ECE 191
Team G - Drone Integration for RF Scanner Payload

Alexander Bergman, Shenghao Jiang, Jingpei Lu, Wilson Tran
A11885149, A91063762, A91033661, A12127824

in collaboration with

Mentor: Kaan Pinar

March 21, 2017

Executive Summary

The project *Drone Integration for RF Scanner Payload* was intended to continue on the work of a previous group in order to automate the process of a wireless connection site survey. Currently, the process for mapping a location for Wi-Fi and cellular signal strength involves employing a technician who manually records signal strength through a RF scanner and associates this with his current GPS coordinates. These signal strength/GPS coordinate data pairs could then be used to generate a heatmap of cellular or Wi-Fi strength over a region. Obviously, this solution is slow, and requires the employment of possibly multiple individuals, who demand significant wage. Replacing this process with a drone which could perform this scan of Wi-Fi and cellular signal strength and automatically associate this with its GPS location would receive the exact same data the technician uses and save time and money for a surveying company.

The goal for our team was to take an existing model of a drone and integrate it with the previous teams work on a RF scanning payload. The drone had to support a payload of 250 grams (the previous group's payload weight) and provide its location in GPS coordinates through an onboard endpoint to the Raspberry Pi which controlled the existing RF scanning payload. This required research of drone models which would work best for both carrying the payload and ease of integration of the drone's GPS system into the existing onboard computing architecture. Through this integration, our project goal was to use the RF scanning payload and drone in conjunction to generate heatmaps of different types of signal strength data in a specific location. This ideally would replace the on site technicians since these heatmaps would provide the same information, namely signal strength data as a function of GPS coordinates.

The results of our research found the DJI Matrice 100 to be the drone that would best serve our purposes. The DJI Matrice 100 offered an onboard and mobile software development kit (SDK), which allowed for development of applications on both the Raspberry Pi and a mobile phone connected to the drone controller. These applications offered a number of services, namely endpoints for receiving the GPS coordinates of the drone and communication methods between the drone and the mobile phone connected to the controller. Due to the communication function in the DJI Matrice 100 onboard and mobile SDK, the RFM69HCW 915 MHz transceiver used to offload signal strength data from the previous group's payload was deemed unnecessary. The RF scanning device from previous group's payload was also replaced due to inaccuracy, and was instead replaced with a phone which could provide the same information. This modified payload consisting of just a phone and the onboard drone SDK allowed transmission of both GPS coordinates and signal strength data from to the mobile phone application connected to the drone controller. On the application side, this information is used in order to display a heatmap of the data as a function of the coordinates.

Table of Contents

1. Introduction

Our project is sponsored by Hoverport, which is a startup that aims to automate Wi-Fi and cellular signal strength surveys by utilizing a drone. The current industry solution as described in [1] to survey a location for Wi-Fi and cellular signal strength is to employ someone to manually travel around the area and scan frequency bands for their respective signal strength. Although often times the person has sophisticated software which can make signal strength readings and associate it with the GPS coordinates, the employed individuals still has to manually travel the area in order to collect this data. Clearly, this process is time consuming and expensive depending on the quality of the software and the employment of the person.

The problem with this setup is that the employed person demands wages, increasing the cost, and the person manually traveling around the location is slow. Additionally, although advanced software can plot heat maps based on the data collected, this data is only as accurate as the person walking around is consistent. Estimates for the cost of on-site wireless surveys are very high for labor and software included, ranging from $2,500 to $10,000 for a building [2]. The motivation for this project is then to create a drone that can replicate and improve upon a human manually doing this survey for a much cheaper cost.

Often times, for areas that are very large and rural it is hard to perform an on-site wireless survey which can identify the locations in need of additional support. As described in [3], 97% of the nation's landmass is rural and 20% of the American population lives in rural areas. Additionally, [3] asserts that access to technology and infrastructure will increase economic opportunities and standards of living in rural areas. As a result, these on-site surveys detecting deficiency in signal strength coverage have the potential to increase the establishment of infrastructure that could increase the economic and standard of living conditions of millions of Americans.

There exist other things that are similar that approach similar problems. For example, Wisp Drone [5] is a company that offers drones for RF work. One product marketed by them, the Air Survey, is a RF survey unit which can scan bands that correspond to Wi-Fi and cellular signal strength. However, Wisp Drone leaves it up to a user on how to integrate this RF scanner with the drone in order to make decisions based on the data, i.e. heatmaps. As a result, Wisp Drone does not completely solve the problem of on-site wireless surveys. Individuals such as the one in [4] have attempted to tackle the same problem of generating a heat map based on signal strength over a target area. The work here asserts that a multi-rotor could be used in order to generate wireless site surveys, but makes no attempt in creating a commercial version of this software in order to effectively and consistently market the product.

2. Technical Background

2.1 Raspberry Pi

As described in [8], the Raspberry Pi is an open-source hardware platform which is similar to a real computer with computer-like architecture. The Raspberry Pi computer can be used to develop programs for processing data and controlling other pieces of hardware. The Raspberry Pi can be communicated through SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver-Transmitter), or other devices communication protocols [8]. Noting its computing power, the Raspberry Pi can be used to develop applications using the DJI Matrice 100 Onboard SDK which take advantage of the drone software and hardware. We decided to use Raspbian as the OS (Operating System) for our Raspberry Pi due to its similarity to Ubuntu and being a relatively lightweight OS.

2.2 Onboard SDK

The DJI Onboard SDK is a key component which provides the tools for development of onboard applications with logic that integrates with a DJI product. A SDK, or a *Software Development Kit*, is a software interface that allows for creation of applications for a specific software framework or hardware system. This onboard SDK could be used in order to take advantage of the DJI drone hardware for transmission of data and retrieval of GPS coordinates. The SDK is organized into objects called *classes* which each contain a set of functions, or *methods*, which take advantage of specifics of DJI drone hardware.

Extended documentation for functionality provided by the Onboard SDK can be found in [6], but important parts of the Onboard SDK are outlined here:

- DJI::OSDK::MobileCommunication::sendDataToMSDK

This OSDK method sendDataToMSDK is offered in the MobileCommunication class. The method implements the functionality that allows for transmission of data from the OSDK running on the Raspberry Pi to the Mobile SDK on the connected phone to the flight controller assuming the receiving method is implemented. This method accepts a maximum of 300 bytes of information which is sent.

- DJI::OSDK::Telemetry::GlobalPosition::latitude

This OSDK value is offered in the Telemetry class, in the GlobalPosition struct. This struct value latitude, which holds the value of the drone's current latitude in radians.

- DJI::OSDK::Telemetry::GlobalPosition::longitude

This OSDK value is offered in the Telemetry class, in the GlobalPosition struct. This struct value latitude, which holds the value of the drone's current longitude in radians.

Any application developed on the Raspberry Pi can take advantage of these methods assuming the onboard SDK is set up with a connection to the drone. This connection is done through a UART cable connection to the DJI Matrice 100 drone and USB to TTL Serial Console cable connected to the onboard computer, in this case, the Raspberry Pi. This connection diagram from documentation in [6] is shown below.

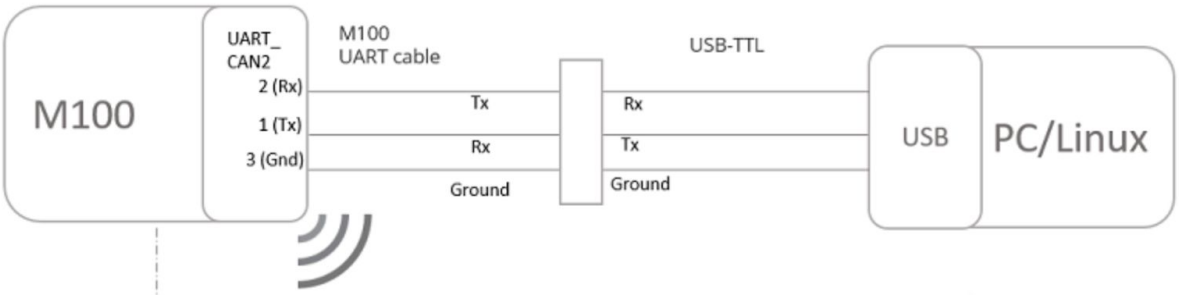

Figure 2.2a) Raspberry Pi and DJI Matrice 100 Connection

As shown in Figure 2.2a, the UART cable that is provided to the drone is currently connected to the USB to TTL Serial Console cable via jumper wires. This connection needs to be secured via heat shrink or soldered for a more permanent solution until further improvements. This connection is necessary in order for the DJI Matrice 100 to provide any support and information via the Onboard SDK to the Raspberry Pi.

2.3 Transmission control protocol

TCP (Transmission Control Protocol) is a one of the main networking protocols. It emphasizes reliability rather than latency. It is implemented in nearly all the devices that can connect to the internet. The procedures of a typical TCP connection:

(1) The client (C) do a three-way handshaking with the server (S) for setting up a connection: C sends a connection request to S. Once S receives the request, it will send an acknowledgement (ACK) for the request from C. Then the C will send an ACK for the request ACK sent by S. Now the connection is set up.
(2) C can send things to S and S can also send things back to C. The packages between C and S will be indexed and error-checked. An ACK will be sent after each package successfully arrives and passes the error-checking. The packages will be sent over and over until an ACK arrives or it reaches the timeout limit.

(3) The connection will be terminated when either C or S sends a CLOSE package or when the connection is idle for a long time.

TCP is a reliable protocol to send messages. We can thus use TCP for setting up communication between the Android RF scanner and the Raspberry Pi.

2.4 Android development

Android Development is a subset of development of mobile applications targeted at Google's Android platform. Extended information can be obtained in the text [9], but it is necessary to understand the information in section *2.3 - Building Blocks*.

Android applications can be broken down into *Activities*, which represent user interface screen logic. Each of these pieces of logic outlines how buttons and text change on the screen, and defines the layout of the screen the user sees in an associated *xml* file. Activities can display anything from buttons to images to maps, and can implement software packages. One such important software package can be accessed through Google's APIs for creating heat maps from a set of weighted latitude/longitude data points.

The switching between *Activities* is initiated by *Intents*, which are "mechanisms for describing a specific action" [9]. These are used to transfer between data between *Activities* and switch the layout seen by the user. Our application will make use of these two building blocks.

2.5 Mobile SDK

The DJI Mobile SDK is a key component which provides the interface necessary for the development of Android applications that can take advantage of the DJI software package. This SDK is used within the mobile application's *Activities* in order to create, display and store data that is available from within the DJI Matrice 100's flight controller software. This mobile application then has access to the *classes* and the *methods* specific to the DJI Matrice 100 which contain information in the flight controller.

Extended documentation for functionality provided by the Mobile SDK can be found in [7], but important parts of the Mobile SDK for our application are outlined here:

- MSDK::FlightController::FlightControllerState::LocationCoordinate3D::getLatitude

This method is located in the LocationCoordinate3D class which is nested within the FlightController class in the Mobile SDK. It returns the FlightController's value for the latitude of the DJI Matrice 100 drone.

- MSDK::FlightController::FlightControllerState::LocationCoordinate3D::getLongitude

This method is located in the LocationCoordinate3D class which is nested within the FlightController class in the Mobile SDK. It returns the FlightController's value for the longitude of the DJI Matrice 100 drone.

- MSDK::FlightController::interface OnboardSDKDeviceDataCallback::onReceive

This OnboardSDKDeviceDataCallback interface within the FlightController class needs to be implemented when a FlightController is instantiated. Within this interface, the onReceive method needs to be implemented, which is called when data is transmitted using the Onboard SDK's sendDataToMSDK method. This method takes in a parameter which is the bytes of data transmitted.

Any mobile application developed on the Android can take advantage of these methods assuming the DJI Mobile SDK is set up and there is a connection to the DJI Matrice 100 controller. This connection is done through a USB cable connected to the DJI Matrice 100 drone controller and mobile android phone.

2.6 Google Maps API

As described in the Android Development section, the *Activities* can use Google's APIs in order to build logic using already implemented features with Google's data. The Google Maps API provides a method for including a Google map *Activity*, and creating a heatmap overlaid on this application. More documentation can be found at [10], but the important components are mentioned here:

- WeightedLatLng

This is how heatmap data is represented in the Google Maps API. Each WeightedLatLng contains a value for latitude, longitude, and weight. Heatmaps are built upon this datatype.

- HeatmapTileProvider

This object is created upon a list of WeightedLatLng points and generates the set of tiles that a heatmap is created on. Creation of this object using a list of WeightedLatLng data is necessary for generating a heatmap using Google Maps API.

- addTileOverlay(HeatmapTileProvider)

This method creates a heatmap using the HeatmapTileProvider created based on the datapoint list. This heatmap can then be added to the *Activity* in the application and displayed to the user.

2.7 RF payload

The RF scanner deployed on the drone is switched from the combination of HackRF and Raspberry Pi to an Android phone. An Android application is installed on the phone to use built-in libraries to extract information about the Wi-Fi signals and the cell signals:

- WifiManager [11]

The method getConnectionInfo() from WifiManager can get all the information of the Wi-Fi signal the phone is connected to. The application on the Android scanner will get the SSID along with the signal strength of the Wi-Fi signal.

- TelephonyManager [12]

The method getAllCellInfo() from TelephonyManager can get all the nearby cell base stations' information. The Android application will pull information of the base station that the phone is registered to. It will get the operator's name, type of the network and the signal strength of that particular station.

There is a lot of unmentioned useful information can be extracted through the libraries. These data can be explored in the future to make the scanner able to accomplish more complicated scanning tasks.

3. System Design

3.1 Wi-Fi and cellular scanning module

Both Wi-Fi and cellular scanning are done by an Android phone that is connected to the Raspberry Pi instead of HackRF and Raspberry Pi from the previous project [8]. This Android phone is taken as the RF Payload that is mounted on the drone, and provides data to the Raspberry Pi on Wi-Fi and cellular signal strength. There are four reasons for the switch:

(1) HackRF is a generalized radio frequency scanner so it sacrifices performance for flexibility. It takes a lot of time to search and analyze the cell signals to provide useful information. However, cell phone is faster in scanning cells since it is optimized for the cell signals. It can automatically search for all the available cell bands and has the baseband chip to process and extract the data.

(2) The signal strength provided by the cell phone is more representative because in some extreme cases, HackRF would be inaccurate and it provide a opposite result compared to the cell phone. Nevertheless, in the end, the phone is the device that utilizes the cell bands. The phone will not work if the signal strength is low on its side no matter what we get from the HackRF.

(3) Cell phone gives us more extendability. We can draw the information of the cell base stations nearby from the cell phone and test the dropped-call rate (DCR) by making phone calls. DCR is an important measurement for the quality of cell signals. We can extend these functionality in the future to make the scanner versatile. HackRF is not capable of doing these.

(4) Moving Wi-Fi scanning from the Raspberry Pi to the phone simplifies the design. Wi-Fi scanning on the phone is similar to it on the Raspberry Pi. If we keep the previous design, the Raspberry Pi will do both data transmission and Wi-Fi scanning. However, with the current design, the phone will be in charge of all the scanning tasks while the Raspberry Pi will only function as a transition center between the phone and the drone.

To complete the cell scanning and Wi-Fi scanning, an Android application is built to pull the desired data. After getting the packed information for the Wi-Fi and the cell, the application will send it to the Raspberry Pi.

3.2 Communication between onboard Android and Raspberry Pi

The communication between Android and Raspberry Pi is accomplished through serial communication via a USB cable. This is one of the possible ways to communicate with the Raspberry Pi and is optimal due to freeing up the General Purpose Input Output (GPIO) pins. There are wireless options but to reduce the interference of the signals we are interested in as much as possible, we decided to go with the wired connection. This wired connection is also more stable than a wireless connection.

Most of modern Android cell phones should support USB tethering. USB tethering is sharing the network that the Android phone is connected to, to the device that the USB is connected to. The phone will function as a router and allocate a local area network (LAN) IP address to the device. We can get a list of IP addresses from the file "/proc/net/arp" in the Android. The file holds a readable dump of the kernel address resolution protocol table used for address resolutions. The LAN IP for the device is between 192.168.42.0 and 192.168.42.255. We can find the LAN IP by this pattern.

After we get the LAN IP of the Raspberry Pi, we can use the famous TCP to communicate between the applications on Android and the Raspberry Pi. The Raspberry Pi will be the TCP server and the Android phone will be the TCP client. The phone will send the signal strengths every 2 seconds and server application on the Raspberry Pi will reprocess the data from the phone with other data that it possesses onboard the drone.

From the previous two sections, we can view the modified attached payload to the drone as in Figure 3.2a. The cellular and Wi-Fi Signal Strength data is sent from the onboard mobile phone every 2 seconds, and this data can be processed in the Raspberry Pi (received via TCP).

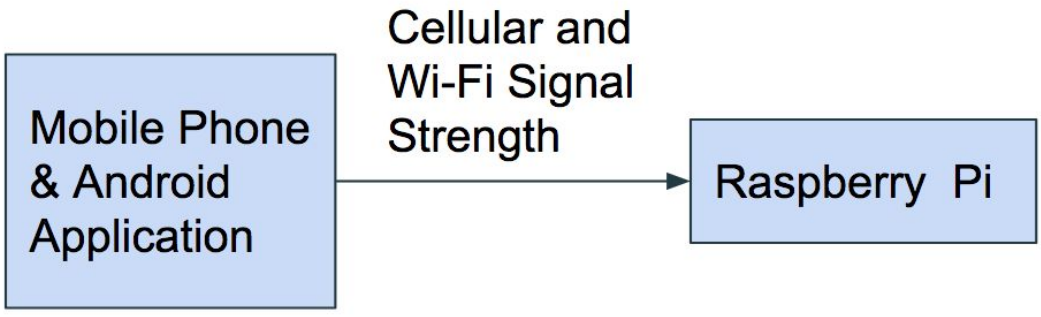

Figure 3.2a) Modified Onboard Payload Communication

3.3 Raspberry Pi and onboard SDK

3.3.1 Setting up the onboard SDK

To setup the Onboard SDK on the Raspberry Pi, pull the latest build version from the DJI-SDK repository. In the case of our application, this was Onboard SDK build version 3.6. The older builds are outdated and have functions that are no longer supported by the latest firmware. After cloning the repository onto the Raspberry Pi, create a build folder in the cloned directory to run and store all the outputs of the `cmake` script. This outputs DJI's sample functions which allow you to ensure the Onboard SDK is correctly setup.

For our application, we set up a Onboard SDK developer ID and developer key in order to register and setup the DJI Matrice 100. This allowed us to create our own programs in the DJI Onboard SDK hierarchy which could use methods provided in the *osdk-core* directory. We added our own file *data-sender* in the *linux* directory of the Onboard SDK environment, and modified the `cmake` script to build our *data-sender* file with the required dependencies in the *osdk-core*. This setup enabled a software connection between the Onboard SDK methods and the *data-sender* program that we could write. The *data-sender* program could then be run using it's main method, at *main.cpp* in the *data-sender* directory.

3.3.2 GPS coordinates

The GPS coordinates are received from the GPS module which comes with the DJI Matrice 100. To acquire the GPS coordinates, we use the DJI Onboard SDK library struct *GlobalPosition* within the *Telemetry* class and obtain the longitude and latitude by accessing these values within the struct *GlobalPosition.latitude* and *GlobalPosition.longitude*. This logic is placed in the *data-sender* main method and runs every 2 seconds, updating the Raspberry Pi program's values for the longitude and latitude respectively.

The longitude and latitude received from this function are in radians. To convert the unit to degree, the value is multiplied by $\frac{180}{\pi}$. Now, the *data-sender* program has an internal representation of the drone's current latitude and longitude in degrees, with a maximum of 2 seconds of staleness. The GPS sensor also must have clear view of a clear sky away from any obstructions and under more than four satellites in order to be most accurate [13].

3.3.3 Data transparent transmission

As described in detail in the background section, data Transparent Transmission serves as a linkage between Mobile SDK and Onboard SDK. Placing transparent transmission in our *data-sender* program running on the Raspberry Pi allows us to use the drone itself as a

communication method between a program interfacing the Onboard SDK and the flight controller.

In our case, the transmission is used in the *data-sender* main method, which executes every 2 seconds. As previously described, this program also receives Wi-Fi and cellular signal strength data via TCP from the connected Android RF scanner, and receives the GPS coordinates from the Onboard SDK itself. This data can be packaged together and transmitted to the flight controller to be accessed by the Mobile SDK.

To transparent transmit, we use the DJI Onboard SDK class *MobileCommunication* and call the method *sendDataToMSDK* to transmit the data. We package the data as a string, which is structured as:

```
line 1: Name of Wi-Fi
line 2: DB Strength
line 3: Cell Signal Name
line 4: DB Strength
line 5: Associated Latitude
line 6: Associated Longitude
```

These string separated with new lines is then converted into a bit array representing the values and sent to the Mobile SDK using *sendDataToMSDK*. This occurs as long as the drone is on and is receiving data from the RF Scanning Android.

Figure 3.3.3 shows the onboard data flow between the *data-sender* program, RF payload implemented as an Android, and Onboard SDK as an interface to the drone itself.

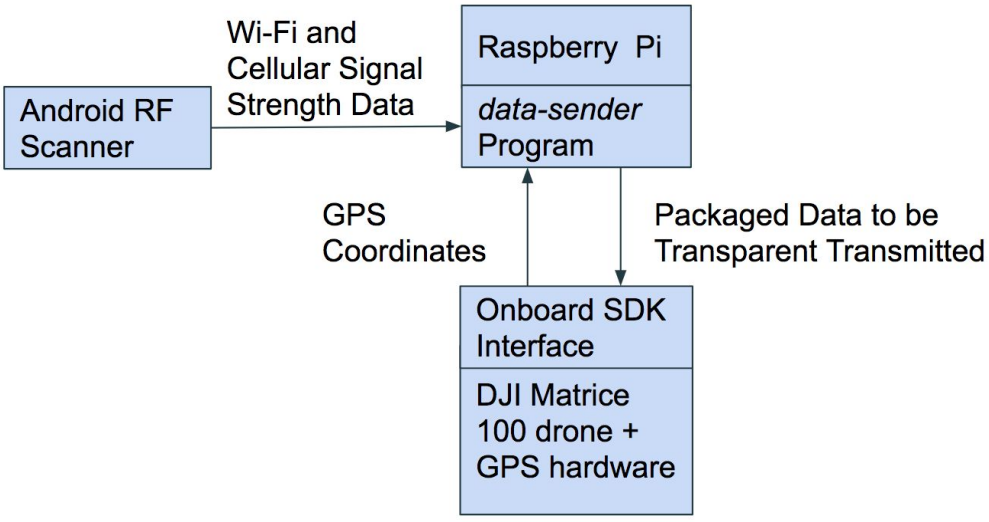

Figure 3.3.3 Onboard System Design

3.4 Mobile SDK

3.4.1 Receiving and displaying data

The Mobile SDK allows for pulling data directly from the controller and receiving data from the transparent transmit. To start, one must registered DJI SDK and develop a DJI Mobile developer application key. Then, one can register the SDK in their application and use the methods offered by the Mobile SDK. These methods are dependent upon a drone being connected to the flight controller and the flight controller being connected to a mobile phone with an application interfacing the DJI Mobile SDK. In our project, we use the Android SDK for developing on an Android mobile phone.

As described in the background on the Mobile SDK section, the Mobile SDK offers methods *getLatitude* and *getLongitude* for a mobile phone interfacing the flight controller. Thus, these were used in order to display the latitude and longitude on the home screen, or *MainActivity* of the application. These can be seen along the top of the screenshot of the *MainActivity* shown in figure 3.4.1a.

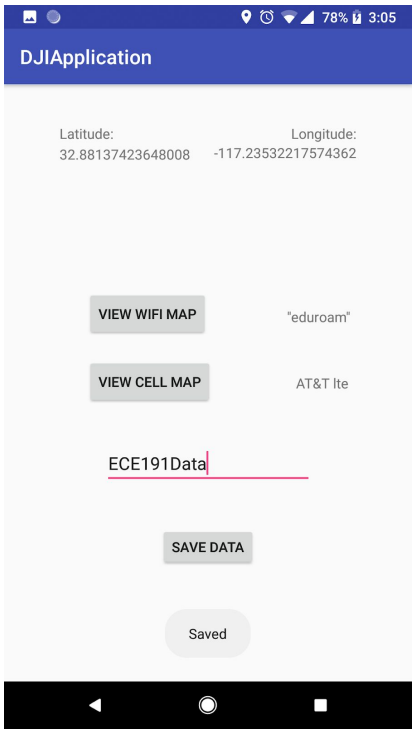

Figure 3.4.1a) *MainActivity User Interface*

Additionally, in the *MainActivity*, the interface OnboardSDKDeviceDataCallback is implemented and the onReceive method is implemented such that all data that is received from the Onboard SDK is stored in a local list. Thus, the *MainActivity* contains two *ArrayList*

structures of *WeightedLatLng* points which are created using the parsed data received in the byte arrays in the onReceive method. In this way, all of the information related to GPS location and Wi-Fi and cellular signal strength coming from onboard the drone is represented in a way that Google's Maps API can use and reflects the information collected on the drone. This connection is shown in figure 3.4.1b.

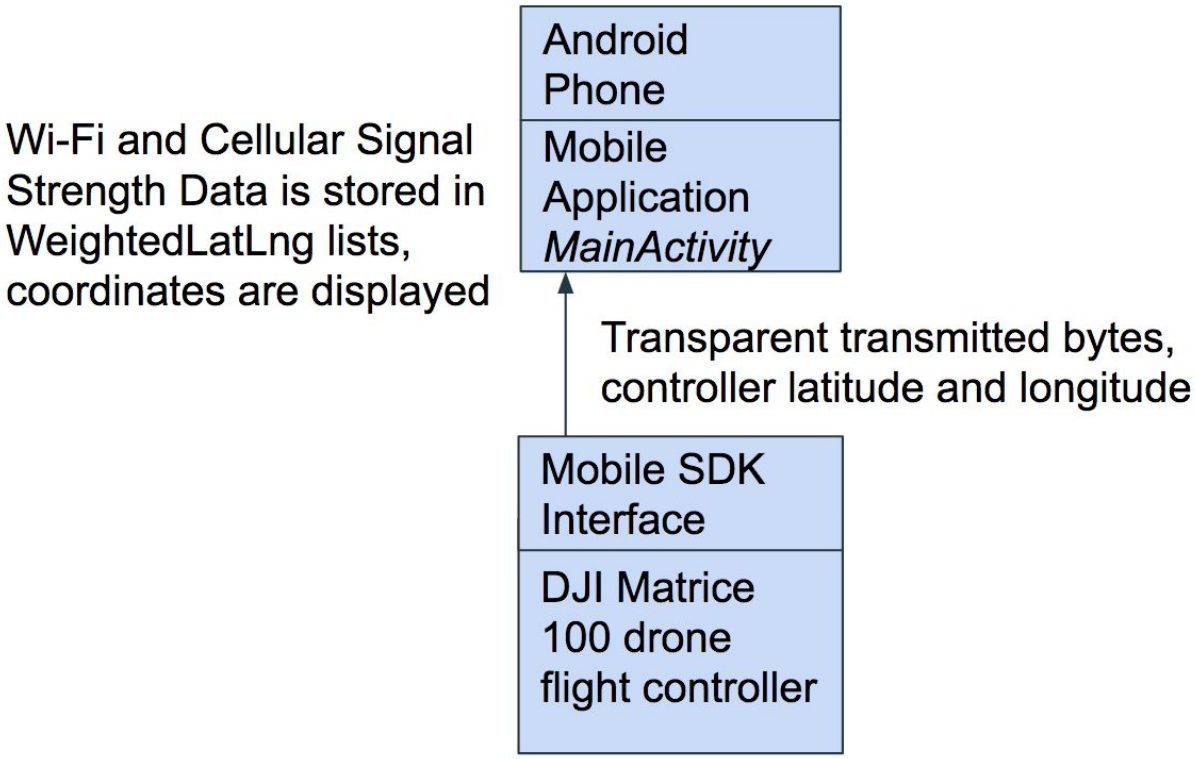

Figure 3.4.1b) *Mobile Application dataflow*

3.4.2 Heatmap construction

Heatmaps are constructed using Google Maps API. Google Maps API provides a visualization to depict the intensity of data at geographical points. When the Heatmap Layer is enabled, a colored overlay will appear on top of the map. By default, areas of higher intensity will be colored red, and areas of lower intensity will appear green.

In our project, we create two separate heatmaps for cellular data and Wi-Fi signal strength data. By providing creating a list of *WeightedLatLng* that is updated whenever the onReceive method is called, we contain a list that Google Maps API can create a Heatmap Layer with. The heatmap is created using a specific *Activity* dedicated to plotting a map. This activity can be accessed by creating an *Intent* based on which view map button is clicked in figure 3.4.1a, and then taking us to an *Activity* that displays a map. This intent also contains a key for which

data the heatmap tiles are to be created for, corresponding to either Wi-Fi or cellular signal strength.

Depending on this key, a *heatmapTileProvider* is created based on the list of *WeightedLatLng* points for either cellular signal strength or Wi-Fi signal strength. This tile provider is then added as an overlay with *addTileOverlay(heatmapTileProvider)*, and can be displayed as a heatmap over the map in the *MapActivity*. This *heatmapTileProvider* is updated every time the dataset receives new points such that the heatmap dynamically updates as data comes in from the Onboard SDK.

This heatmap construction was chosen because it leaves extensive capabilities for future advanced modification and development. We can easily convert the general map view to satellite view and terrain view for the purpose of survey signals in different terrains. We can change the heatmap gradient to other colors for display purpose. We can change the influence of each data point by changing the radius. We can even add more advanced logic for altering the drone flight path in the flight controller through the Mobile SDK based on heatmap results.

3.4.3 Saving data

In order to support further data analytics on the collected signal strength and GPS coordinate data, it is necessary to be able to save the data that these heatmaps were created upon. In the *MainActivity*, a button was created which generates an *Intent* which downloads the list of *WeightedLatLng* points locally to the mobile phone. The user can then take this data and move it to desired locations and perform processing algorithms upon it to derive information from it. Future developers are able to upload these data directly to database, such as Firebase, or run analysis on these data online. In figure 3.4.3, the software structure of the mobile application is displayed.

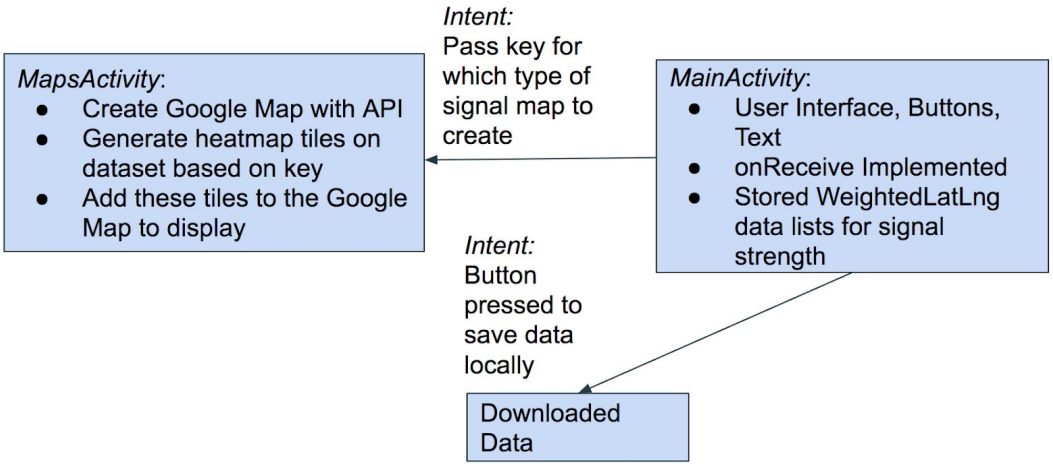

Figure 3.4.3 *Android Application Software Hierarchy*

4. Results

4.1 Phone as RF payload

The Android scanner application will keep pushing data to the Raspberry Pi every 2 seconds. This replacement of the HackRF with the mobile phone clearly worked and it was able to provide the necessary signal strength data with a higher accuracy. The info that the mobile phone provides is shown in figure 4.1, as is the phone settings necessary in order to use the phone as a RF payload.

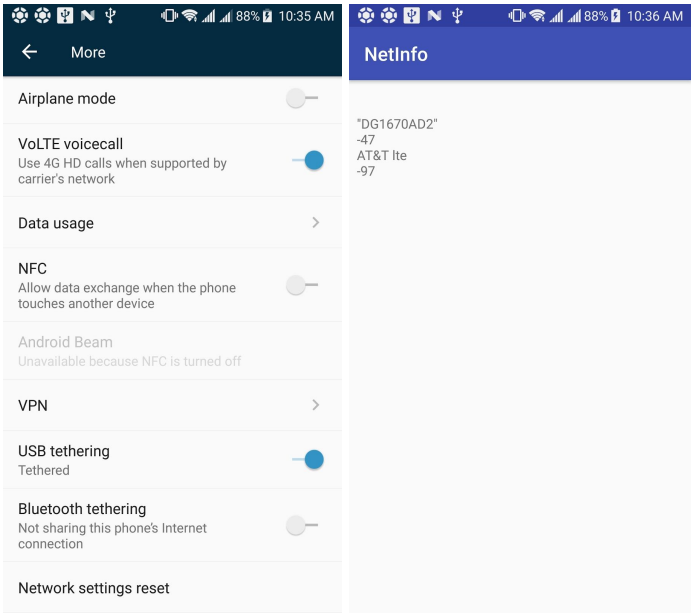

Figure 4.1: [Left: turn on the USB tethering to receive necessary data] [Right: the data that the phone application receives in replacement of the RF scanner]

4.2 Phone App user interface

For a user to view the heatmaps, it was necessary to create an intuitive user interface to navigate the Android application connected to the Mobile SDK. This interface is shown in figure 3.4.1a. The launch page shows the latitude and longitude of the drone, taken from the Mobile SDK. These points are just displayed for the user to keep track of the drone's location. The *View Wifi Map* and *View Cell Map* button will take the user to a screen with a map with the corresponding Wi-Fi or cellular signal strength data plotted as a heatmap over the location. The name of Wi-Fi network and cellular signal are displayed to the right of the buttons. In the edit text box, users are able to type in the file name they want the data file saved as. Users are then able to save the data to their phone by clicking the *Save* button.

4.3 Heatmaps

The ability to plot heat maps of Wi-Fi and cellular signal strength, which was the goal set forth by our sponsor, was met. These heatmaps are shown on the phone by clicking the correct button from the main user interface. These heatmaps plot areas of strong signal strength red, and areas of weak signal strength green, and values between along a color gradient. The prototype works in theory with the drone being carried around, but due to an accident earlier in the quarter we were unable to test the setup with a drone being controlled flying around.

In figure 4.3, a Wi-Fi signal heatmap of "UCSD-PROTECTED" and cell signal heatmap of "ATT LTE" are shown. In the experiment, we carried the drone around Warren Mall at UCSD and plotted these two heatmaps in real time. The Wi-Fi heatmap shows poor signal and disconnected signal strength on the walking path. This result matches expectation, since poor Wi-Fi connection to "UCSD-PROTECTED" is generally experienced here. Additionally, the RF scanning Android was observed to have lost connection to "UCSD-PROTECTED" here. Due to multiple routers on campus, and the ability to reconnect and disconnect, we expect this spotty coverage along our path.

The cell signal heatmap shows strong signal on our walking path. This matches our expectation that usually we receive strong ATT LTE signal on campus. Additionally, the phone was visually observed during this walk and the phone displayed full bars of service all along the walk.

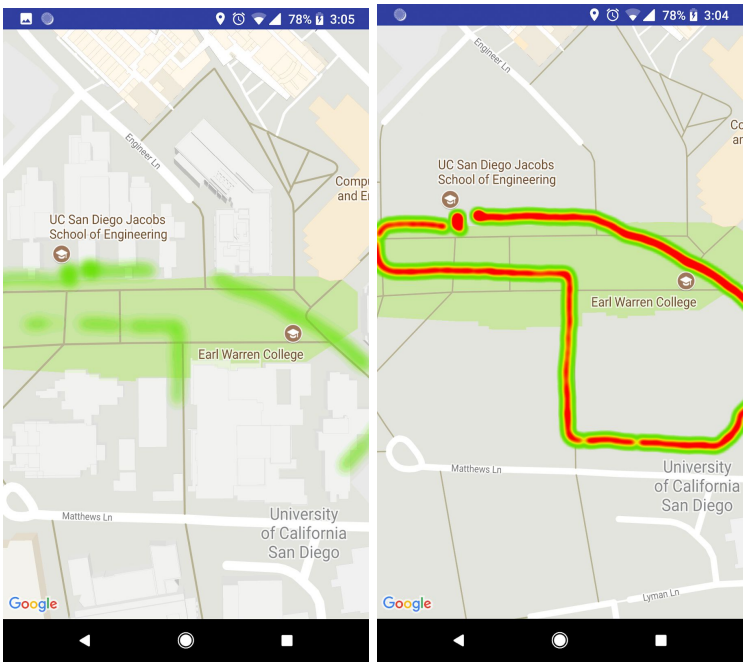

Figure 4.3 [Left: Wi-Fi signal heatmap] [Right: Cell signal heatmap]

4.4 Data saved

Finally, capability is offered to save and offload the data that represents the signal strength reading and GPS coordinate pairs. This is essential to our sponsor Hoverport, since they can perform more advanced analytics upon the data with these files. The file can be named by the user, making smooth transition from phone to file. Inside the file, we are able to check all of the data recorded, such as latitude, longitude, cell signal name, cell signal strength, Wi-Fi name, and Wi-Fi signal strength. This capability is shown in figure 4.4.

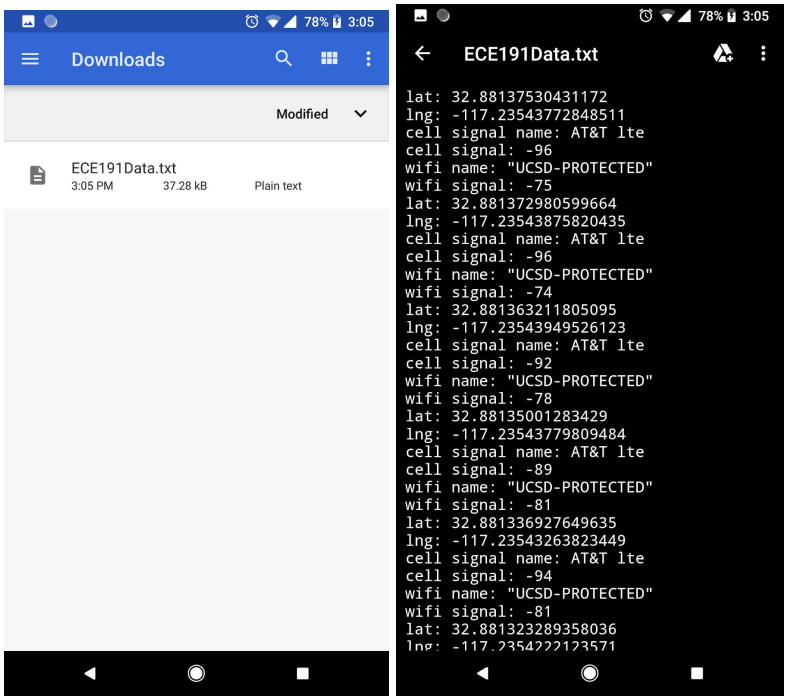

Figure 4.4 [Left: Saved, named file] [Right: Data in the file]

5. Safety, Ethics, and Diversity Issues

The drone must comply with guidelines outlined by the Federal Aviation Administration [15]. These rules restrict the altitudes and locations that drones can be flown in the National Airspace System. These guidelines are created in order to ensure safety for all parties.

The drone needs also needs trained and experienced drone pilot in the case of non autonomous flying, since in the current state of development a flyer is required. The high speed propellors could cause damage if they were to collide with a person.

As mentioned in the introduction and in [3], this product could accelerate the access to high speed Wi-Fi and cellular strength in rural areas. This would quickly improve their standard of living and this would be a differential effect on minority groups but this would be a positive effect.

6. Conclusion

The goal of our project was to integrate a drone with a RF scanner and generate a visual representation of signal strength in a scanned area. By generating a heatmap, we are able to provide a replacement to on-site RF technicians displaying data that they surveyed. Our results consisted of using the various DJI SDKs in order to create programs onboard and on the mobile phone that took advantage of the DJI hardware to package GPS data with signal strength readings from the RF scanning payload and send them to a mobile phone. The RF scanning payload was changed from a HackRF frequency scanner to a mobile phone to make the readings more accurate and more conducive to future use. Finally, the mobile application that received the data through the DJI drone was able to use Google Maps to produce a heatmap displaying data around an area scanned by the drone.

Using a drone for this task of on-site surveying would be faster and cheaper. This would be useful to companies and institutions worldwide who necessarily have to plan their working environments in order to provide good Wi-Fi and cellular signal strength to customers and employees. Additionally, because of the speed and efficiency of a drone, this service could be used to detect deficiencies and vast swaths of area where Wi-Fi or cellular signal strength is very low or nonexistent. Additionally, generalizing the project to a generic "Drone-as-a-Service" project, the RF scanner could be replaced with any data collecting module, and that data could be mapped as a function of location.

The next steps in this project would be to integrate DJI's Guidance system to the drone, enable the DJI Matrice 100 to have data driven flight decisions, and finally incorporate more types of data that Hoverport can use. The Guidance system will enable the drone to have high-precision positioning even without a GPS. This is possible due to the vision positioning system DJI provides and is effective at altitudes of up to 65 feet (20 meters) [14]. By including Guidance, the drone could be flown in much tighter spaces, such as indoors, much more safely. These are critical areas where production of a heatmap of Wi-Fi and cellular signal strength would be helpful to on-site surveyors. Additionally, because the Mobile SDK offers connection to the flight controller, it is possible to process the data received in the app and control the drone as a function of this. This would allow the drone to fly completely autonomously, and explore areas with low signal strength. A SLAM (Simultaneous Localization and Mapping) [16] algorithm could be implemented to explore new areas and make data driven decisions on where to explore. Finally, because the HackRF was replaced with a mobile phone, information such as dropped calls could be recorded. A simple frequency scanner like a HackRF does not have access to all of the parameters that a mobile phone needs to operate smoothly, so the use of a mobile phone itself as the onboard data scanner makes sense. Future groups could work on

implementing these features, along with making overall UI/UX improvements to the application to ensure a satisfying experience when utilizing the product.

References

[1] Dionicio, Rowell. “Performing Wireless Site Surveys.” Network Computing, 8 Mar. 2016, www.networkcomputing.com/wireless-infrastructure/performing-wireless-site-surveys/124269781.

[2] Wegner, Philip. “How Much Does a Wireless Site Survey Cost?” *Securedge Networks*, 24 Jan. 2011, www.securedgenetworks.com/blog/how-much-does-a-wireless-site-survey-cost.

[3] Smith, Diane. “The Truth About Spectrum Deployment in Rural America.” *Mobile Future*.

[4] Pack, Scott James. “Multi-Rotor--Aided Three-Dimensional 802.11 Wireless Heat Mapping.” Brigham Young University, 2014, scholarsarchive.byu.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=5014&context=etd.

[5] “WISP Drone.” *WISP Drone*, wispdrone.net/.

[6] “Onboard SDK.” Onboard SDK Documentation Home - DJI Onboard SDK Documentation, developer.dji.com/onboard-sdk/documentation/introduction/homepage.html.

[7] “Mobile SDK.” *Documentation Introduction - DJI Mobile SDK Documentation*, developer.dji.com/mobile-sdk/documentation/introduction/index.html.

[8] Zhu, Yupei, Liang Wei Nan, Shankar, Srihari, *Continuous RF Scanning with Open Source HW*.

[9] Ed Burnette. *HELLO ANDROID*. Shroff Publishers & Distr, 2015.

[10] *Google Maps Android API | Google Developers*, Google, developers.google.com/maps/documentation/android-api/.

[11] “Android.net.wifi.WifiManager.” *Android Developers*, 7 Mar. 2018, developer.android.com/reference/android/net/wifi/WifiManager.html.

[12] “Android.telephony.TelephonyManager.” *Android Developers*, 7 Mar. 2018, developer.android.com/reference/android/telephony/TelephonyManager.html.

[13] “GPS Basics.” *Sparkfun*, learn.sparkfun.com/tutorials/gps-basics.

[14] “Guidance: A Revolutionary Visual Sensing System for Aerial Platforms - DJI.” *DJI Official*, www.dji.com/guidance.

[15] “Unmanned Aircraft Systems.” FAA, 12 Mar. 2018, www.faa.gov/uas/.

[16] Collier, Jack. “SLAM Techniques and Algorithms.” www.computerrobotvision.org/2010/slam_camp/collier_intro.pdf.

Appendix

Onboard Github repository: https://github.com/alexanderbergman7/ece191-dji-osdk
Mobile App Github repository: https://github.com/alexanderbergman7/ece191-dji-mobile