **Corruption:** The process by which data is altered or made unreadable.
- **Crash:** A software crash; the event in which a piece of software ceases operation unexpectedly.
- **Die:** A process dies when it ceases operation and is removed by the operating system. The difference between dying and crashing is that a crash may cause the program to become non-responsive, whereas dying causes the process to end.
- **GS:** Groundstation, the name of our software.
- **Graphical display:** Data that is displayed using a visualization.
- **Live:** Updated in real-time.
- **Non-volatile storage:** Storage that will not be erased when the system is powered down. For example, a hard drive or flash storage.
- **Page:** A web page that users of GS may connect to in order to view the telemetry.
- **Process:** A running program on a computer.
- **Raspberry Pi:** A small, inexpensive computing platform.
- **Real-time:** Each telemetry datum received from the rocket must be processed and displayed in under one second.
- **Redundant sensors:** Two or more sensors that provide the same type of data.
- **Reliability:** In the event of a software crash, the Groundstation software should automatically start and begin all normal functions in under five seconds.
- **Robustness:** In the event that GS receives data that is garbled or otherwise does not adhere to the protocol, it must continue to receive and display data and not break the real-time requirement.
- **Storage:** A device where data is logged.
- **Telemetry:** Data received from the rocket while the rocket is in flight.
- **Telemetry packet:** The rocket will send a telemetry update once per second. Each one of these updates is a "telemetry packet."
- **Visualization:** Information or data, transformed into an visual context.

## References

[1] Node package manager. [Online]. Available: https://www.npmjs.com
[2] Apache. [Online]. Available: https://www.apache.org
[3] Nginx. [Online]. Available: https://nginx.com
[4] Lighttpd. [Online]. Available: https://www.lighttpd.net/

| | |
|---|---|
| Nancy Squires | Date |
| Natasha Anisimova | Date |
| Terrance Lee | Date |
| Albert Morgan | Date |