Flight Testing of Cell Network-Based Navigation for sUAS

Helen Colleen Kuni

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics & Astronautics

University of Washington

2021

Committee:

Juris Vagners

Christopher Lum

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Flight Testing of Cell Network-Based Navigation for sUAS

Helen Colleen Kuni

Chair of the Supervisory Committee:
Professor Emeritus Juris Vagners
Aeronautics and Astronautics

The operational weaknesses and limitations of a navigation system based solely on GPS have long posed a problem for unmanned aircraft. The lack of any non-GPS-based backup system leaves vehicles vulnerable to sources of signal degradation or loss such as GPS jamming and signal multipathing, which can pose a serious safety hazard. Previous work has been done to develop an alternate navigation system, which generates a position estimate based on signals from cellular and Wi-Fi networks, known as WiCeNav. This study initially attempted to integrate WiCeNav onto an Unmanned Aerial Vehicle, or UAV. However, this proved unfeasible due to the fact difficulty of obtaining sufficient Wi-Fi data in the test environment. An alternate position estimation solution using multilateration of cell signals was therefore devised, and numerous problems associated with prior attempts at developing a system of this type were revisited and investigated in detail, shedding light on many key aspects of the functioning of cellular networks. Flight testing was conducted to investigate the feasibility of using the system for autonomous navigation. This did not produce a positive result, in part due to inaccuracy associated with the model used to relate power received to distance form a given tower, and in part due to inherent limitations of the network itself. However, this research provided valuable insight into the challenges of using cell networks for navigation, and a number of modifications and improvements are suggested which may drastically improve the accuracy and robustness of the system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ADS-B	Automatic Dependent Surveillance - Broadcast
AFSL	Autonomous Flight Systems Laboratory
AGL	Above Ground Level
API	Application Programming Interface
EARFCN	E-UTRA Assigned Radio Channel
FAA	Federal Aviation Administration
FCC	Federal Communications Commission
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSM	Global System for Mobile communication
LTE	Long-Term Evolution
MSL	Mean Sea Level
NED	North-East-Down
PCI	Physical Cell ID
RSSI	Received Signal Strength Indicator
(s)UAS	(small) Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UW	University of Washington
UW-CUTS	University of Washington Carnation UAS Test Site

NOMENCLATURE

A_r	Effective aperture area, receiving antenna
A_t	Effective aperture area, transmitting antenna
c	Speed of light
D	Distance
f	Frequency
G	Antenna gain
P_r	Power received
P_t	Power transmitted
x, y, z	Position of the drone
x_t, y_t, z_t	Position of a cell tower
λ	Wavelength

ACKNOWLEDGMENTS

I would like to thank my advisers, Dr. Juris Vagners and Dr. Christopher Lum, for their many years of support, encouragement, and assistance, both with this project and others throughout my time with the Autonomous Flight Systems Lab. I would also like to thank the following AFSL members for their essential support on this project: Karina Bridgman, Daniel Broyles, Christopher Hayner, Kenneth Wilsey, and Howard Peng. Special thanks also to Sean Murphy from T-Mobile. My work on this project was also supported by the Joint Center for Aerospace Technology Innovation (JCATI) and the Latvian Arctic Pilot - A. Vagners Memorial Scholarship.

Chapter 1
INTRODUCTION

1.1 Problem Statement

Many Unmanned Aerial Systems (UAS) use GPS as their primary sensor for autonomous navigation. However, the circumstances which can cause GPS signals to be degraded or denied are all too common in the most frequent applications of UAS, and many platforms do not have any backup system in place that would allow autonomous navigation to be maintained in the event that GPS is lost. As the use of drones in urban areas for applications such as package delivery increases, operators must worry about signal multipathing: the phenomenon where a GPS signal reflects off of tall buildings or other obstructions before reaching the sensor, thereby making it seem as if the signal originated from the wrong place, and introducing significant and often fatal amounts of error. Degradation of GPS signals may occur in a multitude of rural environments as well, if a connection to a sufficient number of satellites cannot be maintained due to obstructions such as tall mountains or hills. Intentional jamming of GPS signals is a significant concern for drones in military applications; GPS jammers are inexpensive and easy to build, and can be used to render a UAS entirely nonfunctional with no warning. There have even been cases in recent years of aircraft navigation systems being compromised unintentionally by illegally-built civilian GPS jammers on the ground. There is an overwhelming motivation for the development of an alternate navigation method that does not rely on this type of signal, and this need will become even more pressing as the UAS industry progresses in the coming years.

This thesis investigates a navigation system designed for use on UAS which, instead of relying only on signals from a GNSS network, determines the aircraft's position by means

of multilateration of cell signals. The system exists within a lightweight, modular payload that can easily be integrated onto a variety of different types of UAS. Although the system did not prove to be successful, the design and testing process provided valuable insight into what is required in order to make such a system feasible.

1.2 Prior AFSL Work

This research is a continuation of work done by Matthew Davis [1], who created the payload and developed the software that produces position estimates based on cellular and Wi-Fi signals. Davis developed three variations of the system, which accomplish the task of position estimation by different means with varying degrees of accuracy and reliability. To summarize these:

1. TelitHE910D, ATMONI: This solution uses multilateration of cellular signals, and uses the OpenCellID combined with Google's Geolocation and Elevation APIs to obtain tower locations.

2. FONA, ATCIPGSMLOC: This solution is the same as the previous one, but with a different database of cell tower locations and a different configuration used for the cell modem.

3. WiCeNav: This solution combines cellular and Wi-Fi data and queries the Google Geolocation API in order to obtain a position estimate.

Of these three solutions, WiCeNav was found by Davis to be the most accurate. Solutions 1 and 2 were abandoned largely due to difficulties in obtaining the exact position of the cell towers. Most readily available databases of tower location data actually report the approximate position of the center of the cells rather than the actual, physical tower, and obtaining the elevation of the tower is even more challenging. These problems go from inconvenient to disastrous when the system is used in a rural area, where the radii of the

local serving cells are comparatively large (> 1km) and position of the center of the cell may consequentially be nowhere near the actual tower.

The WiCeNav solution, although comparatively better, presents problems of its own. First, the call to the Google Geolocation API results in a significant amount of latency, even in an urban environment with a strong LTE signal. The amount of delay was enough to render the system unsuitable for real-time autonomous navigation. The payload also includes more components than solutions 1 and 2, making it the heaviest, bulkiest solution tested, and introducing many more possible sources of error, interference, and malfunctions. It also relies on having access to not only cell signals, but also Wi-Fi signals near the flight area, which is not something that can be counted upon when flying in rural environments. These problems are investigated in this thesis.

More work has previously been done within the Autonomous Flight Systems Laboratory (AFSL) on other methods of GPS-degraded/-denied navigation. These include TRAPIS (the Transponder-Based Positioning Information System) ([2],[3]), which estimates position based on multilateration of signals from an onboard ADS-B transponder[1], and Visual Anchoring ([5],[6]), which uses a camera to estimate position relative to a visually distinct object. These methods, and any others which are developed in the future, could hypothetically be integrated into the system proposed here, due to the modular nature of the payload and software architecture. This possibility is not investigated here, but is mentioned simply to highlight one of the advantages of the system that has been developed, and point out a potential continuation of this research.

1.3 Prior Work Beyond AFSL

A significant amount of research in the area of cell network based navigation has been done by Zak Kassas at the University of California, Irvine's Autonomous Systems Perception,

[1]A regulation passed by the FAA in January, 2021 forbids operators from equipping their UAS with ADS-B transponders, rendering TRAPIS illegal to use in its current form [4]. This rule was not in place at the time of the referenced research.

Intelligence, and Navigation (ASPIN) Laboratory [7]. Kassas and his team have developed a software-defined radio system that monitors signals of opportunity from a variety of sources, including cellular networks. Several cell providers, including Verizon, Vodafone, and Ericcson, have also begun developing methods for applying their network capabilities to the problem of UAS navigation. The research presented here was done with the benefit of a network subscription (distinguishing it from work done by Kassas), but with far less information about the network than companies such as Verizon developing their own solutions have access to. This solution is also not tied to any specific network, and could easily function using data from a different one simply by the use of a different SIM card.

Chapter 2

FLIGHT TESTING OF THE WICENAV SOLUTION

This section covers the attempted integration of the WiCeNav solution onto an aircraft, and discusses why this solution was not adopted.

2.1 WiCeNav System Overview

The design of the WiCeNav payload and the selection of components was described in detail by Davis, 2020. A summary is provided here for convenience. The components of the payload are described in Table 2.1. Originally, the Raspberry Pi, Wi-Fi scraper, and one of the two cell modems were enclosed in a plastic box, with the remaining components secured to the box's exterior with velcro. The payload uses five antennas: two associated with each of the two cell modems, and one for the the Wi-Fi scraper. Davis also used a GPS unit for truth data; this has been omitted for two reasons. First, it is not necessary when the payload is installed on an aircraft which has its own GPS unit. Second, the GPS unit used by Davis is equipped with a strong magnet, intended to allow it to be mounted on a car for ground

Component	Purpose
Raspberry Pi 4b	Computing
Telit HE910D Cell Modem	Obtaining local cell signals
Charmast USB-C Power Bank	Power
Sierra MC7455 Cell Modem	Provides internet for Google API call
Adafruit Feather Huzzah	Wifi Access Point scraper

Table 2.1: Payload components.

testing purposes. The magnet could not be removed, and it was not possible to install this on the aircraft in a position where it would have strong signal but would not interfere with the function of either of the aircraft compasses. Also note that the Sierra cell modem, which provides internet to the Raspberry Pi via LTE, requires a SIM card which has an active network subscription. The Telit modem, which simply gathers data on the surrounding cells without actually connecting to the network, does not require its SIM card to be activated.

As described in the previous chapter, and in detail in Davis, 2020, the WiCeNav payload obtains a position estimate by sending cellular and Wi-Fi access point data to the Google Geolocation API. The API returns a latitude, longitude, and error radius. The documentation does not provide a detailed explanation of how this position estimate is generated; it is essentially a black box.[8].

2.2 Payload Installation

The physical integration of the payload hardware onto the aircraft proved to be challenging. The most significant problem was the size of the payload, which was too large to be easily integrated onto the aircraft, despite modifications to the airframe and to the payload, and the fact that this aircraft already featured one of the largest payload capacities in the AFSL fleet. The box containing the Raspberry Pi, the Wi-Fi scraper, and one of the cell modems was designed to fit in the payload compartment, but not to pass through the slightly smaller opening of the compartment. Even after this opening was widened, the size of the box did not leave room for the power bank or the second cell modem. It was necessary to transfer the Raspberry Pi to a smaller case and secure other payload components with velcro in order to make installation feasible. The five antennas also proved challenging to install, not only due to their number but also their size. In particular, the antenna for the Wi-Fi scraper is 15.25 inches long, and affixing this to the aircraft in such a way as to provide it with a good signal without interfering with the hand-launch of the aircraft was nearly impossible.

The payload installation is shown in Figures 2.1 and 2.2.

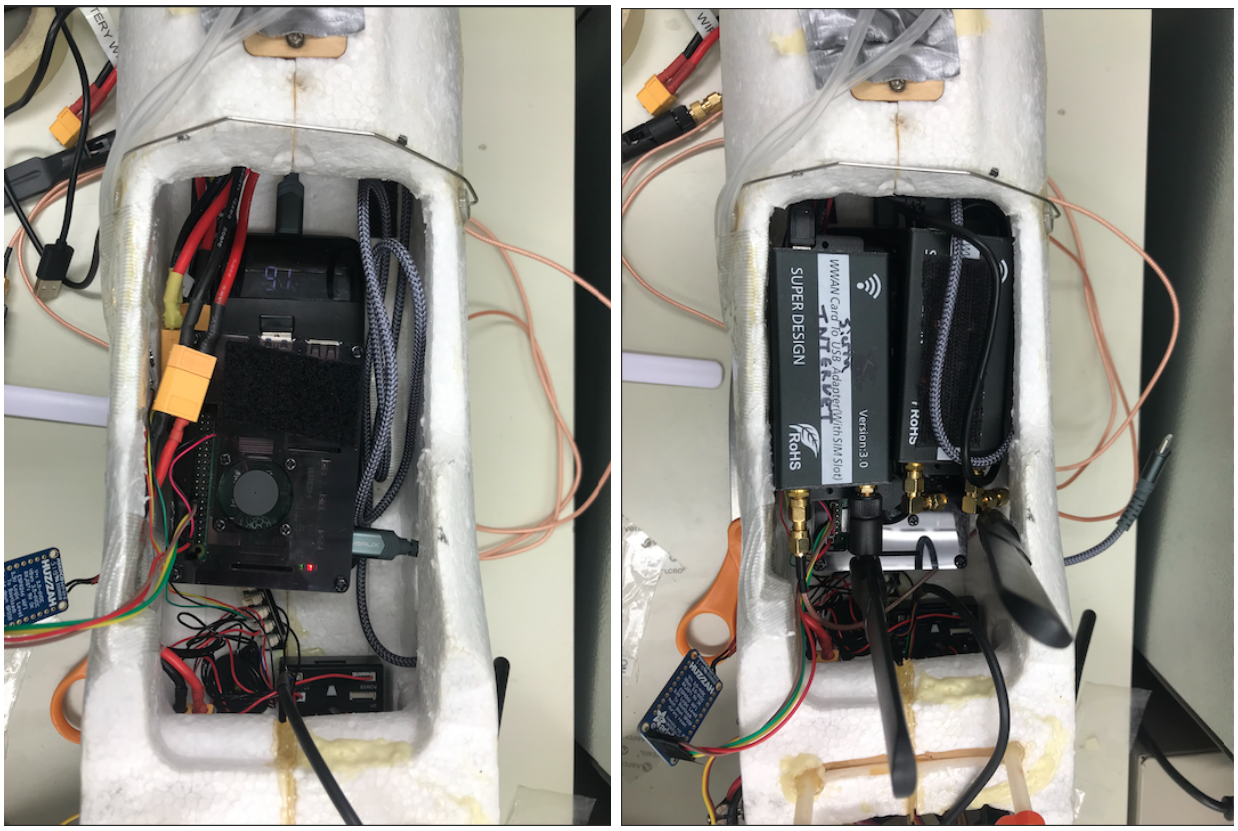

Figure 2.1: Installation of the WiCeNav payload in the aircraft. The two modems are layered on top of of the Raspberry Pi and power bank.

Figure 2.2: Top down view of closed aircraft payload bay containing WiCeNav payload. Note that the WiFi scraper remains fully inside the aircraft during flight, with only the antennas protruding, unlike this image.

2.3 Flight Testing

The work done by Davis did not include flight testing of the WiCeNav payload due to constraints related to the COVID-19 pandemic. The only testing of this solution which had previously been done was a ground test where the payload was mounted on a car and driven through an urban environment. Several flight tests were therefore conducted with the aim of simply gathering data to determine the feasibility of applying this solution to navigation in the air. Unfortunately, these tests did not produce positive results.

Two tests were conducted, and in both cases, the payload was not able to detect enough WiFi networks in the air to produce a position estimation. During pre-flight checks, as many as six WiFi networks could be detected from the ground, and position estimates were being calculated. However, in both tests, the payload stopped returning position estimates as soon as the plane took off. In fact, the system was only able to acquire one unique datapoint both times it was flight tested. Furthermore, despite the purportedly sufficient quantity of data available at ground level, the position estimates that were obtained were highly inaccurate. The estimates from the two respective flight tests are shown in figures 2.3 and 2.4, along with the flight path of the vehicle as recorded by GPS, and the error radii reported by the Google Geolocation API.

Clearly, these are not good results. The radii of uncertainty are massive, with the datapoints being located 2-3 kilometers away from the actual location of the aircraft, and the datasets are not biased in any single direction that could allow for a simple correction.

The vast majority of the problems with this system originate from its reliance on Wi-Fi data. The vast majority of the latency in the system appears to originate from components associated with the use of Wi-Fi (the Wi-Fi scraper and the Google Geolocation API call are the two worst offenders in this regard), even in an urban area. In a rural environment, the problem is compounded by the fact that Wi-Fi is simply not available in most locations. Even when Wi-Fi access points exist near the flight area, the range of most of these networks is not sufficient to sustain a connection with an aircraft flying at any significant altitude.

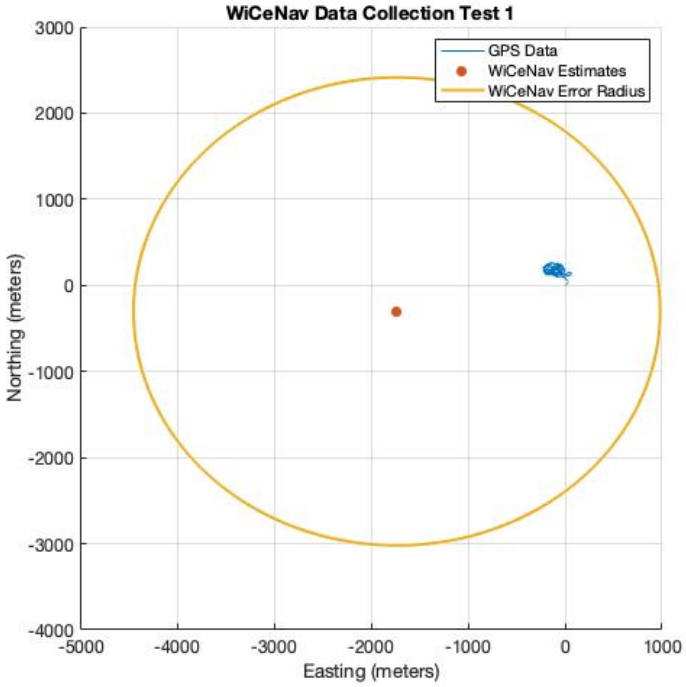

Figure 2.3: Aerial data from first WiCeNav data collection test.

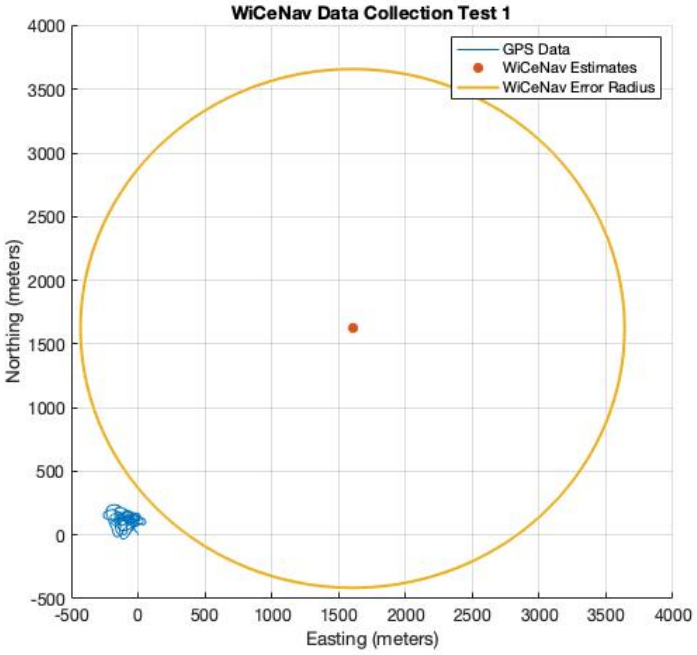

Figure 2.4: Aerial data from second WiCeNav data collection test.

Additionally, as noted above, the Google Geolocation API requires not one, but two or more Wi-Fi access point objects in order to return a position estimate [8], which is a condition that has yet to be observed after takeoff. In an attempt to mitigate these problems, the aircraft was flown at comparatively low altitude (150-200 ft), and Wi-Fi hotspots were set up in the field using the test crew's cell phones. While these measures were deemed reasonable to facilitate a simple data collection test, the system is clearly not robust enough to render autonomous navigation based on the WiCeNav solution a safe or realistic prospect. Moreover, even these workarounds did not render the system usable. The Google Geolocation API does not appear to accept Wi-Fi data associated with mobile hotspots, likely due to the fact that these hotspots obtain their Wi-Fi signal from the local cell towers, and thus represent redundant information.

It is worth noting that flight testing in an urban area was not an option due to airspace restrictions. Even if this had produced better results, it is highly inadvisable to attempt to validate an unproven navigation system in a densely-populated environment. Flight testing is an uncertain and often dangerous enterprise, and even with abundant precautions and rigorous ground testing, it would constitute a significant safety hazard to fly a plane while using this new source of position data for autonomous navigation, as was the original goal of this project, in an urban environment without performing initial flight tests in a more sparsely-populated region. Furthermore, it is wise to test new navigation systems at the highest altitude possible (400 ft AGL), so that in the event of a malfunction, there is sufficient altitude for the pilot to recover the vehicle and prevent a crash. However, increasing altitude was precisely what led to the loss of Wi-Fi signals, rendering the payload useless at high altitudes.

The inability of the WiCeNav solution to produce position estimates in the air in a rural environment due to the inherent limitations of the signals it relies on was enough to justify abandoning further development of this system, and begin pursuit of other avenues. The remainder of this thesis will discuss the alternative solution which was developed using only cell tower data.

Chapter 3
CELLULAR NETWORKS

As the vast majority of the problems with Davis' WiCeNav solution involved the system's reliance on Wi-Fi networks, the decision was made to return to a modified version of one of Davis' earlier solutions using multilateration of cell signals to obtain a position estimate. The central problem with previous attempts at a purely cell network-based solution was the availability of data concerning the exact location of the towers. The solution to this problem and other related issues with the use of cell data are discussed in this chapter, along with relevant information about how cell networks function, the type of data that can be acquired, and how it can be used to formulate a position estimate.

3.1 Cell Data Acquisition

Cellular data can be obtained using a cell modem with a SIM card inserted. The SIM card does not need to be activated unless it is being used for LTE internet access. Any SIM card will be able to simply report which cells are visible to it and the power received from each of them.

One communicates with the cell modem using AT commands, also known as the Hayes Command Set. As the name would suggest, AT commands all begin with the characters "AT", and are executed by sending the command in string format over a serial port and listening for a response. AT commands can be used to make and answer calls, send text messages, retrieve network information, and much more. Only two AT commands were used in the final code for this project. The first is simply "AT", which returns "OK" if the modem is functioning normally; this is used as a simple check for functionality for debugging purposes. The main command the code relies on is AT#MONI, the function of which was

described in detail by Davis, 2020.

AT#MONI works differently depending on the network type (2G vs. 3G vs. 4G), particularly with regards to the set command. For a 2G network, setting AT#MONI= to any number greater than 1 simply dictates the number of neighbor cells you would like to see displayed in addition to the serving cell, up to a maximum of 6. For a 4G network, on the other hand, the set command allows the user to specify what type of neighboring cells they would like to see, with AT#MONI=1 returning intrafrequency neighbor cells and AT#MONI=2 returning interfrequency neighboring cells [9]. Ideally, one would like to see all the visible neighboring cells in order to see as many towers as possible, but this could only be done by setting AT#MONI=1, calling AT#MONI, reading the response, and then setting AT#MONI=2 and repeating the process. This is inconvenient, especially when the time it often takes for the set command to be processed is taken into account. From preliminary ground testing, it was determined that more intrafrequency cells are typically visible than interfrequency cells, so AT#MONI=1 was used by itself. Although the documentation for the LE910-NAG (the 4G cell modem that was used) states that AT#MONI=7 should enable the user to obtain information related to all available LTE neighboring cells, this could not be made to function as expected.

The most important items returned by the AT#MONI call for 4G cells are the cell ID (the precise nature of which will be discussed in detail later in this chapter), the RSSI in dBm, and a number called the EARFCN, or E-UTRA Assigned Radio Channel, which is the industry standard among LTE networks for indicating frequency. Conversion tables for translating between EARFCN and frequency are widely available. [10]

3.2 Tower Locations

In order to perform multilateration, the system must know the exact location in three dimensions of the points from which the cell signals originate: in other words, the cell tower. Obtaining this data was the main obstacle to previous attempts at a multilateration-based solution, and constituted the first and most significant problem to be solved in the course of

developing this system.

A cell is defined as a region of coverage emanating from a particular emitter mounted on a cell tower. Typically, cells are visualized as neat, equally-sized, non-overlapping sectors arranged around the tower, as shown in Figure 3.1. However, cells can and usually do overlap with each other in order to provide continuous coverage as a user moves between serving cells. The neat polygons in Figure 3.1 also fail to capture the effect of terrain on cell reception, which is dramatic. In the Snoqualmie Valley, where the test site used for this project is located, cell reception near the edges of the valley is often blocked by the surrounding hills. One particularly useful resource for visualizing this effect, is CellMapper.net [11], which displays a map of approximate tower locations and cell ranges generated from crowd-sourced data through a mobile application. CellMapper's database is proprietary, and therefore could not be used directly for determining tower locations or correlating cell IDs to towers. However, the data publicly available on their website was a useful resource for verifying the conclusions drawn from other methods with regards to tower location and cell ID assignments.

Although it may seem trivial, it is critical to understand that a cell is not the same as a cell tower. A tower is a structure which may have multiple emitters mounted on it, each of which corresponds to a cell. Several databases, for example OpenCellID, give the approximate location of cells. However, this will not correspond to the location of the tower itself, and thus should not be used for multilateration. The cell location given by databases such as OpenCellID typically lies near the center of the cell, which clearly will not be at (or in many cases, anywhere remotely near) the location of the actual tower. This discrepancy is exacerbated significantly in rural areas, where the size of cells may be on the order of square miles. The results of attempting to use these measurements for navigation are covered in Davis, 2020.

Tower locations, meanwhile, can often be obtained from public records. FCC regulations require any tower taller than 200 ft to be registered, and the records of registered towers are freely available for download from an online database called the Antenna Structure Registration database [13]. One problem is that given that many cell towers are under 200

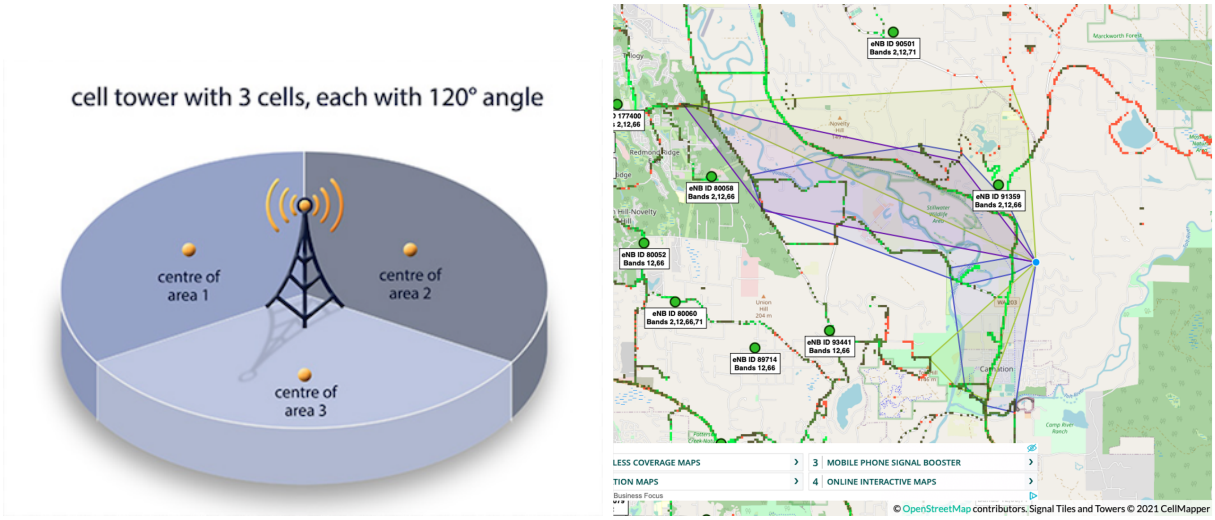

Figure 3.1: At left, an idealized visualization of the arrangement of cells around a tower [12]. At right, a more realistic depiction based on crowd-sourced data displayed on CellMapper's website [11].

ft tall, this database is not necessarily complete. Although T-Mobile appears to be very thorough in registering their towers, there are a handful in the Carnation-Duvall area that are reported by CellMapper to exist, and in some cases can be readily located by driving around the area, but could not be found in the database. In these cases, the approximate tower location was determined using Google Earth, and the height of the tower was assumed to be 100 ft, which is within the range of heights recorded in the FCC database for the registered towers in the area.

Before moving forward, it is important to discuss how cells are identified within cell networks, and the fact that how these identifiers work varies greatly between 2G, 3G, and 4G networks. In particular, there are two different identifiers for each cell in a 4G network: a Global Cell ID and a Physical Cell ID. Global Cell ID uniquely identifies a cell within a network, and does not change. The Physical Cell ID, or PCI, is very different. The PCI of a cell is not unique within the network although it is unique within a given area. It is also allowed to change, and in fact, the network is capable of re-assigning PCIs autonomously

as a kind of self-healing mechanism, in order to accommodate changes or malfunctions in the network in real time; this can happen as frequently as once per day (Sean Murphy 2020, personal communication, 28 April). Unfortunately, for 4G networks, the AT#MONI command only returns the PCI of the neighboring cells. Although other commands can be used to find the Global Cell ID of the serving cell, as of this research being conducted, there is no AT command which will return the Global Cell IDs for neighboring cells.

Thus, there exists a significant problem: in order to use 4G networks and thus have access to the most reliable and up-to-date tower location data, the system must use a type of cell identifier which can change at any moment, and which is only unique within an area of a few dozen square miles. Because this system serves as a proof of concept to answer the question of whether cell networks provide a viable means of navigation for sUAS, the system was designed to use PCIs. However, as a consequence, it should never be deployed outside of the area it was designed for, and operators should always take steps prior to the flight to check whether the PCIs in the local database are up to date. Luckily, throughout the course of flight testing this project, PCI assignments in the region surrounding the test area were not observed to change at all. It must be noted that were one of them to change during the flight, the consequences could be disastrous. If improvements are to be made to this system anywhere, the first priority must be modifying the code and tower database to use Global Cell IDs. Since this is purely an issue of what collection of digits are used to refer to a given signal source, this change would not impact the functionality of the system, only make it safer, more reliable, and possible to use over a much broader area.

No matter which identifier is used, there is still the issue of correlating this with a specific tower. The most straightforward method of correlating cells and towers is to simply collect cell data from nearby each of the towers that are in range of the flight area, then use signal strength to manually correlate cells with towers. This method, although inconvenient and impractical for most applications, proved to be the only feasible way of obtaining the necessary data for the purposes of this project. This method has many disadvantages, the most obvious being that it renders any tests which use this database not easily replicated

at other sites. One would have to repeat this procedure to manually create a different local tower database for every site at which one wishes to fly. The process is also tedious and leaves open the possibility for significant amounts of human error. It is also possible to simply miss a cell, thus resulting in an incomplete database without realizing it. This last concern was partially accounted for by conducting numerous flight tests simply to collect cell data from the field itself and ensure that every cell the aircraft can connect to from its flight path is correlated with a tower whose location is known. But again, this process takes time and additional flight testing, and largely defeats the purpose of having a system which can navigate through an unknown area. However, for a proof of concept of the overall system, this was deemed to be sufficient. The function within the code which retrieves cell tower locations based on PCIs is clearly marked, well commented, and structured in such a way as to make it easy to replace with a more sophisticated method should more complete information become available to future researchers. It should be clearly understood that the current method used for cell/tower correlation is neither optimal nor in any way desirable, and represents one of the key areas for improvement in this project.

The manually constructed local tower database is stored in a CSV file, which can be switched out easily for a file containing data from another region. Sample data from the CSV file is given in Table 3.1. Each tower in the CSV file is given a nickname related to its approximate address in order to make it easier to spot problems without having to memorize which towers are where.

Physical Cell ID	Latitude	Longitude	Ground Altitude (ft MSL)	Tower Height (ft AGL)	Tower Nickname
80	47.67°	-121.901°	148	195.9	328th Ave
288	47.6554°	-121.9647°	212	149.9	Ames Lake Rd

Table 3.1: A sample of the custom database of tower data in the local area of Carnation Farms.

3.3 Summary of Issues

Clearly, there are numerous problems with the methods described in this chapter. It would be beneficial, before moving forward, to summarize these briefly, and discuss why the shortcuts and approximations described here were deemed acceptable in this context.

The central issues are:

1. The use of Physical Cell IDs rather than Global Cell IDs

2. The presence of unregistered towers whose locations and heights must be approximated

3. The use of a local tower database which had to be manually created and is only valid within a certain region

These issues, in particular the third, can be tolerated largely because this system is simply a proof of concept. This phase of the project did not set out to create an optimized version of this system, but simply to explore the matter of how to make multilateration using cell networks feasible without obtaining proprietary data from a network provider, and to answer the question of whether designing a navigation system of this type using publicly available data is possible. In many ways, it would appear that the answer to this question is no. As long as Global Cell IDs for neighboring cells cannot be obtained, and no complete database of tower locations and their associated Cell IDs can be obtained, the system is doomed to suffer from a certain degree of unreliability. However, these are problems that could be solved by the simple use of data from a different source. With even a rudimentary framework such as this one in place, it is possible to begin investigating the accuracy of position estimation using this system.

Chapter 4
MATHEMATICAL THEORY

This chapter describes the mathematics at work behind the CellNav algorithm.

4.1 The Friis Transmission Equation

The Friis Transmission Equation relates the power of a signal to the distance between the transmitter and the receiver. Davis, 2020 used a modified form which simplifies the equation by combining several of the constants. It is given by Equation 4.1, where D is the distance to the tower, $P_r(D)$ is the power received, P_{r_1} is the power received at a distance of 1 meter away from the transmitter, and K is a constant that combines the values of several other constants in more complex forms of the equation. Power here is measured in decibels.

$$P_r(D) = P_{r_1} - Klog(D) \tag{4.1}$$

The difficulty with using this form of the equation is that the constants K and P_{r_1} must be calculated ahead of time using a known range to a given tower in order to use this equation for range estimation. Davis, 2020 determined that it was not acceptable to use the same values of these constants for every tower. Furthermore, it was determined that values of K and P_{r_1} which provide accurate range estimates for certain data points may not work for others.

After seeing limited success using the modified Friis Equation, the system was modified to instead use something closer to the original form of the Equation, given by Equation 4.2 [14]. Many different forms of this equation exist, all of which appear to be equivalent to one another. Unfortunately, they all require some information about the transmitting antenna which is not straightforward to acquire. This form was selected largely due to its simplicity.

Note that in this form of the equation, power is measured in watts.

$$\frac{P_r(D)}{P_t} = \frac{A_r A_t}{D^2 \lambda} \tag{4.2}$$

Solving this for distance results in Equation 4.3.

$$D = \sqrt{\frac{P_t A_r A_t}{P_r(D)\lambda}} \tag{4.3}$$

This form of the equation has more constants, but they are more easily determined. λ is the wavelength of the signal, which is easily calculated from frequency using Equation 4.4, where c is the speed of light and f is the frequency. Frequency is a known quantity whose value is included in the data returned by the AT#MONI command. P_t is the power transmitted by the emitter, which was assumed to be 10 W [15]. A_r and A_t are the effective aperture area of the receiving and transmitting antennas respectively. Aperture area is related to the gain of the antenna and the wavelength of the signal. This relationship is given by Equation 4.5.

$$\lambda = \frac{c}{f} \tag{4.4}$$

$$G = \frac{4\pi A}{\lambda^2} \tag{4.5}$$

Using Equation 4.5, the effective aperture area of the antenna on the payload was determined to be approximately 0.06 m^2. Not enough information was available to determine the effective aperture area of the transmitting antennas mounted on the cell towers, and a trial and error process was used to attempt to determine the value of this constant. This process and the effects of the changes that were made to the value of A_t are discussed further in Chapter 8.

4.2 Multilateration

The CellNav system generates position estimates using multilateration. Several mathematical steps are necessary to convert the data acquired by the cell modem into a position estimate. These algorithms are described in this section.

4.2.1 Range Equations

The principle behind multilateration is that the exact position of the drone can be calculated using the known distance between the drone and at least three cell towers whose locations are known. For each tower, one obtains an equation of the form given by Equation 4.6, where $[x, y, z]$ is the unknown position of the drone, D_i is the distance between the drone and the i^{th} tower, and $[x_{t_i}, y_{t_i}, z_{t_i}]$ is the known position of the i^{th} tower.

$$D_i^2 = (x_{t_i} - x)^2 + (y_{t_i} - y)^2 + (z_{t_i} - z)^2 \tag{4.6}$$

A method for eliminating the nonlinear terms from the resulting system of equations using either an artificially generated pseudo-range or a 4th tower measurement was discussed by Davis, 2020. However, significantly better results were obtained here by solving the full nonlinear system of equations using a nonlinear least squares algorithm. The nonlinear least squares method was discussed in great detail by Davis, 2020. Here, the operation was performed using the scipy package's built-in nonlinear least squares solver (scipy.optimize.least_squares), which by default uses a Trust Region Reflective algorithm to converge on a solution. In order to use it, the system of equations must be converted into the form of a residual, resulting in Equation 4.7.

$$(x_{t_i} - x)^2 + (y_{t_i} - y)^2 + (z_{t_i} - z)^2 - D_i^2 = 0 \tag{4.7}$$

The least squares function also requires the form of the Jacobian as an input, each row of which is given by 4.8

$$\begin{bmatrix} 2(x_{t_i} - x) & 2(y_{t_i} - y) & 2(z_{t_i} - z) \end{bmatrix} \tag{4.8}$$

Note that before any calculations were done, tower locations were projected into a flat-earth, North-East-Down (NED) coordinate system. This process was done automatically using the navpy python package, which also converted the final result back into latitude and longitude.

Chapter 5

HARDWARE

In this chapter the hardware components that make up the payload are described, as well as the aircraft used for this project, and the installation of the payload on the aircraft.

5.1 Payload

The components of the CellNav payload are described in Table 5.1. This payload is far simpler than the WiCeNav payload due to the elimination of components necessary to gather Wi-Fi data and provide the payload with an internet connection. Both of these functions are unnecessary to the operation of the final system, which requires only a cell modem for data collection. The fully-assembled payload is shown in Figure 5.1. It weighs 0.56 kg (1.26 lbs), is relatively small at 14.6 cm × 8.9 cm × 7 cm (5.75”×3.5”×2.75”), and is easily installed on a number of fixed wing and multirotor configurations. All parts are off-the-shelf and relatively inexpensive and easily accessible.

The other significant change which should be noted is the type of cell modem used to collect cell data. The WiCeNav payload used a Telit HE910-D modem for this purpose, which is compatible with 2G and 3G networks. Due to the difficulty of correlating 2G and

Component	Purpose
Raspberry Pi 4b	Computing
Telit LE910-NAG Cell Modem	Obtaining local cell signals
Charmast USB-C Power Bank	Power

Table 5.1: Components of the final CellNav payload.

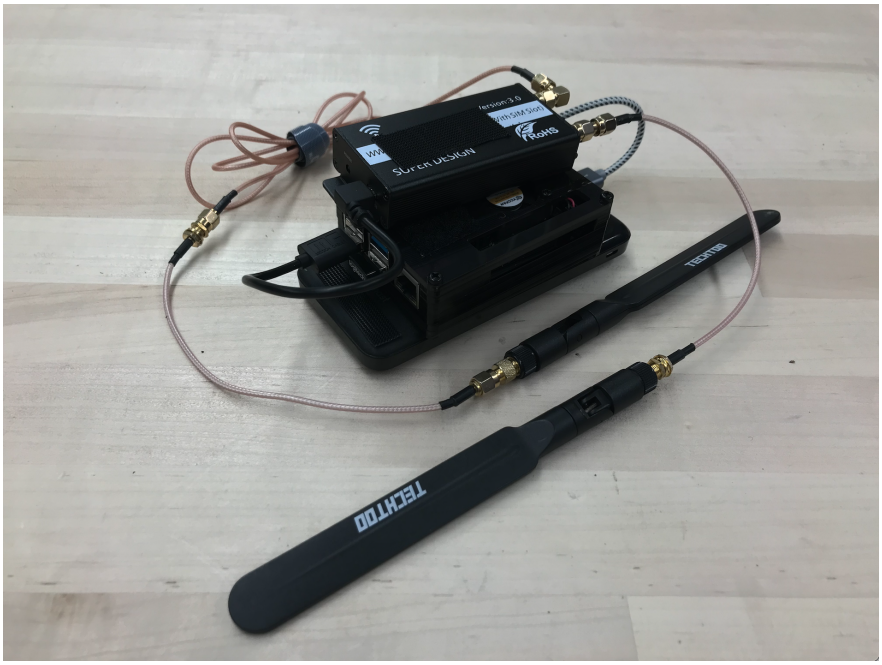

Figure 5.1: The redesigned CellNav payload.

3G signals with tower locations in the Carnation area, and the relative ease of doing so using 4G signals, an upgrade was made to the 4G-compatible Telit LE910-NAG modem. The Sierra MC7455 modem previously used to provide an internet connection to the Raspberry Pi via LTE is also compatible with 4G, and attempts were made to use this modem to gather cell data. However, the AT#MONI command used to gather the necessary data is not supported by the Sierra MC7455, and no equivalent command exists for that modem which would provide not only data on the serving cell, but the neighbor cells as well. The Telit LE910-NAG is compatible with the same command set as the Telit HE910-D, with the simple difference of being able to gather data on 4G networks. The modem used a SIM card provided by T-Mobile which was subscribed to the network, although the payload does function when fitted with a SIM card not tied to a network subscription. It could also function using a SIM card from a different network, if the database of tower locations and cell IDs was correspondingly updated; nothing about the hardware used on this payload is

Figure 5.2: The Finwing Sabre aircraft used for this project, nicknamed Peach.

specific to any one network.

5.2 Aircraft

Two aircraft were used for this project: one fixed wing and one multirotor. The original goal of the project was to integrate the payload with the autopilot of the fixed wing, but for data collection purposes, the multirotor had the advantage of being able to hover and collect data from more precise locations within the test area. The bulk of the preliminary testing of the system was done on the fixed wing aircraft. However the final three flights, which were successful data collection tests using the final form of the system, used the multirotor aircraft due to technical difficulties with the fixed wing.

5.2.1 Fixed Wing

The fixed wing test platform consisted of a Finwing Sabre airframe equipped with a Pixhawk Orange Cube autopilot. This aircraft was also used by Davis, 2020. It has a wingspan of 1.9 m (6.2 ft) and a length from tip to tail of 1.4 m (4.5 ft). With the payload, ballast weight, and battery installed, it weighs 3.45 kg (7.6 lbs). In addition to the previous phase of this project, this aircraft has been used extensively in other GPS-denied navigation projects within AFSL. The configuration easily accommodates many different types of payloads, and the foam structure of the airframe renders it easy to modify if necessary. This aircraft is shown in Figure 5.2. Note the installation of the two antennas, which are secured to either side of the forward section of the fuselage. During ground testing or when flying on a multirotor, antennas were often secured directly to the modem without extension wires. However, the modem demonstrated a consistently poor connection to local towers when the antennas were even partially inside the plane.

The Pixhawk Orange Cube autopilot is equipped with a USB interface which can be used to communicate with the Raspberry Pi, even if the aircraft is also wirelessly connected to a ground control station. While the Pixhawk does save extensive data logs onboard, this USB interface was exploited to separately log the drone's GPS position on every loop of the code, in order to make correlating the plane's true location and the CellNav estimate easier. This connection is also used to obtain a single GPS measurement from which to initialize the CellNav system. This interface is described in more detail in Chapter 6.

5.2.2 Multirotor

A DJI Matrice 600 Pro was used as the multirotor platform for this project. This is a highly robust and reliable aircraft which has been used for a number of other projects within AFSL, most notably for the development of a machine-learning-driven search and rescue platform. It is a hexacopter, and measures 1.6 m (5.25 ft) in diameter with the propellers unfolded. It has a maximum takeoff weight of 15.1 kg (33.3 lbs), although the weight of the

Figure 5.3: The DJI Matrice 600 Pro, nicknamed Aragog, with the CellNav payload (close-up at right) installed.

aircraft plus the CellNav payload is significantly less. The M600 is shown with the CellNav payload installed in Figure 5.3.

The difficulty with the use of the M600 is that it is not equipped with a Pixhawk, but instead uses DJI's own autopilot system. Since CellNav requires a single position measurement from GPS to initialize the system, and is designed to obtain this from the Pixhawk autopilot, this aircraft is only suitable for data collection, and a separate Pixhawk and GPS must be connected to the payload when it is installed on the M600. This is not used for command and control, but simply to provide the initial position estimate, as well as truth data for later analysis that is correlated with the timestamps on the CellNav data. It is worth noting that if collecting truth data on the payload itself is not desired, and because the payload does not currently send any of its position estimates back to the aircraft for use in navigation, it is entirely acceptable to disconnect the Pixhawk after the initial position estimate has been acquired. It remains connected merely for convenience.

Chapter 6

SOFTWARE

This chapter details the development and architecture of the software that defines the CellNav system. The code was written in Python and executed on the Raspberry Pi. It requires a number of Python packages in order to run: pyserial, numpy, scipy, navpy, csv, time, and dronekit. There are also several important restrictions on what *cannot* be running: the Modem Manager daemon must not be installed on the Pi, and the ADS-B receiver built into the Pixhawk Orange Cube must be disabled. Both of these restrictions will be discussed further in this chapter.

6.1 Final Software Architecture

The block diagram of the overall software architecture is shown in Figure 6.1. However, there are a number of initialization steps not shown in the diagram. These are, in order:

1. The payload establishes a serial connection to the Pixhawk and obtains a single position measurement from the aircraft GPS. This represents the "last known position" of the aircraft.

2. Another serial connection with the cell modem is established, and the payload issues the command "AT" to verify that the modem is functioning normally, then "AT#MONI=1" to set the cell modem to the appropriate data collection mode (for more detail, see Chapter 3).

3. If both of the above AT commands return "OK", the payload issues a message stating that the system has been initialized properly, and waits for an input to confirm that the user is ready to start running the main loop.

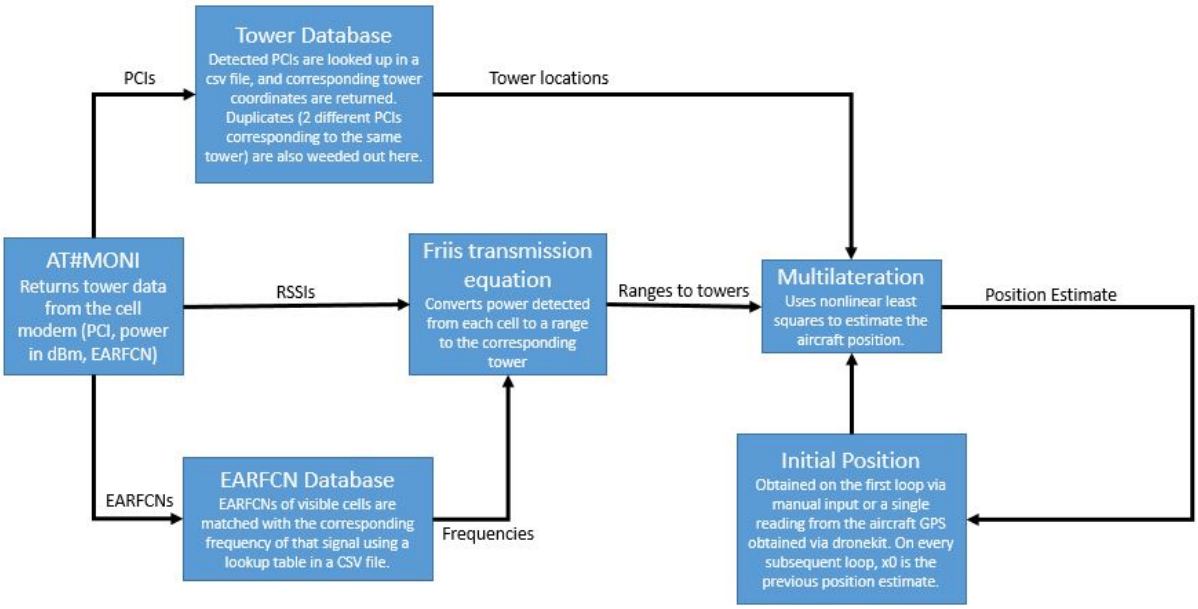

Figure 6.1: Block diagram showing the redesigned CellNav software architecture.

The main loop of the code has four primary stages. First, the AT#MONI command is sent to the cell modem, and the response is parsed into arrays of cell IDs, EARFCNs, and RSSIs. The cell IDs and EARFCNs each then pass through a function that looks each value up in a CSV file and returns the corresponding tower locations and frequencies respectively. As part of this process, the towers are checked for uniqueness, and in the event that more than one cell associated with the same tower has been detected, the second one (which will always have equal or lesser signal strength than the first) is ignored after this stage. The RSSI values are passed through the Friis Transmission Equation (eqn. 4.2), which returns the approximate distance to each tower. This equation requires the wavelength of each signal, derived from the frequencies retrieved from the EARFCN lookup table in the previous step. The locations of the towers and the distances to them are then fed into the multilateration algorithm, which returns an approximate position. On the first loop of the

code, the nonlinear least squares algorithm uses the last known GPS position as its initial guess; on every subsequent loop, the previous estimate is used. It could be argued that this compounds the error of the calculation, however, no significant change in performance was observed when a different initial guess was used. Other sources of error appear to have a much more significant impact on the solution, as we will see.

On each loop of the code, a number of quantities are logged in separate, timestamped CSV files. This is one of the most significant improvements to the system when compared with WiCeNav, which logged all data to the same CSV file, and logged an initial timestamp and the tick time for each loop rather than simply giving a timestamp for each datapoint. This data required significantly more post-processing than the data obtained with the new CellNav system. Currently, the logged quantities are:

- The raw output of the AT#MONI command
- A more readable form of the AT#MONI data, showing only PCI, EARFCN, and RSSI
- The coordinates and PCIs associated with each *unique* tower that was detected
- The position estimate produced by CellNav
- The actual GPS position of the aircraft

The logger does not appear to have a significant impact on latency, meaning that as many quantities as desired can be logged for debugging purposes without any measurable degradation in performance. It should also be noted that the GPS position of the aircraft can also be determined from the Pixhawk's internally recorded Data Flash Logs, or from the Telemetry Logs generated by the ground station. Recording them separately here is redundant, but makes post-processing easier, since the timestamps will be correlated exactly with the position estimates.

6.1.1 Pixhawk Interface

Communication with the Pixhawk was easily achieved using the dronekit python package. Dronekit allows the Raspberry Pi to sustain a connection with the aircraft even if the Pixhawk is also connected to a separate ground station via telemetry radio. Data from the Pixhawk's many sensors can be easily extracted in real time. However, there are a number of idiosyncrasies associated with the use of dronekit. Firstly, if the Pixhawk is equipped with an ADS-B receiver (like the Orange Cube is), this must be disabled in order for the Pixhawk's spare serial port to be available for communication with the Raspberry Pi. Second, some users have reported that dronekit occasionally conflicts with the Modem Manager daemon [16], which was required to run the WiCeNav solution. Modem Manager had to be uninstalled for other reasons, which are detailed in the following section.

6.1.2 Modem Manager

Mention has already been made several times of the Modem Manager daemon and the various problems associated with it. These difficulties were so pernicious that they merit their own section.

Modem Manager is a daemon for Ubuntu that regulates connections with mobile broadband devices [17]. It was used in the WiCeNav solution to enable the Raspberry Pi to access the internet using the LTE connection provided by the Sierra modem. As noted in the previous section, it has been known by some users to cause difficulties with using the dronekit package. However, the central difficulty with Modem Manager encountered in this project surrounded communications with the cell modem. When a 4G modem such as the Telit LE910-NAG is connected to the Raspberry Pi, Modem Manager, if installed and configured properly, will aggressively attempt to use the modem to access the internet. This was observed to occur even if Modem Manager has been disabled, or the corresponding process manually terminated. In fact, Modem Manager will often reactivate itself several minutes after it has been disabled. If the serial port corresponding to the cell modem is in use at that

point by the CellNav code, Modem Manager will shut down that connection in order to take over the port for itself, causing data collection to be interrupted. While it is possible that there is a way to prevent this from happening that was simply not discovered in the course of this research, the most efficient and practical solution is simply to uninstall Modem Manager from the Raspberry Pi altogether. It is not necessary for the operation of this iteration of the system.

Chapter 7

FLIGHT TESTING

Flight testing was conducted at the UW Carnation UAS Test Site (UW-CUTS), located at Carnation Farms in Carnation, WA. The flying field has an area of approximately 0.18 square miles, or 0.46 square km, and consists of flat, grassy terrain. The airspace over the site is Class G up to 700 ft AGL, above which it is Class E. The aircraft was flown entirely below 400 ft. For fixed wing flights, the Mission Planner ground control station was used for real-time telemetry and flight planning. For the multirotor flights, telemetry is received directly by the pilot via the DJI Go app.

A basic flight pattern was established which was used for all autonomous flights, as well as for data collection. This pattern is shown in Figure 7.1. The flight path was designed to demonstrate a variety of both left- and right-handed turns of varying radii, and to pass through regions of varying cellular and Wi-Fi coverage. For some tests, however, the aircraft was flown manually. This was done for a number of reasons, primarily safety concerns due to strong, unpredictable, high-altitude wind gusts or aircraft malfunctions that would have rendered Auto mode unreliable, but did not prevent flight in a manually controlled mode. For these flights, Stabilize mode was used, which allows manual control of the plane, but attempts to keep the aircraft level, and assists the pilot in coordinating turns by mixing aileron and rudder inputs. Takeoffs and landings were always performed in Stabilize mode, allowing the pilot to verify the basic stability and functionality of the aircraft before switching into any autonomous modes.

There are six 4G cell towers associated with T-Mobile located close enough to Carnation Farms to be consistently detected within the flight test area. Reception varies greatly between the ground and the air, as might be expected, with more towers coming into view as the

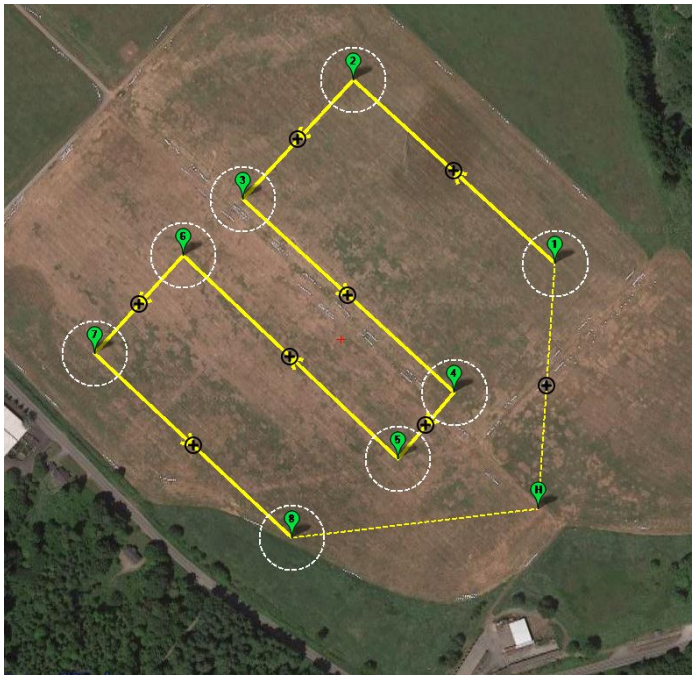

Figure 7.1: Waypoints used for all autonomous flights.

aircraft climbs higher than trees and other terrain obstacles in the area. However, there are only a few regions of the field where three or more towers were consistently detected. The local towers are shown in Figure 7.2, along with the Physical Cell IDs that were associated with those towers during testing. These assignments were inferred, as described in Chapter 3, by observation, and were corroborated by data available on CellMapper's website [11].

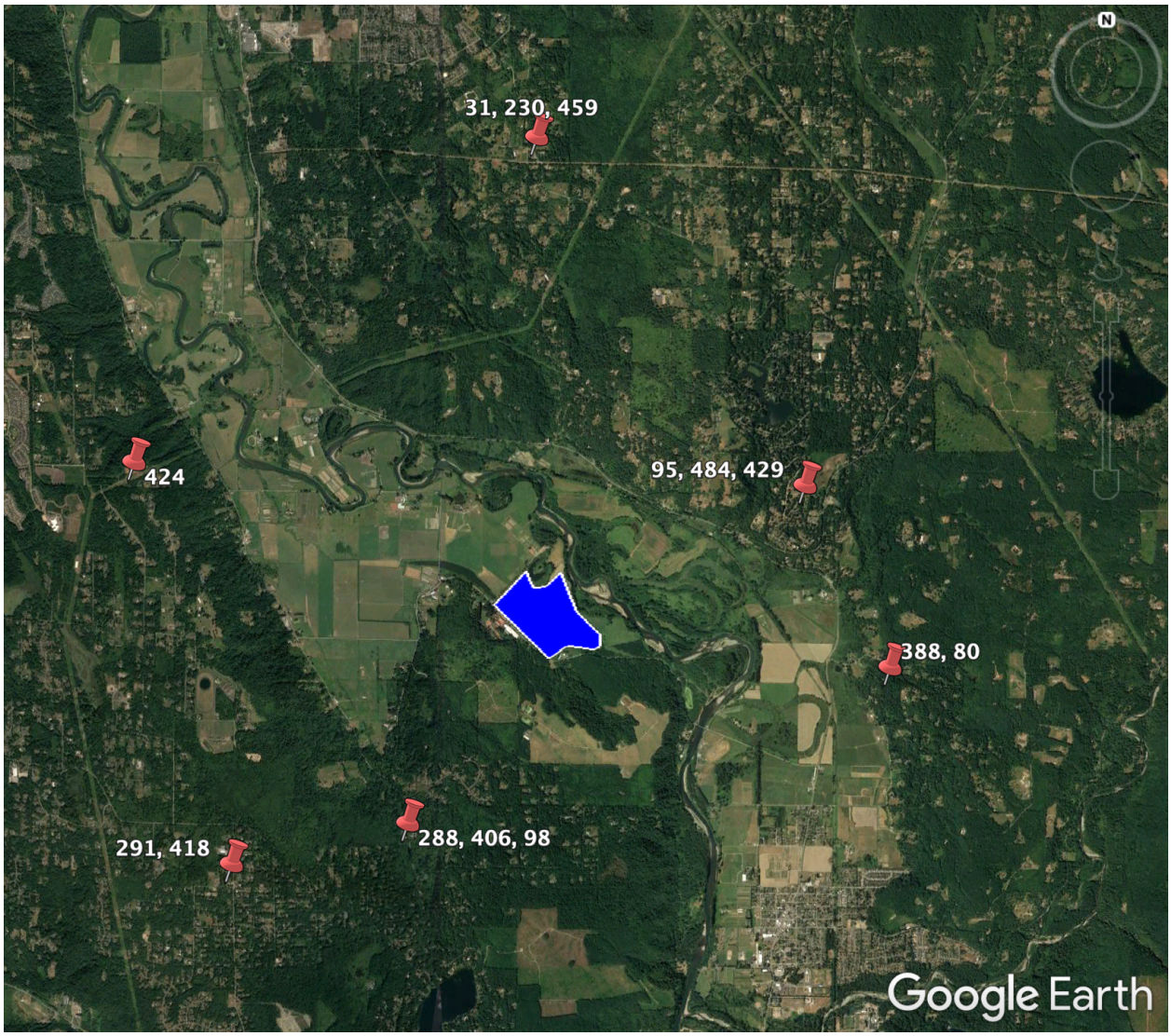

Figure 7.2: Cell Towers in the region surrounding Carnation Farms, and their associated Physical Cell IDs as of May, 2021. The flight test field is highlighted in blue.

Chapter 8

RESULTS

Three data collection runs were completed with the redesigned CellNav payload, all of which were flown using the DJI M600 due to technical difficulties with the Finwing Sabre. The results of these tests display a very poor degree of accuracy and shed light on several aspects of the flight test environment that render this system essentially unusable.

The same code was used on all three flights, except for one small change in the third flight. Due to the poor quality of the position estimates, for the third flight the value of effective aperture area of the transmitting antenna was increased (this is discussed in more detail later in this chapter). Overall performance was not drastically improved, however a small portion of the flight saw slightly better estimates than had been observed on previous flights.

Figures 8.1-8.3 show the flight path of the aircraft recorded by GPS compared to the position estimates gathered by the CellNav payload. The position estimates are color coded according to the number of towers available. The accuracy is generally very poor, which is to be expected to a certain degree, given that only two towers were visible for the majority of the flight path. However, even during portions of the flight where measurements from three unique towers were available, the error remains unacceptably high. Figures 8.4-8.6 show the error between the CellNav estimates and the GPS data for these three flights. The minimum error, which occurred when the aircraft was in range of 3 towers, was 549.5 m (1803 ft).

Note that in Figure 8.2, there is a point where the error appears to have been extremely low. This is due to the fact that when the system initialized on this particular run, only one tower was visible. The system was therefore unable to generate a position estimate and instead returned the last known position of the aircraft, which was the GPS position at which

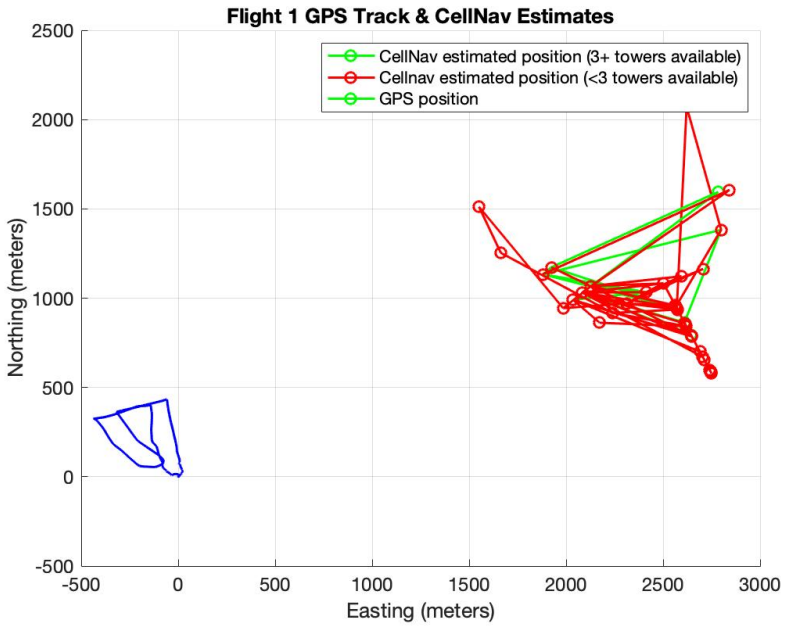

Figure 8.1: Flight path with CellNav estimates, Flight 1.

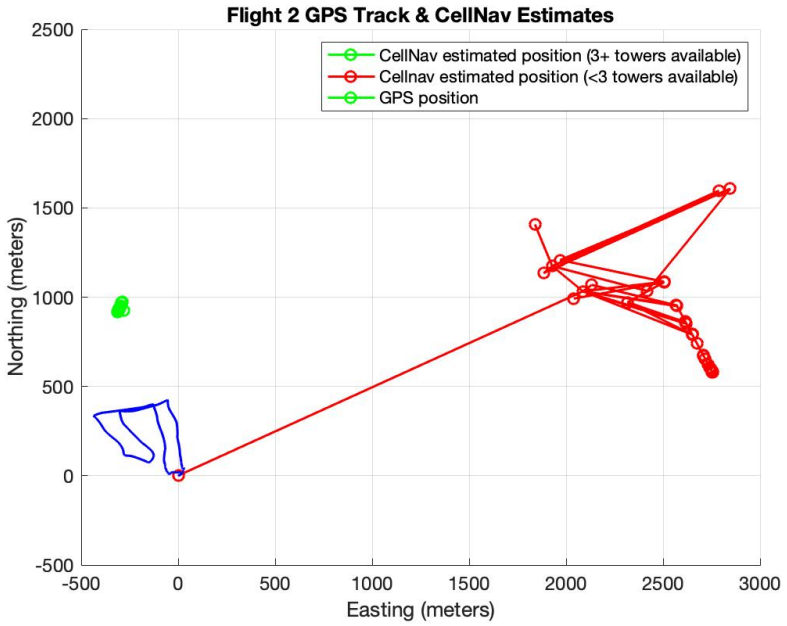

Figure 8.2: Flight path with CellNav estimates, Flight 2.

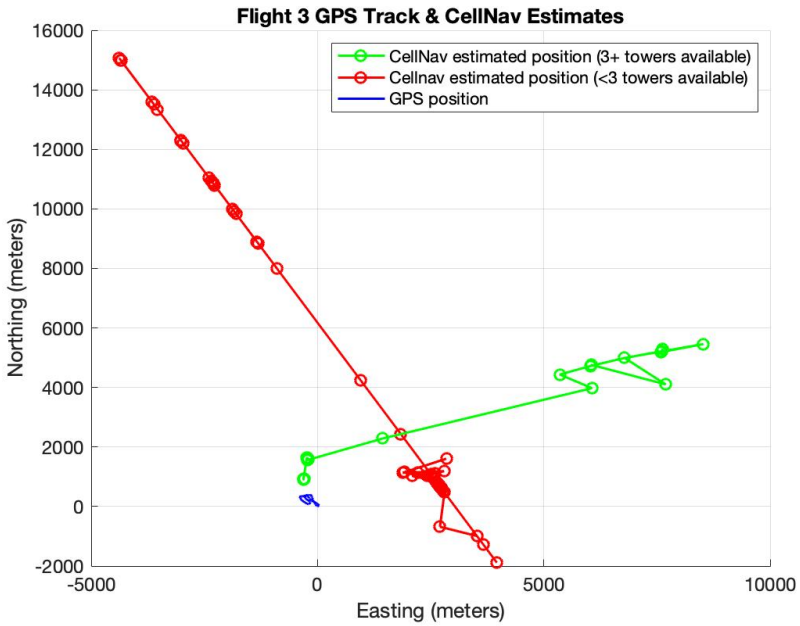

Figure 8.3: Flight path with CellNav estimates, Flight 3.

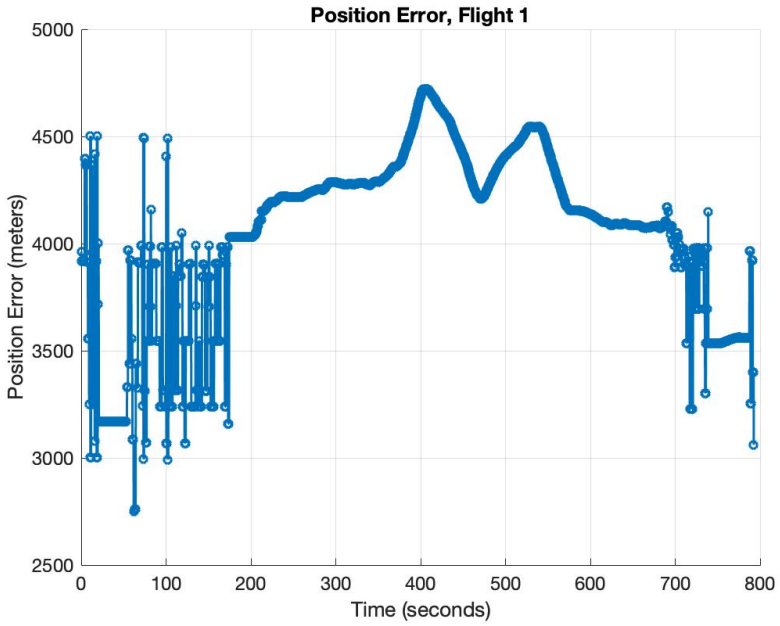

Figure 8.4: Position estimate error, Flight 1.

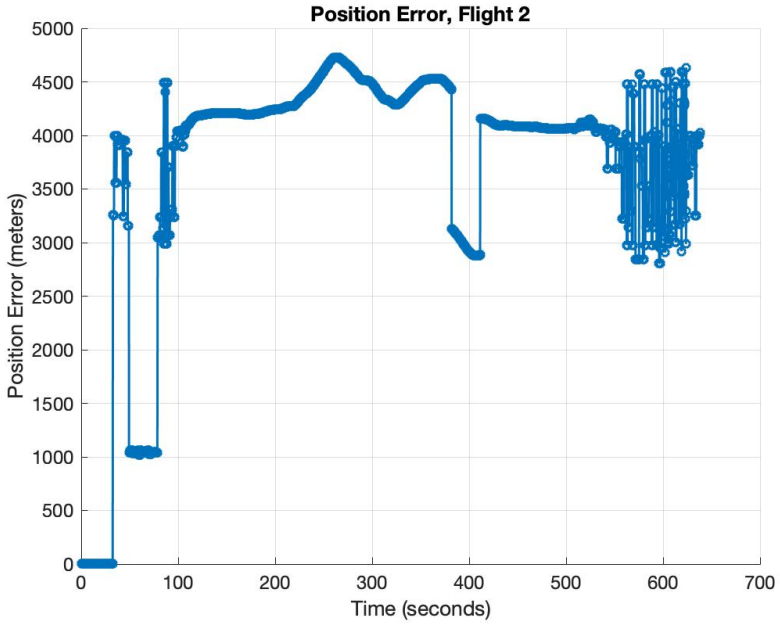

Figure 8.5: Position estimate error, Flight 2.

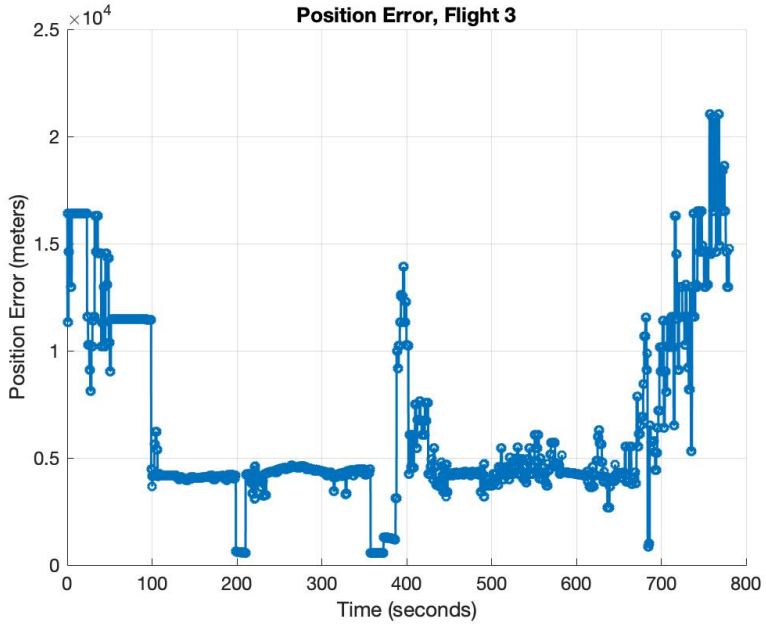

Figure 8.6: Position estimate error, Flight 3.

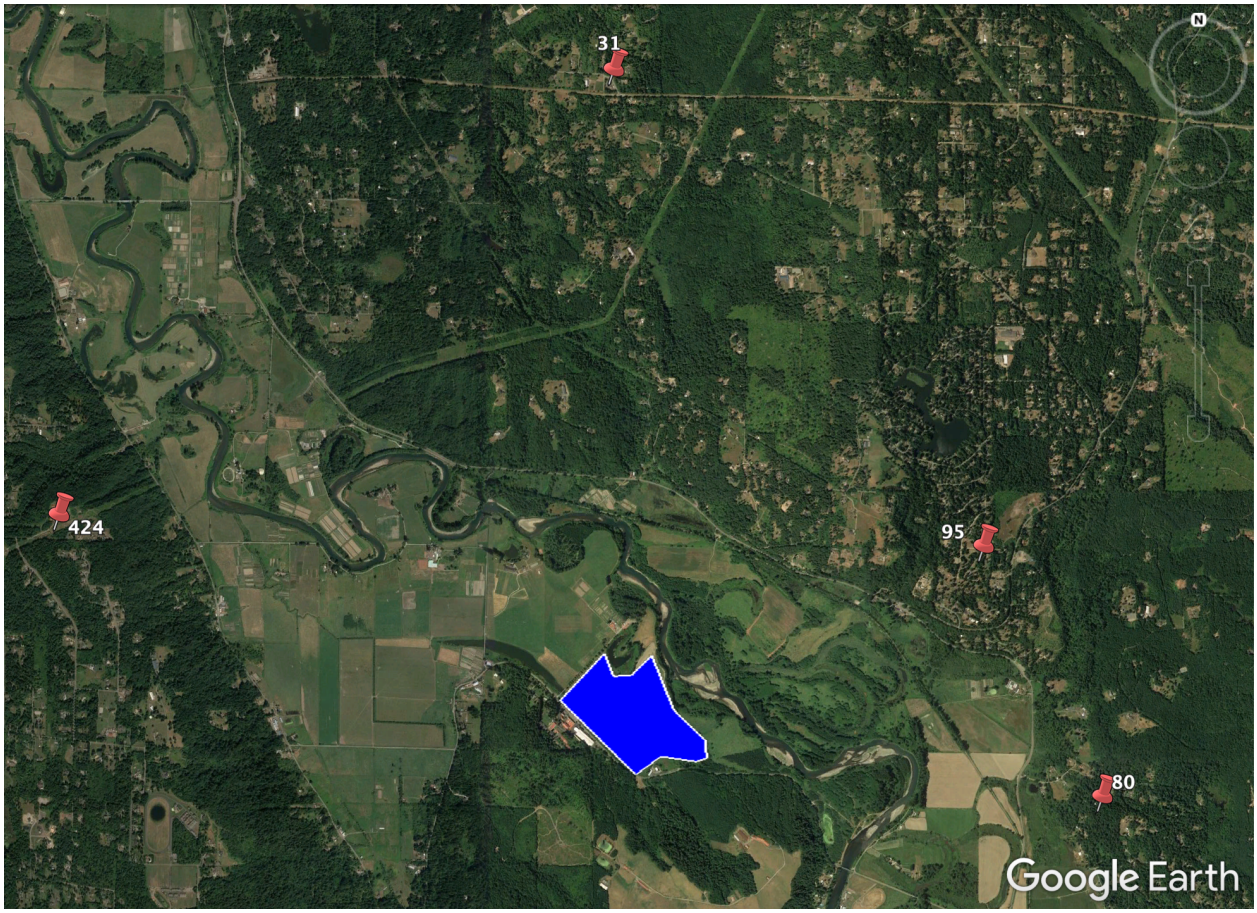

Figure 8.7: Towers detected during the 3 data collection flights using the finalized system design. UW-CUTS is highlighted in blue.

the system was initialized. Since this occurred during the pre-flight phase when the aircraft was not moving, the system appears to be performing well, when in fact it is suffering from a complete deficit of data.

8.1 Sources of Error

The significant degree of error present in the data likely traces back to a number of sources, namely uncertainty in the calculation of range from RSSI, signal attenuation due to terrain features, and uncertainty associated with the specific towers that were detected.

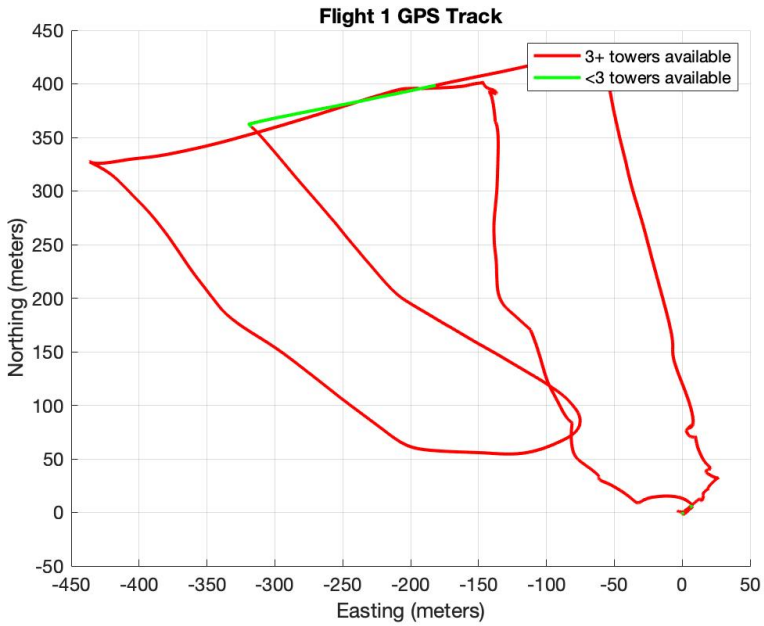

Figure 8.8: Regions of coverage, Flight 1.

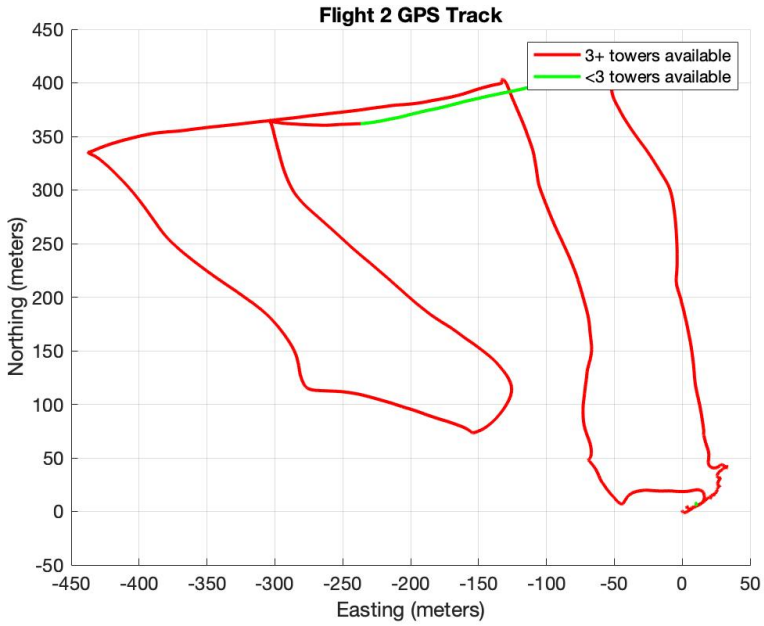

Figure 8.9: Regions of coverage, Flight 2.

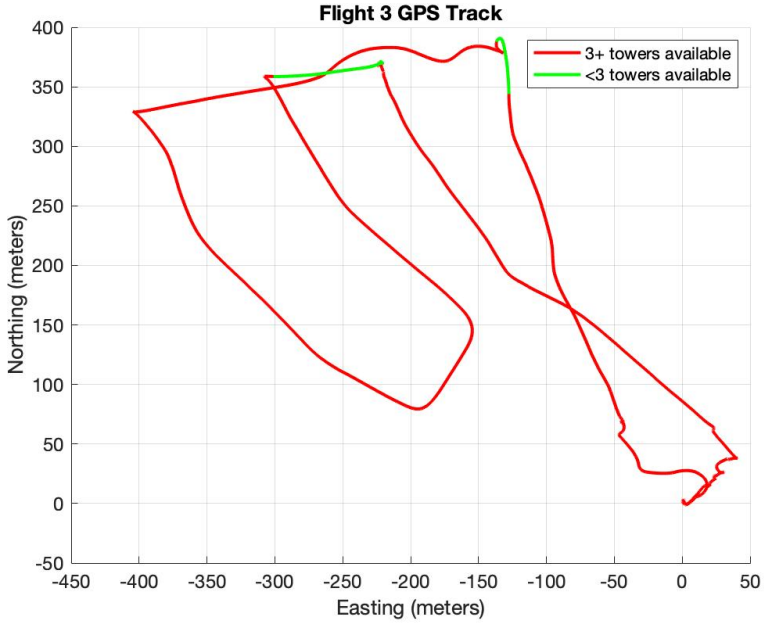

Figure 8.10: Regions of coverage, Flight 3.

Figures 8.8-8.10 show the GPS track of the aircraft, color coded according to the availability of tower data. These figures show that three or more towers were only available in the region near the northernmost corner of the field. Figure 8.7 shows the specific towers the payload detected while in this region. IDs 95 and 80 were detected very consistently throughout the field, with one of these two acting as the serving cell in all portions of the field where the drone had reception. At the northern end of the field, IDs 31 and 424 come into view. It is deeply unfortunate that these four towers specifically are the ones it is possible to detect at this site. The locations of the towers associated with IDs 95 and 424 are uncertain, as these two towers could not be found in the FCC Antenna Registration database. Their existence is supported by data publicly available on CellMapper's website, and their approximate location was determined using Google Earth. The exact height of these towers is not known. The location of the tower associated with ID 31 is known, but not the height of the tower, which was also approximated. A hill whose peak is approximately 350 ft higher than the

ground level at the flight test field separates the field from two towers to the southwest with confirmed locations and heights, and although the drone can legally ascend to 400 ft, the hill is still much taller than either of the towers it is blocking. This is likely what is preventing their signal from reaching the drone. On the other hand, there is a relatively clear line of sight to the towers corresponding to IDs 424 and 31 from the northern end of the field. Even so, these signals (especially ID 31) are likely subject to a significant degree of attenuation due to the densely forested areas they must pass through. Had one of the towers to the southwest of the field been visible, the estimate may have improved.

Why are so few towers visible? The most likely theory is that this is by design. Cell networks are not designed with navigation systems such as this one in mind. They are designed to provide coverage for telecommunications devices which only require a connection to one tower at a time in order to function. In order to facilitate smooth transitions between the regions of coverage of different towers without loss of signal, it is prudent to design the network such that more than one tower is visible at a time. But the construction and maintenance of cell towers is costly, and network providers likely would prefer to put up as few towers as they can while still providing adequate coverage to the regions they wish to service. A phone needs to see one tower in order to function, and it is wise to design the network in such a way that it can see two. But a cell phone does not need to see three towers at once in order to function well and not lose coverage. It makes sense that the design of a cell network would be optimized in a way that takes this into account.

8.1.1 Effective Aperture Area

Recall that one of the required quantities in the Friis Transmission Equation (eqn. 4.2) is the effective aperture area of both the transmitting and receiving antennas. It has already been mentioned that the effective aperture area of the transmitting antenna (that is, the antenna mounted on the cell tower), could not be determined from readily available data, and had to be determined by a process of trial and error. It was found that the range calculations improved if this number was much smaller than it likely is in actuality. Most

runs used a value of 0.0003 for the aperture area of the transmitting antenna, which clearly does not make sense, as it is far smaller than the value for the receiving antenna. In actuality, the transmitting antenna is much larger than the receiving antenna. However, larger values resulted in the tower range estimates being inaccurate by a factor of more than 10 miles. On the final data run, A_t was increased to 0.03, and although the position estimates when fewer than 3 towers were available became significantly worse, the estimates when 3 towers were available improved slightly.

Regardless of this slight improvement, the assumption regarding the value of A_t that was involved calls into question the validity of using the Friis Equation to model range in this context. Davis' multilateration algorithms also suffered from inaccuracy due to this issue, and although modifying the form of the equation did result in an improvement in performance, the continued inaccuracy of the results once again suggests that a different model is necessary to accurately characterize these signals.

8.2 Latency

The latency of the WiCeNav solution was one of the system's most significant problems, and eliminating this issue was a primary concern when designing the new software architecture. Due in large part to a significant simplification of the code, the aim of reducing the latency was accomplished. Figures 8.11-8.13 show the time it took to create each loop of the code for each respective flight, along with the average and maximum values of loop time. The code is still fairly slow when compared with the typical update rate of the aircraft GPS. However, recall that the WiCeNav solution was shown by Davis to have an average latency of approximately 11 seconds. Clearly, the results shown here constitute a significant improvement, and at low enough speeds, provided the accuracy were high enough, would likely have been sufficient to enable autonomous navigation.

The occasional significant drops in latency are easily attributed to portions of the flight where the payload detected no towers at all, or just one tower. This often occurs on the ground, and at certain dead zones in the field while it is flying. When this occurs, the system

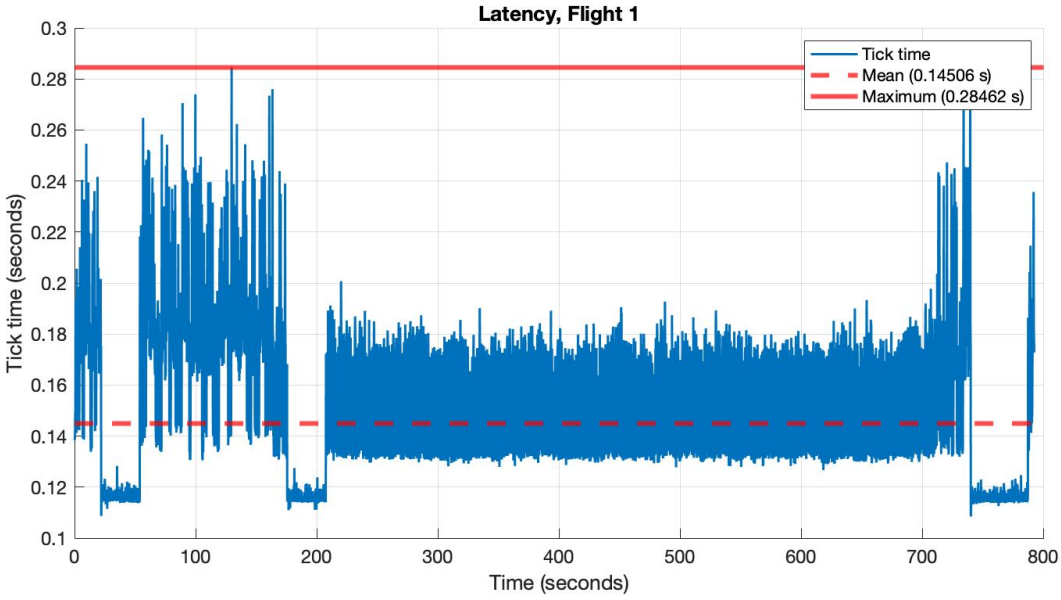

Figure 8.11: Time to complete each loop of the code, Flight 1.

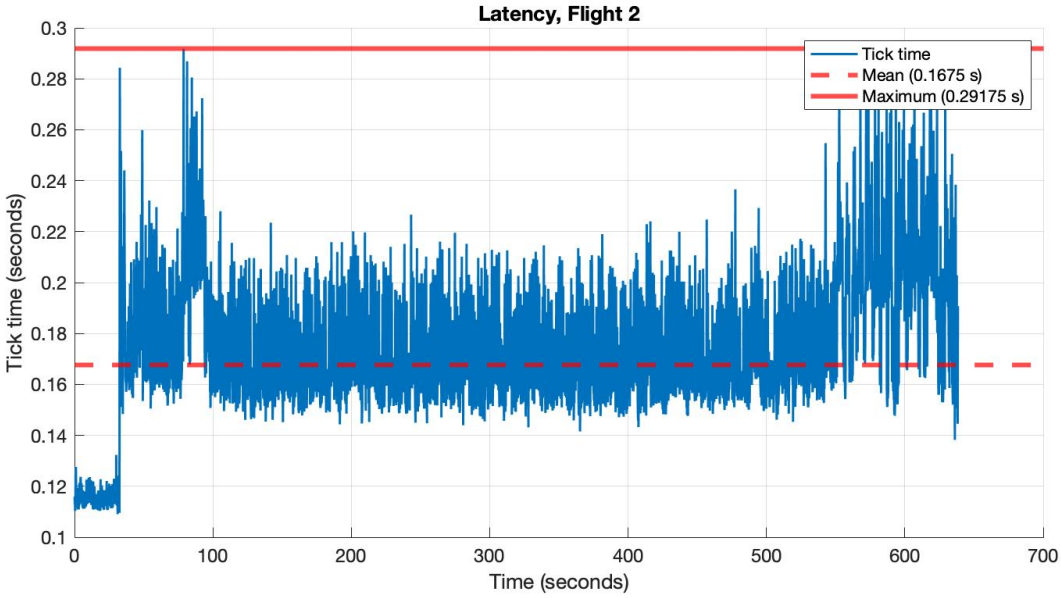

Figure 8.12: Time to complete each loop of the code, Flight 2.

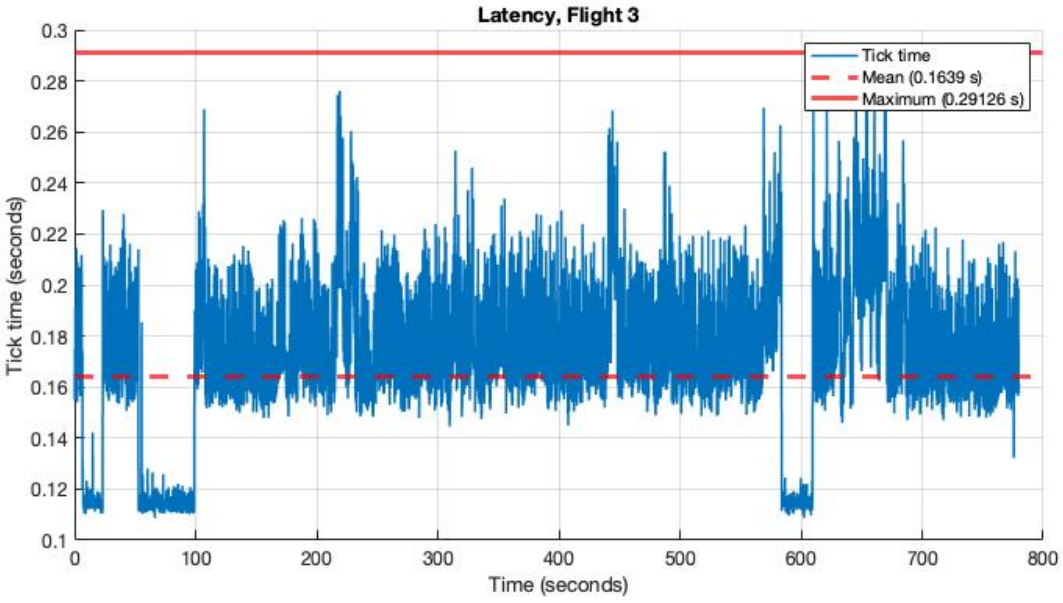

Figure 8.13: Time to complete each loop of the code, Flight 3.

cannot generate a position estimate, and simply returns the last known coordinates of the aircraft, causing processing time during these segments of the mission to noticeably decrease.

Note that when the payload detects only two towers, it still returns a position estimate, albeit a highly inaccurate one. Technically there are two solutions to the system of equations when this condition is met. Had time constraints not restricted the amount of development that could be done in this area, more refinement of the system behavior in this particular case could have been done that may have improved the overall accuracy of the system.

Clearly these results do not support the use of CellNav as a viable alternative to GPS. Even if the error were within an acceptable range, the area of the site where three towers can be detected is so small as to render it nearly impossible to perform adequate testing of such a navigation system at this location, and finding an alternative location to test was not possible within the timeframe of this project.

Chapter 9

CONCLUSIONS

The initial aim of this project was to integrate WiCeNav, a Wi-Fi and cell network based navigation system, onto a UAV. While previous results indicated that the accuracy of this system would prove sufficient for real-time autonomous navigation, this did not prove to be the case, which shifted the focus of the project drastically, and defined a new set of design requirements for the final system: primarily that it must not rely on Wi-Fi networks, which are not available consistently enough outside of urban areas to merit their use in this type of system. The second phase of this research simply set out to answer the question of whether it is feasible to use cell networks and only cell networks as the basis of an aerial navigation system for small Unmanned Aerial Systems, specifically in the case where no "insider" information about the network is available. The answer is a partial yes.

It is often possible to determine the locations of local cell towers from information that is available in the public domain, and although cell ID assignments are considered proprietary, they can be determined by means of simple observations. However, the system did not prove to be accurate enough to be safely used for navigation, primarily due to uncertain knowledge of the constants necessary to accurately determine the range to a tower. In fact, the blatantly unrealistic values that had to be used in the Friis Equation in order to obtain a reasonable estimate of the range are an indication that a completely different model should likely be implemented for this task. This is the single most significant obstacle to the application of a system of this type. Additionally, a number of shortcuts had to be taken in order to accomplish the task of associating cell IDs with towers, resulting in a system which is only useful inside a relatively small, localized region, and only temporarily, as Physical Cell IDs can change.

In summary, this system, while not completely functional, served as an informative demonstration of what information is required in order to design a reliable navigation system using cell signals, and how one might go about obtaining the necessary information from outside the network itself.

9.1 Future Work

The system presented here could benefit from several major improvements should some specific pieces of additional data become available. The most significant of these are:

1. The system currently relies in Physical Cell IDs, which are neither static nor unique. If an AT command becomes available in the future which provides the Global Cell ID for all neighboring cells, this should be used instead.

2. The system currently needs to be initialized with a single position measurement from GPS or some other highly reliable source, and suffers significant degradation of accuracy as the code runs due to accumulation of error in the calculation of the certain constants.

3. A universal method must necessarily exist for determining which cells belong to which towers. This is a database any network provider must maintain. The current system amounts to educated guesswork based on observations and extensive knowledge of the local geography, and renders the system useless outside of the area where it has already been tested, unless the operator intends to spend a day or more collecting local tower data in the region in which they intend to fly.

The second phase of this project also took place under a highly compressed time schedule, due to the amount of time spent attempting to glean useful results from the WiCeNav solution. This drastically limited the amount of testing that could be performed with the final system. If any more testing is to be conducted using the system in its current form, it would be beneficial to focus on refining the values of the effective aperture area constants

in the Friis Equation, or possibly testing other variations of this equation with different constants which may prove easier to determine. Any future research should also involve testing at other sites in order to determine if the problems encountered with the number of towers visible in the air were site-specific, or if this is an inherent limitation of the network, as has been hypothesized.

BIBLIOGRAPHY

[1] M. J. Davis. UAS Position Estimation in GPS Degraded/Denied Skies via WiFi and Cell Network Data (WiCeNav). Master's thesis, University of Washington, Seattle, WA, June 2020.

[2] C. W. Lum, H. Rotta, R. Patel, H. Kuni, T. Patana-anake, J. Longhurst, and K. Chen. UAS Operation and Navigation in GPS-Denied Environments Using Multilateration of Aviation Transponders. In *Proceedings of the AIAA SciTech 2019 Forum*, San Diego, CA, January 2019.

[3] R. S. Larson, J. Winde, and C. W. Lum. UAS Position Estimation in GPS-Degraded and Denied Environments Via ADS-B and Multilateration Fusion. In *Proceedings to the AIAA Information Systems-AIAA Infotech@ Aerospace Conference*, Kissimmee, FL, January 2018.

[4] Federal Aviation Administration. Remote Identification of Unmanned Aircraft. `https://www.faa.gov/news/media/attachments/RemoteID_Final_Rule.pdf`, January 2021.

[5] R. Svitelskyi. A Gimbal-Supported, Mono Camera, Relative Position Measurement System of a Visually Distinct Object for UAV Guidance. Master's thesis, University of Washington, Seattle, WA, June 2019.

[6] R. J. Grimes. Visual Anchoring: Fixed-Wing UAS Orbit Stabilization About a Visual Anchor Point Without GPS Dependence. Master's thesis, University of Washington, Seattle, WA, June 2018.

[7] Z. Kassas. Journal Papers. `http://kassas.eng.uci.edu/publications.html`, 2021.

[8] Google. Geolocation API: Overview. https://developers.google.com/maps/documentation/geolocation/overview, 2021.

[9] *AT Commands Reference Guide*. Telit Wireless Solutions, 14.1 edition, May 2017. https://www.telit.com/wp-content/uploads/2017/09/80407ST10116A_AT_Commands_Reference_Guide_LE9x0_R14.1.pdf.

[10] CableFree. Understanding LTE Carrier Frequency and EARFCN. https://www.cablefree.net/wirelesstechnology/4glte/lte-carrier-frequency-earfcn/, 2021.

[11] CellMapper. CellMapper Live Map. https://www.cellmapper.net/, 2021.

[12] OpenCellID. Frequently asked questions. https://wiki.opencellid.org/wiki/FAQ, 2021.

[13] Federal Communications Commission. ASR Registration Search. https://wireless2.fcc.gov/UlsApp/AsrSearch/asrRegistrationSearch.jsp, 2021.

[14] K. Heurtefeux and F. Valois. Is RSSI a good choice for localization in Wireless Sensor Network? In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pages 732–739, 2012.

[15] Federal Communications Commission. Human exposure to radio frequency fields: Guidelines for cellular antenna sites. https://www.fcc.gov/consumers/guides/human-exposure-radio-frequency-fields-guidelines-cellular-and-pcs-sites, October 2019.

[16] SITL failure - Ubuntu 14.04. https://discuss.ardupilot.org/t/sitl-failure-ubuntu-14-04/6407/6, 2015.

[17] Modem Manager. https://www.freedesktop.org/wiki/Software/ModemManager/, 2021.