Ground Control of Balloon-Assisted Microgravity

Andrew Geltz, Luis O'Donnell, and Priscilla Ryan

Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — This paper illustrates the method in which we address the problem of creating a ground control system for testing microgravity experiments. The system is based off of the MEAN software stack except replacing MongoDB with a relational database of MySQL. The system is to support storage of data from the experiments, as well as start and begin any tests, and display any errors the experiments may have. The front-end site also has authentication to keep the tests private to the users.**

***Index Terms* — Authentication, command and control systems, database systems, ground support, relational databases, web services, web sites.**

I. Introduction

This is a sponsored project by Florida Space Institute. The project is the continuation of a previous project done by an Mechanical Engineering (ME) team at the University of Central Florida (UCF). The previous group had constructed a platform that is attached to a weather balloon that would be lifted into the air and conduct reduced gravity experiments otherwise known as microgravity. The previous team had succeeded in making this platform, thus the next step is to create the ground control system for the balloon.

This paper shall discuss the makings of the entire ground control system that was made to support the balloon in its experiments. Since this is a sponsored project there are some required specifications that are needed for the ground control system. The ground control system must be able to issue commands to the balloon platform. These commands must be along the lines of being able to start and stop tests aboard the balloon platform. The ground control system must also be able to store data from a total of six capsule sensors. These capsule sensors are one Pressure sensor, one Temperature sensor, one Humidity sensor, and three Accelerometer sensors. The ground station must then be able to display the data obtained from these sensors during experiments. The last specification made was the ability to display any errors that are to be transmitted from the capsule on the balloon. One of the freedoms that this group had decided to add onto the project was the addition of authentication for the system. As it seemed important to add it, to be able to prevent unwanted individuals from interfering with the tests.

We were to continue off from where the ME team left off, and build the ground control system for the balloon. It is worth mentioning that another Computer Science team will continue work on this project. This team will be moving back to working on the capsule, the balloon platform and the electronics associated with it, so that the balloon would be able to fully send data to the ground control system. Since it is currently unable to do that, this project shall be tested using a program to send false data into the database, where the front-end web application will display the data. Even after the completion of this portion of the project, the design shall continue development.

II. Ground System Overview

As mentioned previously the ground system must comply with the necessary specifications. These specifications are the ability to submit commands to the balloon platform, store and display sensor data from the capsule, display any errors transmitted from the capsule, as well as have some form authentication on overall system.

To accomplish all these feats the team has decided to incorporate the MEAN stack software bundle. Unfortunately all the data that is to be collected by the sensors and stored in a database would not work well in a non-relational database. So instead of using MongoDB as one would traditionally use in the MEAN stack, it was decided it would be replaced by a MySQL database instead. The MEAN stack is a collection of javascript software bundle that works well together compromising of, MongoDB, Express.js, AngularJS and Node.js. Thus the data that shall be stored will be in a MySQL database, along with the use of the rest of the MEAN stack.This includes the use of Express.js, AngularJS and Node.js. These were all used in tandem together to be able to create the front-end web application of the system.

The use of Express.js, AngularJS and Node.js is the basis of the front-end web application. From using these different javascript based technologies together we are able to view the data from our MySQL database. Through the use of the additional javascript Chart.js technology, we are able to display all of the data on the front-end in a graph that updates in real time. We were then able to fulfill the requirement be able to display the data that was stored in the database. The other specification for displaying errors on our front-end application falls under the same category as all it took was retrieving the error message from the database and having it displayed in a table on our front-end application.

The last thing that was added was the use of authentication on the front-end application. As this was very much needed we were able to find a way of using Passport.js to be able to easily implement authentication. You must be logged-in to be able to perform any commands in the system. Only once the user is logged-in will the user be able to view data from various tests. While if an admin is logged-in they would be able to have access to more options than a regular user, such as being able to

view all users, or edit the sensors in the options menu. The authentication was made to regulate that only people who are performing the experiments have access to viewing the data, as well as starting and ending tests.

So the use of MySQL, Express.js, AngularJS, Node.js, Passport.js and a few other technologies mentioned in other sections is what was used for this system. As it allowed for the resulting design to meet all specifications of the project.

III. The Database

Originally the database that was to be used was that of MongoDB, due to the fact that it works quite well in the MEAN software stack bundle. Unfortunately the data from the microgravity experiments fits more into a relational database than it would a non-relational database. So instead the database that was to be used was that of MySQL.

Upon having decided using MySQL, the team drafted many different types of Schemas to be used as the basis of how we would store our data. The final schema and ultimately the one we chose had eight tables while two of which were not related to anything in the database.

The main part of the database are the tables runs, run_errors, errors, readings, sensors, and sesnor_types. The runs table hosted the information about which run each set of data belonged to. It also held data regarding what errors if any occurred during that specific test run. The run_errors would tell us which error belonged to which test run as well as when they occurred during the test run. While the readings table held all the information about the data that was collected during a specific test run. As the data is collected over a short period of time, there would be multiple readings that belonged to one test run. So we would also keep track of when that data was collected during the test to be able to organize via time in an orderly fashion.

Beyond the readings were the sensors. Each sensor had its own table, which gave a brief description of what it measured. With these sensors we have it relate to which set of data it recorded, so we can specifically find which sensor recorded which data, when looking at a singular test run. Due to having three accelerometer sensors, we decided to have the sensor_type data table as well. This shows which sensors we have in our database that are of the same type, like the accelerometers. This was done to help differentiate between each sensor as well as different sensors of the same type.

The last two tables are incredibly important. The users table contains all of the authentication users information. It contains the users that can sign in along with their hashed password. It even has the admin flag which determines which user is an admin that allows for them to perform certain actions.While the other table is the commands table. This table is in charge of sending commands from the front-end application. The front-end application has the ability to start and begin tests, and this is where the information for these commands is stored. These two tables are extremely important as they perform two of the biggest aspects of the ground control system.

IV. Authentication

As a means of preventing unauthorized access to the experiment interface and the data collected, our team added authentication to the application by securing the Express routes used to serve the data. Being able to limit or prevent access to certain portions of the data and interface allows a piece of mind for the operators and stakeholders of the experiments.

To make the implementation of the authentication system easier, we decided to use an authentication framework called passport.js. This framework allows for quick setup of authentication requirements and procedures. Once the required configuration is created, authenticating requests is rather simple. Express offers access to the REST request stack by means of “middleware”, software that runs between the client side interactions and the bare-bones serving of the data. Authentication with passport.js is accomplished by passing all requests that need to be authenticated through a middleware function that ensure that the user has access or send the appropriate denial response.

To keep passwords safe in the even of a data breach, we decided that all passwords in the system should be hashed. Hashing a password means to take the plaintext (how the user enters the password) string and manipulate it with a function that makes it infeasible to determine what the original password string was. The main benefit of password hashing is that even if a malicious actor gains access to the user database, they do not directly have the passwords of the users. An added benefit is that anyone managing the database also does not have plaintext access to the user’s passwords.

In order to implement password hashing within our application, we used the bcrypt-nodejs module. This module provides easy to use function that handle hashing without having to micromanage the function doing the actual hashing.

V. Front-end Web Application

The web-based front-end of the application is vital to the successful execution of the required functionality. It is also very important to have a User Interface (UI) that is easy to understand while also having enough information on it to be useful to the user. In order to preserve this balance, we chose to break the application into several pages as is common with this type of application. Each page has its own dedicated function and displays information relative to the task that the user needs to accomplish on that page.

The technology that made having a dynamically updating view of the sensor and test readouts is called AngularJS. AngularJS is a framework that tries to extend the basic functionality of html with more dynamic capabilities. Being able to have a UI that reacts easily to the rapidly changing data that is required for an application of this nature was of great assistance to our team.

When the user firsts accesses the application, they are presented with the login page where they can enter their username and password. The application also has an added feature of detecting when no administrator accounts exist and it will create a default user. Once the user is

authenticated, they are redirected to the home page. The home page has several buttons on it for starting and stopping tests as well as viewing the most recent test. This page automatically scales to the number of sensors present in the readings dataset.

Part of our desire to keep the application scalable and dynamic fed the design decision to have the core tables be able to be modified from within the web application. To get to the settings pages, there is a hamburger menu (a menu that expands when clicked on to show several options) that presents the user with a list of pages that they can visit. There is also a button for the user to be able to log out of their session securely.

Inside the hamburger menu, the Users page, is where logins for the application can be managed. A table of users and their administrator status gets automatically populated when the page loads. Next to each user in the table, there is a delete button where an administrator can remove the user from the database. There is also a form where the user can enter information to create a new user.

Also inside the hamburger menu is the Sensors page. This page is where an administrator can dynamically add sensors to be tracked by the application. Once the new sensor is added to the database via a form on the page, a sensor id is generated. The readings from this sensors can then be stored using the API and sensor id. This design structure allows the design and capabilities of the capsule to be modified to fit any changes required without needing to modify the ground control application to be able to display the sensor readings. Also on this page is a table of sensor types. These types are used to group similar sensors. For instance, you can have multiple accelerometers in the sensors table but there will be one accelerometer entry in the sensor type table that ties them all together.

The last settings page is the Errors page. This page allows administrators to create error codes that the capsule can send to the interface. There are several optional, informational fields for each error so the level of detail desired for each type of error can vary according the wishes of the user. These errors are assigned a unique error id when they are created and runtime errors can be entered into the system using this error id. Optionally, the error can also be tied to a specific sensor if desired.

The application would be useful if it could only display the current test runs but it would be more useful if it had a way of comparing runs side by side with other runs. This is what we created the Historical Runs page for. This page presents the user with a list of the runs that are in the database with their descriptions and starting timestamps. The user can select any number of runs and the page will group readings from all the test runs selected by the sensor that the reading was recorded on. For example, all of the sensor id 5's readings across all selected runs, will be on one chart for easy comparisons. Also on this page is a listing of all the errors encountered on the selected runs and the information describing that error including any sensor it was tied to.

VI. Design Portability

Our team wanted to design an application that would be truly useful to the teams that had to work with it. In order to do this, we spent a significant amount of time creating a design plan on how we could make the application fit the changing needs of a developing project. As we would not be working on the project after our graduation, we wanted to allow our design to adapt to the necessary changes that occur during the implementation phase of a project.

In order to manage dependencies and make our application as easy to use as possible, we decided to build a Docker container. Docker is a software that virtualizes application inside of their own sandbox. The benefits of an application like this is that the software dependencies can be installed inside this sandbox without interfering with any other applications on the machine. It also has the added benefit of automating much of the setup process with a docker build script.

Once the application source code is cloned from the Github repository, the user can simply build the the docker script and they will have an image that can be used to launch any number of containers to run the application.

Unfortunately, certain aspects of the project do not mesh well with a dockerized environment. For instance, the MySQL server is just not suited to run in a non-persistent environment. In order to aide the users with this portion of the application, we created a database setup script that creates all the needed tables and stored procedures in the database without any manual intervention. The creation of a user for the database is still needed as well as any necessary configuration for the server to be accessible from any machine other than the host. It is simply, impossible to account for every type of network and server configuration but we believe that the design choices that we made in order to increase that portability of the application have made it relatively easy to set up and utilize.

The configuration files inside the application allow the user to modify the functionality and location of the database server and other useful settings.

VII. Conclusion

In conclusion the specifications for the ground control system that were needed were met. The ability to store data was done via our MySQL database. The MySQL database is where we store all the various types of information such as the data from the readings, the errors that occurred, the sensors that were used, the information for the users authentication, and the commands that would be sent to the balloon. Then the display of all of this data was done all on the front end using part of the MEAN stack of Express.js, AngularJS and Node.js along with the use of Chart.js. After which the authentication for the users was done with the use of passport.js for local log-in and bcrypt-node.js for hashing our passwords. We were able to do all of the specifications needed for our ground control system. Now this project will be left in the hands of the following Computer Science team at University of Central Florida. This follow-up team will continue the development of the overarching project of finalizing the connection between the ground system we created and the Mechanical Engineering teams balloon capsule they created.

Acknowledgement

The authors wish to acknowledge the assistance of the previous ME team at UCF for the creation of the capsule and balloon platform. The authors wish to also acknowledge the following team at UCF that shall work on continuing this project.