# Groundstation: Technology Review

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

**Abstract**

Abstract goes here

Here's how to cite something [1]. Check the LaTeXsource. The actual sources are in groundstation.bib, add new sources as necessary. We can re-use this file so we don't have to rewrite it again in the future, so give your sources useful names. Note that if you don't cite a source from the .bib file, it doesn't show up in the references, which is why we can re-use that file.

Pro-tip: put a tilde in between the cite command and the word before it to prevent LaTeXfrom putting the citation on a new line.

## Contents

## I. Web server

This section will describe the various options for the web server. Author: Albert Morgan.

### A. *Apache*

Apache is the most popular web server option, with over 50% of the market share [1]. Apache uses a powerful and flexible module system that allows for a variety of different behaviors [2]. Each connection to an Apache server spawns a new thread or process, depending on which modules are used. This feature has advantages and disadvantages. Because each connection spawns a new thread, if the thread serving one connection crashes, it won't affect the other connections. However, because each connection spawns a new thread, the overhead for that thread must be paid for each new connection.

### B. *NGINX*

NGINX is 2$^{nd}$ most popular web server on the Internet [3]. Web servers such as Apache that use a one-thread-per-connection model can have problems with large number of simultaneous connections, an issue known as the C10k problem [2] [4]. NGINX solves this issue by having one thread serve many connections. This approach reduces the overhead caused by spawning so many threads, and makes NGINX very lightweight and scalable [5].

### C. *Lighttpd*

Lighttpd, pronounced "lighty", is a web server that is optimized for low memory footprints and fast response. It is optimized for serving a large number of concurrent keep-alive connections, such as serving high-volume AJAX driven web sites [6]. Many web sites use lighttpd for static content delivery while using another web server, such as Apache, for dynamic content [7].

### D. *Conclusion*

All three web servers listed above have advantages for this project. Apache's one-thread-per-connection approach increases stability. NGINX and Lighttpd are both lightweight, which is important for running an application on a low-power system like a Raspberry Pi. However, Groundstation does not have to support more than 20 concurrent users, so the memory and processor savings for choosing a lightweight option will be minimal. Additionally, stability is a primary concern of the High-Altitude Rocketry Challenge. For these reasons, I recommend Apache.

## II. Web backend

This section will describe the options for the web backend. Author: Albert Morgan.

### A. *Node*

Node allows web backends to be written in JavaScript using V8, the same engine that powers Google Chrome [8]. The client side of the web application will certainly use JavaScript, and writing the backend in JavaScript as well will reduce the overhead of learning and programming in new languages. Additionally, because JavaScript would be used on the front-end and backend, it may be possible to reuse code between these two systems.

Node uses an event driven architecture, and is built to work with JavaScript Object Notation (JSON) data. With Node, is it easy to create a web application that response to small events and sends small bits of data to the client [9]. Because Node uses JavaScript, it integrates naturally with JSON

Benchmarks show that Node is over 5 times as fast as PHP, even when using just-in-time compilation [10].

### B. *PHP*

PHP has been around the web since 1994 and is used in over 80% of web sites today [11] [12]. Given its maturity and wide adoption, there are a lot of packages, documentation, and example available for developers. PHP's primary advantage is the ease in which is can be incorporated into an static page. Using PHP tags, HTML and PHP code can be intertwined seamlessly.

### C. *Ruby*

Ruby is an open source programming language designed for simplicity and productivity [13]. It is well-known on the web as part of the framework Ruby on Rails, which allows for embedded Ruby to be inserted directly into HTML [14]. However, compared to PHP, Ruby on Rails has a steep learning curve. Ruby's syntax can be hard for a beginner to read, and it draws some concepts from functional programming, which can be difficult for a new user [15].

### D. Conclusion

Node is fast, which is important for the Groundstation server, which will run on a Raspberry Pi. Node also uses an event driven architecture, which is perfect for our software. Data will be received at regular intervals and need to be pushed out the clients quickly. For these reasons, I recommend using Node.

## III. Logging

This section will describe how the telemetry will be stored. Author: Albert Morgan.

### A. Raw data

The simplest logging method is to save the raw data stream to a file as it comes in. This method would be very low cost, in terms of development time, processing, and memory usage. The data would simply be read from the serial port and saved to a file, where it could be reconstructed at a later time. Saving the data at this point, before any processing is done, will reduce the chance of any corruption.

### B. JSON

JSON is a data format that is easy for both machines and humans read and write. The telemetry may be stored after it is transformed from its raw form from the serial port to JSON. The server will be converting the raw data to JSON already in preparation to send it to the client, so there will be no additional overhead in the conversion. However, it is likely that the JSON data will take up more space than the raw data due to additional formatting, so it will take up more space in the storage media [16].

### C. Database

Storing the telemetry in a database as it comes in will allow for fast random-access reading and writing of the data. This would allow the server to easily grab a certain subset of the data. For example, the server could send the client only data that was generated in the last 2 minutes, or data that comes off of one particular sensor. Data could easily be retrieved using queries in a language such as SQL.

### D. Conclusion

Because the system only needs to store data from a single launch, the size of the data set will be relatively small. The overhead of sending all of the data would be minimal, so the benefits of using a queryable database would be small. In the event that the system goes down, it would need to start back up again and begin operation very fast. Storing the data in JSON format would allow the data to be quickly loaded back into memory without significant processing. Additionally, if the data is stored in JSON format, the server would not need to keep the entirety of it in memory to server it to a connecting client, freeing up resources on the Raspberry Pi. For these reasons, I recommend that the data be logged after it is converted into JSON format

## IV. Data Visualization

This section will describe the various options for the data visualization. Author: Natasha Anisimova.

### A. 2D representation

2 dimensional representations of maps are usually equal in the x and y dimensions. Graphs and charts are representations that most users are comfortable with but they can also be a little dull. For displaying the location of a land bound vehicle or person 2D graph is useful. The ground can be viewed as a 2D plane. With a rocket the height, latitude, and longitude has to be taken into account and displayed. The most important aspects of the software project are to find rockets location in the end and be able to tell if it reached the teams one hundred-thousand-foot goal.

### B. 3D representation

2 dimensional topographic maps are hard to read if one has not had a lot of experience with them. Considering that most of the team will be mechanical engineers and no earth science majors will be participating, a 3 dimensional map makes more sense. Cartographers in the past decade have shifted to creating 3D perspective maps. These types of maps are more expensive and time intensive to produce. The extra time that is put into the visualization makes it more appealing to the readers. It is easy to visualize and understand.

### *C. Numerical Display*

A numerical display is often given when looking for the longitude and latitude of a specific point on a map. It would also be useful when displaying the highest altitude of the rocket. This is a number that would constantly be reviewed and needed. A graph can display the data but it would still take a second to get that information. Having a table will all the most crucial aspects of the project with the values would make the webpage so much more usable.

### *D. Conclusion*

3D visualization for the location of the rocket as well as a small numerical display is the best way to go. Thankfully, Professor Mike Bailey has created software that can create or extract the topology of a certain area making it much more manageable. The High-Altitude Rocketry team will only need the Lucerne Dry Lake area of the Mojave Desert. The images for the area can be pulled from NASAs website and from there they can be texture mapped to the topology. The numerical display will be used to give a more accurate and direct way of giving the data to the users.

## V. User Interface Interaction Model

This section will describe the various options for the user interface interaction model. Author: Natasha Anisimova.

### *A. Unity*

Unity 3D is an engine, not so much an application. This software is going to be used by about thirty to forty people, if not more, and most of them would not have the Unity3D Web-Player installed. Unity is a game development tool and since this project is focusing on just creating a useful visualization of the data that is collected, Unity is not suitable for this particular case.

### *B. OpenGL*

OpenGL has one of the fastest languages to develop with, C++. Double buffering also solves any problems with the program being in the middle of plotting data and the user getting a black screen. Instead it shows the old frame until the new one is ready to be displayed. Unfortunately, OpenGL does not run on the web.

### *C. WebGL*

WebGL is the Javascript API that allows you to create 3D graphics in the browser. Three.js is a tool used on top of WebGL, making it easier to create those 3D graphics. It is used when creating more complicated 3D scenes (textures, models, and rendering). Several different types of browsers support it as well, from Desktop Opera to Firefox and Chrome. WebGL is also the easiest 3D application to use as well as the best tested and documented. Performance wise it is also very fast even though programs are written in JavaScript.

### *D. Conclusion*

WebGL is the most fitting choice to go with for this project. OpenGL is not supported by web browsers, while Unity is not as widely readily supported by web browsers as WebGL. The WebGL application can be used while still having the comfortable environment of the web development environment that HTML and JavaScript have to offer. While there is a high learning curve, there is plenty of support for it at OSU as well as information on the web. The only major downside to it, is that Internet Explorer does not support it but that browser is used by few and far in between. WebGL is supported by mobile devices if the software is expanded to them in the future.

## VI. User Interface Organization

This section will describe the various options for the user interface organization. Author: Natasha Anisimova.

### *A. Paper Prototypes and User Interviews*

In order to understand what are users are most comfortable with it would be most cost effective and time efficient to create paper prototypes. A small sample of the High-Altitude Rocketry Team could be taken through a user study where we would try and figure what would work. Feedback would be taken then and there. In some cases the feedback could be implemented quickly into the paper prototypes so that the user could see how it would affect the layout.

### B. *Flight Simulator Displays*

A lot of flight simulator games efficiently display the information of the planes' or rockets' location, altitude, velocity and status. They are often able to do this with various dials on either side of the screen, much like a dashboard of a car. Having that recognizable display for the users would greatly reduce any misunderstandings that could occur. Units and dials would be labeled for clarity. Colors would be used to help emphasize important goals and perhaps how close they are to being reached.

### C. *Finnish Astronautical Society*

The Finnish Astronautical Society has launch several different rockets since 2011 and successfully displayed the data that they have received from the rockets while they have been in space. As mentioned per their website, most of the data is streamed to a ground station in real-time using a radio link and then from there onto an Internet server. Their solution to the problem of keeping track of their rocket is very similar to what we have planned except that they have a few different sensors and equipment. They have considerably more information about all their launches on their website, making it a bit cluttered but the content of their website is something that would be useful to follow.

### D. *Conclusion*

For the organization of the user interface it would be useful to pull inspiration from websites or implementations that already exist but also have the freedom of easily changing it and being creative. The paper prototypes would be useful for making sure that the software is easily tailored in the early stages of the web page. We have currently been pulling user stories from the team in order to know what would be most important for them to see on the web page. Continuing with that trend, we can continue to create the webpage with their feedback. This would also ensure that they are not shocked or bewildered when seeing the final product before the launch date or on site.

## VII. Retrieval

### A. *Retrieval Options*

For retrieval of data and tracking of the rocket we have decided to use either C, C++, or Python as our choices for the languages. We chose these languages because they can all be used for data collecting.

### B. *Retrieval Goal/s*

Our main goal for this design is to retrieve data from the rockets sensors. These sensors are either being made or purchased by the electrical engineering team. Our global positioning system, accelerometer, gyroscope, and altimeter will provide us the data that we will need, and some of that data will also be used for tracking the rockets position as well.

### C. *Retrieval Criteria*

The criteria for the languages for retrieval are the libraries, and either their references or their resources.

### D. *Retrieval Comparison*

1) The C language has an extensive library which has many choices we can use. The main one for the sensors is the extensive math.h library. It allows us to do any math functions. Plus we can do error conditions in our code if needed. When it comes to references C has many. Since C has had a couple of major programs, for example, Unix was rewritten in C in 1973 many people and companies have used C (Hauben, Part I). This has allowed many references including the internet, books, and instructors. Cs disadvantage is that it is a very low level language and can be security flawed.

2) The C++ library is just as extensive as the C library except that it has made it so that it can be supported in an object oriented programming language. For example, it has to bring out a ¡cmath¿ library so it works with C++ (Schildt 181). C++ is not as old as C. For example, in 1998 an ANSI/ISO standard for C++ was adopted (Schildt xix). It is a favorite to many because of its object oriented programming(OOP). Also, since it is OOP it is easier to write than C but you have a lot more to learn. That is its disadvantage no developer can use its entire set of code that C++ provides. It would take an extremely long time. Just like C, there are many resources for C++ out there if we run into a problem.

3) The Python library is very extensive. The new Python version 3.6 has about 37 main libraries which is sub-sectioned so that when you program you can be more specific (The Python Software Foundation, 3.6.0b4). The main libraries that we need are the data types, numeric, and the mathematical modules. When it comes to references it seems that Python users understand that many schools do not have a class for it. For example, when I was researching the libraries it had a reference page for the library. Especially when it comes to the internet, there are a lot of references to help if you run into issues with coding in Python. Pythons disadvantage is that it is not popular. Not many companies have given it a chance yet, and it hasnt had a big break through.

### *E. Retrieval Conclusion*

All of these options are great. They all have extensive libraries and have great references. If we run into any issues we have plenty of references online, as well as books and actual people we know that we can connect with and speak to about it. I did some extra research and spoke to instructors and people that have used data collecting software in their career field before. I asked them which language they preferred. I got mixed results when it came to which one was the best language to use. I believe that was because everyone has their favorite language that they prefer.

### *F. Retrieval Selection*

After speaking to the team we decided that the best language to use for the retrieval of data would be C. It meets all of the criteria and beyond. It has all of the libraries that we would need to retrieve the data from the sensors on the rocket. If we lose GPS from a sensor we can still calculate the position of the rocket with those libraries as well. With the amount of resources and references that are available to C if we have an issue we should be able to find the solution to our problem quickly so that we do not get stalled on it for long.

## VIII. Interaction Modes

### *A. Interaction Modes Options*

For interaction modes, that we will support so that the high altitude rocketry team can see the data, we decided on either PC, mobile, or PC/mobile. We chose these because everyone is going to have these when we are out at location.

### *B. Interaction Modes Goal/s*

Our goals for this design is to find which would be most functional and accessible to the team when we are out at location.

### *C. Interaction Modes Criteria*

The criteria for the interaction modes are availability, screen size, power usage, and flexibility when coding.

### *D. Interaction Modes Comparison*

1) The PC is our first choice because it is the most available to the teams, it has the largest screen size, and it has the most flexibility when coding. The disadvantage with it is that it is larger than mobile so the team will have to carry it around. It can use up more power than the mobile too, but the battery can be more robust.

2) The mobile mode is also available to most of the people on the team as long as they have a smart phone. The mobile option will give them a lighter option to carry around and will not waste as much power. The disadvantage is that it will have a smaller screen and we have less flexibility when coding. Also, team members will only be able to use android phones with this option. We will not have cell service either when we are out at location.

3) The third option is using both the PC and mobile. With this option we will have all the advantages of both. We will still have a disadvantage which is that we will have to program in multiple languages and formats.

### *E. Interaction Modes Conclusion*

With the PC once the team gets the information they will be able to save it right there on their team computers and move it easily through USB or other devices. With the mobile our biggest issue is not the programming but the lack of cell service. Without cell service, if the team wants to look at any data they are reliant on the Raspberry PI Wi-Fi only. That is not going to be on all the time either. If we use both it has to be as a stretch goal because of the time constraints.

### *F. Interaction Modes Selection*

Since speaking to the team and going over the criteria, the best option for this series that I would recommend would be to use the PC for the interaction mode. It is reliable, most available, flexible when coding, and has a large screen size. If multiple people are looking at the screen they are pushing each other out of the way to see it. When it comes to carrying it around most of us have a computer bag and can carry it around that way. That is not that big of a drawback. With the power issue we plan on bringing a power generator with us to the launch site which the power plug for most computers are compatible. That should not be that big of an issue because we only need to have the computers on for launch and test launch too.

## IX. Toolkit to generate the User interface (UI)

### A. Toolkits Options

For the toolkits to generate the UI the options we decided to use are Prototyping toolkits, jQuery UI, and the Dojo toolkit. These are all part of the top ten JavaScript Web UI Libraries, Frameworks and Toolkits (Deering, JavaScript).

### B. Toolkits Goal/s

Our goal for this design is to have Toolkit that allows us to generate a UI with ease of use and is reliable.

### C. Toolkits Criteria

Criteria is price, ease of use, functionality for our purpose, and documentation.

### D. Toolkits Comparison

1) Our first option is the Prototype UI toolkits. Their advantages are that it is reliable and it has the functionality for our purpose. It will allow us to generate the UI we want. Also it has great documentation. The disadvantage is that they cost money. The cheaper they are the less functionality, and the more expensive the more functionality (Jia, Prototyping Tools). The best one I found was called Mockplus, but its biggest disadvantage besides money is that it has a lack of responsive design features (Jia, Prototyping Tools). One big advantage though is that ease of use due to its drag and drop feature.

2) The second option is jQuery. With jQuery it is free to use, it is easy to use, and has great documentation. One disadvantage with jQuery is that it has limited functionality in some cases (JQuery).

3) Our last option is the Dojo toolkit. The Dojo toolkit is under the academic free license version 2.1 so the license is free (Reliam Inc, Dojo License). Its other advantage is that it is easy to use. Its disadvantages are that it does not have great documentation and its functionality for our purpose. Dojo has a great framework but what we need is something with a good library.

### E. Toolkits Conclusion

They seem to all have good advantages. I feel that Prototype toolkits are for more professional type work. If you were doing a project and you could afford one of the high-end ones they seem like the way to go. For this type of project jQuery and Dojo have the best advantages functionality wise for this scale of this project.

### F. Toolkits Selection

For the Criteria we felt that jQuery was the best way to go. Even though Dojo has great framework. What we need is a really go library for this project. Also the team is very familiar with it and that is a plus. With the limited functionality disadvantage it is only for some cases. We believe that we should not run into that issue because we just need it for some of its basic features.

## Index

## References

[1] (2016) Usage statistics and market share of apache for websites. [Online]. Available: https://w3techs.com/technologies/details/ws-apache/all/all
[2] J. Ellingwood. (2015) Apache vs nginx: Practical considerations. [Online]. Available: https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations
[3] (2016) Usage statistics and market share of nginx for websites. [Online]. Available: https://w3techs.com/technologies/details/ws-nginx/all/all
[4] D. Kegel. The c10k problem. [Online]. Available: http://www.kegel.com/c10k.html
[5] O. Garrett. (2015) Nginx vs. apache: Our view of a decade-old question. [Online]. Available: https://www.nginx.com/blog/nginx-vs-apache-our-view/
[6] Lighttpd. [Online]. Available: https://www.lighttpd.net/
[7] Powered by lighttpd. [Online]. Available: http://redmine.lighttpd.net/projects/1/wiki/poweredbylighttpd
[8] Node.js. [Online]. Available: https://nodejs.org/en/
[9] G. Woodfine. event driven architecture node.js. [Online]. Available: http://garywoodfine.com/event-driven-architecture-node-js/
[10] R. Sanchez. Comparing node.js vs php performance. [Online]. Available: http://www.hostingadvice.com/blog/comparing-node-js-vs-php-performance/
[11] History of php. [Online]. Available: http://php.net/manual/en/history.php.php
[12] Usage of server-side programming languages for websites. [Online]. Available: https://w3techs.com/technologies/overview/programming_language/all
[13] Ruby. [Online]. Available: https://www.ruby-lang.org/en/
[14] Getting started with rails. [Online]. Available: http://guides.rubyonrails.org/getting_started.html
[15] L. Teo. (2016) Ruby on rails vs php - the good, the bad. [Online]. Available: http://www.leonardteo.com/2012/07/ruby-on-rails-vs-php-the-good-the-bad/
[16] Introducing json. [Online]. Available: http://www.json.org/