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 examine nine different components of the system. For each of these components, three different technologies will be described and evaluated for use in this component. Finally, a recommendation will be made about which technology should be used.

I. Introduction

1) Identitication of the SSD
2) stakeholders
3) concerns
4) selected viewpoints
5) design views
6) design overlays
7) rationale

Maybe we don't need all of this stuff in the intro? I think some of it is covered below.

II. Structure View

PC
Entity
Author: Your name here
Type: Component
Purpose: Why does it exists?
Contents: Stuff here.

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: Your name here
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. The backend takes care of all data collection, transformation, logging, and serving that data to the user.
Contents:

Node
Entity
Author: Your name here
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: Your name here
Type: Data store
Purpose: This is where data gets logged
Contents: Stuff here.

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

3.js
Entity
Author: Your name here
Type: Libary
Purpose: UI stuff
Contents: All of your 3 needs

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 will run on the Raspberry Pi. The web server has three primary functions: Groundstation will use the Apache [2] web server.

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: Your name here
Type: Composition
Contents: The web browser runs on the PC

Frontend composition
Relationship
Author: Your name here
Type: Composition
Contents: Stuff

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: Your name here
Type: Composition
Contents: Stuff

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

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.

References

[1] Node package manager. [Online]. Available: https://www.npmjs.com
[2] Apache. [Online]. Available: https://www.apache.org

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