Assignment C

This is a minor thing for this assignment but between the WebSocket and RESTful GET the keys of the objects are Camel Case in the RESTful API and Pascal Case in the WebSocket. In the real world, if you have multiple implantations of the same API, they should have the same data structure between each will cause multiple errors and issues to occur.

To improve the performance of the API, would be to optimize the payload size, using technologies such as GZIP to reduce the size of the JSON being sent, not using unnecessarily long and/or over descriptive variable/key names, however descriptive names are still needed for the user to understand what the data is portraying.

Having good internal documentation can help with this and is a must for larger projects. Such as for this assignment, there was no way of telling if altitude was in feet or metres, same with other values that were of the type `number` or other types if needed.

For improving the API structure, a more sophisticated communication protocol that handles session, message queues, sensor data and communication lost better would be the OPC UA standard.

Proper error handling is needed, but also things such as `messageStatus` needs to have predefined enumeration of statuses and their condition, for example `{ messageStatus: { status: "ok", message: "All systems are okay" }` or ` { messageStatus: "warning", message: "Temperature is too high" }` or adding to documentation or another variable to understand if the readings are in an okay range, such as temperature needing to know if it is too hot.

Encrypting the information for a safer interaction, such as sending data to the rocket. Using an authentication system such as a login system to do actions to the rocket, but also using JWT and/or API Keys when using the API, so third parties cannot reverse engineer the API or affect the rocket in a hostile manner.