Boston University
Electrical and Computer Engineering

EC 464 Senior Design Project

Final Testing Report

4/8/2021

Aerial 5G Network Modeling

Team 17

Team Members
Taehyon Paik ttpaik@bu.edu
Minseok Sakong msakong@bu.edu
Talbot Taylor talbotct@bu.edu
Stanley Zhang zhangs5@bu.edu
Yuan Chi Chung ycc88@bu.edu

1.0 Prototype Setup Summary

1.1 Required Materials

Hardware:

- Android phone equipped with 5G AT&T network
- DJI Flame Wheel Drone
    - F450 Frame Kit
    - 2312E 800KV Motors
    - 30A ESC Brushless Speed Controllers
    - 10 inch FPV 1045 Propeller
    - 2x 3000mAh LiPo battery
    - Flysky FS-i6X Transmitter

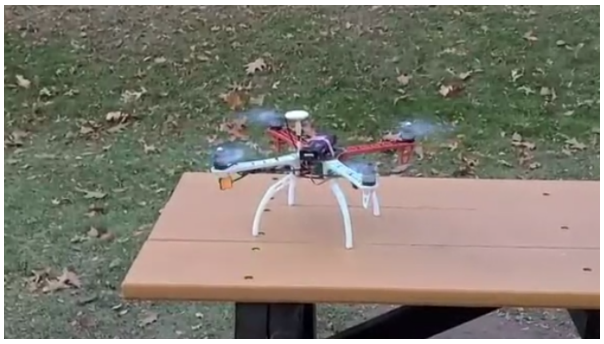


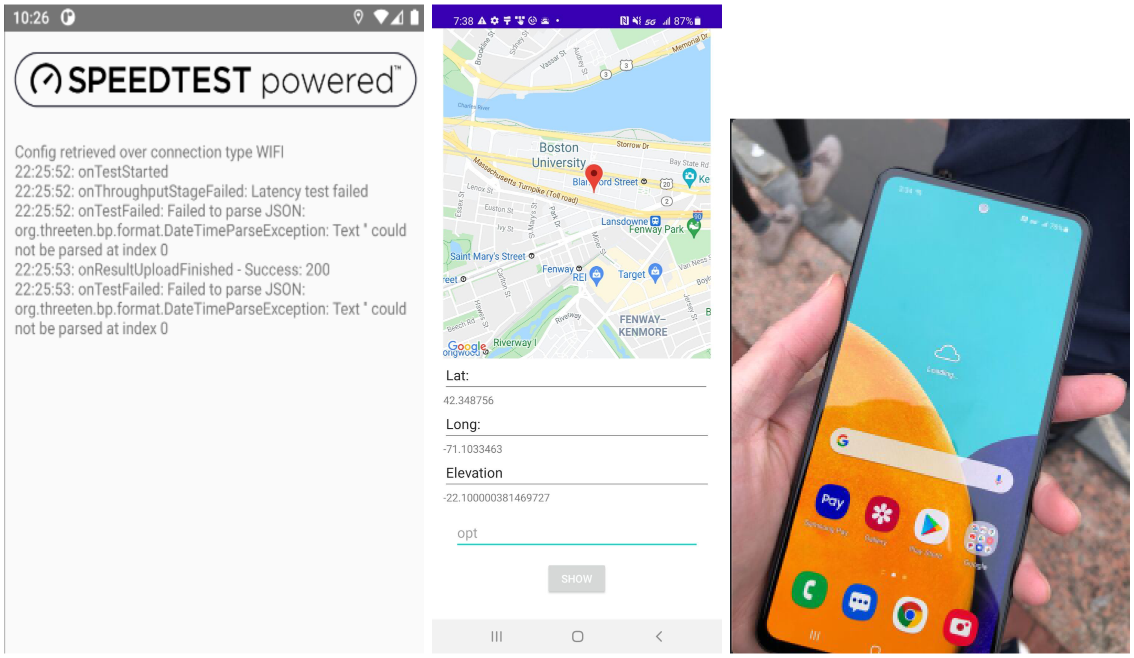
Figure 1.1: DJI Flame Wheel Drone

Software:

- Data collection
    - Created speed testing application (Kotlin)
    - Ookla SDK for created application
    - Google Firebase Firestore

- Data display and management
    - Python3
        - Pandas
        - Plotly
        - Numpy
        - Scipy
        - Sklearn
        - Matplotlib

- Web Application
    - Flask framework, Python, Javascript
    - HTML/CSS
    - Plotly Chart Studio



```
Config retrieved over connection type WIFI
22:25:52: onTestStarted
22:25:52: onThroughputStageFailed: Latency test failed
22:25:52: onTestFailed: Failed to parse JSON:
org.threeten.bp.format.DateTimeParseException: Text '' could
not be parsed at index 0
22:25:53: onResultUploadFinished - Success: 200
22:25:53: onTestFailed: Failed to parse JSON:
org.threeten.bp.format.DateTimeParseException: Text '' could
not be parsed at index 0
```

Lat:
42.348756
Long:
-71.1033463
Elevation
-22.100000381469727
opt

Figure 1.2: Our application based on the Ookla SDK

1.2 Data Collection

Just like our second prototype, we brought our drone in and showed that it was completely built and functional, demonstrating it is able to fly for our data collection. Although we were not able to fly the drone as we were indoors, it was obvious that it works as long as we attach the phone to the drone.

We then displayed the improved mobile application we had built upon from last semester. It is now able to automatically and continuously collect data while in flight. This is an improvement from our last prototype because it uses the Ookla SDK and runs without requiring human interaction. The results from the speed tests were automatically uploaded to a Firebase database and then able to be extracted into a .json file, as we demonstrated. This data is then pipelined into our data visualization and machine learning sections.

Automatically recording the altitude during our tests required additional work. The client requested the altitude in relation to the height off of the ground. Unfortunately, our Google Maps API only provides the WGS84 ellipsoid altitude. In order to determine the Geoid height, the application must request two different altitudes simultaneously: 1. ellipsoid height (h) and 2. geoid height (N). The orthometric height is then determined using the equation: $H = h - N$ to find mean sea level (orthometric height), the altitude that the client requested.

1.3 Data Visualization

We put this collected data into our Python script that displays the information in a geographical heatmap to indicate how fast the network performed at the locations of interest. We were able to display both the newly collected data as well as previous tests conducted. The data was collected from various places on campus, including the Student Village dorms and the Photonics Building.

We also presented other maps to show upload speeds, download speeds and latency to properly evaluate network performance across available factors. This format aids in visualizing the range of data. Our current methods create various 2D maps based on clusters of similar altitudes. We are finalizing the ability to create a 3D visual of the data so that the data points could factor in altitude as well.

1.4 Machine Learning Modeling

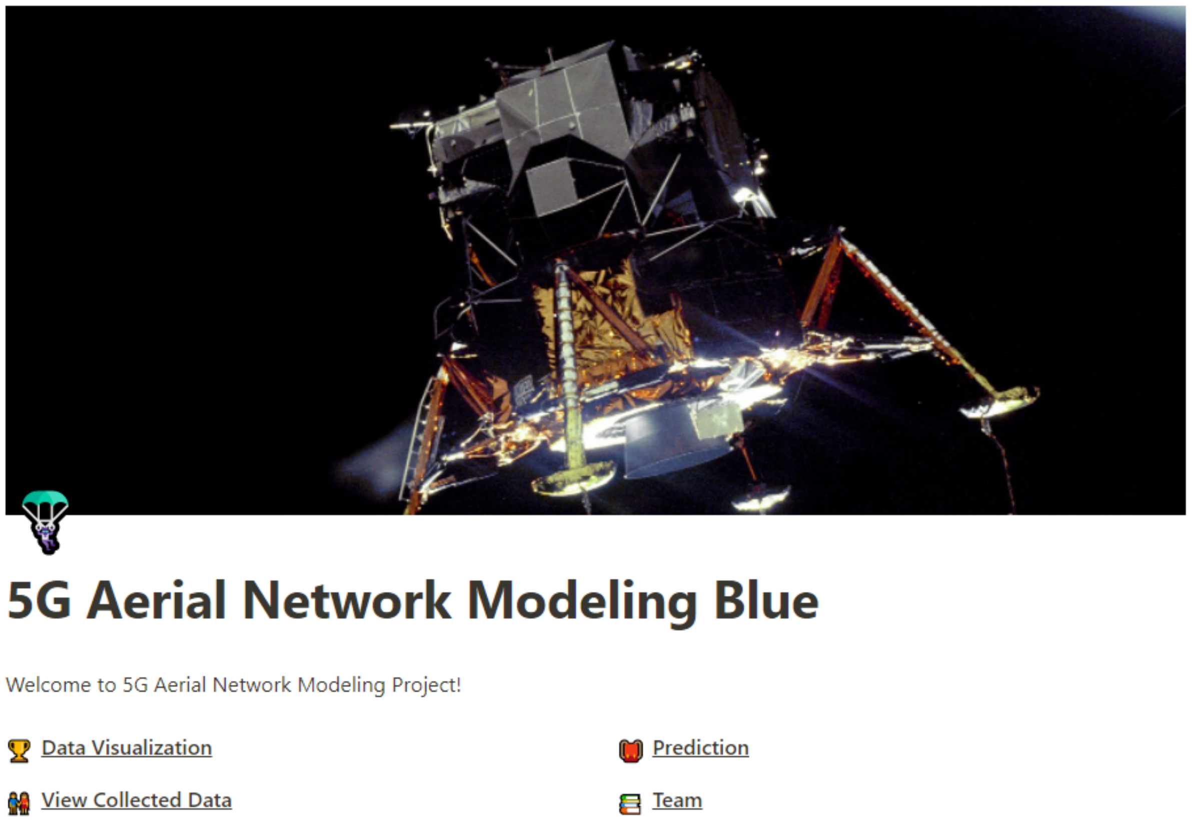
For our machine learning component, we displayed our new and improved regression model for our expanded data set. This utilizes a K Neighbors Regressor and imports many libraries such as pandas, numpy, matplotlib, and sklearn. The python code is done through a series of processes that implement this model:

- Setting/splitting datasets
- Clustering datasets with respect to similar altitudes
- Fitting the multiple variables into regression model
- Model evaluation using different error rate calculations.

The model is shown to input three features: the altitude, the latitude and longitude. When it came to evaluating the model, we used the mean squared error, the root mean squared error, the mean absolute error and the R squared metric. We also output our predictions on a small subset of our dataset that became part of our testing data.

1.5 Web Application

For our web application, we transitioned from an original ASP.NET application using C# to an application built around the Flask framework to utilize both Javascript and Python. We showed our newer run application that allows users to navigate to the visualization and prediction pages. The visualization makes use of the method described earlier that is imported into HTML and the prediction page allows users to click into a map hosted by the Google Maps API and predict the network speeds at that specified latitude and longitude.



5G Aerial Network Modeling Blue

Welcome to 5G Aerial Network Modeling Project!

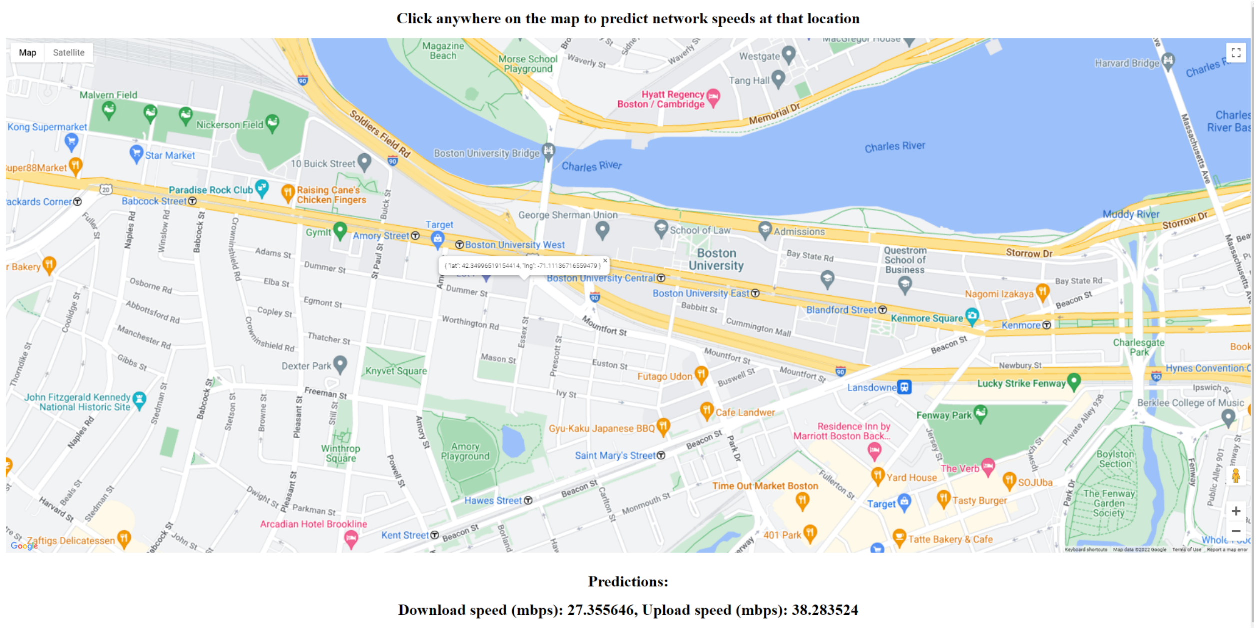
Data Visualization

Prediction

View Collected Data

Team

Figure 1.3 Web Application using Flask Framework



Click anywhere on the map to predict network speeds at that location

Predictions:

Download speed (mbps): 27.355646, Upload speed (mbps): 38.283524

Figure 1.4 Prediction page of web application

2.0 Measurements

Based on our criteria for successful running and displaying, our testing procedure went well in being able to repeatedly and automatically collect data from the phone and send that to our database. This data can be seen in the .json files in our Firebase. As for visualization, this same data can be seen in our Plotly code that displays a well organized and labeled map showing the connection speeds at each location. The machine learning model was able to predict accurately with an impressive R Squared score of over 0.79, showing a good fit. The error rates are also below 5 (MAE, RMSE) and below 30 (MSE). The web application also displays relevant information and incorporates machine learning for prediction results.

3.0 Conclusion

3.1 Results

- Our prototype performed as expected in the data collection, data visualization, predictive modeling, web application portions of our testing. Each team member took part in the presentation and shared the aspects they were able to contribute to since the last demonstration. We demonstrated that we had all our functional components working and future improvements will allow us to expand our visualization tools which will better represent our data. It is promising that our clustered regression model has improved and enables us to focus on refining other aspects of the project.

3.2 Future Plans

- Since the major functioning components of our project are now complete we plan on polishing and putting our final deliverables together for the client over the next couple of weeks. We will continue to improve the usability of our new Web Flask application and collect additional data for our model. We will also add the new plots and maps we have been working on to the application. This will improve our final product without needing to change the architecture of our design. We will also clean up the GitHub documentation for ease of use for the client. Finally, we will make the required changes to the user manual for the client.