IMPLEMENTATION OF AN UNMANNED AERIAL VEHICLE FOR NEW GENERATION PETERBILT TRUCKS

VENKATESH SRINIVASAN K, B.E.E.C.E.

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

May 2016

APPROVED:

Yan Wan, Major Professor

Xinrong Li, Committee Member

Shengli Fu, Co-Major Advisor and Chair
of the Department of Electrical Engineering

Costas Tsatsoulis, Dean of the College of
Engineering and Interim Dean of the Toulouse
Graduate School

Srinivasan K, Venkatesh. *Implementation of an Unmanned Aerial Vehicle for New Generation Peterbilt Trucks*. Master of Science (Electrical Engineering), May 2016, 80 pp., 58 figures, 34 references.

As science and technology continue to advance, innovative developments in transportation can enhance product safety and security for the benefit and welfare of society. The federal government requires every commercial truck to be inspected before each trip. This pre-trip inspection ensures the safe mechanical condition of each vehicle before it is used. An Unmanned Aerial Vehicle (UAV) could be used to provide an automated inspection, thus reducing driver workload, inspection costs and time while increasing inspection accuracy. This thesis develops a primary component of the algorithm that is required to implement UAV pre-trip inspections for commercial trucks using an android-based application. Specifically, this thesis provides foundational work of providing stable height control in an outdoor environment using a laser sensor and an android flight control application that includes take-off, landing, throttle control, and real-time video transmission. The height algorithm developed is the core of this thesis project. Phantom 2 Vision+ uses a pressure sensor to calculate the altitude of the drone for height stabilization. However, these altitude readings do not provide the precision required for this project. Rather, the goal of autonomously controlling height with great precision necessitated the use of a laser rangefinder sensor in the development of the height control algorithm. Another major contribution from this thesis research is to extend the limited capabilities of the DJI software development kit in order to provide more sophisticated control goals without modifying the drone dynamics. The results of this project are also directly applicable to a number of additional uses of drones in the transportation industry.

ACKNOWLEDGEMENTS

I would like to thank my parents back in India, Geetha Srinivasan and Dr. Rajagopala Srinivasan for always being there for me and supporting me throughout my life and education. I would also like to thank Vaibhavi Srinivasan for being a nice and loving little sister.

I would like to offer my gratitude to Peterbilt Motors Company, Denton, Texas for funding my thesis research project. I would also like to thank Mr. Wes Mays, Director, Omnitracs and Mr. Drew Harbach, Group Lead, Peterbilt Motors Company for their constant support and technical contribution to this project.

My sincere thanks goes to my advisor and mentor Dr. Yan Wan for her support and guidance during my thesis research and for providing me with various opportunities. I would also like to offer my gratitude to Dr. Shengli Fu and Dr. Xinrong Li for providing their technical expertise and serving as my thesis committee members. I would like to thank my Department of Electrical Engineering for providing me with financial support and the staff members Peggy Foster, Jason Meiritz, and Nick Thompkins for their support and help.

I am very grateful to Dr. Miguel Garcia-Rubio and Dr. Murali Varanasi for their encouragement and guidance throughout my studies. I would like to thank all my close friends Mitch Grabner, Annika Nelson, Jeffrey Smith, Jesse Stauffer, Rahul Shukla and my Denton family friends including Mark and Dawn Sandel, Mel and Gloria Achterberg, and Jorge and Patty Urbina for their meaningful contributions to my education.

At last, my sincerest gratitude goes to my American parents Dr. Claire Sahlin and Dr. Jon Nelson for providing me a loving home and an affectionate family in the United States during the pursuit of my graduate studies at the University of North Texas.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation and Background

As technology evolves, one sees an increased implementation of robotics to perform innovative research with extremely higher precision and accuracy. One area of robotics in particular, unmanned aerial vehicles (UAV), has gained a lot of popularity. UAV applications have grown over the years with increased utilization in the defense and military sector, search and rescue operations, surveillance, hobbyist aerial photography, and various other applications.

One of the extremely innovative and interesting uses of a drone includes the conceptual prototype of a technologically advanced car developed by a French multinational automobile manufacturer Renault called the Kwid Concept [1]. In this prototype, the compact smart car is installed with an unmanned aerial vehicle to help in traffic maneuvering [1]. There has been various work in the field of UAVs, article [2] and [3] gives a good understanding, modelling and results on designing a safe controller for micro UAVs, which generates safer, optimal and smooth trajectory in the strictly constrained environment. The experimental results and simulations shows the efficient execution using micro UAVs that supports the models and design provided in the paper. In articles [4-6], vision based state estimation has been explained which helps in the autonomous maneuvering of UAVs in complex indoor and outdoor environments using onboard camera and the inertial sensors which has given us the reference for the development of vision based maneuvering which will be developed later. Article [7], explains the multi-sensor fusion for the robust autonomous flight for indoor and outdoor environment.

Over a year ago, Peterbilt Motors Company, a corporation that manufactures heavy-duty trucks headquartered in Denton, Texas, came up with the idea of implementing a drone on their flagship new generation trucks. The engineers at Peterbilt made a conceptual video of the installation of a Parrot Incorporated quad-rotor drone on their Model 579 trucks. This drone is intended to help both in navigation and parking the truck in difficult parking locations. While I was working at Peterbilt as an intern in the summer of 2014, Peterbilt engineering manager Mr. Wes Mays suggested that this project might provide a good topic for my master's thesis. I concur because I envision that in few years UAVs will be widely used to perform a variety of operations in the automotive industry, including, but not limited to, inspection, parking, automatic truck maneuvering, and data logging.

1.2 Research Objective

Previously at Peterbilt, the engineers only envisioned the use of drones for traffic and parking related applications. However, Peterbilt wanted to explore the possibility of using UAVs to perform pre-trip inspections (Visual Inspection) on the trucks. Truck pre-trip inspections include confirming turn signal, head light, and wind shield wiper functionality and a visual investigation of other parameters that may cause accidents on the road. While traditional visual inspections require two people, a drone-assisted, automated inspection could be performed by just the driver, thus reducing inspection costs, saving time and increasing inspection accuracy. The idea to implement a drone to perform an autonomous vision-based inspection makes this process more effective and innovative.

The scope of this thesis research is to develop a stable infrastructure and find the necessary design requirements for the development of a pre-trip inspection project. Another major contribution from this thesis research is to extend the limited capability of the used DJI software development kit for more sophisticated control goals, while not modifying the drone dynamics. The DJI software development kit used in this thesis does not allow to control the vertical speed of the drone for throttle control, I have overcome this problem by introducing delay in each of the throttle execution. The entire flight control developed in this thesis has been built around the drone stability control made by DJI. In order to find the solution and pave the road for the future completion of the pre-trip inspection on trucks, this thesis research work will focus on the following objectives:

1. Research and find the best unmanned aerial vehicle, micro controller boards, sensors, cameras, and other supporting accessories for the UAV. Which would effectively satisfy all the requirements for the main project.
2. Develop an android application that controls the drone and provides flight information, performs stable take-off and landing, provides real-time video from the drone, and automates flight control including throttle up/down, controlling roll, yaw, and pitch of the drone.
3. Develop a hardware-software infrastructure to use the laser rangefinder for altitude measurement. Use the laser rangefinder for building the height algorithm that uses the laser sensor and by-passes the use of pre-installed pressure sensor on the drone. This height algorithm will control the drone with specific height accuracy. Implement a P-Controller on the height algorithm.
4. Simulate the drone environment in MATLAB, develop a height algorithm in Matlab and obtain the results.
5. Execute flight tests, receive the resultant data and compare it with the simulation data.

1.3 Thesis Organization

This thesis first provides a high level overview of the pre-trip inspection project in Chapter 2. It also provides the larger context for the project, emphasizing how this research provides a fundamental infrastructure that contributes significantly to future pre-trip autonomous inspection projects.

Chapter 3 focuses on the hardware requirements for this thesis research by comparing various drones and their capabilities and limitations. This chapter also provides detailed information regarding the specific hardware components utilized and their contribution to the project.

Chapter 4 explains the various software requirements and the software development kits that have been used in this project. It provides a high-level view of the various classes and functions incorporated in the development of the android application and its communication with the drone.

Chapter 5 explains a block diagram of the prototype setup as well as an overview of the android application. This chapter describes the prototype by explaining the UAV-Phone communication, the model of this project, and also by providing a thorough outline of how the android application is working.

Chapter 6 contains the results of the various flight tests, development of the simulation model on Matlab, data received from the simulation, and comparison of the data received from the field tests and the simulation.

Chapter 7 discusses the conclusion of this project by overviewing all the results achieved from this project and how this framework will be used for the effective future development of the pre-trip inspection on the trucks and its contribution to the scientific community on drone research.

CHAPTER 2

AUTOMATED PRE-TRIP INSPECTION AND THESIS RESEARCH OVERVIEW

2.1 Introduction

The pre-trip inspection process focuses primarily on inspecting the exterior of a truck. The exterior inspection guidelines provided by the Division of Motor Vehicles (DMV) [8], involves the examination of automotive parts including lighting systems, tires, body damage, windshield wipers, and other components that are critical to truck operation. The various test requirements for exterior inspection are provided in Appendix 1.

When UAV automated inspection is fully developed there will be two modes of operation: Automated Inspection and Semi-Automated Inspection.

- Automated Inspection Mode: The automated inspection mode of operation will be fully automated. If the driver selects this mode, the UAV will fly around the truck and perform the entire inspection process without driver assistance. At the end of this process, the UAV creates the inspection report and saves it to the remote server.

- Semi-Automated Inspection Mode: In this mode of operation, the UAV will fly around the truck and transmit the video from various locations to the display on the dashboard. While the drone transmits video, the driver checks to see whether each of the test scenarios are performed correctly and decides whether to pass or fail each test scenario. In this mode of operation, the location of the UAV will be shown at all times on the display from the camera installed on top of the truck. The UAV will perform the data logging at the end of the inspection process and save it to the remote server.

The following diagram provides a high level implementation of this automated pre-trip inspection using the UAV. The diagram highlights the four primary aspects of this project. The user interface block on the diagram is the interface that will be shown on the truck's dashboard display.

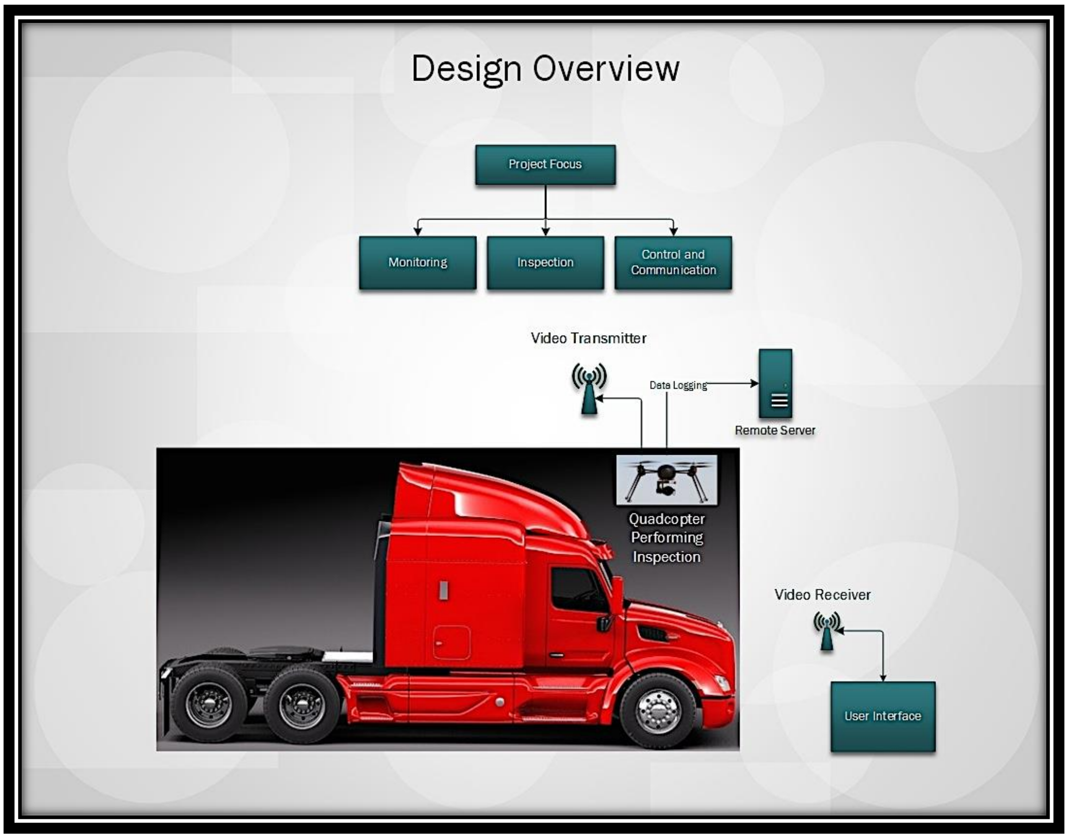

Figure 1. High-Level Automated Pre-Trip Inspection [9]

2.2 Truck Inspection

This implementation of the Semi-Automated Inspection will use a flat surface on the prototype Peterbilt 579 NGP [9] model truck as the UAV launching station. The UAV will be programmed with a region of operation, which is the area around the truck where the UAV will fly in order to perform the inspection.

The camera installed on the UAV will capture video, which the UAV will transmit in real-time to the Cab Electronic Control Unit (CECU) which is the micro controller that controls all the truck operation. The video will be displayed on the truck dashboard through an android application. The driver will visually inspect and verify each of the aspects of the exterior inspection to determine whether the test scenarios pass or fail. The UAV will perform the data logging automatically onto the server if the truck is connected to the server using 4G LTE or onto any storage device on the truck.

The UAV could also be used to detect whether everything in the trailer is in order or to ensure that there are no unwanted objects on or around the vehicle that could create road hazards. In the automated inspection mode, digital image processing or digital video processing will be used to parse the video and conduct the inspection automatically.

2.3 Drone System Monitoring

This component focuses on the monitoring of the UAV. In order to effectively implement the UAV in the truck, constant monitoring of the proper functioning of the UAV and its location is very important, so as to avoid any potential failure by transferring the automatic control of the UAV to manual control. In order to monitor the UAV, markers may be installed on the truck and an algorithm will be developed to detect the distance of the UAV from the markers. This information will be combined with the GPS data to accurately pinpoint the location of the UAV.

2.4 Drone Communication Control

An unmanned aerial vehicle will be chosen with sufficient flight time and good stability and controllability. The communication between the UAV and the ground-control unit will be implemented using Wi-Fi, which follows the IEEE 802.11 standard and operates at a frequency of 2.4 GHz. The UAV will have a high quality camera, which will be necessary for transmitting the video to the truck's display for performing the inspection.

I will also develop automatic UAV control for accurate landing and take-off from the launching pad of the truck as well as all flight maneuvers mandated by the inspection requirements. This will involve very meticulous programming since efficient maneuvering of the UAV is a crucial aspect of this project.

The UAV will be connected to the truck's computer at all times using a Wi-Fi network, while the truck's computer will be connected to the remote server at a dealer location using a 4G LTE network. Upon the completion of the inspection process, the UAV will create and transmit a report file. The file will be transmitted to the truck's computer, which will be enabled to send the report file to the remote server using the 4G LTE network.

2.5 User Interface

An android based Graphical User Interface (GUI) will be developed for the Paccar 2.0 dashboard display. This interface will display video received from the UAV as well as the location of the UAV that is captured from the camera located on the truck. In addition, this interface will have all the functionality required for the control of the UAV.

2.6 Thesis Research Concentration for Pre-Trip Inspection

The ultimate goal of this thesis project is to develop a very stable and a robust framework for the automated pre-trip inspection on which the rest of the development can be successfully implemented. The outcome of this thesis will not only develop the infrastructure for the pre-trip inspection but also provide an android based application that can control the drone and provide stable height control using a laser sensor, and real-time video.

The different phases of implementation of this thesis research was as follows:

Phase 1 Research conducted on hardware and software requirements.

i. Selection of the drone, micro-controller board, and other supporting hardware sensor accessories.

ii. Preliminary research study on the Parrot AR Drone.

iii. Determination of the best Software Development Kit(SDK) for use in creating an android control application.

Phase 2 Prototype Setup and Verification

i. Prototype development using Arduino boards and laser sensor.

ii. Arduino board and XBee setup for receiving laser rangefinder altitude readings from the drone and transmission of the readings over Bluetooth.

iii. Manual flight test performed for a stable setup verification.

Phase 3 Android Application Development and Flight Test

i. Precise control for stable take-off and landing in complicated outdoor settings from the application.

ii. Real-time video transmission on the android application.

iii. Complete drone flight control including throttle up/down, roll, pitch, and yaw.

Phase 4 Height Control Algorithm on Android Application

i. Developing height control algorithm, using laser rangefinder sensor which records the drone altitude.

ii. Performing flight tests to effectively execute the control.

iii. Addressing height control challenges.

Phase 5 Simulation and Testing

i. UAV environment simulation on the Matlab.
ii. The challenges encountered during flight tests were simulated in Matlab to find the best problem resolution in real time flight.
iii. Obtaining results and comparison with the real time flight data.

CHAPTER 3

HARDWARE REQUIREMENTS

With the growing popularity of UAV technology, there are a number of hardware options to consider for use in an autonomous pre-trip inspection. These options include a variety of drones as well as additional required hardware. The current chapter discusses the various hardware components required for this thesis research. This hardware includes not only a drone that features appropriate features necessary for this research, but components such as micro-controller boards, sensors, XBees, Bluetooth modules.

3.1 Unmanned Aerial Vehicle Selection

3.1.1 Introduction

Unmanned aerial vehicles (UAVs) are also referred to as drones, remote controlled aircrafts, aerial robots et cetera. An UAV, as the name suggests, is an aeronautical and electrical system without an on-board pilot that is controlled using a wireless remote controller or via software control in which the drone is programmed to perform autonomous operations.

Drones' were initially developed for use in special military operations including reconnaissance missions, search and rescue missions, and remotely controlled bombing mission. In each of these cases, UAVs can execute missions without jeopardizing the safety of the aircraft pilots and the troops on the ground. However, in the past few years' drone use has become much more prominent in civilian applications such as aerial photography, surveying, hobbyist flying, and academic research.

The types of drones range from the famous MQ-1 Predator and MQ-9 Reaper (Military Grade Drones) to small scale civilian drones such as the DJI Phantom Series, DJI Inspire, Parrot

Bebop drones, and 3DRobotics drones. Unmanned aerial vehicles also come in two different varieties: Fixed wing drones and multi-rotor drones.

- Fixed-Wing Drones: Fixed wing drones are similar to airplanes in that they fly using wings and the forward airspeed generated by the rotors mounted on the wings or on the nose of the aircraft. These fixed wing drones are installed with a microcontroller that is programmed to be controlled using a remote controller. Fixed wing drones benefit from the capability to glide without any power. Fixed wing drones also have the capability to carry heavy payloads. However, fixed wing drones cannot be used for applications that require great precision and accuracy because they need air moving around their wings to have lift. In other words, they cannot hover at one position.

- Multi-Rotor: Multi-Rotor drones have multiple rotors present in their structure. Currently available multi-rotor drones include tri-copters, quad-copters, hex-copters, and octo-copters. The most commonly used multi-rotor drones are quad-rotor drones. Because of their flexibility and durability, quad-rotor drones are considered in a lot of applications.

3.1.2 Unmanned Aerial Vehicle (UAV) Selection

For this thesis project we need basic hovering functionality to perform a thorough investigation on a truck, therefore we cannot choose a fixed-wing drone. This research also requires a drone that has a camera and has enough thrust to carry some additional sensors and microcontrollers, after detailed study I decided to use a multi-rotor drone in this project. The various drones that I considered for this research include the following:

3.1.2.1 AR Drone 2.0

AR Drone 2.0[10] is a small quad copter robot as shown in figure 2 that can be controlled via Wi-Fi using a smartphone or a tablet. It is a very efficient drone that has many sensors and has complex designed electronics just like any full blown quad rotor. This quad copter can be used indoors and outdoors.

Figure 2. AR Drone 2.0 [10]

This drone is capable of taking high definition videos and capturing images. It can fly outside and inside, however after various tests performed by me. I found that because of a very light structure this drone does not provide good performance outdoors as it is easily blown off course by gusts of wind. The AR Drone can be flown using the AR Drone application from a smartphone or a tablet in which it has virtual joysticks and other features to control the drone. Though ultimately not selected for this research, I used this drone as a platform to learn drone dynamics and mechanics with the help of the Matlab and Simulink Development Kit developed by Peiter Mosterman and David Sanabria et al [11].

3.1.2.2 DJI Flame Wheel F450 and F550 [12]

DJI is a Chinese drone manufacturing company that produces quality drones for hobbyist photography and academic research. DJI sells the Flame Wheel kit which is a do it yourself

(DIY) setup. It comes in two different kinds, quad-rotors and a hex-rotor drones. DJI Flame Wheel F450 and F550 is shown in the following figures:

Figure 3. DJI Flame Wheel F450 [12]

Figure 4. DJI Flame Wheel F550 [12]

These drones have high strength arms. It also provides ultra powerful frame boards. However, the Flamewheel kit only provides the basic drone structure. The user needs to also obtain flight controllers and other required hardware components in order to operate these drones efficiently. These drones require a lot of initial setup and require the purchase of additional hardware components. Given both time constraints and financial limitations, I decided not to use this drone for this thesis project.

3.1.2.3 DJI Spreading Wings S800 [13]

DJI Spreading Wings S800 is a hexa-rotor drone that is both highly powerful and very durable. As the website indicates, it has a circuit-integrated center frame integrated with ESCs, efficient carbon fiber propellers, and high performance brushless motors that can lift a heavy payload of up to 10lbs. The drone image is shown in the following figure:

Figure 5. DJI Spreading Wings S800 [13]

Just like the DJI Flame Wheel kit this drone also requires the user to obtain a flight controller and other hardware for it to become flight capable. Also this drone is very big in cross section. Though it has many desirable features, this drone will not fit on the roof of the truck cab and thus it is not suitable for this thesis project.

3.1.2.4 DJI Phantom 2 Vision+ [14]

After a thorough investigation of various drones and their features, I chose DJI's Phantom 2 Vision+ drone for this research because of its excellent flight stability, longer battery life, and its payload capability. Phantom 2 Vision+ is an easy to fly quad-rotor drone with extremely stable 3-axis gimbal and 14-megapixel camera.

Figure 6. DJI Phantom 2 Vision+ [15]

Phantom 2 Vision+ features an extremely stable flying environment, with a powerful DJI 5200mAh LiPo Battery with a maximum flight time of 25 minutes [3]. It features a maximum ascent and descent speed of 6m/s and 2m/s. It comes with standard radar positioning and return home facility. This feature enables the drone to come back to its home location and land if it goes out of range or loses its connectivity with the remote control. The remote controller works at an operating frequency of 5.7GHz-5.8GHz with the communication range of 800 meters in an open area. The Phantom 2 Vision+ comes with four powerful motors which gives a maximum thrust of 200grams/motor which makes it possible to add more equipment (sensors and micro-controllers) on the drone. Phantom 2 Vision+ has a hover accuracy of Vertical 0.8m and horizontal 2.5m [14]. It uses a Naza-m[16] based microcontroller.

3.2 Sensor Components

In this thesis research there are various hardware sensors that have been used. Phantom 2 Vision+ features a lot of sensors that are inbuilt in the drone. The sensors include a 3-axis gyroscope, barometer, compass, and GPS system.

In this project, an external laser sensor has been added which provides accurate distance measurement that is used in the measurement of accurate height data for control. For this purpose, Lightware's SF02 laser rangefinder [17] has been selected. This laser sensor is a laser distance sensor with very high accuracy. The diagram of the laser sensor is shown in the following figure 7.

Figure 7. Laser Range Finder SF02[17]

It can measure the distance up to 40 meters (130 feet). The main reason for using the laser sensor is that it doesn't get affected by the environmental conditions. Its readings are always accurate because it uses optical physics for measurements.

It can provide up to 12 readings per second which is suitable for implementation in many controlling algorithms. This sensor could be powered with the voltage between 6.5-9VDC or could be powered from Arduino boards with a 5V DC.

3.3 Microcontroller Boards

3.3.1 Arduino UNO Board [18]

Arduino is an open source hardware and software company based in Italy. The Arduino programming environment is very user friendly and provides an efficient environment for programming sensors, actuators, and other devices. Arduino is also a very affordable programming environment. Arduino’s most famous micro-controller board is called Arduino UNO which is the easiest programmable board.

Arduino UNO is an Atmega 328P microcontroller. It has 14 digital input/output pins and 6 analog inputs. It is a lightweight controller with a powerful capability to be used to control and sense a variety of sensors. Arduino UNOs are widely used in applications because of its efficient and easily programmable board supported by a very large open source community. The Arduino UNO board is shown in the following figure:

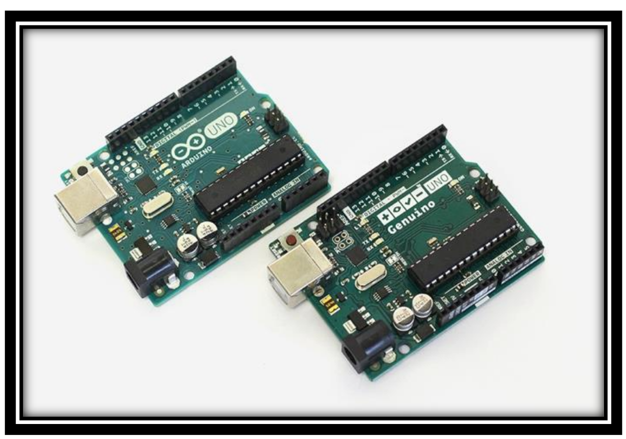

Figure 8. Arduino UNO Boards [18]

In this project, two Arduino UNO boards are used. One Arduino board is mounted on the quad-copter and connected to the laser sensor to obtain and transmit the distance reading.

Another Arduino board is used to wirelessly receive the laser data and transmit over a Bluetooth channel to an android cellular phone.

3.3.2 Arduino Wireless Shield [19]

An Arduino wireless shield is an extra add on module that is used to equip the Arduino controller boards. This shield comes with the flexibility to communicate among Arduino boards wirelessly using xbee modules. This module can communicate up to 100 feet indoors or 300 feet outdoors.

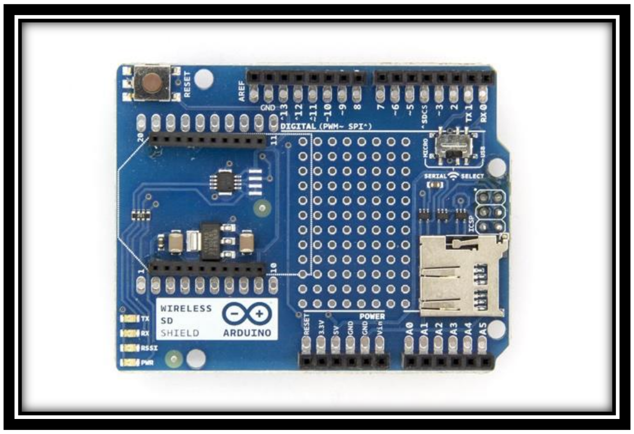

Figure 9. Arduino Wireless Shield [19]

3.4 XBee and Bluetooth Modules

In this project Xbee modules are used to receive the data from the laser sensor and the Bluetooth module is used to transmit the received laser data over Bluetooth to the phone.

3.4.1 XBee Modules

XBee radio is a Digi International product used for wireless radio applications. It follows the IEEE 802.15.4 standard. It transmits data at a frequency of 2.4GHz with a range of 300 feet. For this project, XBee Series 2 [20] has been used because it allows the user to create a network easily. XBee Series 2 has a built in antenna with a 250kbps data rate as shown in the following figure:

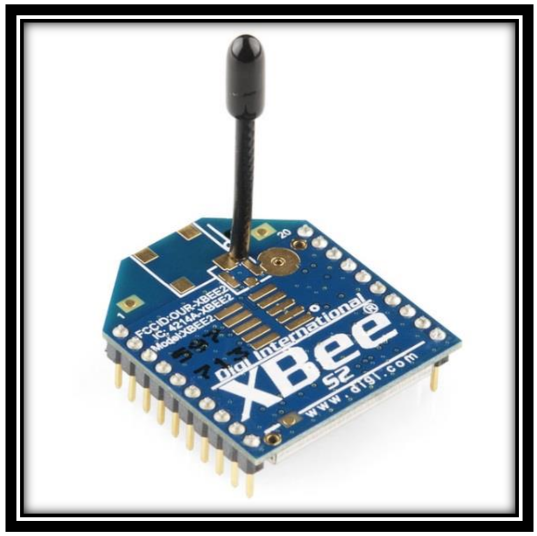

Figure 10. XBee Series 2 [20]

In this thesis research, one XBee module is installed on the Arduino connected to the laser sensor which acts as a transmitter/receiver of the laser readings. Another Arduino board on the ground is installed with an XBee module that acts as a receiver for the laser data. The two XBees are calibrated to be a transceiver using XCTU software [21] which helps in setting up XBees.

3.4.2 Bluetooth Modules

Bluetooth technology, IEEE 802.15.1 standard, works at a frequency of 2.4GHz and has a limitation to exchange data up to 100 meters. A Bluetooth module is used in this project to transmit the received data in the Arduino receiver setup to the android phone over a Bluetooth channel. BlueSMirf silver [22] is a Bluetooth module developed by Sparkfun Electronics. This device is very easily programmed and can transmit data over a Bluetooth channel up to 18 meters. It creates a very robust link. This module is shown in the following figure.

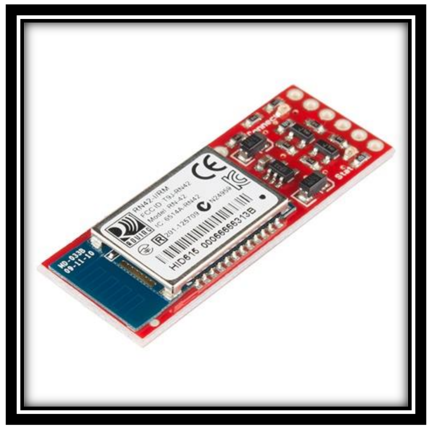

Figure 11. BlueSMirf Bluetooth Module[22]

CHAPTER 4

SOFTWARE REQUIREMENTS

This section focuses on the various software needs and requirements for this thesis project. As discussed previously, this project requires the use of several different software development environments and tools.

4.1 DJI Software Development Kit [23]

4.1.1 Overview and Features

DJI, the company whose product Phantom 2 Vision+ is chosen for this project, also released a software development kit (SDK) for its products in the fall of 2014. This SDK provides platforms for iOS and Android developers to make applications. This thesis project includes the development of an Android application, therefore the android SDK played a useful role in the software development. Indeed, some of the capabilities and features offered by this SDK informed the decision to select the DJI drone for this thesis project.

The significant features offered by the SDK [24] include:

- Access to Real-Time Video from the High Definition (HD) camera for on screen display within the application.
- Camera access for capturing HD images and downloading them to the phone.
- The Flight System feature can receive real-time flight data from the drone microcontroller.
- The gimbal, which is a camera stabilizer, can be controlled using the SDK thus providing a tilt control and stable image capture.

- Battery information can be received from the drone microcontroller in real-time.
- Ground station and flight control features that are used for route planning can receive full telemetry data. The ground station features allow one to upload and download waypoints, pause/continue the mission, and establish automated “go home” features. Flight control features also include one touch take-off and landing and also make it possible to develop joystick control features in the android application.

4.1.2 DJI Software Development Kit

The SDK thus provides the framework for developing applications in android and iOS environments, making it possible to control DJI drones using cellphones. The SDK is composed of various important packages and classes that assist in the development of control applications. SDK Classes and packages are the core of the application development process. The important SDK classes are as follows:

4.1.2.1 DJI Drone Class

In the SDK, DJI Drone class is the heart and brain of the SDK. This class connects each component, interchanges information and also sends and receives command signals to and from the drone. This class performs a variety of tasks. It verifies the application key provided by the user to activate the app. This class helps in connecting and disconnecting the drone from the phone.

4.1.2.2 DJI Battery Class

This class helps in building a connection between the app and the battery of the drone. This class could be used to program the application to receive the current battery status and the current battery information to plan the mission for the drone according to the remaining battery power.

4.1.2.3 DJI Camera Class

The DJI Camera class is very important in that it helps in getting and setting the camera functions, taking photos, recording videos, and receiving the data in the SD card. This class also lets the developers set the photo size parameters, file name, set white balance, and other graphic file settings.

4.1.2.4 DJI Gimbal Class

This class helps in controlling and setting the drone’s view. This class helps the developers to make apps with first person view to see how it looks from the drone’s perspective. The main function of the gimbal is to stabilize the camera when the drone is in motion. In the phantom 2 vision+ drone the gimbal only provides the control for the pitch of the camera. The roll and yaw can be controlled by changing the heading of the drone.

4.1.2.5 DJI Ground Station

This class is the heart and brain of this project. The majority of the controls utilized in the Android application relied greatly on the capabilities of this class. There are specific steps that need to be performed before any of the control command is sent to the drone from the application [25].

1. The Open ground station method is used to open the connection between the app and the drone.
2. The Upload ground station method is used for uploading the tasks that the user needs to perform.

3. The Start ground station method is used for starting tasks.
4. The Pause ground station task method is used to pause tasks until the resume ground task is invoked. All ongoing tasks will be saved until resumed.
5. The Resume ground station method is used in resuming the ground tasks.
6. The Close ground station method is used in closing the connection between the drone and the application.

In this project we have used this class to perform a variety of functions including:

- Take-off: Take-off of the drone can be performed using one-key-fly function from the ground station class. This function makes the drone rise to a specific height after the function is invoked. In the case of Phantom 2 Vision+, the drone flies up to approximately 5 meters after the function is invoked.
- Aircraft Yaw Speed: The drone’s yaw speed can be set using the set aircraft yaw speed function provided by the ground station task. The yaw rotate speed is set between -1000 to 1000 which is the angular speed of -150 degrees per second to 150 degrees per second.
- Aircraft Pitch Speed: The drone’s pitch speed can be set using the set aircraft pitch speed function provided by the ground station task. The pitch rotate speed is set between -1000 to 1000 which is the angular speed of -150 degrees per second to 150 degrees per second.
- Aircraft Roll Speed: The drone’s roll speed can be set using the set aircraft roll speed function provided by the ground station task. The roll rotate speed is set between -1000 to 1000 which is the angular speed of -150 degrees per second to 150 degrees per second.

- Aircraft Throttle: The drone's vertical movement can be controlled using the aircraft throttle function. There are three ways to control the throttle of the drone. Using Command "0" for throttle stop, the drone stops and hovers at the height and location at the time that the function was invoked. Using Command "1" for throttle up, the drone will gain height at a constant velocity. Command "2" invokes throttle down, causing the drone to lose height at constant velocity.
- Waypoints: The developer can program the drone to follow a specified predefined path using this function. This waypoint function consists of several user inputs such as altitude, latitude, longitude, speed, heading, max reach time, and stay time. The user can plan the mission using the various functionalities for the drone to perform a trajectory-based mission. The drone uses its GPS to execute these tasks.
- Hover: The ground station pause method is used to make the drone hover at a position.
- Go Home function: When the drone is turned on, it waits for the GPS signals to be locked and then records and saves the home location latitude and longitude. The go home function is invoked if the drone goes out of the radio range or if a low battery level is sensed. This function is used to provide a fail safe method. When this function is invoked the drone rises to an altitude of 66 feet, returns to the previously saved home location, and lands.
- Ground Station Flight Information: This class provides the flight information to the user. The flight status information provided by this functions include altitude,

latitude, longitude, GPS status, pitch, roll, yaw, latitudinal velocity, longitudinal velocity, and altitude velocity.

4.1.2.6 DJI Main Controller

This class provides the users with real-time information from the drone about the main controller status. The users can receive data every 1000 ms. This class provides a series of information functions to control the drone.

- Get/Set Go Home Altitude: The user can receive the go home altitude and also change the go home altitude in case the go home function is invoked. The go home altitude can be set between 20 m and 500 m.
- The user can turn on or off the motors, perform automated landing and takeoff, calibrate the compass, and saving flight data to the external drive.
- In the DJI Main Controller class, there is another sub class called the main controller system state which provides the microcontroller system information. This sub class provides state information such as altitude, compass status, drone latitude, longitude, flying status, motor status, pitch, roll, yaw, speed, and velocities in x, y, and z directions.

4.1.2.7 DJI Media

This class is present in the SDK for saving the information of the image file in the SD card. The data includes the file name, file size, data type, and the date it was created.

4.1.2.8 DJI Interfaces

This class provides various interface callback functionalities that receive the data from the drone and send commands generated by the user. This class basically creates an interface between the drone and cellphone application.

4.2 Android OS and Android Studio

Android [26] is a prominent cell phone operating system within the smart phone industry. It is a Linux based operating system developed by Google. As of 2015, android is believed to be the worlds largest operating system being installed on cellphones [26]. Given the pervasive use of the android operating system, it was selected as the application development platform for this project.

Android application development is performed using android studio [27] which provides an integrated development environment. Android application development is comprised of application is programming in java and layout design work in xml. Once the development of the application is complete it can be packaged into ".apk" format and can be installed on android based cellphones or can be pushed out to the google store for users to download the application. Android project structure [28] is comprised of various elements that play a vital role in the development of the android applications.

- Java: Java code is used to create application functionality. The java files are stored in the src/main/java/com directory.
- Resources: The user interface style and design includes folders containing images, icons, layout, menu, and values within the resources. The user interface design is programmed in xml.

- Android Manifest: The android manifest provides important information about the application to the android operating system. As it is stated in the android developer tutorials [29], the android manifest does the following things:
    - It creates a package name which is the unique identifier for each application.
    - It explains the various components of the applications, helping the android operating system understand when and how this application works.
    - It also states the processes in which user permission is required, such as whether or not Bluetooth needs to be turned on.
    - It states the level of android versions the app can be run on.
    - It also lists the libraries the application might be using.
- Intent: As defined in the android developer tutorial website [29], intent is a description of the operation to be performed. Intent can be used to launch an activity, broadcast it to any interested receiver components, or to communicate with a background service. Intent acts as a bridge between various different activities.

In this project the android application is chosen because the application needs to control both the drone and the dash board software in the Peterbilt trucks which are android based.

4.3 Arduino Development Environment

Arduino is an open source hardware and software company based in Italy. The Arduino programming environment provides an easy and efficient environment for programming sensors, actuators, and other devices. Arduino is a very affordable programming environment. Arduino's most famous micro-controller board is called Arduino Uno.

Arduino IDE [30] software is an open source application written in java and used as an interface to write the code for Arduino boards. Arduino programs are written in C or C++. The Arduino IDE comes with a function for adding several different libraries that can be called within a function. Usually, the Arduino IDE has a setup function in which the user initiates the methods at the start of the program. Also, the Arduino IDE has loop function, in which methods that need to be initiated and performed concurrently and repeatedly are placed.

4.4 XCTU

XCTU [21] is free open source software developed by Digi international. It consists of tools used to configure and calibrate Digi International radio frequency devices such as the XBee and other RF devices. It is a software environment in which the XBees can be programmed to work and calibrated to perform wireless communication tasks.

CHAPTER 5

PROJECT IMPLEMENTATION

5.1 Introduction

A fairly extensive configuration process is required to implement this research project. This includes configuring hardware devices, programming for both Arduino and Android, and coordinating wireless communications for device control. This chapter provides these configuration details and the procedures followed to obtain positive results.

5.2 Project Implementation

The following figure 12 provides a block diagram that illustrates the high level component implementation required for this thesis research.

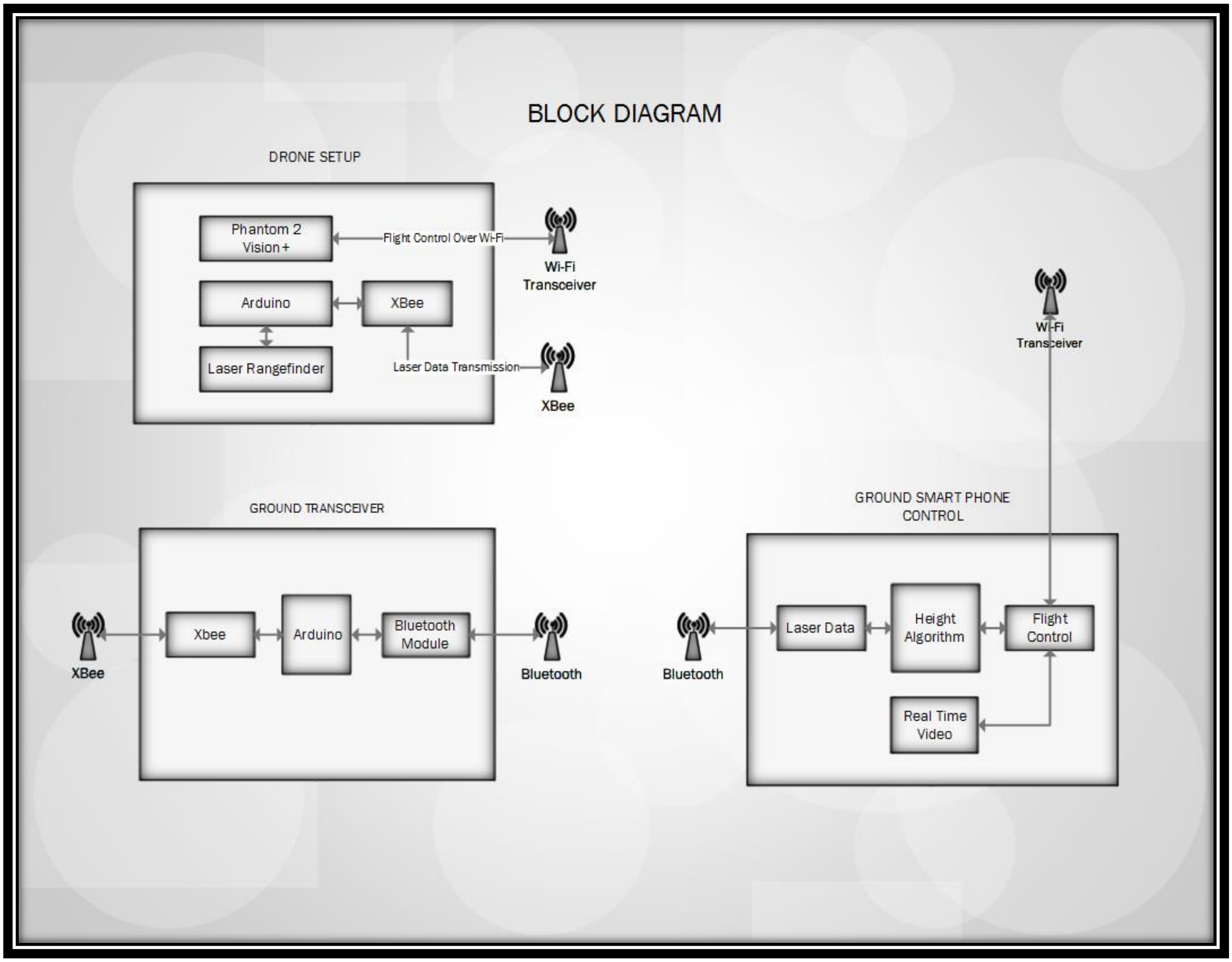

Figure 12. High Level Project Overview

5.2.1 Drone Setup Block

Like the majority of drones on the market, Phantom 2 Vision+ uses a pressure sensor to calculate the altitude of the drone for height stabilization. However, these altitude readings do not provide the precision required for this project. Rather, the goal of autonomously controlling height with great precision necessitated the use of a laser rangefinder sensor. The need to access this height data from a remote controller also required the use of wireless communications. As a result, the drone setup as seen in the block diagram above includes a Phantom 2 Vision+ that includes a high-definition camera, a remote controller, an Arduino Uno board, a laser rangefinder sensor, an XBee transceiver, an Arduino wireless shield, propeller guard, and a 9V battery. The closer look of the drone setup block from the project prototype is shown in figure 13 and figure 14.

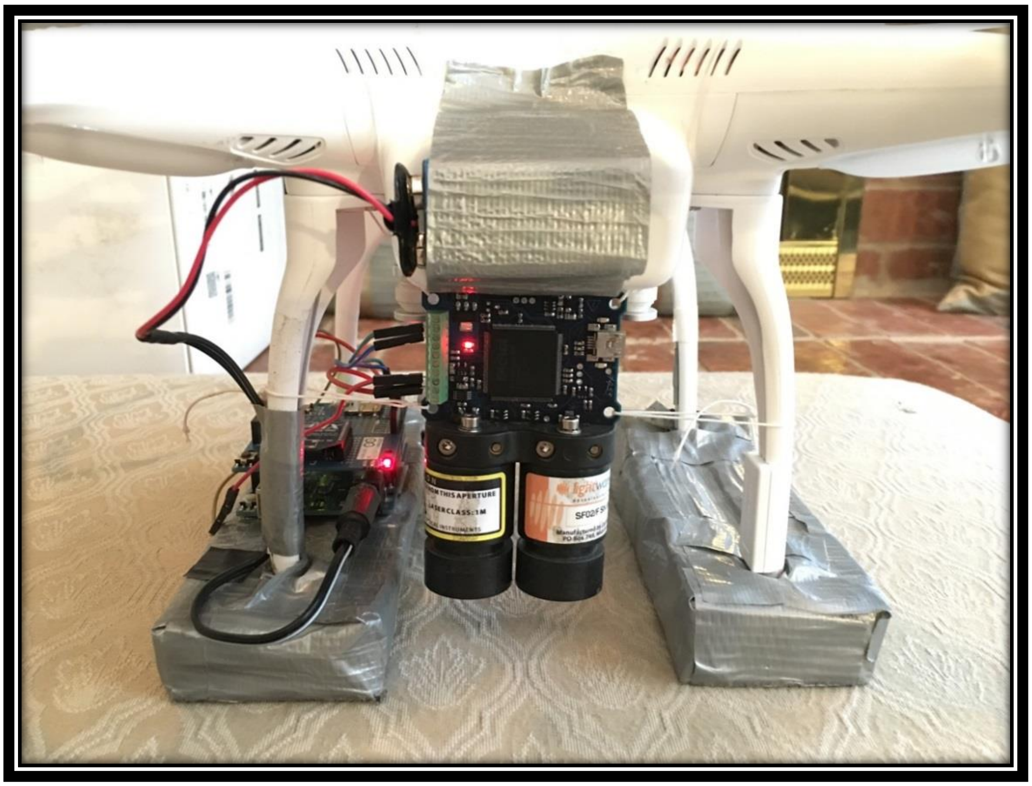

Figure 13. Drone Setup Block

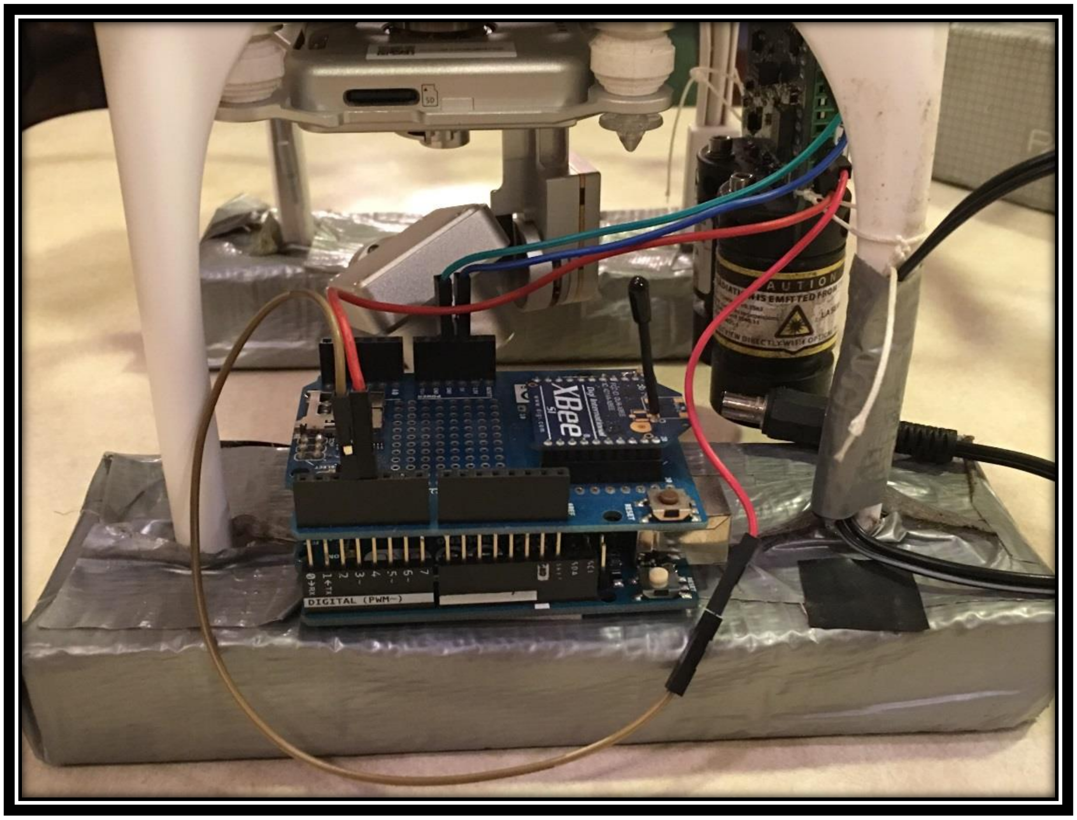

Figure 14. Onboard Arduino Laser Setup

It can be seen from figure 13 and figure 14 that the laser rangefinder sensor is placed in the center of mass of the drone for maintaining the balance while flying. At first, when I installed the laser sensor the laser pointing ends were too close to the ground with just the pre-installed landing gear. Therefore, I had to design a Styrofoam based landing gear which lifted up the drone body and the laser sensor also provided the cushion for landing. It can also be seen that the Arduino board with XBee shield is placed on one of the landing legs, which is connected to the laser rangefinder sensor. The 9V rechargeable battery is used to provide power to the Arduino-laser rangefinder setup.

The Phantom 2 Vision+ is typically controlled manually by using a flight remote controller. The remote controller includes a range extender that creates a Wi-Fi network. This Wi-Fi network channel is used as the communication channel that controls the drone's flight from a smartphone. The same channel is used for transmitting and receiving the camera's video feed.

As mentioned in chapter 4, the DJI software development kit provides limited functions for controlling the drone. Regrettably, this SDK does not allow the programming of the on board flight controller installed on the Phantom 2 Vision+ drone. Consequently, the laser rangefinder sensor cannot be utilized by the onboard microcontroller. Thus, I decided to receive the laser sensor data using an Arduino Uno board and transmitting it to the android application that will be discussed later.

The laser rangefinder sensor is capable of calculating 12 readings per second. In this thesis project, the Arduino is programmed to only receive 2 height readings per second from the laser rangefinder in an effort to maximize efficiency of data communication flow through the Arduino Uno board. The Arduino Uno board is installed with a wireless shield and an XBee transceiver. The laser data received from the laser rangefinder is transmitted from the drone via XBee to the XBee transceiver which is installed on the ground receiver.

5.2.2 Ground Transceiver Block

The ground transceiver block functions primarily as a communications routing station and as such is a relatively simple configuration as shown in figure 15. This setup includes an Arduino Uno board installed with a wireless shield that holds a XBee transceiver, a rechargeable Anker portable battery pack [31], and a Bluetooth module. The ground station Arduino is programmed to receive the laser data transmitted from the drone's onboard Arduino board via Xbee and transmits this received laser data through Bluetooth, which is received via Bluetooth by the android application.

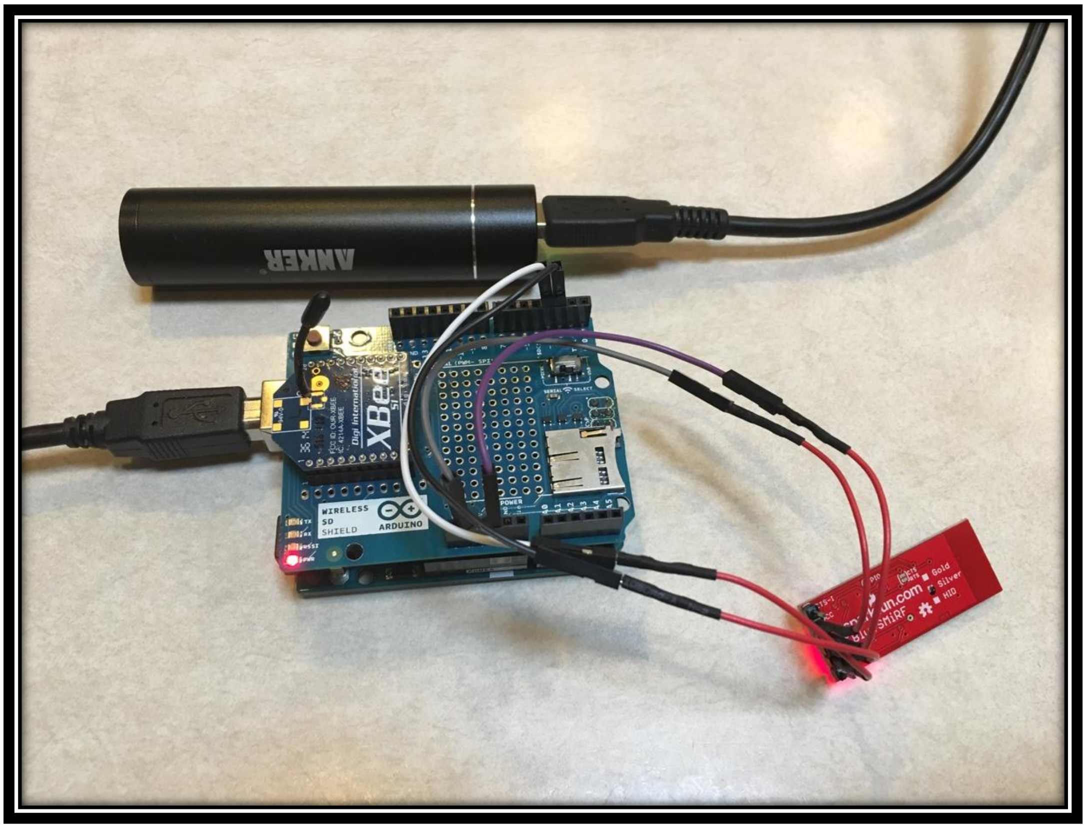

Figure 15. Ground Station Transceiver

Initially, this ground station transceiver was conceived as including an on-the-go USB cable instead of the Bluetooth module. In this configuration, the female USB port would connect to the Arduino's USB port and the mini-USB port would connect to the phone's mini-USB port. Although this would have reduced the potential for wireless communications to be overloaded, it created insurmountable Android programming problems. The Android library could not support the on-the-go USB cable, but could support Bluetooth communications. This mandated the use of the configuration indicated in the block diagram.

5.2.3 Ground Smart Phone Control Block

Autonomous control of the DJI drone is possible through the use of a smart phone application designed to both receive drone data and send data to the drone. Using the configuration indicated in the block diagrams above, it is possible to have the drone perform autonomous tasks that are initiated from a smart phone. The application developed for this thesis provides a variety of used controls while receiving height and video data.

5.2.3.1 Execution Process

The android application is the main controller unit in this project. As such, it performs a variety of operations. The android application layout is shown in the following figure.

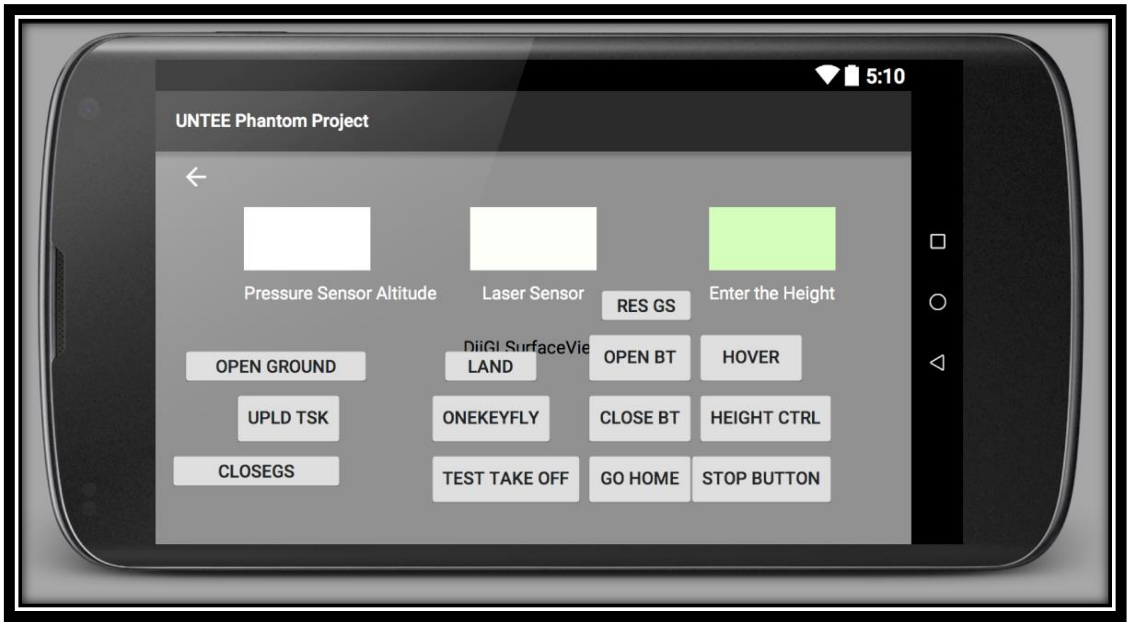

Figure 16. Android Application "UNT Phantom Project"

The android application establishes the connection with the drone using Wi-Fi. When the phone is connected to the drone, the application will alert the user with the following message on the screen “Drone is Connected.” Similarly, when the drone is not connected to the phone it will display “Drone is Not Connected.” When the drone is connected to the phone, it immediately starts transmitting the video from the camera. The android application is programmed to display the video in the background of the application window.

The android application on the phone also receives laser height measurements from the ground station via Bluetooth. It displays the received laser data in a text view on the application. Also, for the sake of understanding how the laser sensor attitude measurement varies with the pressure sensor altitude measurement, the application also has the text view that displays this data.

The following figure shows the demonstration of the android application while working. As it is shown in figure 17, the application is displaying real time video in the background. It shows the laser data and the pressure sensor height measurement and includes several flight control buttons. The application also has an editable text field in which the user can provide a specific target height for the drone to reach.

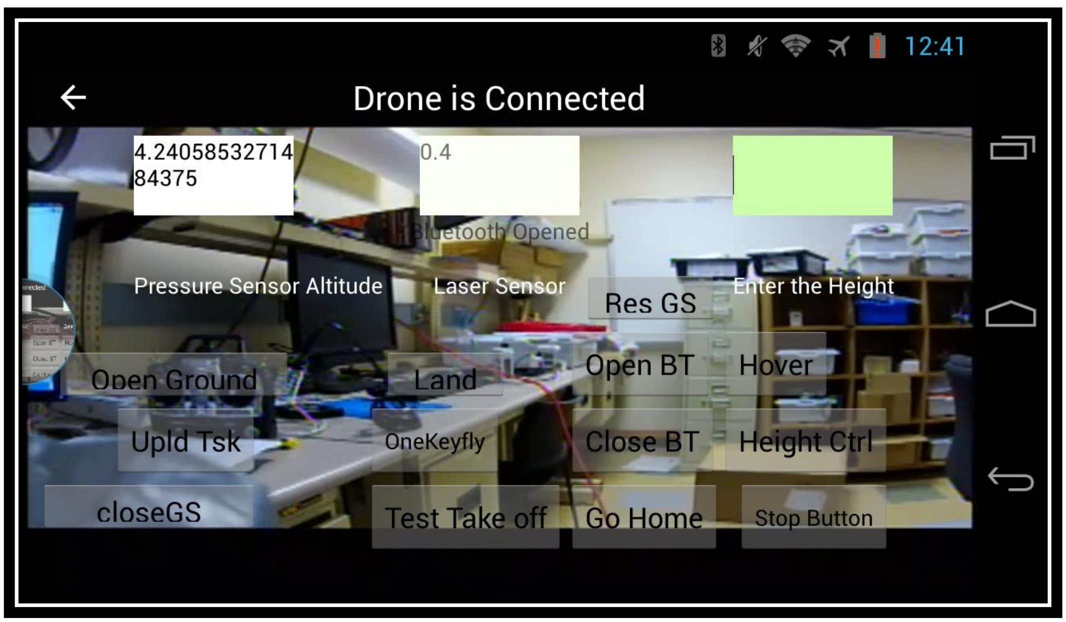

Figure 17. UNT Phantom Project Application

5.2.3.2 Flight Control

The flight control commands include controls that are used for drone navigation from the smartphone. The application includes flight controls for take-off, landing, emergency go home, throttle up, throttle down, and throttle stop. Of these controls, the throttle commands play a very important role in the development of the height algorithm that will be discussed later in this chapter. There are specific steps involved in the proper execution of the flight control. In order to start the execution of commands using the smart phone the following operations need to be performed.

- The user should press the open ground station button to create the control channel between the phone and the drone, thus enabling the transmission of commands. Once the channel is created, the Upload task button needs to be pressed in order to send commands from the phone to the drone. When the user is done sending commands, to safely close the connection between the phone and the drone, the user needs to press the close ground station button.

- After the connection is opened between the phone and the drone, the user should press the take-off button for the drone to fly. The take-off command is pre-programmed to ascend to 5-meters. After the drone performs the successful take-off, the user needs to select the resume ground station task button to start sending further control commands to the drone.

- The Software Development Kit does not really provide a landing function for the Phantom 2 Vision+, though it does provide landing functionality to other DJI drones. To overcome this challenge, I used a delay function in the throttle down command that executes a short throttle down followed by throttle stop (hover). More specifically, the drone will throttle down for 500 milliseconds and then throttle stop for 2000 milliseconds. The throttle down throttle stop durations were determined to be optimum values as a result of extensive delay variable tests. This function provides a functional and effective landing function.

The go home command is used to safely bring the drone back to its initial location and land. When the drone is switched on, it locks the GPS longitude and latitude of that place as the home location. When the application go home button is pressed the drone returns and lands at the take-off position. In the android application this command has been used as the emergency maneuvering protocol that can help avoid a crash in the case of a command failure. The state of the drone can be also received on the phone. In this android application, the pressure sensor height value received is displayed on the text view of the application window.

5.2.3.3 Height Algorithm

The height algorithm developed is the core of this thesis project. In order for the drone to perform a stable maneuver around the truck, the drone has to maintain its height accurately. Therefore, height stabilization is a critical fundamental feature of the android application.

Height control is performed using the up and down throttle command on the drone. As mentioned previously, the throttle control provided by the SDK is very limited. The throttle control on the drone using the SDK is not variable, providing no flexible means of changing the rotor speed. There are only three modes of throttle control: throttle up is executed when the value “1” is sent to the drone from the smartphone, throttle down is executed when the value “2” is sent to the drone from the smartphone, and throttle stop (hover) is executed when the value “0” is sent to the drone from the smartphone. This limitation provides challenges to autonomously control the drone height with great precision.

5.2.3.3.1 Up and Down Height Control

The height algorithm was developed in three major stages. Initially, the algorithm contained a relatively straightforward if-else loop during which the drone would ascend until the desired height range was sensed. For example, setting the height range from 10.0 to 10.5 meters, the drone would execute throttle up, sense the height, and when the height exceeded 10.0 meters throttle stop would be executed. If the drone height exceeded 10.5 meters then throttle down would be initiated and throttle stop again executed when the height sensor indicated the drone descended below 10.5 meters. However, the communications delay that resulted from transmitting the height information to the phone, analyzing the height, and then sending a throttle stop command back to the drone fluctuated between 0-4 seconds. Consequently, the delays created significant height inaccuracy and often the drone would oscillate between heights below and above the desired hover range.

5.2.3.3.2 Up and Down Height Control with constant delay

In an effort to address this problem resulting from the communications delay times and the lack of a variable throttle control, a timer delay was introduced into the height algorithm. Introducing a constant sample delay to all three throttle functions provided much greater stability, accuracy and predictability in the height algorithm. Nonetheless, height deviations were still present after the implementation of the delay.

5.2.3.3.3 Up and Down Height Control with a P-controller

The final stage of height algorithm development included adding a p-controller to the delay. For this algorithm a p-controller was added based on the difference between the current and desired height. If the height difference was large the delay produced from the p-controller will be a small. Consequently, the drone would ascend or descend faster with little or no delay. If the difference between the height is very small the p-controllers produced a longer sample delay, thus prompting the drone to ascend or descend slowly until it reaches the desired height with great accuracy. A more detailed explanation of the algorithm ensues.

5.2.4 Project Prototype

The following figures 18 and 19 show the prototype project configuration. Here, the laser rangefinder is seen installed under the drone's battery facing downward to record and measure the altitude. This laser rangefinder is connected to an on board Arduino-XBee setup that is mounted on the landing gear. This records the laser data and transmits it to the ground receiving station (front right), which then transmits the laser data via Bluetooth to the smartphone for performing flight control using the height algorithm.

Figure 18. Project Prototype Setup

Figure 19. Project Prototype Flying Setup

CHAPTER 6

FIELD TEST AND SIMULATION RESULTS

6.1 Introduction

A variety of tests and simulations were performed in the development and successful completion of this study. These tests and simulations resulted in the successful development of a height control algorithm, real-time video transmission, autonomous flight control, and flight status reporting. As this research study is performed on an UAV, extensive field tests have been performed in order to verify the implementation. The field tests and simulation will focus on the height control algorithm for the drone.

6.2 Android Application Code

Height control serves a primary function that is required for any autonomous drone control. As a result, the development of the height control algorithm plays a fundamental role in this research. The development of the android application utilized a variety of height control operations for respective height control tasks. As indicated in chapter 4, the SDK includes some limited functions that enable stable maneuvering of the drone from an android application.

The functions incorporated in the application development are seen in the following code examples provided by demo DJI SDK application [32]. The first function that is initiated within the application is a check home location test method. This checks whether or not the drone has locked and saved the home location before executing any flight control tasks as shown in the following figure 20.

```
private boolean checkGetHomePoint(){
    if(!getHomePiontFlag){
        setResultToToast("Home Point Location is not locked");
    }
    return getHomePiontFlag;
}
```

Figure 20. Check Home Location Function

In the android application, as soon as the phone is connected with the drone via Wi-Fi the video transmission from the drone will be displayed on the smartphone screen. The code for this function is shown in figure 21.

```
mDjiGLSurfaceView.start();
mReceivedVideoDataCallBack = new DJIReceivedVideoDataCallBack(){

    @Override
    public void onResult(byte[] videoBuffer, int size)
    {
        // TODO Auto-generated method stub
        mDjiGLSurfaceView.setDataToDecoder(videoBuffer, size);
    }

};

DJIDrone.getDjiCamera().setReceivedVideoDataCallBack(mReceivedVideoDataCallBack);
```

Figure 21. Video Transmission Android Code

The code for throttle up, down, and stop are initiated by a call back function which is performed by the DJI Ground Station task as shown in figure 22, 23, and 24. The set throttle codes are 0-Stop, 1-Throttle Up, and 2-Throttle Down. The go home function method is explained in figure 25.

```
void ThrottleUp(){

        DJIDrone.getDjiGroundStation().setAircraftThrottle(1, new DJIGroundStationExecutCallBack() {

            @Override
            public void onResult(DJIGroundStationTypeDef.GroundStationResult result) {
                Log.v(TAG,"Throttle Up Function Initiated");
            }

        });

}
```

Figure 22. Throttle Up Android Function

```
void ThrottleDown() {

        DJIDrone.getDjiGroundStation().setAircraftThrottle(2, new DJIGroundStationExecutCallBack() {

            @Override
            public void onResult(DJIGroundStationTypeDef.GroundStationResult result) {
                Log.v(TAG,"Throttle Down Function Initiated");

            }

        });

}
```

Figure 23. Throttle Down Android Function

```
void ThrottleStop(){

        DJIDrone.getDjiGroundStation().setAircraftThrottle(0, new DJIGroundStationExecutCallBack() {

            @Override
            public void onResult(DJIGroundStationTypeDef.GroundStationResult result) {
                Log.v(TAG,"Throttle Stop Initiated");
            }

        });

}
```

Figure 24. Throttle Stop Android Function

```
void goHome(){
    if(!checkGetHomePoint()) return;
    DJIDrone.getDjiGroundStation().goHome(new DJIGroundStationGoHomeCallBack() {

        @Override
        public void onResult(DJIGroundStationTypeDef.GroundStationGoHomeResult result) {
            Log.v(TAG,"Fail Safe Initiated");
        }

    });

}
```

Figure 25. Go Home Fail Safe Command

6.3 Field Test and Simulation Results

The use of the limited functions of the software development kit provide the fundamental building blocks utilized in this application. The absence of variable throttle control required the implementation of a number of field tests and simulations in an effort to obtain the most precise height control possible. The following code examples provide an overview of height control commands used within the various modes of height operation within the application.

6.3.1 Up and Down Height Control Field and Simulation Results

In this stage of application development, the android application was programmed to perform a height stabilization using an if-else statement to have the drone descend from 5 meters to a height between 3 and 3.5 meters as shown in figure 26. When this was tested out in the field, the drone throttled down as expected but fell below the specified height. When falling below the desired height it would again ascend, but would often ascend above the desired height, thus fluctuating between 0.5 meters to 4 meters. This can be seen in Figure 27. Consequently, using this simple if-else loop, the drone was not stopping within the desired height range.

```
public void HeightControl1(final String Data) {
    if(!checkGetHomePoint()) return;
    float height = Float.valueOf(Data);

    if (height > 3.5) {
        ThrottleDown();
    }
    else if(height<3){
        ThrottleUp();
    }
    else {
        ThrottleStop();
    }

}
```

Figure 26. Android Height Stabilization Statement

Figure 26 shows the code for the general height control statement, this method first checks if the drone has locked its GPS home location for emergency reasons, then it receives the laser rangefinder data through Bluetooth and performs the height check. It can be seen that this method checks if the height is greater than 3.5 meters, it initiates the throttle down function. If the height is less than 3 meters it initiates the throttle up function. If the height is between 3 and 3.5 meters it initiates the throttle stop function.

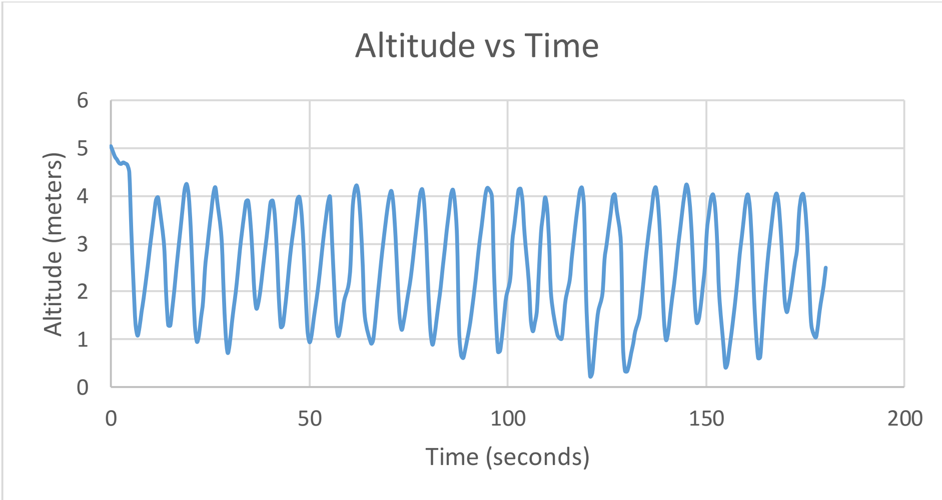

Figure 27. Fluctuating Flight Result

After performing multiple field tests, I was able to determine the cause for this fluctuation of height. The first reason was the delay between the phone command to the command execution of the drone. After recording the height measurements from the laser sensor, and studying them thoroughly I was able to determine that the delay between the phone command to the drone execution carried within a range of 0 – 4 seconds. Another reason for the overshoot in the height was that the drone was going up too fast when it was throttling up and was coming down too fast when it was throttling down. As a result, it gained too much momentum to be able to stop within the desired height range, resulting in the fluctuations seen in Figure 27.

This was a very crucial problem in order to move ahead for the development of the height control algorithm for this thesis. In order to come up with a solution to this problem, I decided to simulate the entire drone operation in Matlab to better understand and determine the solution. To understand the drone behavior for simulation I performed various flight tests to measure the drone's response to the throttle stop to see when the drone was actually stopping after receiving the command. Initiating the command to throttle up at 2-meters height, the test data shows that the drone's average throttle up velocity was 0.68-meters/second. The throttle stop command was initiated when the drone was at 10-meters height and the drone stopped at 14.5 meters. I plotted the altitude verses time and sampling speed, to better understand the drone behavior. This is shown in figure 28 and 29. In another flight test, throttle down was initiated from a starting height of 15 meters. The drone dropped to a height of 5 meters at an average velocity of -1.05 meter/second.

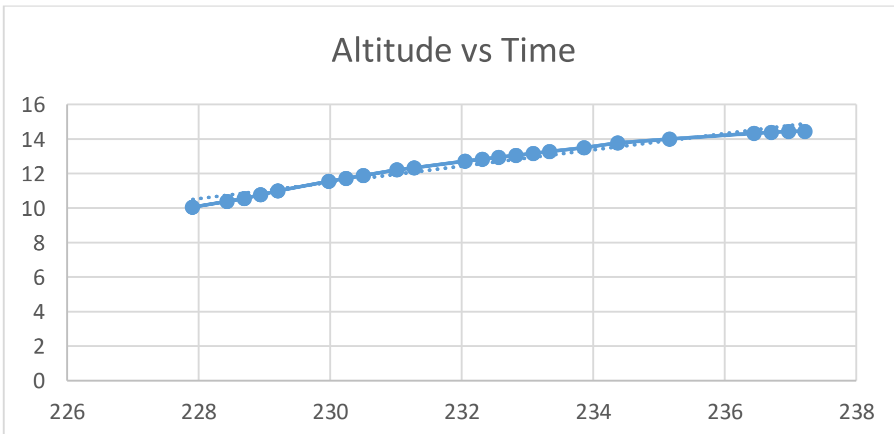

Figure 28. Altitude Vs Time Plot

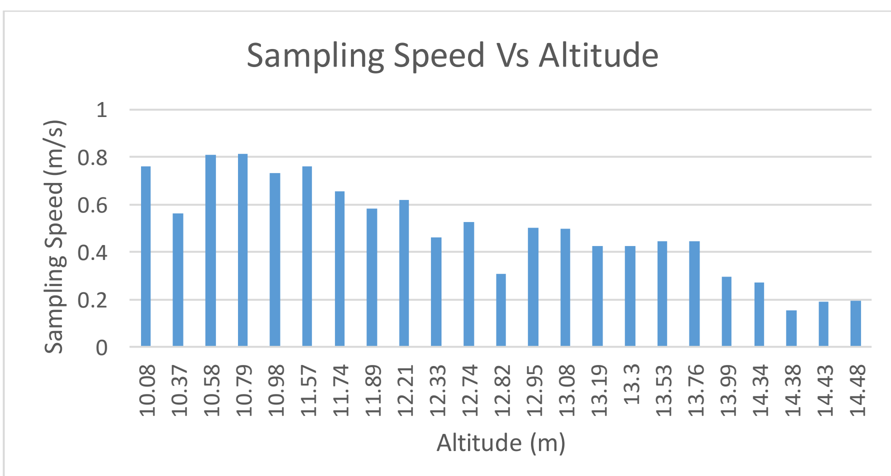

Figure 29. Sampling Speed Vs Altitude

A more accurate height algorithm must compensate for both the communications delay and the surpassing of the target height due to momentum. The data obtained from each of the flight tests was helpful in the development of the simulation algorithm that could help determine the best settings to ensure more accurate height control. The simulation algorithm was developed in Matlab to emulate a real time flight scenario. The data obtained from the simulation and the flight test was plotted together for better understanding as shown in figure 30.

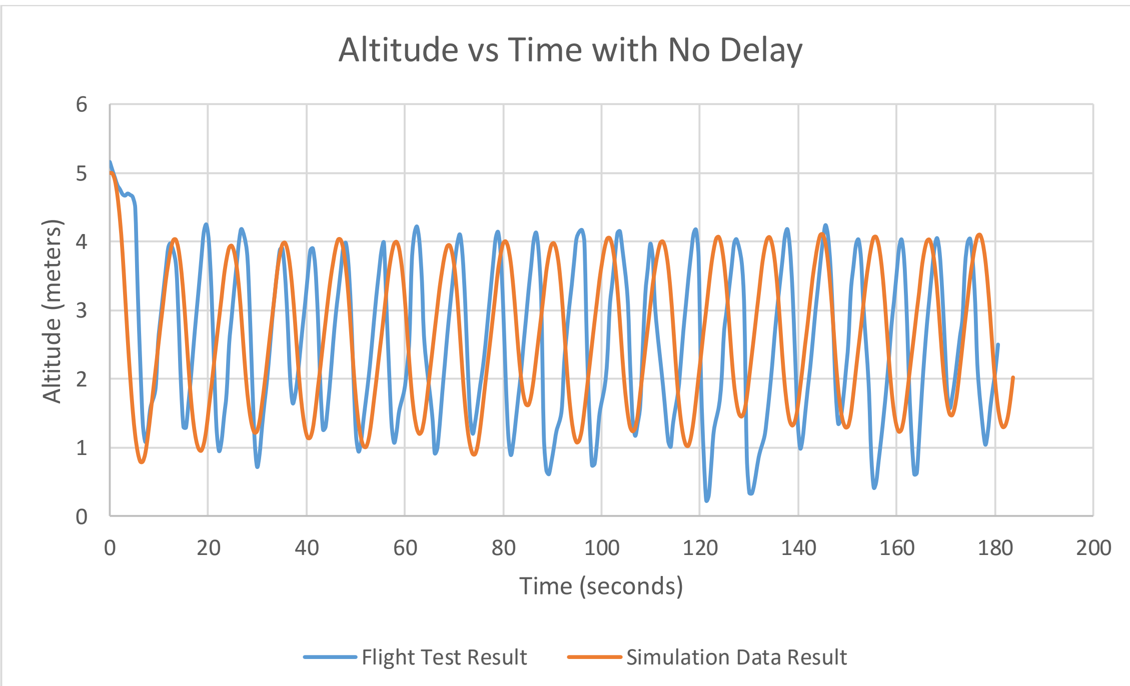

Figure 30. Altitude vs Sample with No Delay

Given these momentum and communications delay issues, I determined that the best means of addressing the problem was to introduce a delay in every throttle up, down, and stop command execution. Introducing this delay into the height control algorithm ensures that the drone will have ample time to execute the height control with much greater accuracy.

6.3.2 Up and Down Height Control with Delay Simulation and Flight Test Results

Introducing delays on the various throttle commands provides much greater height control accuracy in this algorithm, addressing the problems presented by momentum and systems communications delays. For example, to better control autonomous drone ascent one can execute a throttle up command for one second followed by a 3 second throttle stop command. This will effectively slow the ascent, minimizing the momentum causing the drone to surpass the desired height. When using the android application to control the drone, the drone will execute any throttle command until another one is called. Hence, the system delay must be taken into consideration when creating an effective height control algorithm. The simulation can provide data that will help determine the optimum delay times to choose for use within the algorithm.

Therefore, I decided to fix the throttle momentum issue which was causing the fluctuation, by introducing delay for few seconds on every throttle command, for example, if we want the drone to go up, the throttle up command will be executed for example, 1 second and throttle stop command will be executed for 3 seconds. One more property of the throttling using SDK is that if there is a command a throttle up is executed from the phone, the drone will be going up until throttle stop or throttle down is called, which means that the drone will keep executing the function until another function is called. Before I can implement it on the drone, I wanted to find the overshoot and the rise time from the simulation in order to determine the best delay to choose for execution on the drone.

In order to design the simulation to find the overshoot and rise time, a random delay model was needed. In this delay model the drone will operate during the operator sampling time which is named "*n*" and the system or sample delay time is named as "*m*." In this simulation *m* is assumed to be 8 samples, or 2 seconds, and n is assumed to be a random variable between 1 and 45 samples. The target height in the simulation was set as a range from 10 to 10.5 meters.

$$\text{Ratio of the sampling period or operation delay "}L\text{"} = \frac{n}{n+m} \qquad (1)$$

Before, I explain further details about the simulation, I would like to explain the overshoot and the rise time. Overshoot is defined as the percentage calculation of the maximum number of times the system fluctuates around the final value. In other words, it is the difference between the fluctuating value and the step value divided by the step value. Rise time is defined as the amount of time taken by the system to ascend from 10% of the final steady state value to the 90% of the final steady state value. Sometimes in the simulation there is no final value. In that case the final value is assumed to be 10.25. The curve obtained after the simulation in Matlab is shown in figure 24.

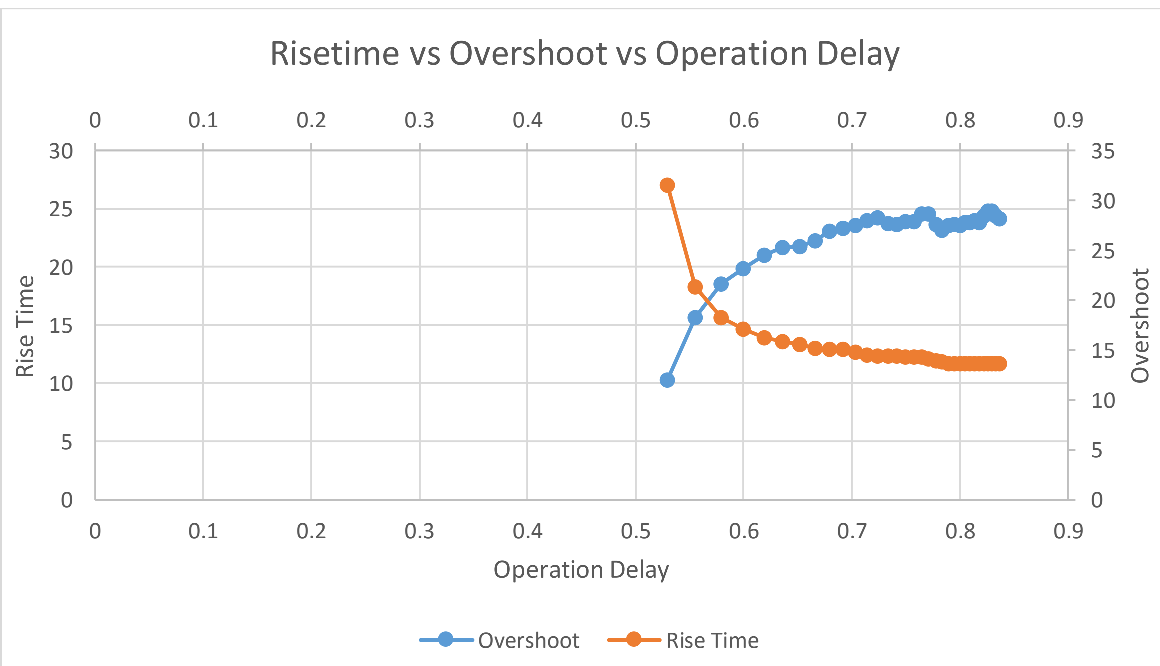

Figure 31. Overshoot & Rise time vs Operation delay

The overshoot in figure 24 is not a linear line because there was no final value and the final value was assumed to be 10.25. This simulation was run for 500 samples using the Monte Carlo simulation method. In the Monte Carlo simulation the same system function is run several times. This method is useful in complex modeling because it provides consensus at the end. The random delay model used in the simulation is explained as follows

$$X[t] = V[t - K] \qquad (2)$$

$$Y[K + 1] = Y[K] + T * X[K] \qquad (3)$$

SD stands for system random which is a value between 8 and 16 samples. Also, "K" is the sample time at which the drone receives the command from the phone.

$$K = K' + SD \qquad (4)$$

The velocity for throttle up and throttle down is studied from the sample data from the flight test and is simulated. The velocity in the real flight is variable when the drone is throttling up or throttling down as seen in figure 32 and 33.

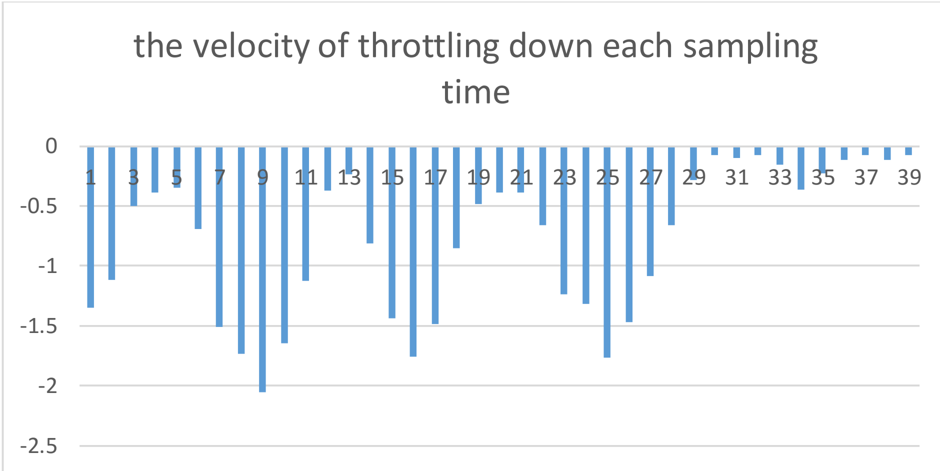

Figure 32. Throttle Down Variable Velocity

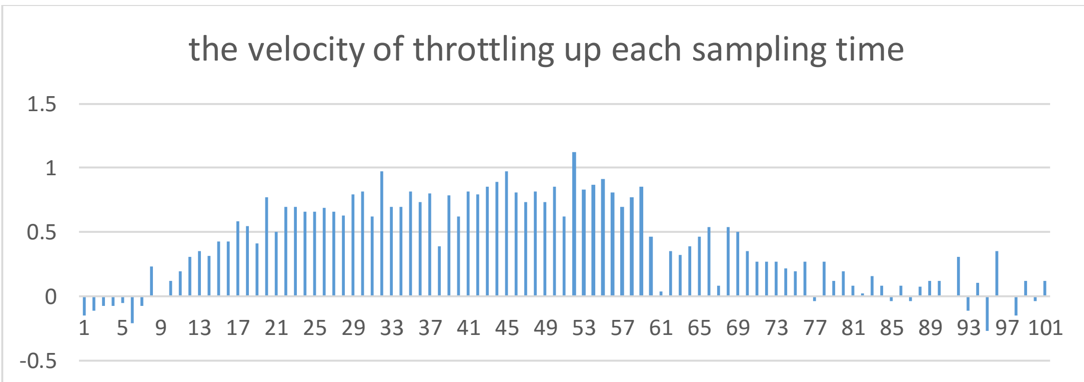

Figure 33. Throttle Up Variable Velocity

The height algorithm on the android application was modified with various delays and the plots from these delays will be discussed in the following section. The assumed height to reach was between 3 meters to 3.5 meters. The height algorithm with delay was initiated from 5-meters height. This delay was introduced in the throttle up/down using a timer function. The throttle up/down and throttle stop android application code can be seen in figures 34-37. In this code, the timer task is used as a function that schedules tasks to execute concurrently. The delay that is mentioned in the timer schedule expresses delay time in milliseconds.

```java
public void task1up(){
    timerup=new Timer();
    final Handler handler = new Handler();
    TimerTask dothrottle = new TimerTask() {
        @Override
        public void run() {

            handler.post(() → {
                    try {

                        DJIDrone.getDjiGroundStation().setAircraftThrottle(1, new DJIGroundStationExecutCallBack() {

                            @Override
                            public void onResult(DJIGroundStationTypeDef.GroundStationResult result) {

                                Log.v(TAG, "Throttle Up Function Initiated");

                            }

                        });
                    } catch (Exception e) {

                    }
            });

        }
    };
    timerup.schedule(dothrottle,0,delay);

}
```

Figure 34. Throttle Up with Delay Function

```java
public void task2stopup(){

    final Handler handlerstop = new Handler();
    timerstopup=new Timer();

    TimerTask dothrottle1 = new TimerTask() {
        @Override
        public void run() {
            handlerstop.post(new Runnable() {
                @Override
                public void run() {
                    try {

                        DJIDrone.getDjiGroundStation().setAircraftThrottle(0, new DJIGroundStationExecutCallBack() {

                            @Override
                            public void onResult(DJIGroundStationTypeDef.GroundStationResult result) {

                                Log.v(TAG, "Throttle Stop Up Function Initiated");

                            }

                        });
                    } catch (Exception e) {

                    }
                }
            });
        }
    };

    timerstopup.schedule(dothrottle1,500, delay);
}
```

Figure 35. Throttle Stop with Delay

```java
public void task3down(){
    final Handler handlerstop = new Handler();
    timerdown = new Timer();
    TimerTask dothrottle1 = new TimerTask() {
        @Override
        public void run() {
          handlerstop.post(new Runnable() {
                @Override
                public void run() {
                    try {

                        DJIDrone.getDjiGroundStation().setAircraftThrottle(2, new DJIGroundStationExecutCallBack() {

                            @Override
                            public void onResult(DJIGroundStationTypeDef.GroundStationResult result) {

                               Log.v(TAG, "Throttle Down Function Initiated");

                            }

                        });
                    } catch (Exception e) {

                    }
                }
            });
        }
    };
    timerdown.schedule(dothrottle1, 0, delay);
}
Handler mHandler = new Handler();
```

Figure 36. Throttle Down with Delay

```java
public void task2stopdown(){
    final Handler handlerstop = new Handler();
    timerstopdown=new Timer();
    TimerTask dothrottle1 = new TimerTask() {
        @Override
        public void run() {
           handlerstop.post(new Runnable() {
                @Override
                public void run() {
                    try {

                        DJIDrone.getDjiGroundStation().setAircraftThrottle(0, new DJIGroundStationExecutCallBack() {

                            @Override
                            public void onResult(DJIGroundStationTypeDef.GroundStationResult result) {

                                Log.v(TAG, "Throttle Stop Function Initiated");

                            }

                        });
                    } catch (Exception e) {

                    }
                }
            });
        }
    };
    timerstopdown.schedule(dothrottle1, 500, delay);
}
```

Figure 37. Throttle Stop with Delay

The height algorithm utilizes an if-else statement that uses the aforementioned timed tasks for executing the delayed throttle functions as shown in figure 38.

```
if(height>3.5){
    if(height<height-- && height>3.5)return;
    else if (timerdown!=null && timerstopdown!=null)return;

    if(timerup!=null) {
        timerup.cancel();
        timerup.purge();
        timerup = null;
    }
    if(timerstopup!=null) {
        timerstopup.cancel();
        timerstopup.purge();
        timerstopup = null;
    }
    task3down();
    task2stopdown();
}else if(height<3){
    if(height>height-- && height<3)return;
    else if(timerup!=null&&timerstopup!=null)return;
    if(timerdown!=null) {
        timerdown.cancel();
        timerdown.purge();
        timerdown = null;
    }
    if(timerstopdown!=null) {
        timerstopdown.cancel();
        timerstopdown.purge();
        timerstopdown = null;
    }
    task1up();
    task2stopup();
}
else{
    ThrottleStop();
    if(timerdown!=null) {
        timerdown.cancel();
        timerdown.purge();
        timerdown = null;
    }
    if(timerstopdown!=null) {
        timerstopdown.cancel();
        timerstopdown.purge();
        timerstopdown = null;
    }
    if(timerup!=null) {
        timerup.cancel();
        timerup.purge();
        timerup = null;
    }
    if(timerstopup!=null) {
        timerstopup.cancel();
        timerstopup.purge();
        timerstopup = null;
    }
}
```

Figure 38. Android Height Algorithm

By flight testing the various delay parameter's responses I could readily determine and implement the best delay parameter for each height algorithm. Flight testing was performed on each parameter three times and the plot obtained showed the same response.

For 500 millisecond throttle up/down and 1 second throttle stop, the height the height control algorithm executed was at 10.5 meters the following response is obtained:

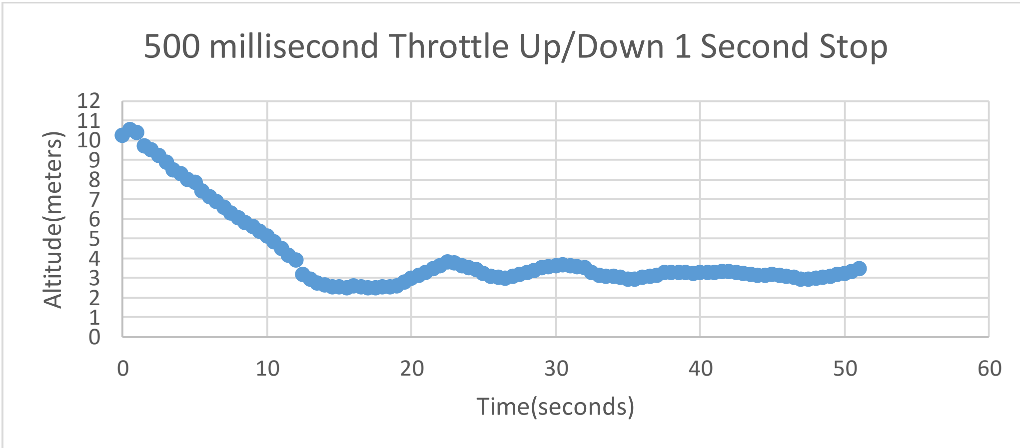

Figure 39. 500 millisecond Throttle Up/Down 1 Second Stop

For 500 millisecond throttle up/down 1.5 second throttle stop, the drone height when the algorithm executed was 9.3 meters the following response is obtained:

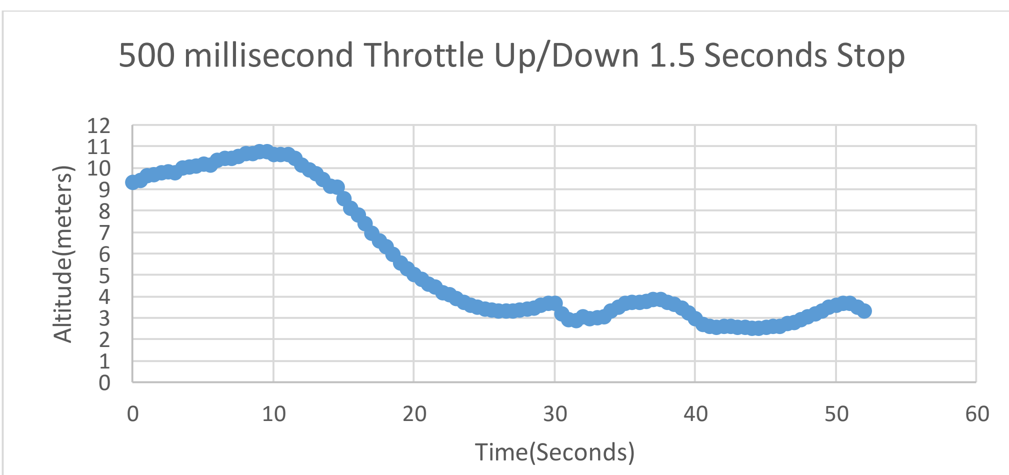

Figure 40. 500 millisecond Throttle Up/Down 1.5 Seconds Stop

For 500 millisecond throttle up/down 2 second throttle stop, the drone height when the algorithm executed was 5.2 meters the following response is obtained:

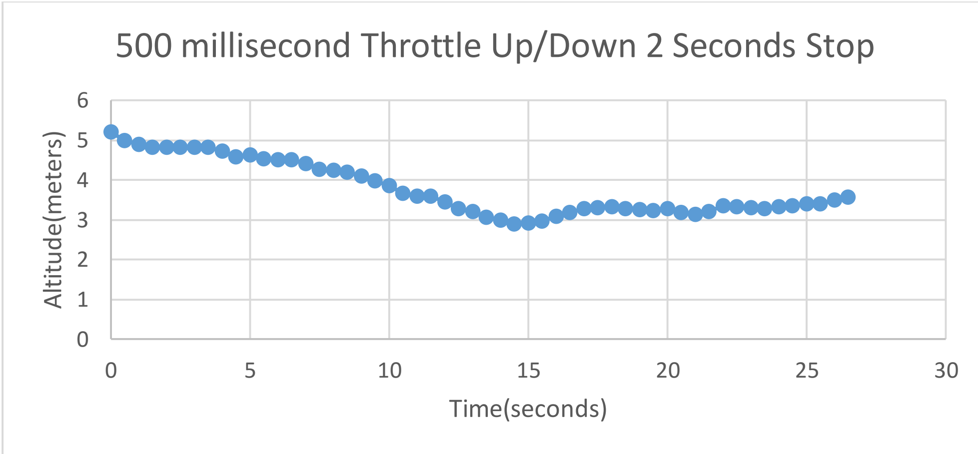

Figure 41. 500 millisecond Throttle Up/Down 2 Seconds Stop

For 1 second throttle up/down 1 second throttle stop, the drone height when the algorithm executed was 10 meters the following response is obtained:

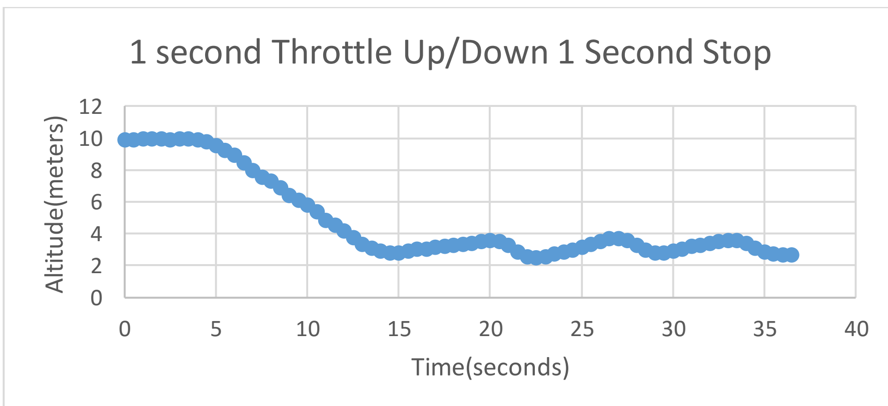

Figure 42. 1 second Throttle Up/Down 1 Second Stop

For 1 second throttle up/down 1.5 second throttle stop, the drone height when the algorithm executed was 10 meters the following response is obtained:

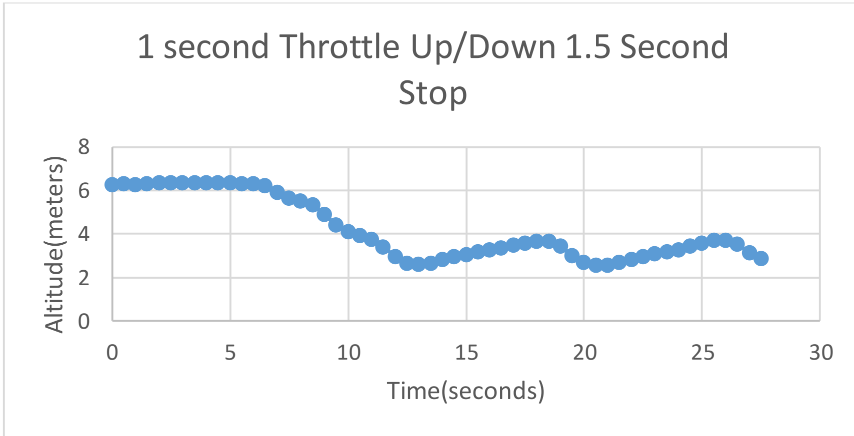

Figure 43. 1 second Throttle Up/Down 1.5 Second Stop

Figures 39-43 show the flight response of various delay parameters. All these tests have been performed three times. The delay parameters range from 0.5 second through 2 seconds for throttle up/down and throttle stop varied from 0.5 through 4 seconds. The response to some of these parameters for different delay combinations have been provided in Appendix 2.

It is also very useful to compare the flight response data with the response from the simulation for the same delay. Through evaluating this data, the delay parameter 500 millisecond throttle up/down and 2 second throttle stop provided the optimal response on the simulation as well as on the flight test which can be seen from the following figure 44.

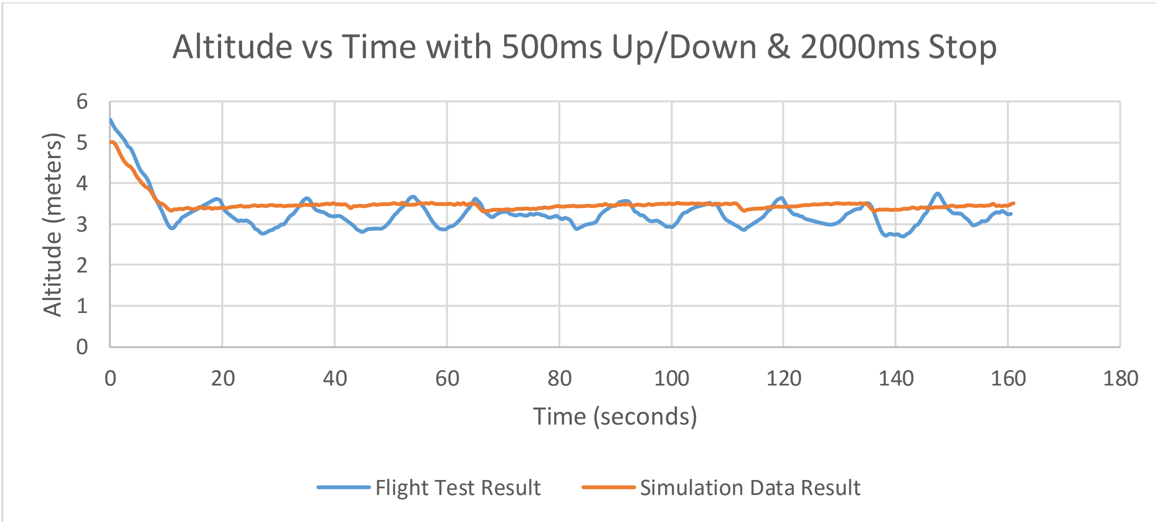

Figure 44. Altitude vs Sample Time with 500 ms Up/Down & 2000 ms Stop Response

6.3.3 Up and Down Height Control with P-Controller Results

In the up/down height control with delay, we have seen that even though it has greatly reduced height fluctuation and provided a stable height maneuvering, there is still some overshoot that can be seen in the response plot. This overshoot can be further reduced by optimizing the algorithm with the installation of a p-controller on the delay for determining a variable delay for optimal performance. The design of a p-controller on the delay can be seen in figure 45.

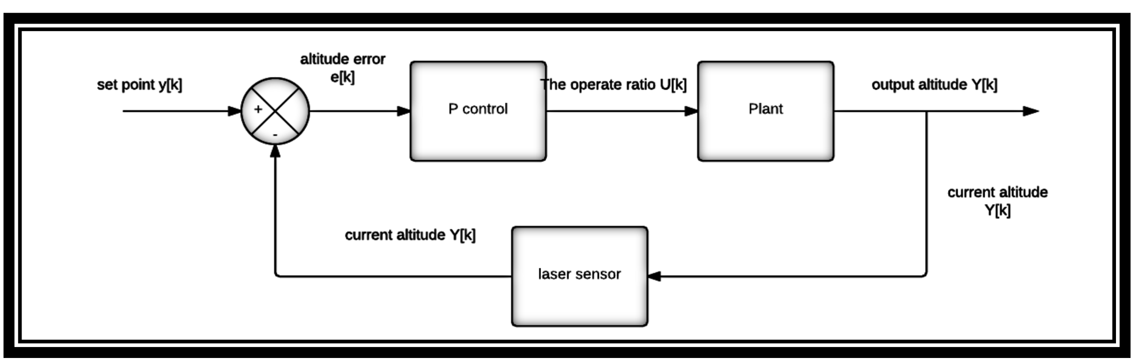

Figure 45. P-Controller for Delay

The operation delay ratio can be written as

$$Operation\ Delay = \frac{Operation\ n}{Operation\ n + delay\ m} \tag{5}$$

The altitude error can be explained as

$$e[k] = Y[k] - y[k] \tag{6}$$

The operation ratio U[k] can be written as

$$U[k] = P * e[k] \tag{7}$$

$$U[k] = P(Y[k] - y[k]) \tag{8}$$

The p-controller on the delay has been implemented on the drone simulation as well as on the android application the code can be shown in the following figures 46 and 47. In the simulation, K represents the sample delay. If K is greater than the operation delay L, then the altitude error is the absolute value of the difference of the middle value of height which is 3.25 meters and the final height value. The p-controller provides a variable "m" which represents the period of time that the drone will throttle. However, if m is greater than 9 it will be assumed to be 9 and if the m value is smaller than 3 it will be assumed as 3. The same logic for the p-controller has been implemented in the android application and the delay obtained from this is fed in to the height control algorithm mentioned in figure 38. The simulation and flight test response has been shown in figure 48 and it can be noted that the performance of the height control has been significantly improved using a p-controller.

```
if K>L
    e=3.25-Y(K);
    e=abs(e);% the altitude error
    n=1;% the flying sampling number
    m=round(abs(1-e*p)/(e*p));% the delay sampling number.
    if m>9
        m=9;
    end
    if m<3
       m=3;
    end
       L=L+n+m;    % the period sampling number
end
```

Figure 46. P-Controller Simulation Code

```
 float height =Float.valueOf(Data);
 heighterror=abs(3.15-height);
 Log.v("Height Error", String.valueOf(heighterror));
 double kp=0.1;
 SystemDelay=round(abs(1-(heighterror * 0.1))/(heighterror*0.1));
 Log.v("System Delay", String.valueOf(SystemDelay));
if (SystemDelay>9){
     SystemDelay=9;
 }
 else if(SystemDelay<3){
     SystemDelay=3;
 }
 delay= SystemDelay*1000;
 Log.v("Real Delay", String.valueOf(delay));
```

Figure 47. Android Application P-Controller Code

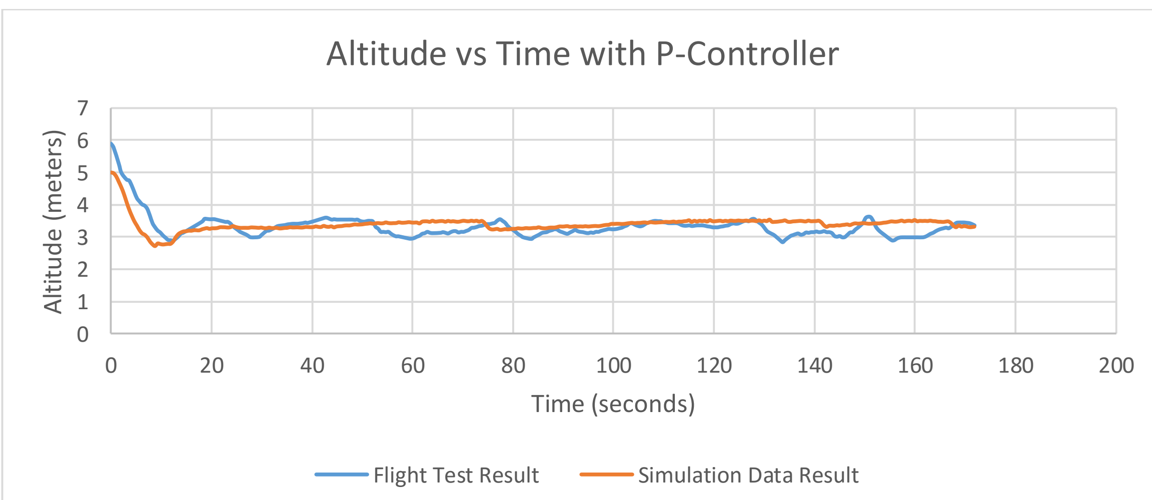

Figure 48. Altitude vs Sample Time Response with P-Controller

6.4 Simulation Pseudo Code

Here the height control algorithm and its model that have been used will be explained in each of the different cases.

6.4.1 Height Control Algorithm with No Delay

A. Simulation Initialization and Execution

a. Declare the drone altitude *Y*, Command, sample time, and Drone Velocity *X* as an array for *K*=1, … ,400 according to the following:

i. Initialize the constants, Velocity Up=0.44, Velocity Down=-0.45, Height limit between low limit=3 and higher limit=3.5 meters. Velocity to Stop while coming up VuStop=-0.1, Velocity to stop when throttling down VdStop=0.1, T=0.5. Set the drone starting altitude Y=5 meters, the initial velocity X=0.

ii. For K= 1 to 400

System Delay between the drone and smartphone is initialized as a normal random distribution between 0 to 4 samples.

Height Check is executed.

Graph is plotted between altitude Y and the sample time.

End

B. Height Check Method

a. Evaluation of the altitude for the drone is performed here and the height check is performed using if-else statement.

b. If-else Statement

i. If Y(K) < lower limit

Command array is initiated as "1" (Throttle up)

End

ii. If (Y(K) >= higher limit) and (Y(K) <= lower limit)

Command array is initiated as "2" (Throttle Stop)

End

iii. If Y(K) > lower limit

Command array is initiated as "3" (Throttle Down)

End

c. Command Array Execution

0= No Command, 1=Throttle Stop, 2=Throttle Up, and 3=Throttle Down

a(K) is the velocity which is determined based on the command

If command = 0, 1, 2, 3

X(K+1) = X(K) + a(K)*T +h

Y(K+1) = Y(K) + X(K) * T + 0.5 * a(K) *(T^2) + h*T

End

d. System Delay Setup

i. There is always a delay between the drone and smartphone. This is assumed to be a normal distribution between (0, 4 Seconds).

ii. Also "h" is a noise which is added in to the simulation because the drone would perform the command given to it in the previous state if the drone doesn't receive a new command.

1. If Y(K) <= higher limit height

   h=noise, a random variable speed.

   End

2. If Y(K) > higher limit height

   h=-abs (noise), a random downward speed

C. Plotting and Storing the Data in Array

a. Sample time can be implemented as

i. Sample Time K+1 = S at K + T

b. Plot the graph between Sample Time and the Altitude Y.

6.4.2 Height Control algorithm with Constant Delay

A. Simulation Initialization and Execution

a. Declare the drone altitude *Y*, Command, sample time, and Drone Velocity *X* as an array for *K*=1, … ,400 according to the following:

i. Constant delay scenario there are operation delay "n" in which the throttle operation is performed, sample delay "m" when the drone performs the stop operation, and the total delay L= m + n.

ii. Initialize the constants, Velocity Up=0.44, Velocity Down=-0.45, Height limit between low limit=3 and higher limit=3.5 meters. Velocity to Stop while coming up VuStop=-0.1, Velocity to stop when throttling down VdStop=0.1, T=0.5. Set the drone starting altitude Y=5 meters, the initial velocity X=0.

iii. For K= 1 to 400

System Delay between the drone and smartphone is initialized as a normal random distribution between 0 to 4 seconds.

Constant Delay Execution with Throttle Command.

Height Check is executed.

Graph is plotted between altitude Y and the sample time.

End

B. Constant Delay Execution with Throttle Command

a. The height check method is executed only when K<= L-m during the command operation period.

b. The stop command is executed during "m".

c. Performed flight tests proves that 500 millisecond throttle up/down and 2000 millisecond throttle stop command proves to be the best analysis. Hence, in the simulation n=1 and m=4 has been used as the delay to get the response.

Section C "Height Check Method" and section D "Plotting and Storing Data in Array" would remain the same from the height control algorithm with no delay.

6.4.3 Height Control algorithm with P-Controller

A. Simulation Initialization and Execution

a. Declare the drone altitude *Y*, Command, sample time, and Drone Velocity *X* as an array for *K*=1, … ,400 according to the following:

i. In this scenario there are variable operation delay "n" in which the throttle operation is performed, sample delay "m" when the drone performs the stop operation, and the total delay L=m+n.

ii. Initialize the constants, Velocity Up=0.44, Velocity Down=-0.45, Height limit between low limit=3 and higher limit=3.5 meters. Velocity to Stop while coming up VuStop=-0.1, Velocity to stop when throttling down VdStop=0.1, T=0.5. Set the drone starting altitude Y=5 meters, the initial velocity X=0.

iii. For K= 1 to 400

System Delay between the drone and smartphone is initialized as a normal random distribution between 0 to 4 seconds.

P-Controller Output Delay Execution with Throttle Command.

Height Check is executed.

Graph is plotted between altitude Y and the sample time.

End

B. P-Controller Delay Execution with Throttle Command

a. The height check method is executed only when K<= L-m during the command operation period.

b. The stop command is executed during "m".

c. The p-controller is applied to create only variable "m", when the drone executes the stop command as explained in the following:

i. If K>L

Error= 3.25- Y(K)

Error Absolute= absolute value of Error

Operation Period is always "1"

m=rounded value ((1-Error*0.1)/(Error*0.1))

if m>9

m=9

else if m<3

m=3

end

L=L + n + m Total Sampling period

End

Section C "Height Check Method" and section D "Plotting and Storing Data in Array" would remain the same from the height control algorithm with no delay.

CHAPTER 7

CONCLUSION AND FUTURE WORK

This research has resulted in the definition of design requirements as well as the creation of a stable framework necessary for the development of a pre-trip inspection project. The research included the determination of the best UAV that could be utilized for such a project. It also included software development to build a height control algorithm implemented in an android application. Another major contribution from this thesis research is to extend the limited capability of the used DJI software development kit for more sophisticated control goals, while not modifying the drone dynamics.

The Phantom 2 Vision+ which was used as a UAV in this project was a stable, robust, and durable drone. Its payload capability facilitated the effective placement of the laser rangefinder for altitude measurement and transmission of laser data to the ground station transceiver. The android application that was developed provided an effective controller unit that is capable of performing crucial operations such as, stable take-off and landing, real time video transmission, gathering real-time flight data, and accurate height control.

The height control algorithm was developed successfully and implemented within an android application. The various flight tests performed verify the efficient implementation of the height control algorithm from the android application. In addition, the entire drone's operation was successfully simulated in Matlab to determine the best delay that could give the lowest overshoot and fastest rise time. This research has resulted in a stable and a durable framework that can be used as the basis for additional work to develop autonomous pre-trip inspections for trucks.

Although this research provides a solid foundation for autonomous UAV control, further development would be beneficial. Specifically, if additional funding is available a long range Bluetooth module could be implemented on the onboard Arduino. This would effectively replace the ground transceiver that was utilized in this research. Next steps in the development of this project would also include the development of vision based maneuvering. The onboard camera can be combined together with the laser rangefinder data to develop a vision positioning system, which could help in flying the drone efficiently in an indoor environment. The onboard camera can also be used for foreign object detection on the trucks by using image processing algorithms. In addition, the android application can be modified further to provide drone system monitoring. As the SDK gives the longitudinal and latitudinal data the drone's location can be plotted on the map on the display of the android application. This will help the user in determining the drone's location when the inspection is carried out.

I believe that the successful implementation of the height control algorithm, the hardware design prototype, and the android application developed for this thesis provides a solid foundation leading toward a more full development of autonomous pre-trip inspections for commercial trucks. I envision that the utilization of drones to assist in these inspections will become a reality in the near future.

BIBLIOGRAPHY

[1] "Renault Kwid concept car picture and information", URL: https://www.renault.co.uk/discover-renault/innovation-and-technology/concept-cars/kwid.html.

[2] Daniel Mellinger & Vijay Kumar, "Minimum Snap Trajectory Generation and Control for Quad-rotors," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2520-2525, 2011.

[3] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, & Vijay Kumar, "Towards a swarm of agile micro quad-rotors," in *Springer Science Business Media New York*, pp. 287-300, 2013.

[4] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-Based State Estimation for Autonomous Rotorcraft MAVs in Complex Environments," in *Proceedings of the IEEE 2013 International Conference on Robotics and Automation - ICRA'13*, Karlsruhe, Germany, pp. 1758-1764, May 2013.

[5] K.Mohta, V. Kumar, K. Daniilidis, "Vision Based Control of a Quadcopter for Perching on Lines," in *Proceedings of the IEEE 2014 International Conference on Robotics and Automation - ICRA'14*, Hong Kong, China, pp. 3130-3134, June 2014.

[6] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quad-rotor," in *Proceedings of Robotics: Science and Systems*, June 2013.

[7] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-Sensor Fusion for Robust Autonomous Flight in Indoor and Outdoor Environments with a Rotorcraft MAV," in *Proceedings of the IEEE 2014 International Conference on Robotics and Automation - ICRA'14*, Hong Kong, China, pp. 4974-4981, June 2014.

[8] “Pre-trip inspection”, URL: https://www.dmv.ca.gov/portal/dmv/?1dmy&urile=wcm:path:/dmv_content_en/dmv/forms/dl/dl65prt1.

[9] “Peterbilt 579 NGP Specifications and Image”, URL: http://www.peterbilt.com/products/on highway/579/.

[10] “AR Drone Specifications and Pictures”, URL: http://ardrone2.parrot.com.

[11] “Matlab and Simulink development kit”, URL: http://www.mathworks.com/matlabcentral/fileexchange/43719-ar-drone-simulink-development-kit-v1-1.

[12] “DJI Flamewheel F450 and F550”, URL: http://www.dji.com/product/flame-wheel-arf.

[13] “DJI Spreading S800 EVO”, URL: http://www.dji.com/product/spreading-wings-s800-evo.

[14] “DJI Phantom 2 Vision+ Specifications”, URL: http://wiki.dji.com/en/index.php/Phantom_2_Vision%2B.

[15] “DJI Phantom 2 Vision+ Images and Features”, URL: http://www.dji.com/product/phantom-2-vision-plus.

[16] “DJI Naza m controller”, URL: https://www.dji.com/product/naza-m-v2.

[17] “Lightware’s SF02 Laser Rangefinder Specification and images”, URL: https://www.parallax.com/product/28043.

[18] “Arduino UNO Board Specification and Images”, URL: https://www.arduino.cc/en/Main/ArduinoBoardUno.

[19] “Arduino Wireless Shield”, URL: https://www.arduino.cc/en/Main/ArduinoWirelessShield.

[20] “XBee Series S2 with Antenna Specifications”, URL: https://www.sparkfun.com/products/10414.

[21] “XCTU Software”, URL: http://www.digi.com/products/xbee-rf-solutions/xctusoftware/xctu.

[22] “BlueSMiRF Silver Specifications and Images”, URL: https://www.sparkfun.com/products/12577.

[23] “DJI Software Development Kit”, URL: www.dev.dji.com.

[24] “DJI SDK Features”, URL: https://developer.dji.com/mobile-sdk/features/.

[25] “DJI SDK Android Procedure”,URL: https://developer.dji.com/mobilesdk/documentation/android/.

[26]“Android Operating System”, URL: https://en.wikipedia.org/wiki/Android_(operating_system).

[27] “Android Studio Features”, URL: http://developer.android.com/tools/studio/index.html.

[28] “Android Tutorial and Project Structure”, URL: http://www.raywenderlich.com/78574/android-tutorial-for-beginners-part-1.

[29] “Android Studio Tutorial”, URL: http://developer.android.com/training/index.html.

[30] “Arduino Development Software”, URL: https://www.arduino.cc/en/Main/Software.

[31] “Anker Portable Battery Pack”, URL: http://www.ianker.com/product/A1104041.

[32] “Android SDK Demo Application Code”, URL: https://developer.dji.com/mobile-sdk/downloads/.

[33] "3DRobotics Open Source Community", URL: http://ardupilot.com.

[34] "University of Pennsylvania Robotics Lab", URL: http://www.kumarrobotics.org.

Appendix 1

The following section explains the various test scenarios that are required for pre-trip inspections by the Federal Government.

Exterior Inspection

Headlights: Both high and low beams must be checked

Turn Signals: Front and back pairs of signals must work.

Hazard Lights: Hazard lights must be operational.

Back-up Lights: Back-up lights must work if transmission is shifted into reverse.

Mirrors: All mirrors must be present, unobstructed, and adjusted to the person who will be driving the vehicle.

Windshield Wipers: Windshield wipers must work at all settings. Wiper fluid pump should also be tested.

Windows: Windows must be secure and in good operating condition.

Tires: All tires should be visibly inspected for inflation and tread wear.

Body Damage: Any kinds of body damage should be reported.

Lift/Ramp Operation: All lifts must be checked before the vehicle is put into service, whether or not the lifts are intended to be used.

Appendix 2

Flight Data Test Response Height Algorithm with delay

For 500 millisecond throttle up/down 2.5 second throttle stop, the drone height when the algorithm executed was 7 meters the following response is obtained:

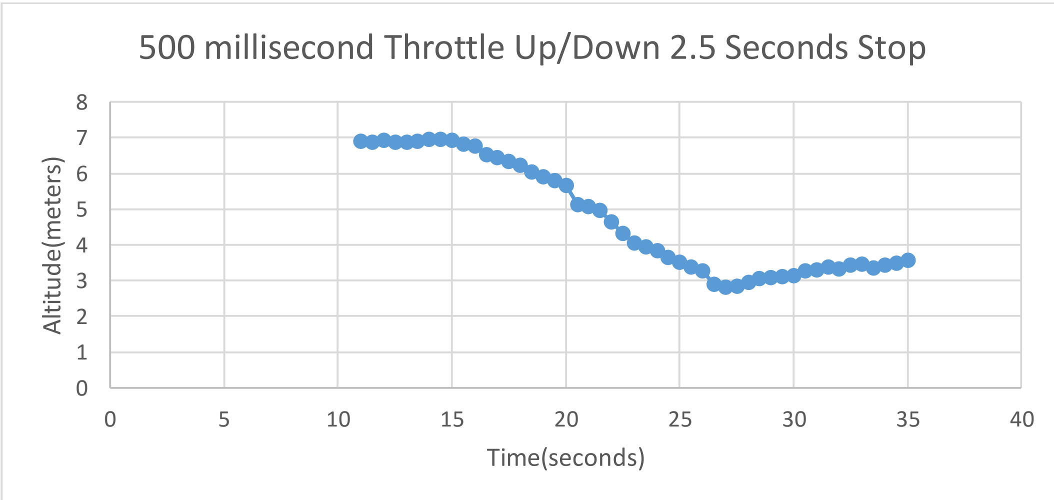

Figure 49. 500 ms Throttle Up/Down 2.5s Stop

For 500 millisecond throttle up/down 3 second throttle stop, the drone height when the algorithm executed was 6.5 meters the following response is obtained:

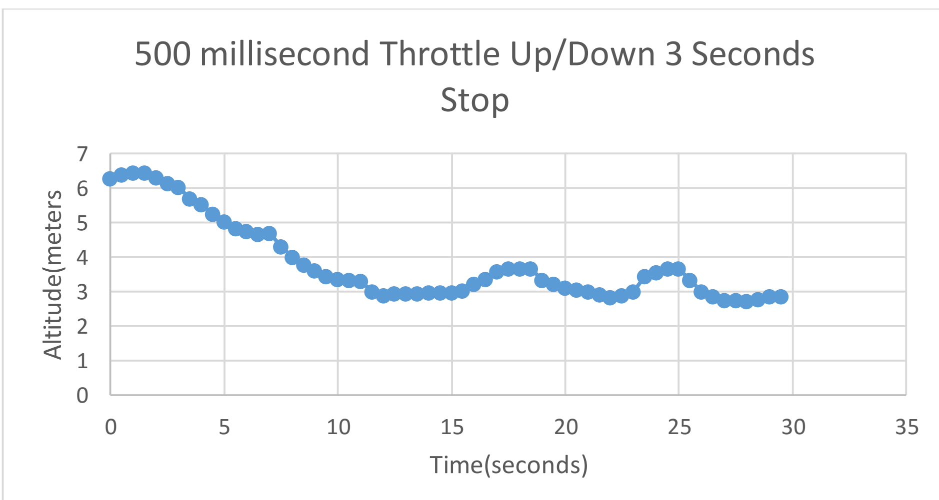

Figure 50. 500 ms Throttle Up/Down and 3s Stop

For 500 millisecond throttle up/down 3.5 second throttle stop, the drone height when the algorithm executed was 11.5 meters the following response is obtained:

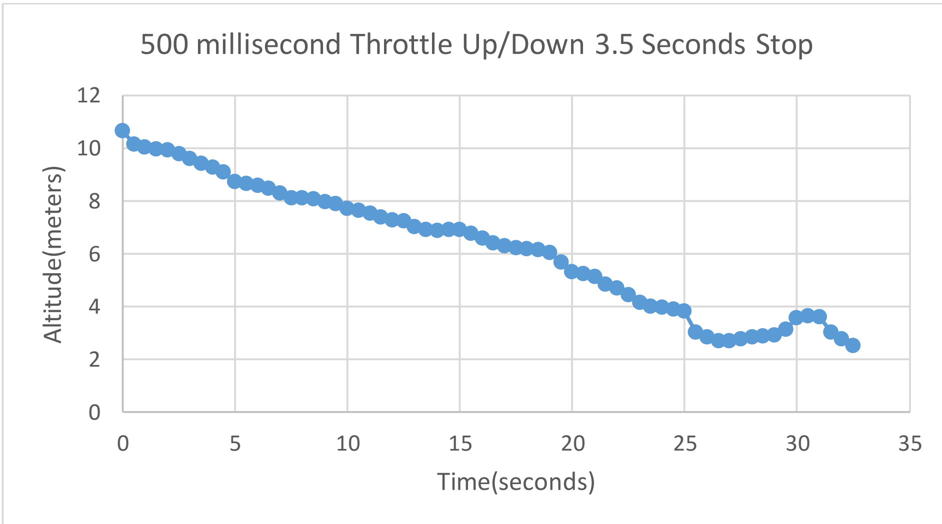

Figure 51. 500 ms Throttle Up/Down 3.5s Stop

For 500 millisecond throttle up/down 4 second throttle stop, the drone height when the algorithm executed was 9.5 meters the following response is obtained:

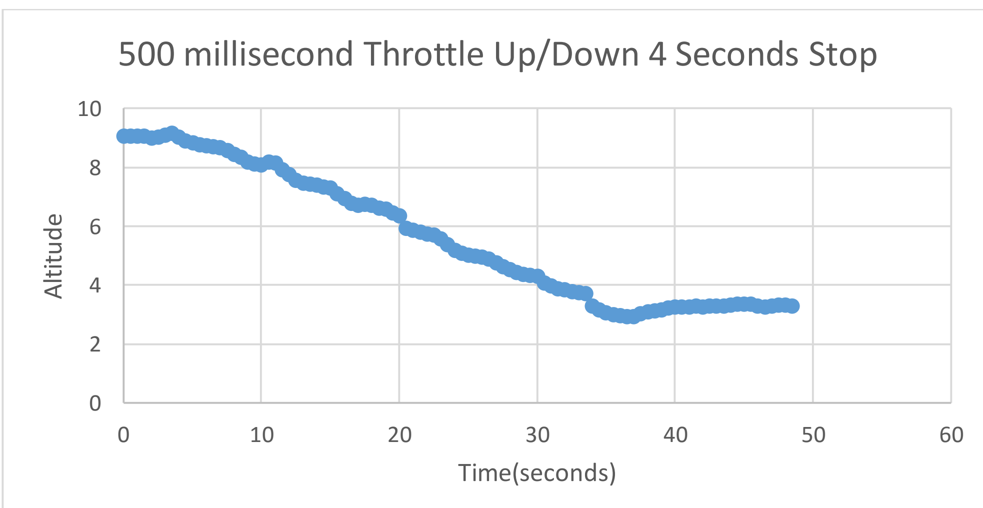

Figure 52. 500 ms Throttle Up/Down 4s Stop

For 500 millisecond throttle up/down 4.5 second throttle stop, the drone height when the algorithm executed was 8 meters the following response is obtained:

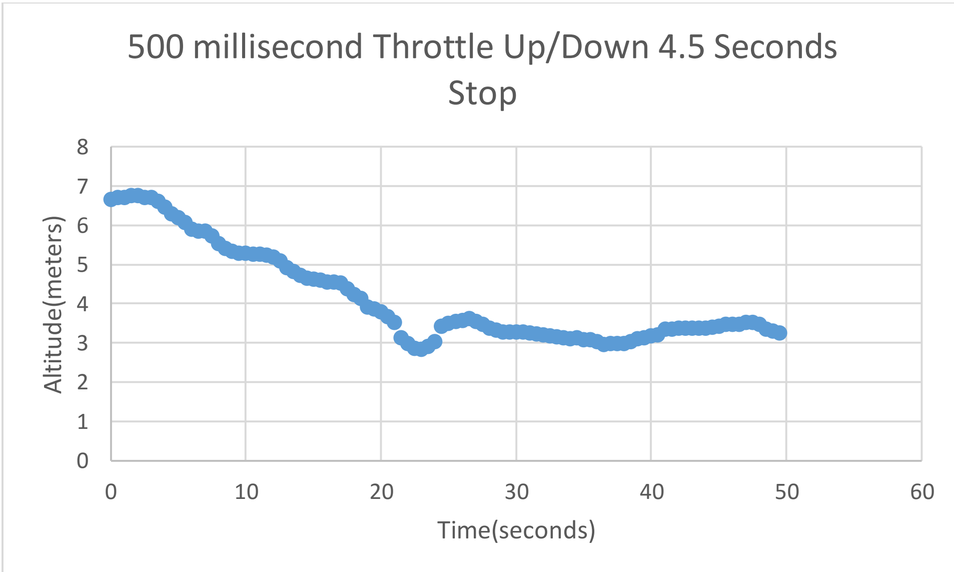

Figure 53. 500ms Throttle Up/Down 4.5s Stop

For 1 second throttle up/down 2 seconds throttle stop, the drone height when the algorithm executed was 7.5 meters the following response is obtained:

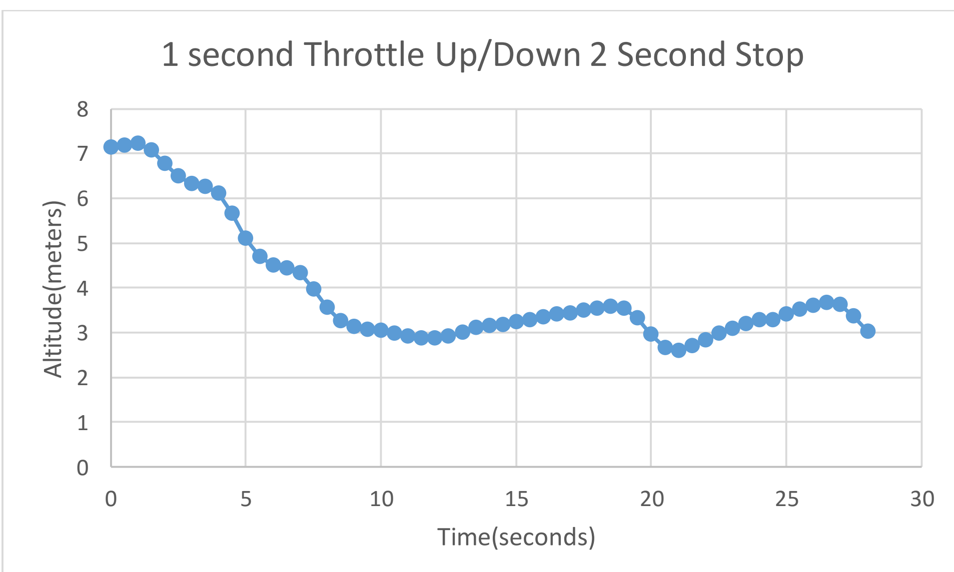

Figure 54. 1s Throttle Up/Down 2s Stop

For 1 second throttle up/down 2.5 seconds throttle stop, the drone height when the algorithm executed was 6 meters the following response is obtained:

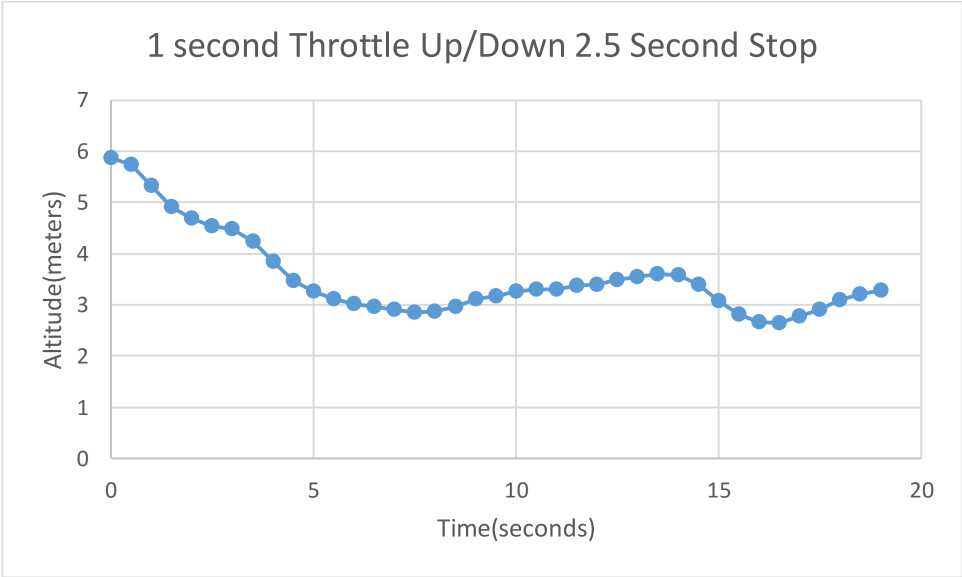

Figure 55. 1s Throttle Up/Down 2.5s Stop

For 1 second throttle up/down 3 seconds throttle stop, the drone height when the algorithm executed was 5.8 meters the following response is obtained:

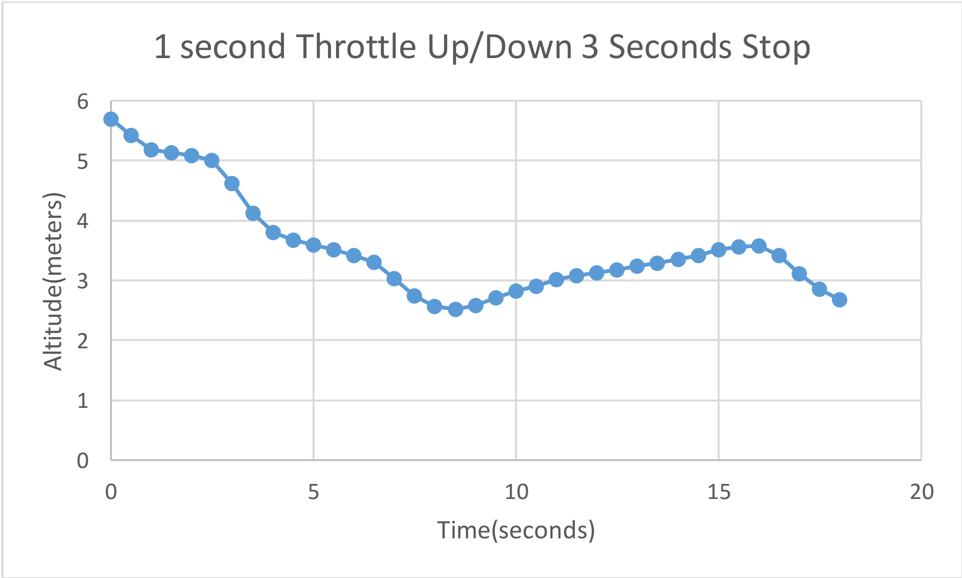

Figure 56. 1s Throttle Up/Down 3s Stop

For 1 second throttle up/down 3.5 seconds throttle stop, the drone height when the algorithm executed was 8.8 meters the following response is obtained:

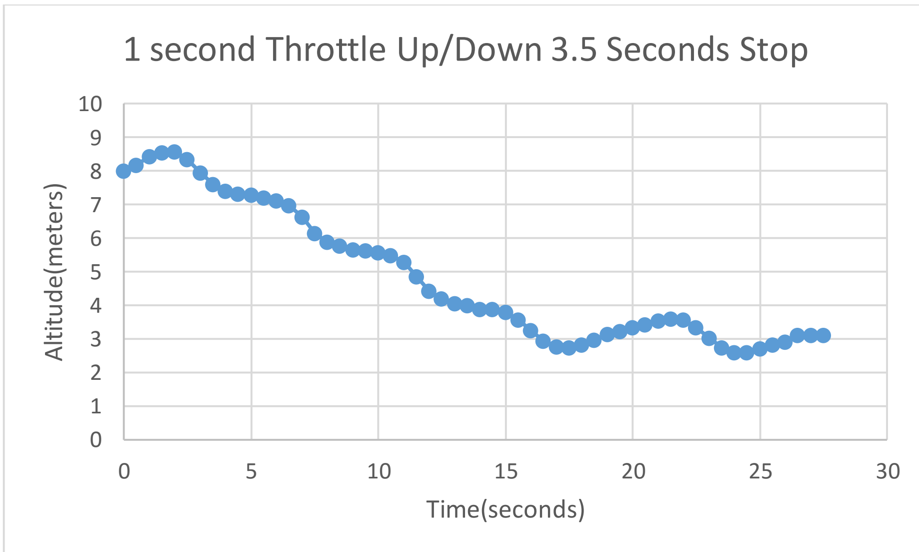

Figure 57. 1s Throttle Up/Down 3.5s Stop

For 1 second throttle up/down 4 seconds throttle stop, the drone height when the algorithm executed was 7 meters the following response is obtained:

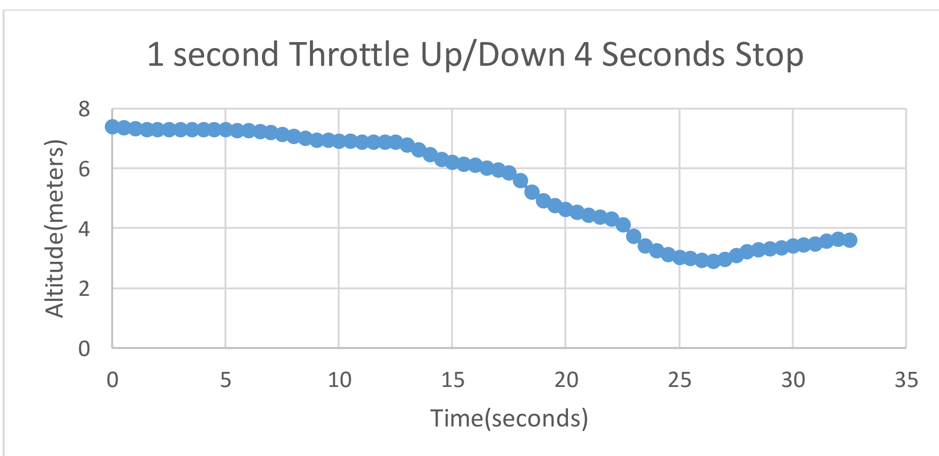

Figure 58. 1s Throttle Up/Down 4s Stop