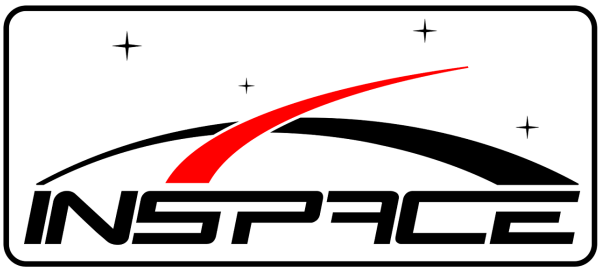
Ground Station

Report of Findings

Authors:
Thomas Selwyn
Matteo Golin



Date: January 20, 2023
Carleton University,
Faculty of Engineering and Design

Table of Contents

1 Ground Station

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2 User Interface

The user interface is very cool. Its design is very human.

2.1 Dashboard

Lorem ipsum doler sit amet.

2.2 Map

Lorem ipsum doler sit amet.

2.3 UI Internal Architecture

The UI receives data via a websocket in order to preserve the real-time nature of the data being transmitted. When the data is received, it is distributed amongst all the dashboard charts and the map for a live visual of the rocket's status. The UI stores data locally in the browser for historical data display.

2.3.1 Connection Handling

The UI will maintain connection with the backend system even with interruptions.

While disconnected, the UI continues to poll the backend system every second until a connection is established. If at any point this connection is severed, the UI will beging polling for a connection again.

When the UI reconnects to the backend, it issues an *update* command, which prompts the backend to send the most recently received data packet. This ensures that the UI displays the latest data with minimal delay.

The websocket connection allows multiple clients to be connected at once. This means that the rocketry team could choose to have multiple instance of the user interface running at once on different devices. This would allow a full ground station control team to monitor the rocket flight, similar to corporations in industry.

2.3.2 Receiving Data

Data is received in JSON format by the frontend via a websocket. The data may not contain all different packet types at one time (i.e. not every websocket message will contain altitude information).

When data is received by the frontend, it is unpacked and written to the browser cache (local storage). The browser cache contains several different buffer arrays, one for each type of data packet (altitude data, satellite data, etc.). Each buffer contains n of the most recent packets corresponding to its data type (altitude, satellite, etc), where n is the number of historical data points that should be displayed on the dashboard charts. The current software stores 20.

Because not all types of data are sent on each websocket transmission, separate buffers allow the data to be stored for each category out of sync with the other categories. Altitude data is transmitted most frequently, so its buffer will fill up and refresh faster than the buffer for satellite data.

When data is received and written to the local storage, a browser event is issued. This browser event causes all charts on the dashboard and the map to update their display with the most recently received data.

2.3.3 Static Display

The buffers containing each packet type are used to maintain a static display when the connection between the UI and backend system (or the connection between the rocket transmitter and the ground station board) is interrupted. The UI will maintain a display of its reserve data until connection can be re-established.

2.3.4 Websocket Commands

There are several commands available to the ground station operator in order to communicate with the backend system.

- **Update:** The update command retrieves the latest packet from the backend.

2.3.5 Mission Replays

To allow for analysis of previous rocket flights, system testing and ground station operator training, the UI implements mission replay functionality.

Mission replays allow the ground station operator to select a recorded mission (stored telemetry data from a previous rocket flight) and play it back through the graphical interface. The replay has several controls, not unlike video playback:

- **Play:** Continues a currently paused mission.
- **Pause:** Pauses the current replay.
- **Stop:** Stops the current mission replay and resets all cached historical data.
- **Fast forward:** Doubles the current replay speed, to a maximum of 4x.
- **Rewind:** Halves the current replay speed, to a minimum of 0.25x (does not play replay in reverse).

The replay controls are accompanied by a replay speed indicator. Each of the controls above matches to a specific websocket command.

- **Play:** `telemetry replay play`
- **Pause:** `telemetry replay pause`
- **Stop:** `telemetry replay stop`
- **Fast forward:** `telemetry replay speed 2x`, where `x` is the current speed
- **Rewind:** `telemetry replay speed 0.5x`, where `x` is the current speed