Boston University
Electrical & Computer Engineering
EC463 Capstone Senior Design Project

Second Prototype Testing Report

5G Network Performance Testing - Sky Seer

Team #16
5G Drone Team Red

Team Members:

Ryan Mondonedo, mondor28@bu.edu
Peter Wallace, peterjw@bu.edu
Jong Whee Jeon, crm04140@bu.edu
Dohun Ji, kaahsh38@bu.edu
Jungin Chang, changju@bu.edu

Setup and Materials

The setup that was carried out was consistent with our test plan. Our Android phone was connected to AT&T's 5G network, and we ran our application using Ookla's speedtest SDK to collect upload speed, download speed, latitude, longitude, and gps altitude. Since our first prototype testing, we have acquired a SIM card from AT&T, so our testing is now valid for 5G network performance.

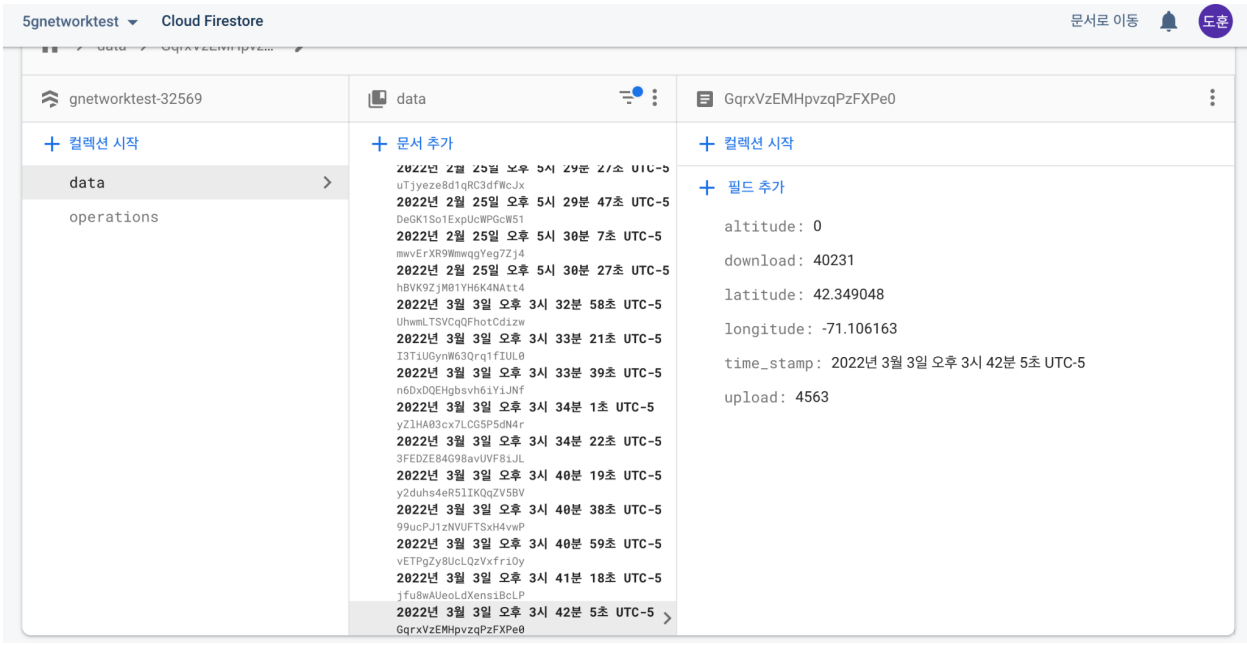
For our prototyping, the physical materials we needed were our Android phone with a 5G SIM card and our Android application installed, a laptop to run our Flask application to control data collection and visualize results, and a laptop to view our Firebase Firestore database as well as to run the machine learning model, and the drone. As far as software materials, we used a Firebase firestore database for our data storage, PyCharm for running our python machine learning code, and our custom Android application using the Ookla Speedtest SDK for data collection. The GPS altitude that the Android Location library gives is the altitude in relation to the WGS84 ellipsoid, a mathematical representation of the surface of the Earth that is inconsistent in some places with the mean sea-level (MSL) altitude (which is exact). This is due to the inconsistency in the Earth's curvature, and can be corrected by using an aviation-grade altimeter that uses barometric pressure. This is not feasible for our project due to budget constraints, runtime constraints, hardware constraints (we are limited to our phone's low quality barometer), and the variability of barometric pressure due to weather conditions seeing as we are going to be flying only at low altitude.

Measurements Taken

The measurements we took are as follows:

- Does the drone turn on?
    - Results: yes, the motors spin the propellers as expected
- Can the drone be controlled by the RC transmitter?
    - Results: yes
- Can the phone stay secured to the drone on the phone mount/clamp?
    - Results: yes, the phone does not move once it is put in place
- Can the network speed be measured by the phone?
    - Results: yes, upload and download speeds could be collected. We measured the upload and download speed at 12295800 bits/sec and 3091980 bits/sec, respectively
- Can GPS coordinates and altitude be measured by the phone for every test run?
    - Results: yes, new GPS coordinates (latitude and longitude) and altitude was measured for every test run
- Can the data be pushed to the database?

- Results: yes, data was pushed into our database (number 31)
  {‘id’,’download_speed’,’upload_speed’,’lat’,’lon’,’alt’}



- Can 5G network performance test operations can be remotely controlled on the web application?
  - Results: yes, user can start or stop network test on mobile application remotely
- Can the data be imported into the machine learning model?
  - Results: yes, we were able to pull data from Firebase to run the machine learning model locally on one of our team member’s laptop
- Can the machine learning model calculate predictions based on the data?
  - Results: yes, we used a multi-out regression model from SciKit Learn on python, with a measured accuracy against test labels of 67%

```
Sample predicdtion for latitude:  42.350872  longtitude:  -71.125286  altitude:  -8.944273
Upload speed:  24285.86 Download speed:  55564.14
Model Acurracy:  0.67
```

Fig 1. Plot produced by model, predicted speed vs actual speed through linear regression

Conclusions

Hardware (drone and 5G phone):

The drone and phone were brought to the lab just so that the hardware apparatus could be demonstrated. Our plan was to show that the motors worked and that we could control the drone with the RC transmitter. We also planned to show how the phone will be attached to the drone by our installed phone clamp. There were not any actual measurements necessary for the hardware other than these qualitative checks.

In the lab, we could only show that the drone can turn on and that the motors work as expected since the lab space was crowded. However, flight testing has already been carried out prior to the demonstration in the lab, so we know that the drone can fly. The drone has a slight drift due to a slight weight imbalance from the equipment attached to the drone. However, the magnitude of the drift is minimal and can be compensated for by the drone controller. For future demonstrations, such as ECE Day, we will create a video of the drone flying as proof of successful operation.

There was not much to show for the 5G phone other than the phone clamp that will be keeping the phone on the drone. The drone is able to support the weight of the phone and clamp, and the phone can be kept secured to the drone by the clamp. From all of this, it seems that there are no significant issues with our current setup and we can move forward with all of our data collection.

Data collection:

We were able to obtain accurate results for upload speed, download speed, longitude, latitude, and altitude (in reference to the WGS84 ellipsoid) in our test. Since the network speed tests were taken in the same place several times, the collected data had a low variation over several network speed tests. The runtime of our tests were also very low compared to our first prototype testing, due to the integration of the Ookla SDK into our project and running everything directly on Android, as opposed to the open source speedtest-cli we were previously running on Python via Termux.

Every collected data for each network speed test including gps coordinate (latitude, longitude), upload speed, download speed, and altitude was successfully pushed into the database and synchronously updated.

Web application:

We were able to control the network performance test operations on the mobile application with the web application. Corresponding to an user selection on a type of operation, the application automatically uploads the new operation command on the database. The mobile application asynchronously catches the updated field value and starts the corresponding command. Furthermore, the web application is integrated with the machine learning model so that the user can train the model whenever new data is updated on the database and predict an upload speed and download speed based on the user input of GPS coordinates and altitude. We also implemented Google Maps on our application so that a user can simply choose or find the GPS coordinates with visualization.

Machine Learning:

Instead of sample data we used from the first prototype test, the machine learning model was trained with our own collected data. By exporting data from Firebase, we were able to access data with separated columns: upload, download, latitude, longitude, and altitude. Data preprocessing using dataframe takes place to make it a trainable set for our model.

The result predicted through the collected dataset has an error range of 100kbps, which is less than the standard deviation of the original dataset. We achieved model accuracy of 67%, which is lower than the sample dataset trained model. We only have 185 tests to train the model, and that is certainly not enough data. More data collection to enlarge the dataset is what needs to be done next to improve the accuracy of the model. For this current model, we produced a graph showing linear regression between predicted and actual labels, which is seen in Fig 1. Going forward in this project, we can work to optimize accuracy of the machine learning model even more.