Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

Drone 5G Network Performance Modeling

Submitted to:
AT&T Labs: Timothy Geraghty
tgeragh@bu.edu

by

Team 16
Sky Seer

Team Members

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

Submitted: 18 April 2022

User's Manual

Table of Contents

Executive Summary (Ryan)

As commercial implementations of aerial drone fleets become more prevalent, there is an increasing need for beyond line of sight autonomous operations. These would rely on communications networks, such as 5G, and would require minimal latency for precise control. Currently, there is minimal understanding of whether 5G network connectivity can support drone operations at altitudes up to 400 feet above ground level, which is the airspace allotted to drones. To obtain a better understanding, this project models 5G network performance and represents a framework that can be used for further studies. We installed a 5G cell phone on a quadcopter drone. The cell phone conducted network speed tests and pushed the results to a cloud database. The collected data was then input into a machine learning algorithm to model 5G performance at different altitudes. The end results of this project are a data collection application, a machine learning model, and a visualization of the data via a web application.

1 Introduction (Ryan)

This is a research project exploring the feasibility of autonomous unmanned aerial vehicles (UAVs) playing a larger role in everyday life. More companies are experimenting with UAVs for automation and services, such as Amazon's Prime Air aerial package delivery. Autonomous UAVs would rely on existing telecommunications networks for connectivity and control and UAVs losing connectivity during operations could potentially be catastrophic. If a UAV were to crash there could be damage to the UAV itself, damage to its payload, damage to other property, and even injury to people in the area. This means that it is paramount to ensure telecommunications networks can support future UAV operations.

5G networks are the forefront of contemporary telecommunications development and implementation, so their ability to support UAV operations is of particular interest. However, the problem is that there is currently insufficient understanding of the quality of 5G network connectivity at the various altitudes that UAVs would operate at because 5G networks are primarily designed to serve users at or near ground level. Our client, AT&T Labs, is interested in better understanding the performance of current 5G networks in the air. By doing so, AT&T Labs would be better informed of what future research is needed to ascertain whether current telecommunication systems are adequate or if changes need to be made to support future UAV operations.

The end result of this project represents a proof of concept for such further study. The project uses machine learning to model 5G network upload and download speeds up to 400 feet above the ground, which is the maximum height the US Federal Aviation Administration (FAA) allows UAVs to operate at. The machine learning model is visualized through a web app so users can view the results. A relevant dataset for 5G upload and download speeds in the air did not yet exist, so a dataset had to be created from scratch. The data was collected by a 5G Android phone attached to a quadcopter drone. The phone ran an app developed from the Ookla software development kit (SDK).

The modeling demonstrated in this project is not necessarily generalizable to a wider geographic area since the data collected to conduct the modeling is limited in scope. However, this is compensated for by the fact that the end product is a modular software framework for users to conduct their own modeling of whatever location they desire. A user simply needs a drone to fly a 5G Android phone. Our Android app and web app handle the rest by conducting the data collection and visualization.

The remaining sections of this manual contain more detailed information on the overall system, how the product is operated, the technical background, the relevant standards the project adheres to, the cost breakdown, the specifications, and the team members.

2 System Overview and Installation (Do Hun, Jung In, Ryan)

2.1 Overview block diagram

As seen in Figure 2.1 below, the end product consists of a hardware and software system. While the main deliverables of the project are software in the form of a data collection application, a machine learning model, and the model visualization, a drone and a 5G Android cell phone were necessary to generate the 5G performance dataset for the machine learning. The phone connects to the 5G network in the area and attaches onto the drone however the user desires. We used a commercial off the shelf phone clamp screwed onto the drone, but since the drone platform is irrelevant as long as the phone does not fall off during flight, one can even use cable ties or tape. As the drone is flown to different altitudes, the phone runs our data collection software. The collected data is automatically pushed to a cloud database we have set up and the saved data is imported into a Python machine learning algorithm to produce our model of 5G network upload and download speeds for the different altitudes.

The data collection software is an Android app. The app uses the Ookla mobile speedtest SDK to perform network speed tests and uses the phone's built-in sensors along with Kotlin libraries for the altitude and position coordinates. The database is a Google Firebase Firestore database. The web app enables remote interfacing with the phone while collecting data by containing commands for executing speed tests and retrieving the results. The app can also display the data.

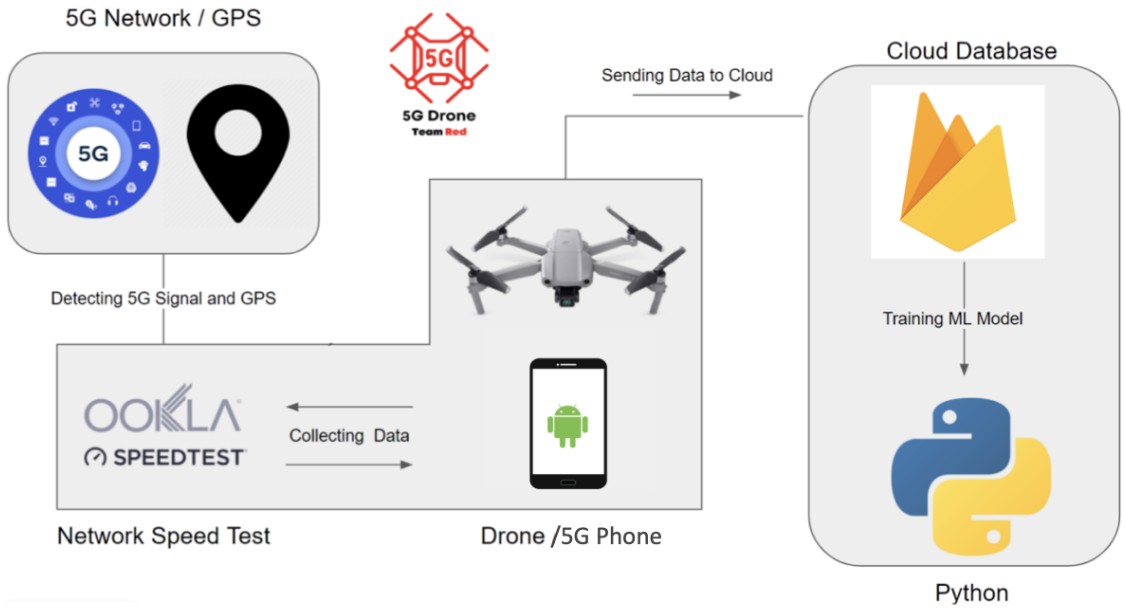

Figure 2.1. Overall system diagram.

2.2 User interface.

The user interfaces of this project are within the web app and the Android app. Web app screens can be seen in Figure 2.2.1, 2.2.2, and 2.2.3. Android app screens can be seen in Figure 2.2.4 and 2.2.5.

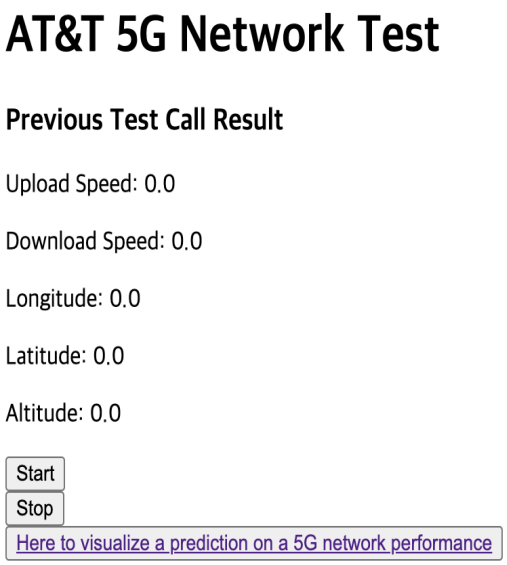

Figure 2.2.1. On this screen of the web app, the user is able to view the results of the previous network test, remotely control the app on the phone (start and stop speed tests), and navigate to the model visualization.

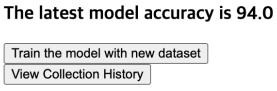

Figure 2.2.2. On this screen, the user can interact with the model directly. The user can also train the model with the most recent data and track locations of any collected data via Google Maps.

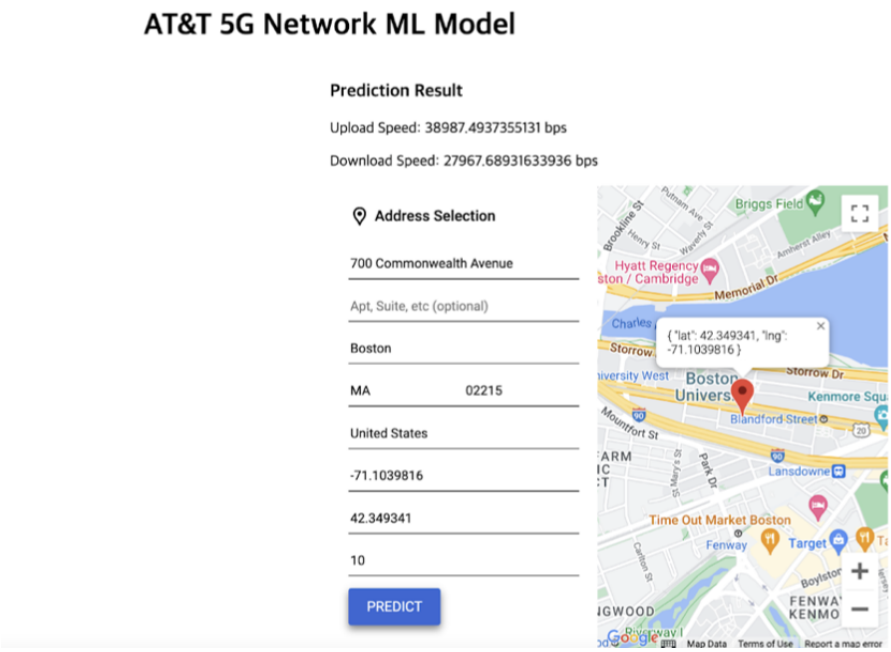

Figure 2.2.3. This is an example of viewing data via Google Maps on the web app.

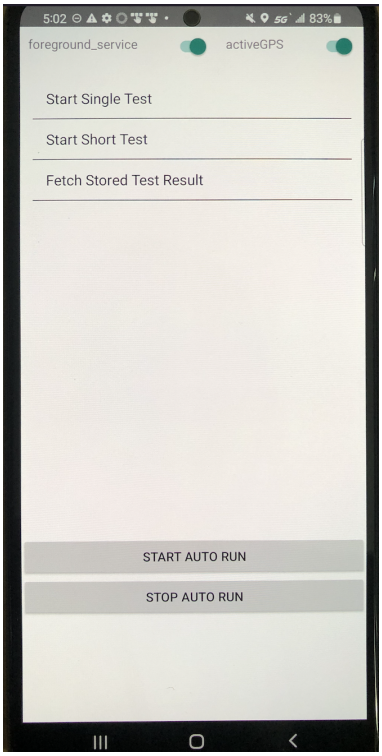

Figure 2.2.4. This shows the opening screen of the Android app. The speed tests can be run manually one by one or an auto run can be started and stopped via the app.

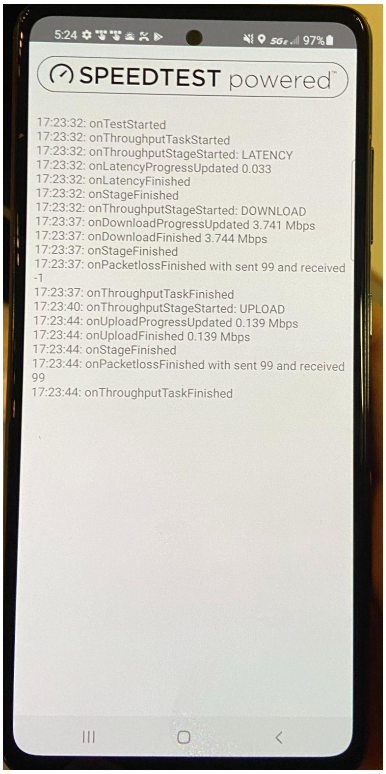

Figure 2.2.5. This shows the app running the network speed tests. The main data points (latitude, longitude, altitude in meters, and upload/download speed) are displayed for the individual test run afterwards.

2.3 Physical description.

The hardware portions of this project consist of a drone and a 5G Android phone and were used only for data collection. Notably, the phone and drone platform are not considered part of the project deliverables as a user is meant to use whatever drone and phone they desire for their own data collection. The specific drone used for the development of this project is pictured below in Figure 2.3.1 and a schematic can be seen in Figure 2.3.2.

Figure 2.3.1. Image of the drone. This was a DIY DJI F450 Quadcopter Pixhawk kit. The phone is attached on top via a car phone mount and the battery is attached below in a pouch. The specific setup may be different depending on the drone model the user decides to use for their own data collection.

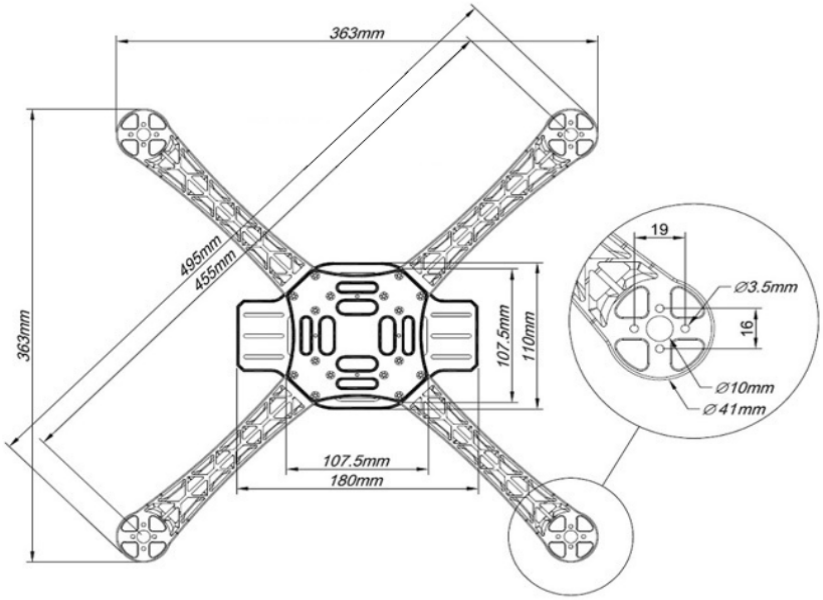

Figure 2.3.2. This is a schematic of the drone frame illustrating the dimensions of the drone.

2.4 Installation, setup, and support

If a user wishes to perform additional data collection, as was done in our project, he or she will need a drone to carry a 5G cell phone. There are many additional components and information that one must familiarize themselves with to operate a drone safely and successfully. First, one must obtain a drone. A ready to fly drone or DIY drone kit can be used. Regardless of what the user chooses, they must ensure the drone platform can carry the cell phone payload, have a compatible RC transmitter, receiver, and battery for the drone, and comply with US FAA drone regulations. The details of such aspects are beyond the scope of this manual and instead can be found in tutorials on the internet. Regarding the cell phone, the user simply needs to install our application. Afterwards, the user can run the data collection script and gather data at their discretion. To install the mobile application, the software needs to be cloned from our GitHub repository (to be provided to customers upon request), and the user needs to have Android Studio installed on their computer. Once the repository is local, navigate to the './att_network_test' folder and open the nested folder 'app' in Android Studio. With the project open in Android Studio, connect a 5G-enabled cell phone via USB to the computer, and click the play button at the top of the screen in Android Studio. If prompted, select to run the

application via the connected phone. Once the application is run through this method, it will be accessible locally on that phone by tapping the Android logo for the application.

Our modeling software will also need to be cloned from our git repository (same as above), which will be provided to any clients upon request. Users will need to have Python 3.9 installed on their machine along with the concurrent Pip version. The necessary packages and libraries will be documented in a requirements.txt file, from which the user can automatically install them using Python Pip. The command for this in a Unix system is *pip3 install -r requirements.txt.* Once installed, users can access our web application via URL and then remotely control the training of their model from the web app and perform predictions based on input longitude, latitude, and altitude data points. The model can also be built and run locally on a Python 3.9-enabled machine by doing the following: navigate to the ./execution folder and run *python3 run.py*. This command will build the model and run a prediction, which can be changed subsequently by editing the longitude, latitude, and altitude fields in the md.predict() function in the main function of the script.

All our components for the web application are included in a repository "Network_test_web_app". We highly recommend constructing and running our web application under a virtual environment which can be initiated with ". *venv/bin/activate"* Unix command because the app requires several dependencies and there's a possibility of conflict if not using a virtual environment. Once under a virtual environment, a user can download all necessary packages and libraries using Python Pip with "pip install -r requirements.txt" command. After all the setup, a user can initiate the application with a command "python3 app/app.py" which will run the server on the local machine.

3 Operation of the Project (Author: Peter)

3.1 Operating Mode 1: Normal Operation

The normal use of this project is handled automatically by the various systems that were implemented by the team. Data collection is performed automatically because many data points need to be collected consecutively at a rapid pace. Users can initiate and end data collection by tapping a button on the Android app. Users can also initiate and end the collection remotely through the Flask web application. Use of the Flask application comes down to the ability to access the URL for the machine it is hosted on, which continuously runs at our discretion. Upon installation this functionality will be passed on to the client. Our Python software is debugged, functional, and will be delivered alongside the web application. Thus, as long as the web application is operational, users will be able to train their model and perform predictions.

Additional complexity arises when the user is conducting their own aerial data collection. In these cases, users must ensure the phone is properly secured to the drone during flight and that the drone flies successfully. The proper operation of a drone is beyond the scope of this manual and such information can be readily found online.

3.2 Operating Mode 2: Abnormal Operation

The main abnormal scenario that the team foresees for customers is the loss of network connectivity during the operation of the data collection application. In this scenario, the network speedtest will time out. Due to our database connection relying on a network connection, the application will also be unable to push any resulting Null or 0 values into the database. Furthermore, any Null or negative upload speed and download speed tuples are removed from the dataset for training and testing via the data preprocessing step in the Python code.

Outside of the above, the team has not encountered any abnormal user modes. Any abnormalities introduced to our software are introduced by users, which is why proper documentation will be provided to prevent users from introducing bugs into the codebase. Because the web application is hosted locally by users, they will not need to worry about issues created by hosting applications with cloud services.

3.3 Safety Issues

There are no safety issues inherent to the software of this project. Safety issues only arise if a user is conducting their own data collection by flying a drone. Safely operating a drone at different altitudes can be difficult, especially if there is wind,

obstacles, or people in the area. Therefore, it is important for users to conduct multiple practice flights to not risk damage to their hardware, damage to any surrounding property, injury to themselves, or injury to other people.

4 **Technical Background** (Authors: Jung In, Do Hun)

Data collection

Android Application:

The Android application is written in Android Studio using the Kotlin language. Kotlin was chosen over Java due to the Ookla mobile speedtest SDK being written in Kotlin. Our final application is a stripped down version of the Ookla sample speedtest SDK application, with scripting added for the results retrieval, the database connection, and the transfer of data to the database. This was done using the Firebase library for Android: com.google.firebase.ktx.Firebase and com.google.firebase.firestore.ktx.firestore.

GPS location:

GPS location is determined using the location provider client developed within the Ookla SDK. Coordinates are retrieved in real time from the Ookla results database to then be stored in the Firebase database.

Altitude:

Altitude is fetched from GPS data. GPS uses satellite signals to measure the distance from a device to a GPS satellite and then compares that distance to the global ellipsoid, which is a mathematical model of the Earth's surface. The altitude used in this project is an approximation, therefore, as the ellipsoid is not truly equivalent to Earth's surface due to inconsistent curvature. Future iterations of this project can incorporate a high quality aviation-grade altimeter that can correct for this inaccuracy in real time using barometric pressure.

Database:

The relational database is used to categorize all features of collected data. Using Firebase Firestore, each test run by the Android app is stored with unique test keys and the data are split into longitude, latitude, altitude, timestamp, upload, and download speed in each test.

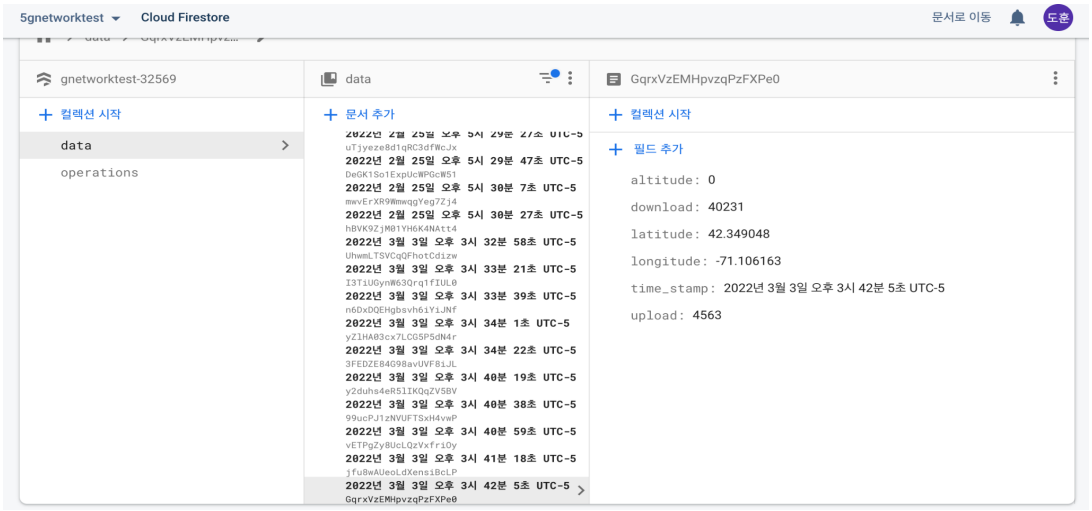

Figure 4.1. Example of database usage.

Data Prediction

Machine Learning model:

The training values are longitude, latitude, and altitude. The target values are upload speed and download speed. All exported data from the database are converted to dataframes for data pre-processing and training.

In the data pre-processing stage, GPS coordinates are clustered into zip codes. Then, the altitudes for each zip code are grouped into 50 meter ranges and the average of the network speeds is calculated for each range. The data are normalized using the calculated upload and download speed averages to avoid underfitting in the model. Finally, the data are split into x and y values and a training and test dataset.

Since there are multiple targets to predict, we fit one regressor per target. The multi-out regression model from the Python package sklearn fits a linear model with a coefficient to minimize the sum of squares between targets in the dataset and targets predicted by a linear approximation.

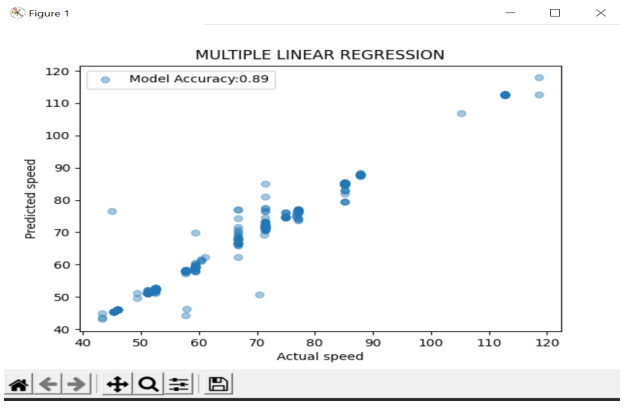

Figure 4.2. A plot produced by the model depicting the predicted speed vs the actual speed through regression.

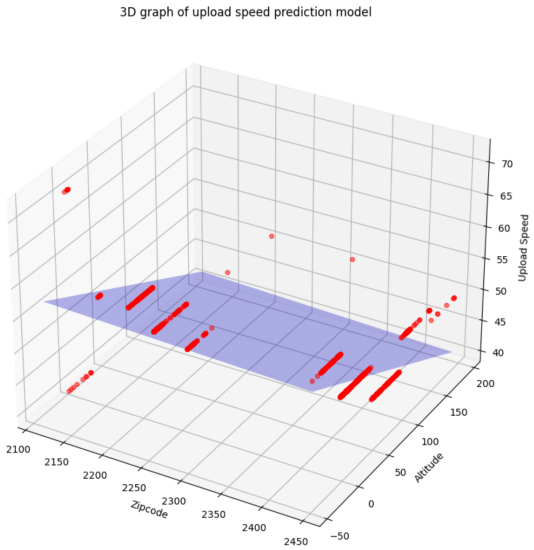

Figure 4.3. 3D plot to visualize the upload speed prediction model based on the zip code and the altitude.

Web application:

The web application is developed upon a framework called Flask and includes HTML, CSS, JavaScript, and Ajax for several functionalities. Overall, Python has mainly been used as the programming language. A key functionality of the web application is to allow a user to remotely operate data collection and prediction on a single platform. Since a user will not have the ability to control data collection nor to monitor the status when the phone is on the drone, real-time data transfer is crucial.

Real-Time Data Transfer

The first task was to establish a reliable connection between the software installed on the Android phone and the web app on a local machine. The real-time database (Firebase Firestore) enables real-time data transfer between two remote devices. Whenever a new data point is pushed to the database, the web app marks the changes by observing certain fields in the database with the function onSnapShot (Document), which asynchronously watches the database.

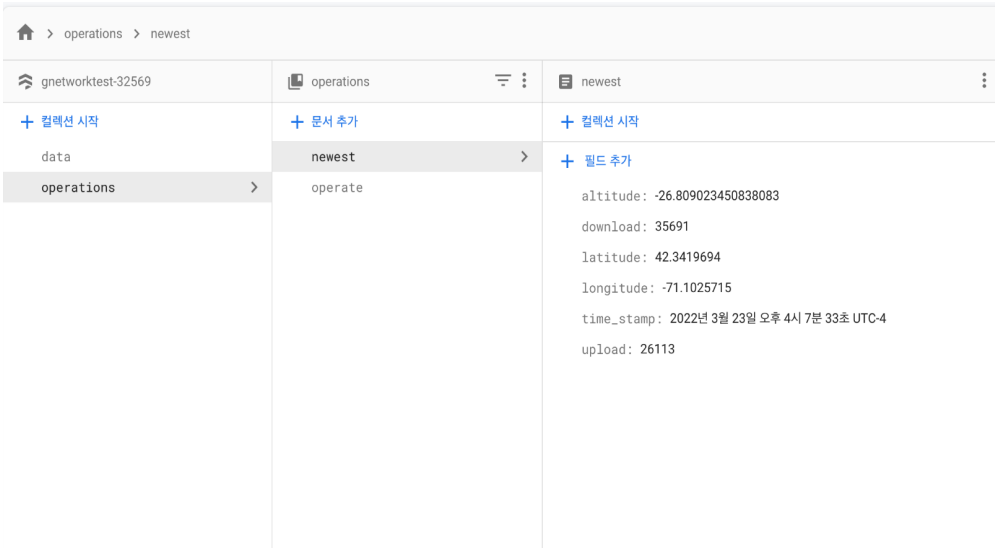

Figure 4.4. Realtime data transfer as shown in the database.

Real-Time Data Visualization

Once the change has been detected, the data are stored in the JSON data format and get POSTed to a certain route. The web app catches POSTed data every two seconds and replaces the previous data with the updated data on the main page, which displays the status of network connectivity tests.

5 Relevant Engineering Standards (Author: Peter)

Our project follows a set of modern software development standards. The codebase was developed with our team working in an agile environment to ensure proper pacing and peer review. We conducted team code reviews at the end of our sprints, which ensured proper code style and documentation within our libraries. Additionally, we have provided extensive documentation in our Git repository using GitHub Wiki, which is still being updated and will be finalized prior to delivery to our client. Over the course of the project we have had a set of rotating Team leaders who have essentially acted as Scrum Masters in each phase of the project.

For the software itself, our mobile application was built entirely in Kotlin using Android Studio, while our machine learning engines and web application were developed in Python. In the Android mobile app, we employed standard Android native libraries to gain access to our phone's GPS capabilities. For the machine learning and web app, we used the SciKit Learn library and Pandas for database manipulation, and used the Python Flask library to build a remote server on which to deploy the website. These libraries are all open source, and their repositories will be linked in the README of the final project. Each library meets the standards for open source development as defined by the O.S. community, and are therefore reputable and reliable tools to use for the project.

6 Cost Breakdown (Author: Jong Whee, Ryan)

The final product of this project is a software framework and therefore does not have any explicit material production costs. All of the expenses for this project are for the hardware necessary for data collection. If users are not conducting any additional data collection, there will be no actual cost for them, as listed in Table 6.1 below. The software has been developed already, so the user just needs access to it to use it.

If users wish to conduct additional 5G network performance data collection themselves, such as mapping a location of their choice, the product cost will be much more, as listed in Table 6.2 below. Users would need a 5G cell phone with a 5G SIM card and a drone to collect 5G data at different altitudes. In the development of this project, we only needed to obtain a 5G cell phone because the drone components and SIM card for the phone were provided to us at no cost. An additional LiPo battery for the drone was acquired as a spare, but this was not absolutely necessary. Users would not be given any components as we were in the development of this project, thus there would be a greater hardware cost for them. Finally, one should note that the cost of a drone and a 5G cell phone can vary considerably depending on the type of drone and phone obtained. The DIY drone kit used in this project is just one type of drone. Users can use other drone models of different sizes and features and they may cost more. This project used an entry-level 5G cell phone, but a user can choose to use a more expensive model.

Project Costs for Production of Beta Version, No Additional Data Collection By User				
Item	Quantity	Description	Unit Cost	Extended Cost
1	1	Only need access to the web application via a computing device	N/A	N/A
			Beta Version-Total Cost	$0

Table 6.1. The total product cost for a user not gathering any additional data. In this case, the user would simply use our web application to view the 5G network performance model for any data already collected.

Project Costs for Production of Beta Version, User Can Collect Additional Data				
Items Provided by Client, Not Considered Part of the Budget for the Project				
Item	Quantity	Description	Unit Cost	Extended Cost
1	1	DJI F450 Quadcopter Pixhawk Drone Kit	$230.00	$230.00
2	1	RC Transmitter and Receiver	$200.00	$200.00
3	1	LiPo Battery Charger	$40.00	$40.00
4	1	LiPo Battery	$68.00	$68.00
5	1	5G SIM Card	N/A	N/A
			Subtotal	**$538.00**
Items Procured for Project, Actual Project Budget				
Item	Quantity	Description	Unit Cost	Extended Cost
1	1	5G Android Phone	$297.49 (phone used here, may vary)	$297.49 (phone used here, may vary)
2	1	LiPo Battery	$68.00	$68.00
			Project Budget Total	**$365.49**
Beta Version Total Cost (Client provided and procured items, minus one LiPo)				**$835.49**

Table 6.2. The total product cost for a user that will be gathering 5G network performance data. To use our software framework (Android data collection app and web app) for 5G modeling at various heights above the ground, users will need a UAV and a 5G Android Phone with a SIM card. Note that the Beta Version Total Cost does not include one of the two listed LiPo batteries given that only one battery is necessary at a minimum to have a working drone.

7 Appendices (Author: Jong Whee)

7.1 Appendix A - Specifications

Requirement	Value
Additional Drone Payload (for DIY F450 drone kit)	~ 1.0 lb, light enough for the drone model since the drone can successfully fly
Drone Flight Time on a Battery Charge	15 min, plus or minus 2 min
Phone Network Connectivity	5G
Position Coordinate Precision	At least 4 decimal places
Machine Learning Model Accuracy	At least 80%
Data Collection Run Time Per Trial	Less than 20 sec
Overall System Cost	Less than $1000.00

Table 7.1. Specification information.

7.2 Appendix B – Team Information

Name	Degree	Email	Hometown	Post Semester Plans
Ryan Mondonedo	B.S. in Electrical Engineering	mondor28@bu.edu	New Milford, Connecticut	Commissioning into the USAF
Jong Whee Jeon	B.S in Electrical Engineering	crm04140@bu.edu	Republic of Korea	Job Search
Do Hun Ji	B.S. in Computer Engineering	kaahsh38@bu.edu	Republic of Korea	Job Search
Jung In Chang	B.S. in Computer Engineering	changju@bu.edu	Republic of Korea	Job Search
Peter Wallace	B.S. in Computer Engineering	peterjw@bu.edu	Encinitas, California	Associate Software Developer at Publicis Sapient

Table 7.2. Team information.