UNMANNED AERIAL VEHICLE FLIGHT SYSTEM FAILURE AND RADIO RANGE ANALYSIS

by

William Willie Wells

A PROJECT

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Carrick Detweiler

Lincoln, Nebraska

November, 2016

UNMANNED AERIAL VEHICLE FLIGHT SYSTEM FAILURE AND RADIO RANGE ANALYSIS

William Willie Wells, M.S.

University of Nebraska, 2016

Adviser: Carrick Detweiler

An increasing global trend is the use of unmanned aerial vehicles (UAV) by hobbyists, researchers, and businesses. Various manufacturers produce their unique UAVs either as assembled models, as a collection of parts requiring assembly, or as a combination of both. There are risks associated with the operation of UAVs. Potential risks include motor failures, lost communications, and abnormal execution times. Vehicle malfunction can be costly. This work closely analyzes the rates and types of failures through repeated flight. A series of tests were devised and conducted to determine the most common physical and cyber failures. Flights were performed indoors using a motion capture system for position estimates while the vehicles were connected to a DC power supply to enable quick repetition. Robot Operating System (ROS) was used to parse data received by a connected XBee-Pro and compute a control input that was transmitted to a receiving XBee-Pro on a vehicle. Over 900 test flights with an execution time of 65 seconds using 6 vehicles were performed with over 500 flights performed on a single vehicle for endurance testing. Out of 923 tests 140 (15%) flights failed. Individual vehicle failure rates ranged from 4-33%. Packet delay variation control messages accounted for 125 failures, which is the cause of the majority of failures. System designers and users should aim to mitigate the impact of timing problems. Communication interruption failures accounted for 8 failures. Hardware failures accounted for 5 failures. Position system lost track of the vehicle on 45 flights.

Most UAVs require radio communication with a ground station. Precise adaptive autonomous control is necessary to enable effective coordination between UAVs. Many man-

ufacturers of UAVs use 900 MHz or 2.4 GHz ZigBee radios due to their low cost, low power consumption, and compact form factor. The international XBee-Pros tested have a specified outdoor range of up to 750 m. A range of 140 m outdoors was achieved using 3.5 inch antennas on a pair of XBee-Pros. Achieving these range limits depends on radio placement and antenna orientation. This initial work is a start on an analysis of the effective range of vertically oriented paired XBee-Pros with two different antenna lengths.

DEDICATION

This work is primarily dedicated to David Michael Wells Junior. This work is secondly dedicated to Matthew Alan Wells, Violet Lenae Wells Bean, Symone Marie Wells Mell, and Gabriel Dean Wells.

ACKNOWLEDGEMENTS

The supervision of Dr. Carrick Detweiler throughout this process and the guidance of Dr. Sebastian Elbaum at the beginning of this process are greatly appreciated. The service of Dr. Byrav Ramamurthy and Dr. Justin Bradley on my committee and their valuable feedback is appreciated. The suggestions and assistance of everyone in the NIMBUS Lab is much appreciated.

GRANT INFORMATION

This work was partially supported by United States Department of Agriculture grant USDA-NIFA 2013-67021-20947 and National Science Foundation grant NSF IIA-1539070.

Any opinions, findings, conclusions, or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of these agencies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

An increasing global trend is the use of unmanned aerial vehicles by hobbyists, researchers, and businesses. Various manufacturers produce their unique unmanned aerial vehicles either as assembled models, as a collection of parts requiring assembly, or as a combination of both. There are risks associated with the operation of unmanned aerial vehicles. Potential risks include motor failures, lost communications, and abnormal execution times. Vehicle malfunction can be costly. Testing was conducted on quadcopters to determine characteristics that cause failure of normal unmanned aerial vehicle operation. Monitoring of parameters was conducted such that the cause of any failure could be recorded and could be analyzed. The total failure rate for all test flights is 15%, 140/923. Communication only faults consisting of interrupted serial communication accounted for 8 failures, 6% of all failures. The positioning system lost track of the vehicle on 45 flights, 5% of all flights. Time delayed control messages accounted for 125 failures , 89% of all failures. Hardware failures accounted for 5 failures , 4% of all failures.

The placement of electronics on an unmanned aerial vehicle with limited payload and wing span is crucial in order to receive good radio signal strength. The furthest range tested

was 140 m in Euclidean 3-space outdoors and this was with one degraded XBee-Pro in a pair of receivers. With two good XBee-Pros a range of 130 m in Euclidean 3-space with 70 dBm signal strength outdoors was the furthest range tested. Further range is achievable though probably not up to the full 750 m specification. A degraded XBee-Pro significantly limits achievable range.

1.2 Contributions and Lessons Learned

The flight system failure analysis conducted involved data from nearly a thousand individual experiments on six UAVs and two different UAV systems. While some of the results are specific to our particular setup and vehicles, there are a number of factors that we believe may influence many other UAV systems. The key contribution of this work is that this work is one of the first detailed studies of failure modes for small UAV systems under controlled conditions. The design and details of these tests can be used as a model to test other UAVs. The key finding that communication is the cause of most failures is also significant as system designers and users should aim to mitigate the impact of communication problems. More specifically, we found:

- Packet delay variance of sensor request and control input packets in autonomous systems result in observable abnormal behavior that is not intended by the designer of the control software. Specifically, Ascending Technologies recommends using a control rate at or above 10 Hz. A sensor request update rate of 0.8 Hz and a control input rate of 4 Hz are the minimum respective rates at which an Ascending Technologies Hummingbird will follow a designed trajectory, exhibit expected behavior.

- Communication protocol checksums occasionally fail packets, which is a source of undesired control system time delay. A control system should be designed to be robust enough to handle such occurrences. Undesired time delay in a control system transpires. Out of 923 flights 89% of errors involved time delayed packets. This time

delay will cause the controlled system to exhibit unexpected behavior, therefore, to ensure safe flight, actively monitor system time delay and make a decision to perform a safety action based on the measured time delay. Perform a pre-flight check that control inputs, sensor requests, and positioning system updates are being updated at the desired rate.

- An improperly configured network device caused 4 test failures. Therefore, prior to flight properly configure all devices operating on the same network(s) accessed within the flight system control loop. Perform a pre-flight check of devices connected to the indoor positioning system network and implement an online warning system that actively monitors for other devices connected to the positioning system network. Serial communication was interrupted during 6 flights. If using a USB radio device, actively monitor for pipe halts and/or stalls. Send a warning if this occurs.

- Positioning system failures can occur. Indoor positioning, motion capture system cameras, can be blocked or markers can be obfuscated and likewise outdoors a GPS denied scenario can occur. So, perform a pre-flight check that verifies proper operation of the utilized positioning system. Perform active monitoring that monitors for a change in control input and that monitors sequential position system estimates. If the difference between sequential position system estimates exceeds the positioning system accuracy plus the control input, perform a safety action.

- Hardware such as XBee-Pros and SparkFun XBee Explorer Dongles exhibit performance degradation over time. The placement of a radio on an unmanned aerial vehicle with respect to the surrounding electronics effects received signal strength of the receiver. Thus, the placement of a radio on an unmanned aerial vehicle should be carefully considered to minimize interference effects of neighboring electronics. Perform a pre-flight check of received signal strength indication, adjust radio placement and/or additional electronic devices as required. Actively monitor RSSI for abrupt

changes in signal strength and for maximum radio RSSI values, initiate a warning if these situations are indicated.

1.3 Structure

The remainder of this report is organized as follows. Chapter 2 discusses the flight system failure analysis test setup, the developed testing procedure, the types of failures encountered, causes of the failures encountered, solutions to reduce or eliminate the types of failures encountered, analyses of vehicle position error, vehicle power consumption, vehicle status flags, vehicle execution time, and flight performance of the various vehicle types. Then, radio range analysis testing setup, procedure, and results are discussed in chapter 3. Testing viability of integrating an RTK GPS system with an Ascending Technologies Hummingbird is explored in chapter 4. Chapter 5 discusses future work and conclusions.

Chapter 2

Flight System Failure Analysis

2.1 Motivation

Flight system failure is costly whether manned or unmanned. The utilization of unmanned aerial vehicles has recently increased partially due to the potential advantages of utilizing unmanned systems. These advantages include decreased man-hours required to perform a task, decreased human risk in hazardous environments (increased safety), and decreased economic cost per task. Thus, to fully reap the maximal potential benefits of UAV deployment to society the types of failures that occur with UAV flight systems most be known and then measures must be engineered to reduce and/or eliminate the occurrence of such flight system failures.

2.2 Related Work

Portraying an actively controlled unmanned aerial vehicle flight system in the context of a cyber physical system [12], a cyber request for sensor data (physical system state) is transmitted from a controlling station through a communication medium (radios), a cyber response is transmitted back to the controlling station, a cyber control input is transmitted to control actuators (physical devices that change system state), and actuators apply the

control input.

One of the more prolific individuals involved in studying UAV fault analysis, Youmin Zhang, has study primarily actuator and secondarily sensor faults [88] [87] [86]. This branch of research is separated into two coupled sets of algorithms, algorithms for fault detection and diagnosis and algorithms for fault tolerant control [39] [56] [53] [63] [85]. Simulation of fault tolerant control algorithms for actuator faults is widely studied [7] [47] [64] [70] [74]. There are even those who have implemented an observer used in experimental tests of actuator fault tolerant control analysis [16] [34] [69]. There are passive and active fault tolerant control algorithms. Active fault tolerant controllers for actuator faults are most often proposed [2] [16] [17] [18] [29] [48] [68]. Though not as thoroughly analyzed as actuator fault detection and control, sensor faults are studied as well [89] [61] [20] [34] [62].

Studies of UAV reliability are conducted by the Department of Defense [73] [28]. One such study [28] found that flight control systems were at fault for 20% of UAV mishaps and recommended the creation of self repairing flight control systems. Manufacturers of UAVS incorporate general safety procedures [42] [4].

Reliability of manned aerial vehicles has focused on reliable guidance, navigation, and control algorithms [22] [67] [23] [80] [40]; reliable fault tolerant multiprocessor algorithms [38]; simulations of fault tolerant control algorithms [82] [32] [45] [55]; actuator, sensor, and component fault analysis [50]; decreasing spacecraft failure rates [35]; and simulations for extended duration spaceflight [10]. Manned aircraft failure testing often involves rigorous structural testing procedures [43] [37] [36] [25]. This is impractical for UAV failure testing.

Something that is lacking is a systems approach to failure analysis [51] and system design [11]. This work analyzes types of flight system failures and the occurrence of such failures with respect to number of flights. In addition to analyzing flight system failures as a complete system, this work analyzes failures caused by unanticipated time delays. While there are those that have acknowledged that variable time delays could cause their

algorithms to be unstable such as in [16], analysis of the effects of variable time delay in a flight system has not been widely studied. This work proposes implementing an observer that monitors for time delay of key control messages in the system and that subsequently enacts a safety measure based on observed time delay.

2.3 Overview of Procedure

Figure 2.1: Herbie–Ascending Technologies Hummingbird with ARDrone propellers connected to switching DC power supply after manual flight test.

A switching DC power supply is used to enable repetitive testing without the need to replace and exhaust the applicable unmanned aerial vehicle's supply of batteries. This power supply is switched on. A manual flight of the vehicle to be tested is conducted. The power supply is connected to the vehicle under test and the vehicle is turned on. Figure 2.1 shows the physical test configuration after the completion of these steps.

Required software, Vicon Tracker and Robot Operating System (ROS), is started. The controlling station is connected to the power supply to enable receipt of sensor values. Communication, transmission and receiving of data packets, between vehicle under test and controlling station is established. Figure 2.2 shows the physical test configuration after the completion of these steps.

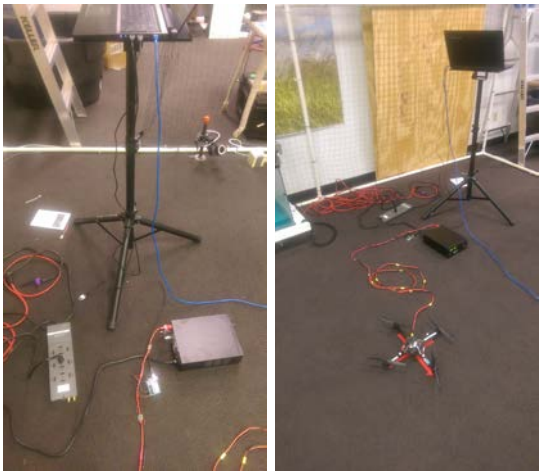


Figure 2.2: Physical test setup.

Testing software is launched. The vehicle under test is monitored as it performs predefined maneuvers. After the completion of the current test, the next test is prepped. Figure 2.3 shows four snapshots of the test procedure in progress as the unmanned aerial vehicle performs the predefined maneuvers.

Figure 2.3: Herbie–Ascending Technologies Hummingbird with ARDrone propellers progressing through the maneuvers of the flight test.

2.4 Components and Configuration

2.4.1 UAV Platforms

Ascending Technologies Hummingbirds provide access to control packet data through wireless serial data links (2 UART's) as well as programming the high level processor directly for automatic missions. The former control method is used exclusively during the tests described here. The on-board computing platform is the AscTec Atomboard based on the Intel Atom but utilizes two ARM7 microprocessors for IMU processing [4]. The ARDrone is designed for mobile control with a hand held device such as a phone or tablet. Wi-Fi 802.11b/g is provided as a wireless interface for the ARDrone. An ARM processor is utilized as the on-board computing platform for the ARDrone [3].

2.4.2 Test Environment

All tests are performed indoors, inside a PVC pipe and wire mesh cage. After the first test day, strips of tape were placed on the carpeted cement to provide a uniform starting position for each unmanned aerial vehicle for each test run.

2.4.3 Switching Power Supply

A switching DC power supply [9] was purchased to enable repetitive testing without the need to replace and exhaust the applicable unmanned aerial vehicle's supply of batteries. At the beginning of testing for the day, the switching DC power supply is switched on, voltage reading is verified to be 12 volts, and current limit is verified to be 30 amperes. A battery is placed inside the unmanned aerial vehicle's on board battery holder but is not connected to simulate the normal weight. A cable connecting the switching DC power supply is connected to the power terminal of the vehicle to be tested. Unmanned aerial vehicle under test is powered on.

2.4.4 Controlling Station

A Futaba controller is used for all Ascending Technologies vehicles to set the vehicle's controlling station [31]. A XBee-Pro via a USB cable is physically connected to the controlling computer. The virtual machine user profile on the controlling station is given read and write access to the corresponding USB port. A communication node in ROS handles the serial data receiving and transmission processing. The vehicle is flown in manual mode once prior to conducting the automatic tests to determine if the vehicle is in need of repair prior to testing. Computer control is then enabled on the Futaba controller. A wireless connection is established using the vehicle specific IP address 192.168.2.5 to connect to the ARDrone. The communication node is replaced by an ardrone_driver node to handle communication, packet transfer between the vehicle and the controlling station.

2.4.5 Vicon Motion Capture System

Vicon Tracker along with 12 strategically mounted cameras and Vicon markers are used to track objects inside the cage. This tracking approach is a passive optical approach which uses retroreflective markers and cameras to capture motion. The retroreflective material reflects light back to the cameras with minimal scattering. The cameras are designed for high resolution motion capture [78]. Vicon tracker is started on the Vicon computer. A roscore node and Vicon connection is established on the controlling computer. Through this Vicon connection a vicon_bridge node is started that publishes vicon position in the format: vicon/subject_name/segment_name (subject_name and segment_name are equivalent names in the system used).

2.4.6 Voltage and Current Monitoring

An Arduino with a pair of voltage and current sensors attached between it and the switching DC power supply is connected to the controlling computer. The Arduino is programmed to record voltage and current readings from two AttoPilot 45A sensors [5]. The AttoPilot sensors are in series with the main output ground and power ports terminating in a Dean's T connector. A current, voltage, and ground line are connected from the AttoPilot sensor to the Arduino. The voltage and current lines from the first in series AttoPilot are connected to analog inputs A0 and A1 of the Arduino while the voltage and current lines from the second in series AttoPilot are connected to analog inputs A4 and A5 of the Arduino.

2.4.7 ROS Configuration

A program to test basic flight maneuvers (launch, pitch, roll, rotations around z-axis, position changes in three dimensional space, land) was written to run in VMWare, Ubuntu 14.04 with ROS Indigo.

A Vicon position node is started first using Vicon Bridge. A node that converts Vi-

con position to subject pose is started. A node that connects to the XBee-Pro radio is started along with any additional telemetry nodes, unmanned aerial vehicle platforms other than Ascending Technologies vehicles may require additional communication nodes. Four nodes required to operate a minimal graphical user interface are started. Message protocol, subject status, state control, and translator of control input messages nodes are started in that order. These nodes are the minimal nodes required for basic controlling station flight in the system used. Position control input nodes are started which includes an arbitration node. Creation and recording of topics is supported by ROS. Each topic may have and usually has multiple fields. A record node is started. Two nodes used to communicate and parse the data from the Arduino that is processing power consumption parameters are started. Finally, the node that runs through the flight maneuvers is started.

2.5 Testing Procedure

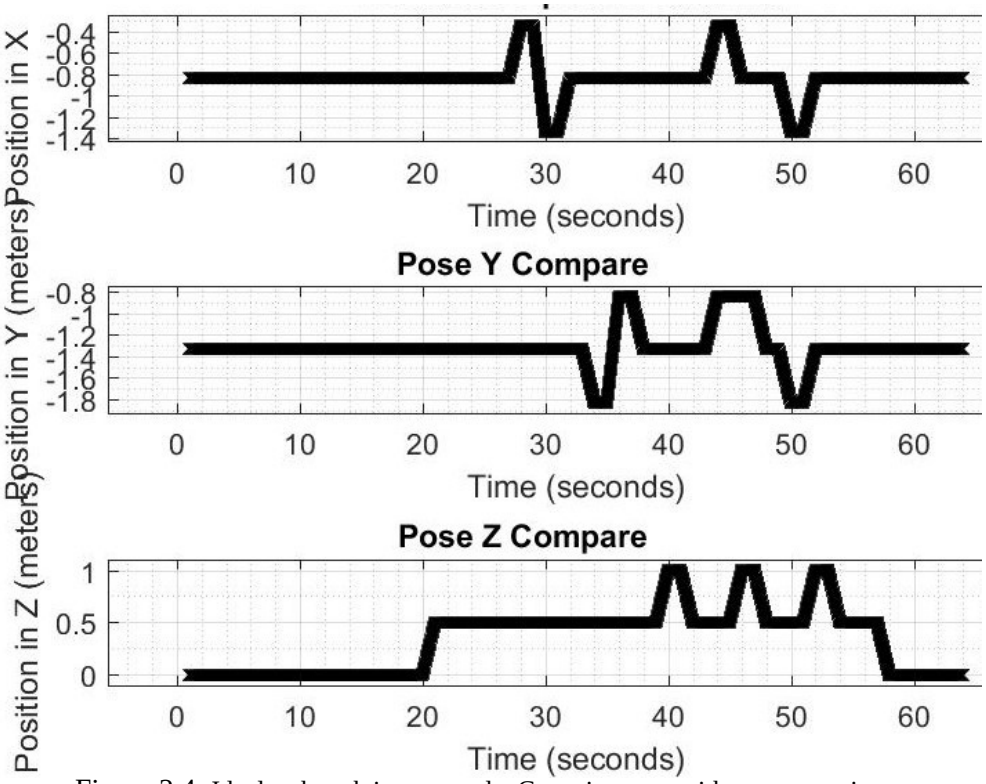


Figure 2.4: Ideal task path in x, y, and z Cartesian axes with respect to time.

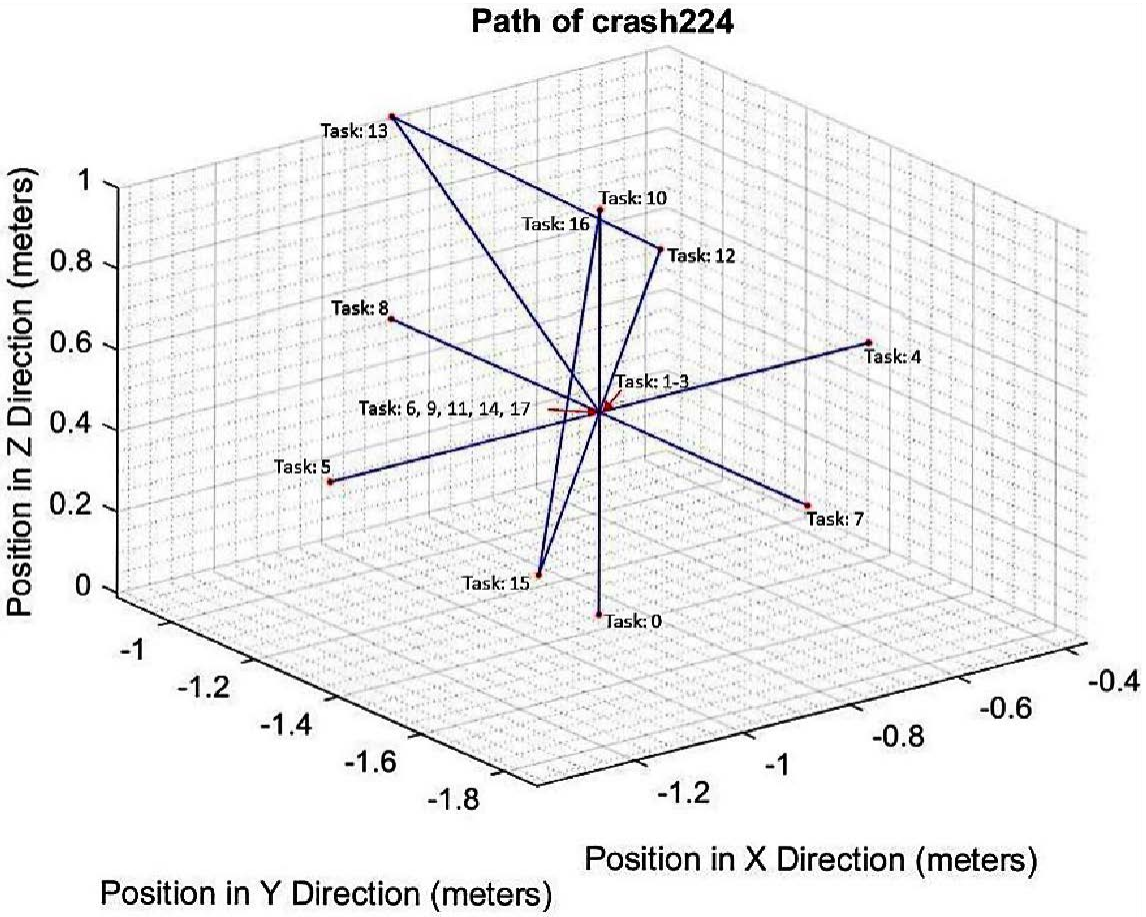


Figure 2.5: A 3 dimensional plot of the ideal task path without time.

2.5.1 Description

Flight testing launch file is launched. Launch height and a uniform magnitude for the base translation value (δ) can be specified in the launch file. The test maneuvers themselves are as follows. Turn motors on; test operation of motors by performing the maneuvers: pitch forward, pitch backward, roll left, roll right; launch; perform this rotation sequence while maintaining position $\pi \rightarrow \pi/2 \rightarrow 3\pi/2 \rightarrow 0(2\pi)$; translate along each Cartesian axis in 3 dimensional space with and without rotations; translate to positions that require change in multiple Cartesian axes coordinates with and without rotations; land; and turn motors off. The turning on of the motors is essential for the rest of the maneuvers and is the first visible state transition. Pitch and roll tests indicate vehicle orientation as well as indicating individual motors are operating properly. Launching enables testing translations and rota-

tions while in flight. While on the ground and during launch the vehicle is initially rotated near π . A 90 degree counter clockwise (clockwise) rotation is equivalent to a 270 degree clockwise (counter clockwise) rotation. The selected rotation tests rotate the vehicle so that the vehicle faces positive y (North), negative x, positive x (East), and negative y this demonstrations the vehicle's ability to rotate to these facings while maintaining the same translation. The 180 degree rotation demonstrates the vehicle's ability to handle a maximum rotation as well as the quality of the gyroscope measuring yaw. Translations along each Cartesian axis demonstrate the vehicles ability to translate to specific positions in a Cartesian space. Simultaneous rotation and translation commands indicate the vehicle's ability to handle changes in both rotations and translations while implementing a single tasked waypose. Each pair of Cartesian translations (x and y, y and z, z and x) are tested once with a 90 degree rotation and once without a rotation. The last two flight maneuvers are a change in translation for all Cartesian axes with and then without rotation. The final translation and rotation is designed to be the same position and facing as the target launch position and facing. While maintaining this position and facing the vehicle is commanded to land. Landing is a prerequisite to turning off the motors. Turning off the motors places the vehicle in a safe condition prior to starting the next test.

During the test any abnormal behavior is annotated in a testing response form using Google Sheets. A safety operator ready to take manual control is always attentive during each flight. The output ROS bag file is updated. Bag files are in the format vehicleType_vehicleNameTestNumber.bag. Vehicle is reset to start origin, launch file is started again, and the above maneuvers are repeated.

At the end of the day's testing or when ever convenient thereafter the bag files are converted to .csv files using bag2csv [1]. The .csv files are then imported into MATLAB for analysis. An updated version bag2matlab [2] was created by Dave Anthony in June of 2016. This version is the version used for data analysis after the indicated date.

[1] https://github.com/unl-nimbus-lab
[2] https://github.com/unl-nimbus-lab/bag2matlab

2.5.2 Analysis

Testing setup may include delays based on being able to connect to Vicon and other device connection issues. The test procedure itself is streamlined for efficient repetition.

The maneuvers themselves are designed to test vehicle response to common maneuvers. The pre-flight sequence is split into three stages: start up of the motors, testing of the motors, and launching of the vehicle. The pre-flight test of the motors is the only place in the procedure that tests maximum pitch and roll. An initial position in the horizontal plane is set once during the first state change. All subsequent changes are based on this, a rotation value of 0.0, and a launch height, which is set in the launch file. Slight modifications to the control software's Proportion Integral Derivative (PID) settings may further minimize position errors. A counter keeps track of the latest target in flight maneuver. Each in flight maneuver has a move to position then hover component. After witnessing and being required to use this move implementation style in a previous project, it was deemed a better, stabler, implementation than a continuous move implementation style. When the maneuver counter exceeds a certain value a land request is set and the landing sequence begins.

The flight testing node is shutdown after an electrically stopped state is reached. The rest of the nodes started by the cascaded series of launch files are shutdown as well. Automatic shutdown of running nodes provides a reliable way to extract a measure of test execution time. Without this measure in place topics may be recorded continuously beyond the actual length of the test. Launch file cascade structure used involves three layers. The innermost layer contains the nodes related to communication, the middle layer contains nodes related to basic position control, and the top layer contains the nodes specific to the flight testing project and specific to that particular test launch. Vehicle under test, vehicle type, vehicle type specific middle layer launch file, and recorded bag file name are the commonly modified fields in the top layer launch file. A subset of the available topics to record is recorded to reduce bag file size. The topics recorded are: target state, quad and robot control input, robot inertial measurement unit, robot and subject status, voltage and

current sensor messages, actual state, subject position, target position, and Vicon position.

The power cord connected from the switching DC power supply to the power terminal of the vehicle under test caused more delays than any other single factor in the test procedure. The power cord tends to get tangled and may interfere with propellers during the testing of the motors. This happened more frequently at the beginning of the series of tests conducted but is still a viable concern. Power cord twist occurs due to the rotational tests in the test procedure. Severity of power cord twists is related to which direction counter clockwise or clockwise the vehicle rotates during the 180 rotations.

2.6 Experimental Results

Vehicle	Vehicle Type	Propeller	Failed Tests	Total Tests	Failure Rate (%)
1	AscTec Hummingbird	AscTec Safety	78	510	15.294
2	AscTec Hummingbird	AscTec Safety	4	100	4.000
3	AscTec Hummingbird	AscTec Safety	1	3	33.333
4	AscTec Hummingbird	AscTec Normal	33	110	30.000
5	AscTec Hummingbird	ARDrone	14	100	14.000
6	ARDrone	ARDrone	10	100	10.000

Figure 2.6: Experimental results and the respective failure rate of each vehicle.

Vehicle	Vehicle Type	Propeller	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
1	AscTec Hummingbird	AscTec Safety	10	100	100	200	100	0	0	0	0
2	AscTec Hummingbird	AscTec Safety	0	0	0	0	0	0	0	100	0
3	AscTec Hummingbird	AscTec Safety	0	0	0	0	0	0	0	3	0
4	AscTec Hummingbird	AscTec Normal	10	0	0	0	0	100	0	0	0
5	AscTec Hummingbird	ARDrone	10	0	0	0	0	10	80	0	0
6	ARDrone	ARDrone	0	0	0	0	0	0	0	0	100

Figure 2.7: Number of tests performed per day.

Over 900 total flight tests were performed with at least 100 tests performed on each working vehicle and 510 tests performed on a single vehicle for endurance testing. A total of 200 tests were performed on that single vehicle in one day over a 6 hour span. Six vehicles were tested. The first vehicles tested were Ascending Technologies Hummingbirds with three different propeller types: Ascending Technologies safety propellers, Ascending Technologies normal propellers, and ARDrone propellers mounted on an Ascending Technologies Hummingbird. An additional Hummingbird with Ascending Technologies safety propellers was tested to provide a comparison across vehicles with the same propeller type. An ARDrone was tested next. The vehicle that flew 510 tests was an Ascending Technologies Hummingbird with Ascending Technologies safety propellers. The vehicle with Ascending Technologies normal propellers in the first batch of tests flew 110 test missions. The remaining three vehicles mentioned were tested 100 times each. A sixth vehicle, a Hummingbird with Ascending Technologies safety propellers was tested thrice. Testing on this vehicle ceased after the vehicle flipped prior to launch on the third test.

2.7 Types of Failures and Their Causes

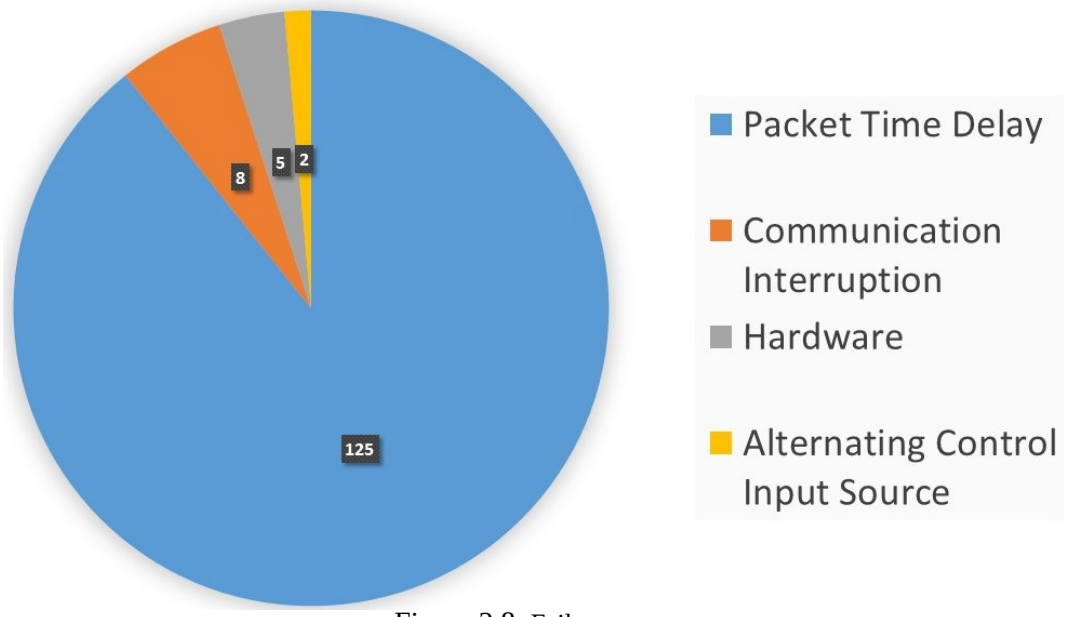


Figure 2.8: Failure types.

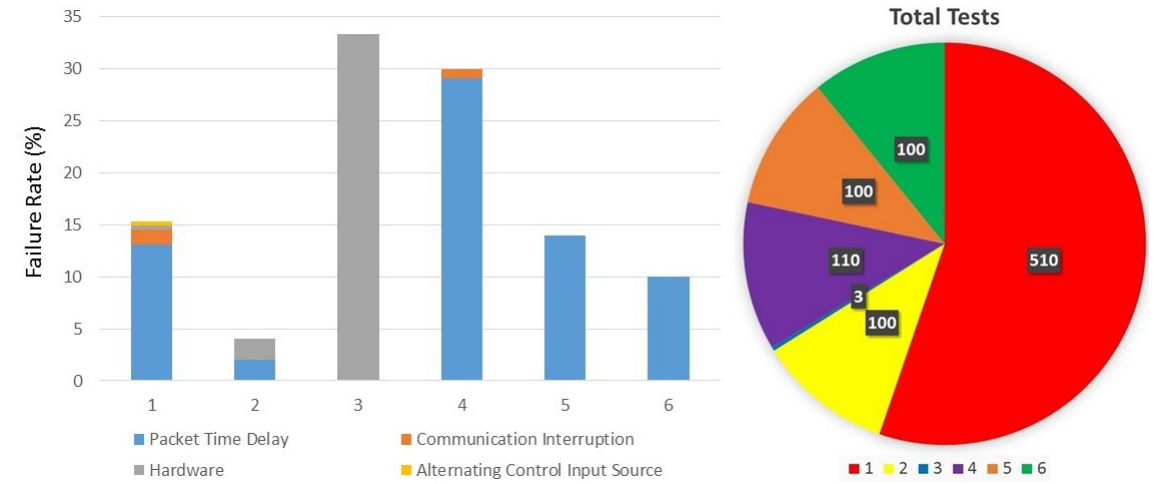


Figure 2.9: Failure rates.

2.7.1 Symptoms

A Google Forms form was modified from a pre-existing NIMBUS Laboratory flight tracking form to record human observer responses to the observed tests. Two human observers, Levi Amen and the author, observed the tests. If any abnormal behavior was witnessed on a test, the abnormal behavior was described in the response form. The fields of the Failure Testing Response Form include: Tester, Observer, Test Version Number, Vehicle Name, Log File Name, Test Completion, Response Timestamp and Abnormal Behavior Description. The responses of the observers were compared to plots generated from the recorded data to identify the cause of observed abnormal behavior. A test that included abnormal behavior was considered a failed test, since a controlled system should respond to a control input within expected response behavior. The expected response behavior was limited to the controlled system performing the maneuvers described in section 2.5.1 within the time allotted. Thus, failure symptoms observed during the tests were:

1. the UAV landing prior to receiving the landing task and recovering,
2. additional time taken to perform a maneuver,
3. quicker than normal transition through a maneuver,

4. a maneuver was not observed at all,

5. performance of additional maneuvers than those tasked,

6. motors remaining on after automatic shutdown commanded.

During some tests manual control was taken due to observing the above abnormal behavior, though the unmanned aerial vehicle may have recovered and finished on with the test sequence.

The types of failures were separated into four high level categories: packet time delay (jitter), communication interruption, hardware, and alternating control input source. This is visualized in Figures 2.8 and 2.9. Figure 2.9 further breaks down failure rate per type and per vehicle. The total tests per vehicle is given in Figure 2.9 for a comparison of individual vehicle failure rates with the amount of tests performed by the vehicle. Table 2.1 displays a further breakdown of failure type into 15 sub-categories, some of which are overlapping. Sections 2.7.2 through 2.7.11 describe in detail the 10 non-overlapping failure type sub-categories that occurred.

Vehicle	1	5	4	2	3	6
Propeller (type)	AscTec Safety Propellers	ARDrone Propellers	AscTec Normal Propellers	AscTec Safety Propellers	AscTec Safety Propellers	ARDrone Propellers
Manual Control (count)	1/510 (0.196%)	0/100 (0.000%)	3/110 (2.727%)	1/100 (1.000%)	1/3 (33.333%)	0/100 (0.000%)
Serial Communication Interrupt (count)	5/510 (0.980%)	0/100 (0.000%)	1/110 (0.909%)	0/100 (0.000%)	0/3 (0.000%)	0/100 (0.000%)
Extra Task (count)	24/510 (4.706%)	6/100 (6.000%)	5/110 (4.545%)	1/100 (1.000%)	0/3 (0.000%)	9/100 (9.000%)
Missing Task (count)	13/510 (2.549%)	6/100 (6.000%)	8/110 (7.273%)	1/100 (1.000%)	0/3 (0.000%)	0/100 (0.000%)
Extra State (count)	29/510 (5.686%)	6/100 (6.000%)	10/110 (9.091%)	2/100 (2.000%)	0/3 (0.000%)	2/100 (2.000%)
Missing State (count)	7/510 (1.373%)	0/100 (0.000%)	5/110 (4.545%)	0/100 (0.000%)	0/3 (0.000%)	0/100 (0.000%)
Quad Control Input Delay (count)	58/510 (11.373%)	5/100 (5.000%)	29/110 (26.364%)	0/100 (0.000%)	0/3 (0.000%)	0/100 (0.000%)
Command State Delay (count)	61/510 (11.961%)	7/100 (7.000%)	22/110 (20.000%)	2/100 (2.000%)	0/3 (0.000%)	1/100 (1.000%)
Tasked Pose Delay (count)	57/510 (11.176%)	9/100 (9.000%)	21/110 (19.091%)	0/100 (0.000%)	0/3 (0.000%)	1/100 (1.000%)
Subject Pose Delay (count)	57/510 (11.176%)	6/100 (6.000%)	27/110 (24.545%)	1/100 (1.000%)	0/3 (0.000%)	43/100 (43.000%)
Vicon Delay (count)	15/510 (2.941%)	5/100 (5.000%)	5/110 (4.545%)	2/100 (2.000%)	1/3 (33.333%)	9/100 (9.000%)
Vicon Lost Object (count)	0/510 (0.000%)	17/100 (17.000%)	6/110 (5.455%)	20/100 (20.000%)	1/3 (33.333%)	1/100 (1.000%)
Motors Stayed On (count)	12/510 (2.353%)	0/100 (0.000%)	1/110 (0.909%)	4/100 (4.000%)	0/3 (0.000%)	0/100 (0.000%)
Vehicle Flipped (count)	0/510 (0.000%)	0/100 (0.000%)	0/110 (0.000%)	1/100 (1.000%)	1/3 (33.333%)	0/100 (0.000%)
Subject Status (count)	40/510 (7.843%)	3/100 (3.000%)	12/110 (10.909%)	1/100 (1.000%)	0/3 (0.000%)	100/100 (100.000%)

Table 2.1: Failure types per vehicle with amount of occurrences per failure and the associated percentage of failure type per vehicle.

2.7.2 Control Packet Delay

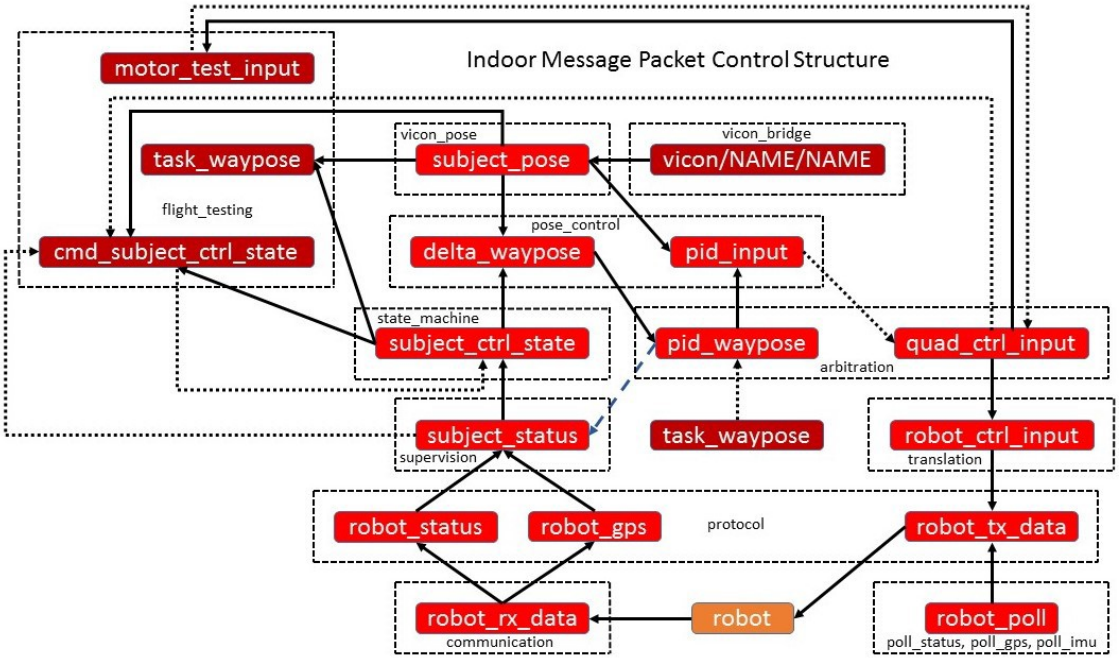


Figure 2.10: Control message flow loop.

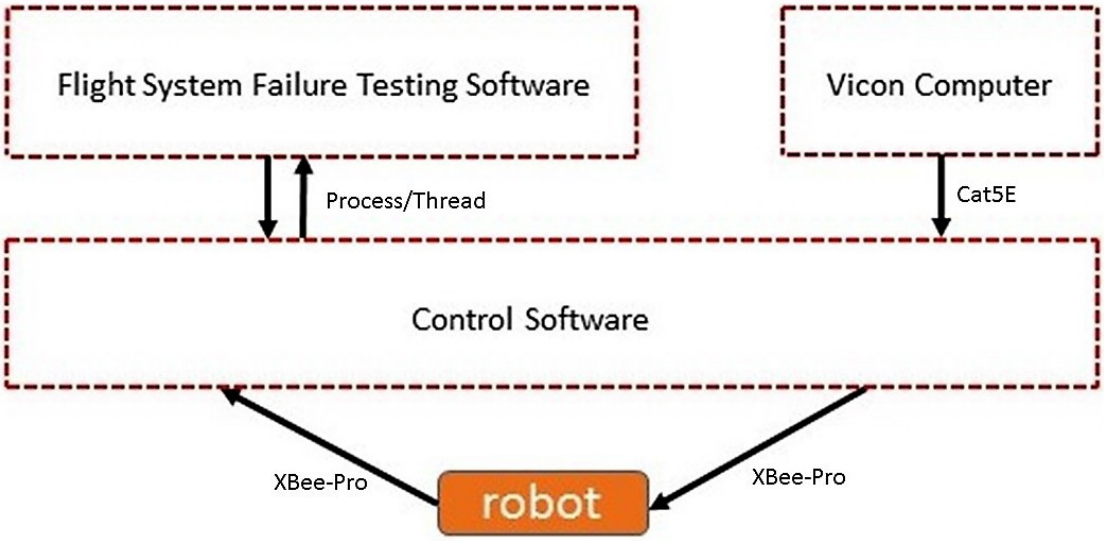


Figure 2.11: A simplified control flow diagram.

Delays in packets other than packets used to directly control the unmanned aerial vehicle were recorded such as delays in subject status, Vicon object, subject pose, and subject control state. Additional packets to analysis in the future include robot status, robot rx data, robot tx data, robot poll, and robot control input. Analysis of these topics may reveal a closer relationship between propagating packet delays. Delays in robot rx data may indicate

a failing component onboard the quadcopter. There may be delays that result from near simultaneous publishing times of robot poll (1 Hz, 4 Hz, 15 Hz) and robot control input (30 Hz) messages. With the topics analyzed so far this cannot be determined. The delays recorded may be the fault of the controlling computer or either XBee device as well as the quadcopter. Most of the queues in the control loop are of size 10 with the serial transmission topics being the exceptions.

A diagram showing the control flow of data from the unmanned aerial vehicle and back is illustrated in Figure 2.10. Solid lines from topic A to topic B indicate topic A is a publishing dependency for topic B. A dashed line from topic A to topic B indicates that topic A is subscribed to but not required for publishing of topic B. A dotted line from topic A to topic B indicates that topic A provides data that results in the published data of topic B changing or that topic A is one of several state dependent message sources for topic B in the case of the arbitration node. Nodes are identified by dashed boxes and their node names in the figure. The topic robot poll is the only topic directly published to bymultiple nodes (poll gps requested at 4 Hz, poll imu requested at 15 Hz, and poll status requested at 1 Hz). Figure 2.11 is a simplified version of Figure 2.10 and contains the communication medium used between the separate parts of the control system.

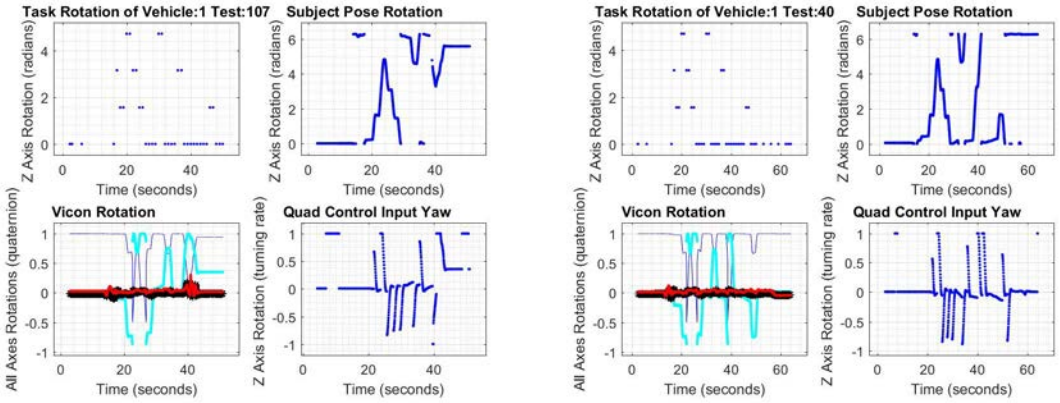


Figure 2.12: Rotation plots for both failed test: (Vehicle 1, Test 107) and successful test: (Vehicle 1, Test 40).

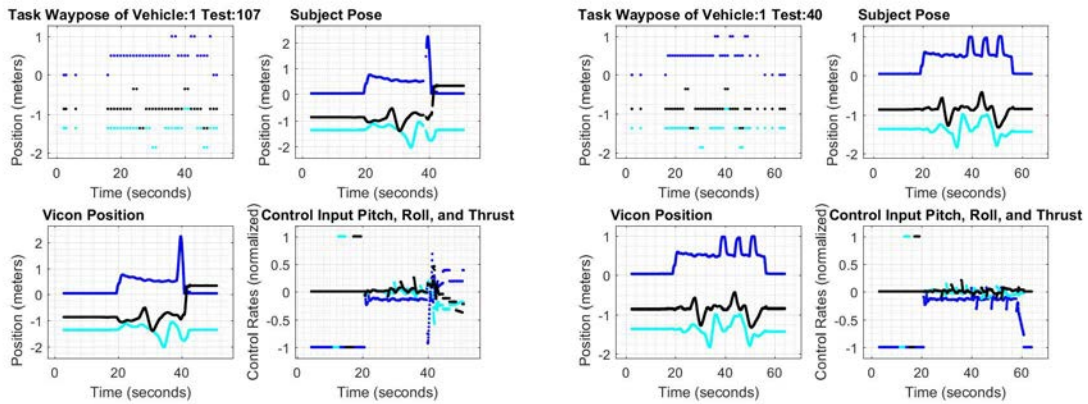


Figure 2.13: Position plots for both failed test: (Vehicle 1, Test 107) and successful test: (Vehicle 1, Test 40).

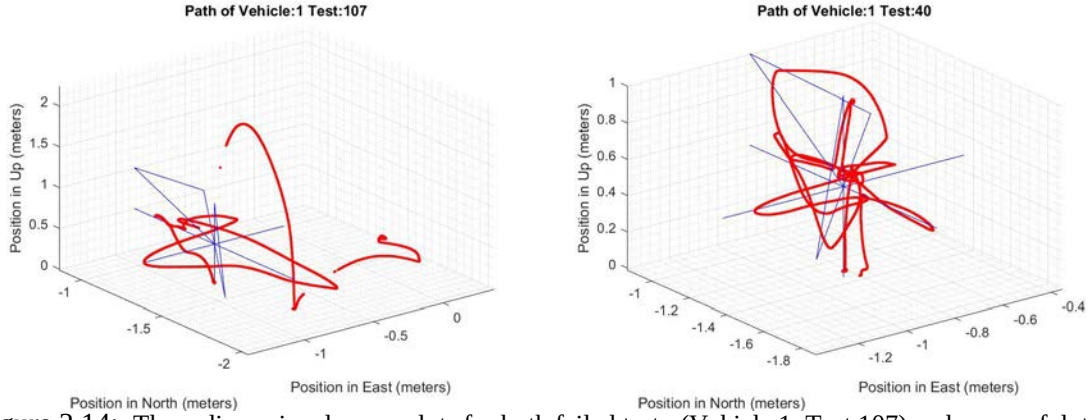


Figure 2.14: Three dimensional space plots for both failed test: (Vehicle 1, Test 107) and successful test: (Vehicle 1, Test 40).

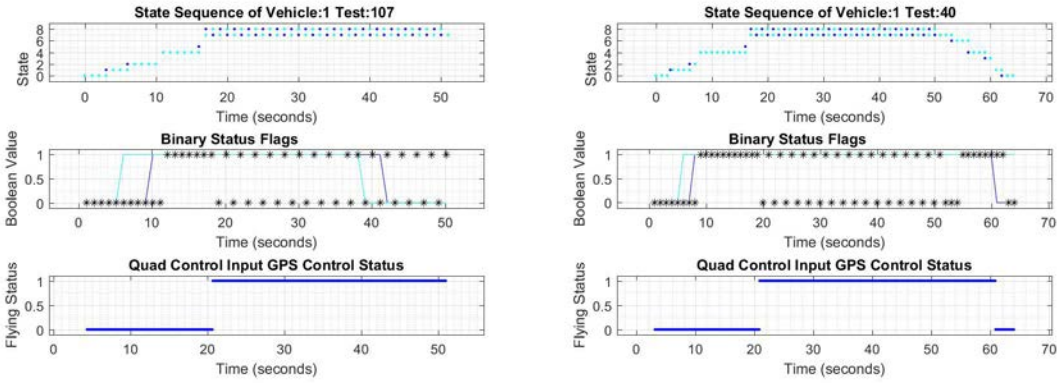


Figure 2.15: State and status plots for both failed test: (Vehicle 1, Test 107) and successful test: (Vehicle 1, Test 40).

Symptoms 2, 3, 4, and/or 5 are observed. A maneuver is not observed at all, a quicker than normal transition through a maneuver is performed, additional time is required to

perform a maneuver, and/or performance of additional maneuvers than those tasked is witnessed[3].

Command subject control state messages along with tasked waypose messages are topic messages used to control the behavior of the unmanned aerial vehicle while in flight. Each in flight task, move and hover, requires one second for an Ascending Technologies Hummingbird to complete. ARDrone move tasks required two seconds and hover tasks required one second to complete. Since each task takes a second for an Ascending Technologies Hummingbird to complete, packet delays greater than 1.300 seconds for Ascending Technologies Hummingbird tests and packet delays greater than 2.300 seconds for ARDrone tests are recognized as either control state delays or tasked waypose delays respectively. Packet delays are computed by taking the difference between successive packet time stamp updates. In the case of tasked waypose or control state delays, a failure is annotated.

The first packet delays that were analyzed from the collected data were subject pose delays. When it was noted that subject pose delays did not seem to correspond with a majority of the above symptom types other message packet delays were analyzed. The topics analyzed were added in this order: subject pose, Vicon, subject control state, task waypose, subject status, quad control input, and command subject control state. A correlation between delays in the control packets was noticed while delays in subject control state (not a control packet), subject status, subject pose, and Vicon were rarely sufficient in the absence of other delays to correspond with the above symptoms. As noted in the Vicon Communication Delay section, delays in Vicon cause symptoms 2 and 3 but are often barely noticeable and excessive delays in Vicon cause symptom 5, when delays exceed 6 times the control rate. Delays in subject status have no affect. Subject control state and subject pose delays together corresponded with a missing task on test: (Vehicle 1, Test 111) but either without other delays do not correspond with any abnormal behavior. Delays in task waypose, command subject control state, and subject control state occurred for 1.527

[3]Further description of the individual plots are under Figure Descriptions in the Appendix.

seconds at approximately 51 seconds into test (Vehicle 5, Test 1) with a delay at approximately 52 seconds into the test of subject pose lasting 0.539 seconds. This resulted in the vehicle exceeding the required height maneuver, reaching within 0.400 meters below the top of the cage, a symptom 5 failure. A task waypose delay of 1.663 seconds starting at 22.022 seconds into the test, a command subject control state delay of 1.665 seconds starting at 22.022 seconds into the test, a quad control input delay of 0.826 seconds starting at 23.205 seconds into the test, a subject control state delay of 1.201 seconds starting at 22.001 seconds into the test, a subject pose delay of 0.630 seconds starting at 23.582 seconds into the test, and a subject status delay of 1.520 seconds starting at 21.682 seconds into the test resulted in an extra tasked waypose message rotation to π and an extra move command subject control state message on test (Vehicle 4, Test 77), which physically exhibited symptom 2, additional time than normal to perform a maneuver.

All of the instances where abnormal translations occurred due to delays of control packets the vehicle eventually recovered and finished the task set except for those instances in which manual control was taken. The first excessive positive height change witnessed was an occasion where manual control was initiated. Whether repetition or skipping of a task occurs depends on the delays associated with subsequent control packets: tasked waypose, command subject control state, and quad control input packets. The tasked waypose and command subject control state packets are updated at 1 Hz. This is due to a dependency on poll status updates. This dependency on the slowest update rate is propagated through from poll status to pid waypose. The time between pid input packet updates is the maximum of time difference between subject pose updates and the period corresponding to the pid input control rate parameter. The time between quad control input packet updates is the period corresponding to the arbitrated input's control rate parameter.

A delay of tasked waypose packets lasting greater than 1.300 seconds while in the hover state results in the vehicle skipping the next maneuver, the maneuver is not observed. This occurred on test (Vehicle 1, Test 264), which had a tasked waypose delay of 2.000 seconds

starting at 40.340 seconds into the test. A delay of tasked waypose packets lasting greater than 1.300 seconds while in the move state followed by a delay of tasked waypose lasting between 0.000 and 0.700 seconds of the next hover task results in a quicker than normal transition through a maneuver such as in test (Vehicle 1, Test 167). This is an instance where three consecutive task waypose messages were published within their respective 1.000 second time slots but the difference between publishing times of the consecutive messages was significantly different than 1.000 second. A delay of tasked waypose packets lasting between 1.300 and 2.000 seconds while in the hover state followed by a delay of tasked waypose packets lasting between 0.000 and 0.700 seconds of the next move task results in additional time required to perform a maneuver such as on test (Vehicle 4, Test 61). In addition to maneuver timing, the duration of a delay of tasked waypose packets between 0.000 and 2.000 seconds while in the move state determines the amount of actual translation performed by the vehicle. The resulting translation is directly proportional to the time delay. Delays of between 0.000 and 2.000 seconds result in translations between 0.000 and $2 * \delta$ (where δ is the tasked change in position). If the delay is greater than 1.300 seconds, an additional maneuver that was not tasked is observed. An extended delay during a negative height change may result in the vehicle landing. An absolute initial position is received from the first subject pose callback but subsequent positions after launch are relative to the absolute target launch position. Thus, if an extended delay lasting 2.000 seconds occurs during a positive height change of δ then the vehicle will end up moving $2.000 * \delta$ upwards. The frequency of occurrence of this failure was: 57/510 (11.176%), 9/100 (9.000%), 21/110 (19.091%), 0/100 (0.000%), 0/3 (0.000%), 1/100 (1.000%). The ratios relating this type of failure compared to total failures per vehicle are: 57/78 (73.077%), 9/14 (64.286%), 21/33 (63.636%), 0/4 (0.000%), 0/1 (0.000%), 1/10 (10.000%).

A delay of command subject control state packets greater then 1.300 seconds results in an extra command subject control state message, a repeat of the particular state the vehicle was in during the delay. On test (Vehicle 1, Test 296) a command subject control state

delay of 1.911 seconds starting at 17.145 seconds into the test resulted in an extra move command subject control state message and an extra task waypose message. Exceptions to this behavior are witnessed in the presence of simultaneous delay in subject control state packets or simultaneous delay in tasked waypose packets. The former combination results with a normal amount of commanded and actual states but the vehicle exhibits increased time in the applicable delayed state followed by a quick transition through the next state this behavior is the more common behavior in the presence of multiple delayed topics where the delay exceeds 1.300 seconds but is less than 1.900 seconds for state, task, and status packets and/or the delay exceeds ten times the control rate for position and quad control input packets, while the latter situation results in a missed task packet and a missed command state packet. If an additional move state is received, the vehicle moves back towards the target position, this is notable in the cases where the vehicle drifted during the delay between reaching the target position and receipt of the second identical move command. The frequency of occurrence of this failure was: 61/510 (11.961%), 7/100 (7.000%), 22/110 (20.000%), 2/100 (2.000%), 0/3 (0.000%), 1/100 (1.000%). The ratios relating this type of failure compared to total failures per vehicle are: 61/78 (78.205%), 7/14 (50.000%), 22/33 (66.667%), 2/4 (50.000%), 0/1 (0.000%), 1/10 (10.000%).

Quad control input uses an arbitrator node to select between control input topics using a state machine. While flying the position control node converts subject position and tasked position messages into quad control input messages, which are converted to robot control input messages by a translation node. Robot control input messages are then converted to transmission packets by a protocol node. If the difference between successive quad control input time stamp updates is greater than ten times the control interval in seconds then a failure due to quad control input is annotated. Each vehicle should possess an exact minimum control interval. If not documented, this parameter can be approximated by testing with different control intervals. Such a test could be expensive if the vehicle is not constrained during the tests.

An absence of quad control input packets lasting greater than ten times the control interval without any other control packet delay results in abnormal not commanded translation. Quad control input packet delays are usually accompanied by subject position packet delays but a few exceptions such as (Vehicle 1, Test 399) and (Vehicle 4, Test 19) exists. A quad control input delay of 0.631 seconds starting at 36.049 seconds into the test, a subject pose delay of 0.860 seconds starting at 38.228 seconds into the test, and a subject status delay of 1.547 seconds starting at 35.133 seconds into the test was recorded during test (Vehicle 1, Test 107). The frequency of occurrence of this failure was: 58/510 (11.373%), 5/100 (5.000%), 29/110 (26.364%), 0/100 (0.000%), 0/3 (0.000%), 0/100 (0.000%). The ratios relating this type of failure compared to total failures per vehicle are: 30/78 (74.359%), 5/14 (35.714%), 29/33 (87.879%), 0/4 (0.000%), 0/1 (0.000%), 0/10 (0.000%).

Create a node that monitors for time lapse between control node updates. If the time lapse exceeds the expected time lapse in this case ten times the control interval for quad control input and 1.000 ± 0.300 seconds for command subject control state and tasked waypose, normal mission execution is paused and the vehicle is tasked to hover. Then after a reasonable time within the pause/hover state, e. g. 1.000 second, either normal execution is resumed, the vehicle lands then resumes, or the vehicle lands, shuts down, and the mission is restarted at the beginning. This monitor node should run at a higher control rate than the mission control nodes to minimize packet delays associated with the processing of this node. This is a proposed solution and not a solution that is currently implemented.

On the vehicle side as a safety measure if at all possible to reprogram, the vehicle should implement a hover maneuver whenever a control input is delayed beyond the minimum control input rate while in autonomous flight mode.

The histograms of Figures 2.16-2.21 use 10 bins. The unit for the bin values is seconds. The bin values were derived from scatter plots that visualize the same data. The x-axis is on a logarithmic scale to capture both small variations around the nominal value and to

capture large outliers. The y-axis is the number of tests that had a packet delay variation value closest to the respective bin.

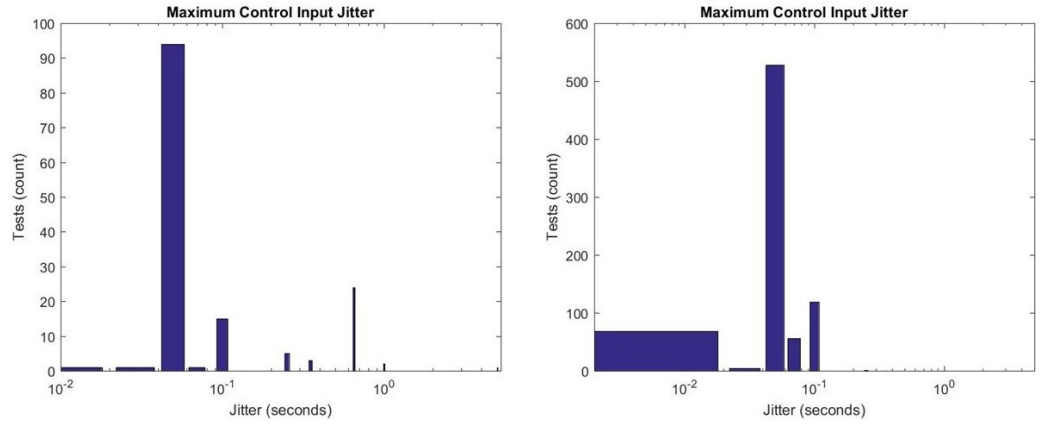


Figure 2.16: Maximum control input packet delay variation plots for failed (left) and successful (right) tests.

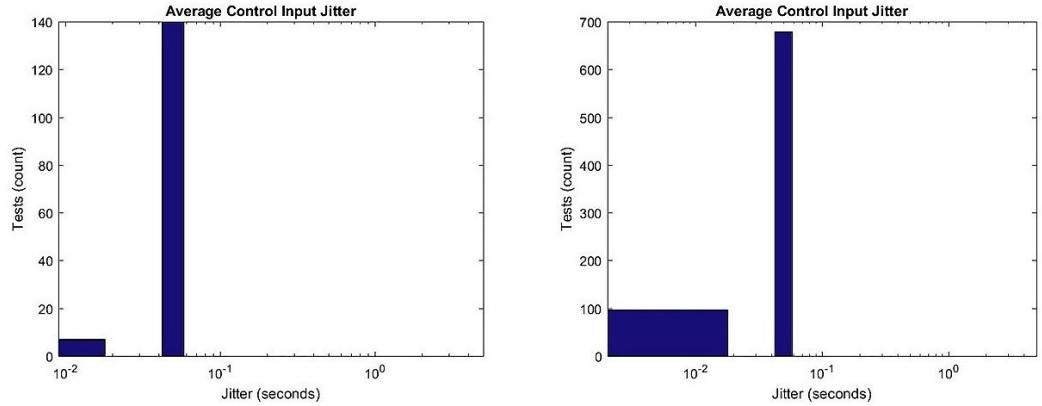


Figure 2.17: Average control input packet delay variation plots for failed (left) and successful (right) tests.

Figures 2.16 and 2.17 use these bins: 0.01 0.03 0.05 0.07 0.1 0.25 0.35 0.65 1.0 5.0. The left histogram of Figure 2.16 displays the maximum quad control input packet delay variation for the duration of each test between start of the launch command and prior to receiving the land command for failed tests while the right histogram displays this data for successful tests. There exist some variation around the nominal 50 ms for both successful and failed tests. The nearly 100 value around 10 ms is due to utilizing a 10 ms control interval for the ARDrone. Though difficult to see, in the left histogram of Figure 2.16 there is one failed test with a value in the 5.0 second bin. The relatively few failed flights with packet delay variations $>$ 0.1 in Figure 2.16 indicates that the majority of tests that failed due to packet delay variation failed due to sensor request delays which are reflected

in Figures 2.18 and 2.20. The histograms of Figure 2.17 illustrate the average quad control input packet delay variation for the duration of the test between start of the launch command and prior to receiving the land command for failed (left) and successful (right) tests. The average does not vary significantly for either failed or successful tests.

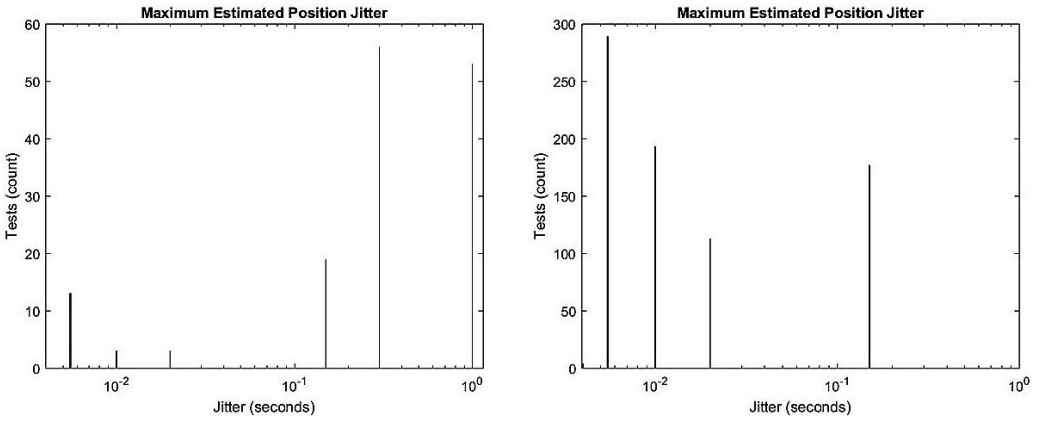


Figure 2.18: Maximum position estimate packet delay variation plots for failed (left) and successful (right) tests.

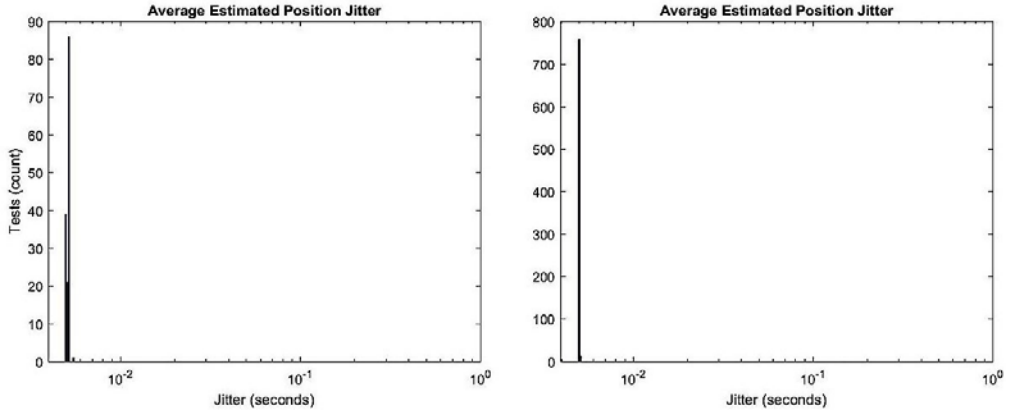


Figure 2.19: Average position estimate packet delay variation plots for failed (left) and successful (right) tests.

Figures 2.18 and 2.19 use these bins: 0.004 0.005 0.0051 0.0052 0.0055 0.01 0.02 0.15 0.3 1.0. The histograms of Figure 2.18 show the maximum subject pose packet delay variation for the duration of each test between start of the launch command and prior to receiving the land command for failed (left) and successful (right) tests. The nominal value is 5 ms. Over 100 failed tests include maximal position estimate packet delays that exceed 250 ms. Approximately 200 of the successful tests included maximal position estimate packet delays near 150 ms; this packet delay variation value of position estimate packets

did not result in discernable abnormal behavior to the observers. The histograms of Figure 2.19 show the average subject pose packet delay variation for the duration of each test between start of the launch command and prior to receiving the land command for failed (left) and successful (right) tests. The left histogram of Figure 2.19 demonstrates small variations in the average for failed tests; the existence of such variation in the generated scatter plot is why the respective bins were chosen.

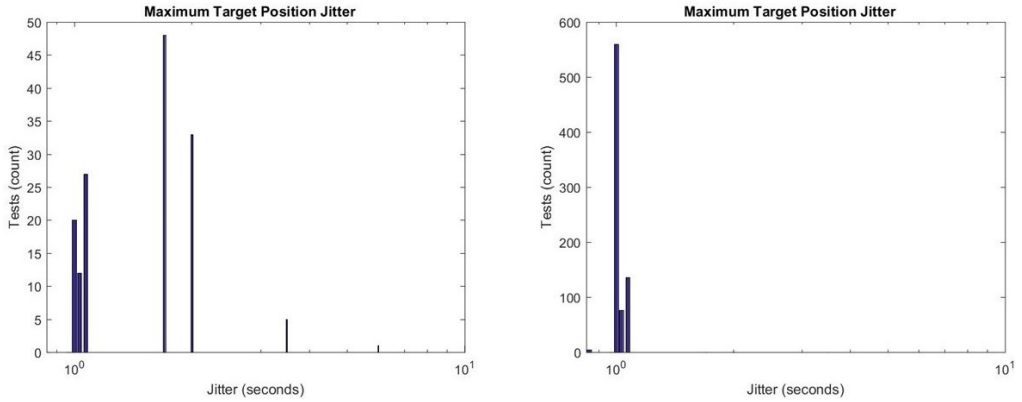


Figure 2.20: Maximum position target packet delay variation plots for failed (left) and successful (right) tests.

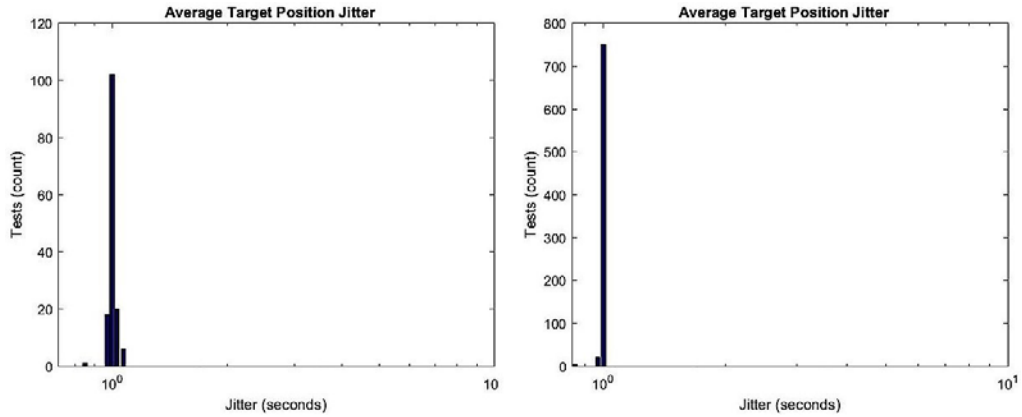


Figure 2.21: Average position target packet delay variation plots for failed (left) and successful (right) tests.

Figures 2.20 and 2.21 use these bins: 0.85 0.97 1.0 1.03 1.07 1.7 2.0 3.5 6 10. The histograms of Figure 2.20 show the maximum task waypose packet delay variation for the duration of each test between start of the launch command and prior to receiving the land command for failed (left) and successful (right) tests. The nominal value is 1 s. Around 80 tests include maximal task waypose packet delay variations that are $>$ 1250 ms. Some successful tests include maximal task waypose packet delay variations that are $<$ 1250 ms

but > 1 s. The histograms of Figure 2.21 display the average task waypose packet delay variation for the duration of each test between start of the launch command and prior to receiving the land command for failed (left) and successful (right) tests. The left histogram of Figure 2.21 demonstrates small variations around the nominal value for failed tests.

With a 5 ms transmission interval using a Cat5e cable the average packet delay variation is within ± 1 ms. With a 1 s transmission interval using paired XBee-Pros the average packet delay variation is within ±50 ms. Figure 2.22 illustrates these results.

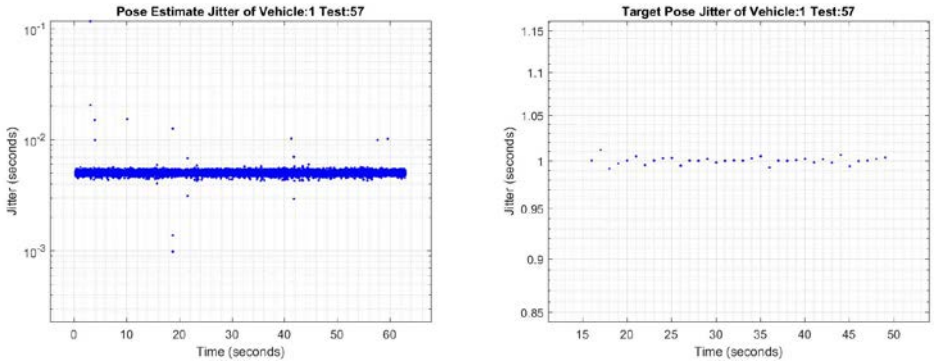


Figure 2.22: Position estimate and target, packet delay variation plots for successful test: (Vehicle 1, Test 57).

Abnormal behavior is observed for an Ascending Technologies Hummingbird if any one of these conditions are met: the average control interval jitter exceeds ± 2% of the desired control interval, the instantaneous control input packet delay variation is ≥ 250 ms, the average estimated position jitter exceeds 150 μs from the nominal 5000 μs, the instantaneous estimated position packet delay variation exceeds 250 ms, or instantaneous subject status packet delay variation exceeds 1250 ms.

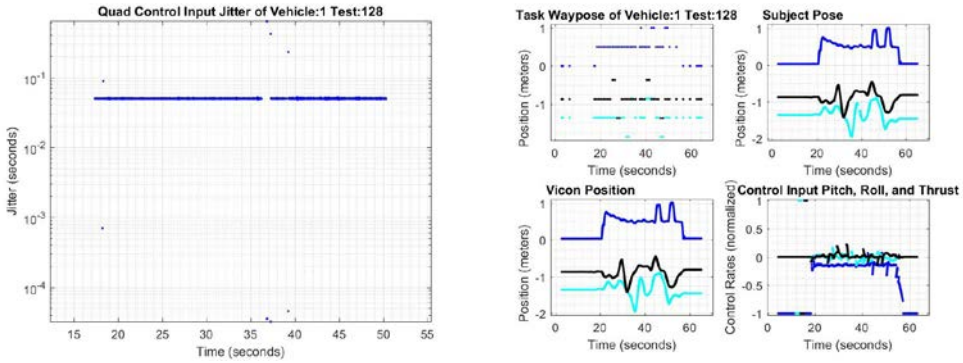


Figure 2.23: Control input packet delay variation and position plots for failed test: (Vehicle 1, Test 128).

2.7.3 External Polling Device

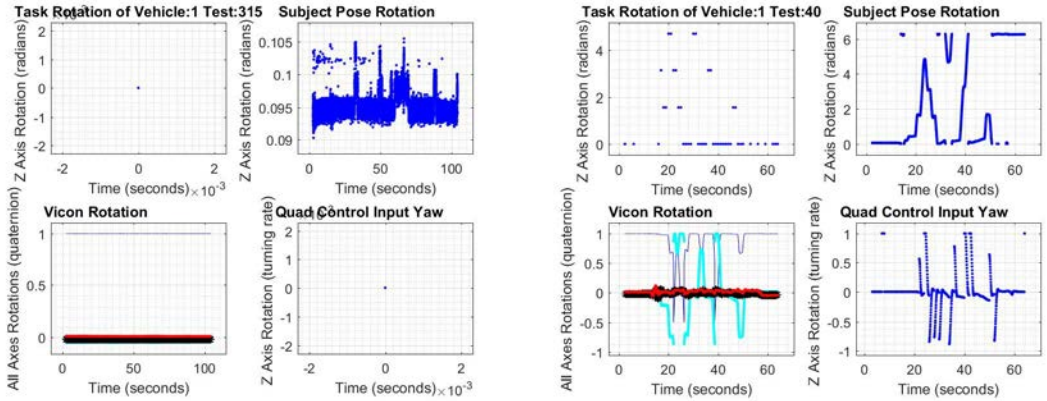


Figure 2.24: Rotation plots for both failed test: (Vehicle 1, Test 315) and successful test: (Vehicle 1, Test 40).

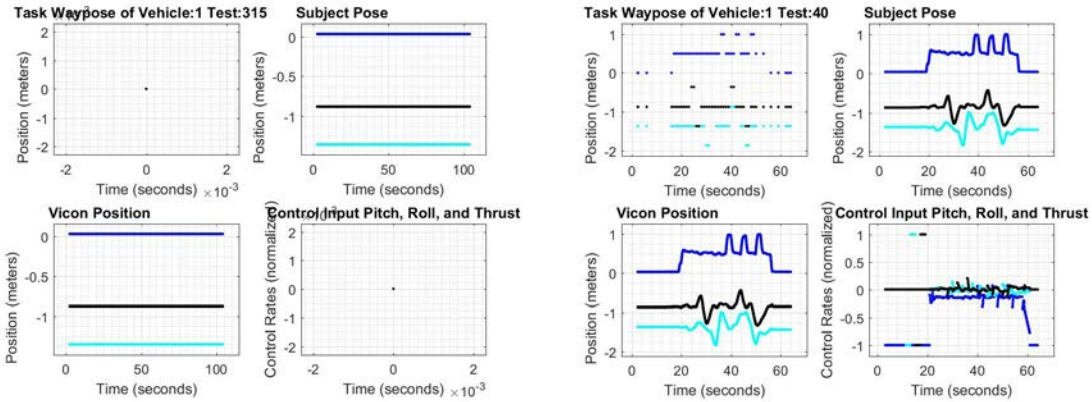


Figure 2.25: Position plots for both failed test: (Vehicle 1, Test 315) and successful test: (Vehicle 1, Test 40).

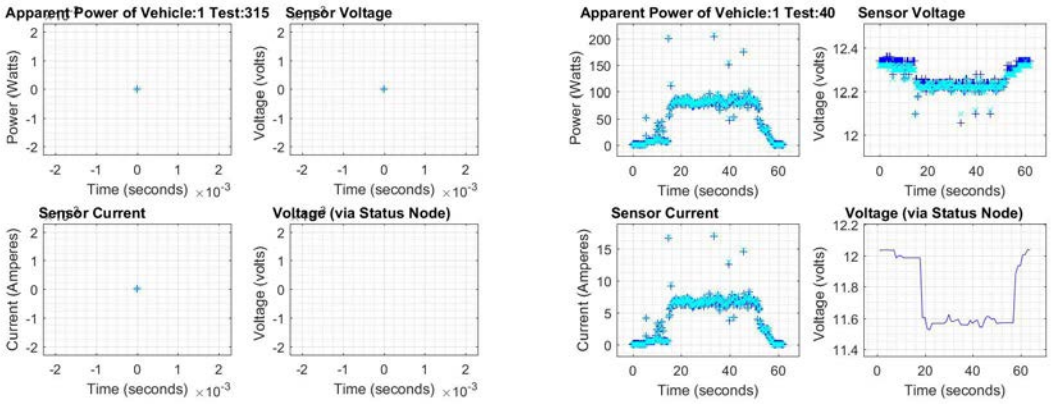


Figure 2.26: Electrical characteristic plots for both failed test: (Vehicle 1, Test 315) and successful test: (Vehicle 1, Test 40).

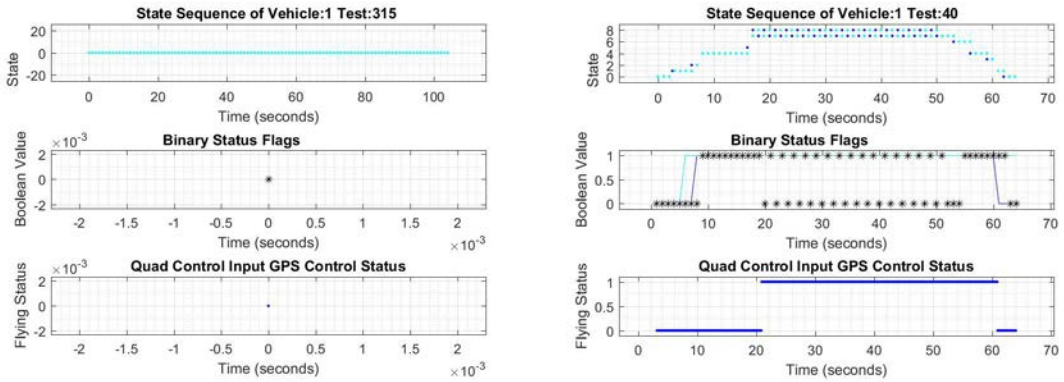


Figure 2.27: State and status plots for both failed test: (Vehicle 1, Test 315) and successful test: (Vehicle 1, Test 40).

Symptom 4 is observed. Motors turning on not observed at all and lack of battery or state change on the gui. This was witnessed on test (Vehicle 1, Test 315). There were 4/339 (1.180%) tests affected while this failure had a non-zero probability of occurring.

Th external polling device failure was caused by two misconfigured machines. The first of these machines was the virtual machine on the controlling computer. The networking settings of the virtual machine on the controlling computer were configured incorrectly. The VMWWare Network Adapter VMnet8 was bridged with the Ethernet adapter in Network Connections along with bridging the VMWare and Ethernet adapter in the network settings of VMWare. Only the latter of these two things are required. If the former is performed, the virtual machine will act as an IP administration machine on the network connected to the Ethernet adapter.

The second machine, a device in the laboratory, was polling for an administration machine and blocked communication from the controlling computer to the XBee-Pro transmitter during the third day of tests, the vehicle under test was a Hummingbird with Ascending Technologies safety propellers. The above plots show the link between the Vicon station and the controlling computer remaining active while no data was transmitted between the controlling computer and the XBee-Pro transmitter. Communication setup was verified as connected correctly and additionally it was verified that no other changes to the test environment were introduced between test (Vehicle 1, Test 314) and test (Vehicle 1, Test 315).

Two other graduate students were discussing problems with the device in question. Tester and the senior graduate student in the laboratory troubleshooted the device. After performing the following solution, tests were able to be executed, thus this was assumed to be the cause.

In the software associated with the device the target administrator IP address was included. This solution was implemented within 2 hours after the failure was detected. The networking settings of the virtual machine on the controlling computer remained misconfigured until attempts to integrate the NIMBUS laboratory autopilot with an ArduCopter were performed. The fix was conducted a short while after a demo resulted in an hexacopter, Ascending Technologies Firefly, crashing. No further failures of this type were recorded. As a precautionary measure against such re-occurrences a pre-flight check of devices connected to the indoor positioning system network and an online warningsystem that actively monitors for other devices connected to the positioning system network should be implemented.

2.7.4 Motor Failure

Performance of additional maneuvers than those commanded: unmanned aerial vehicle flips around one motor.

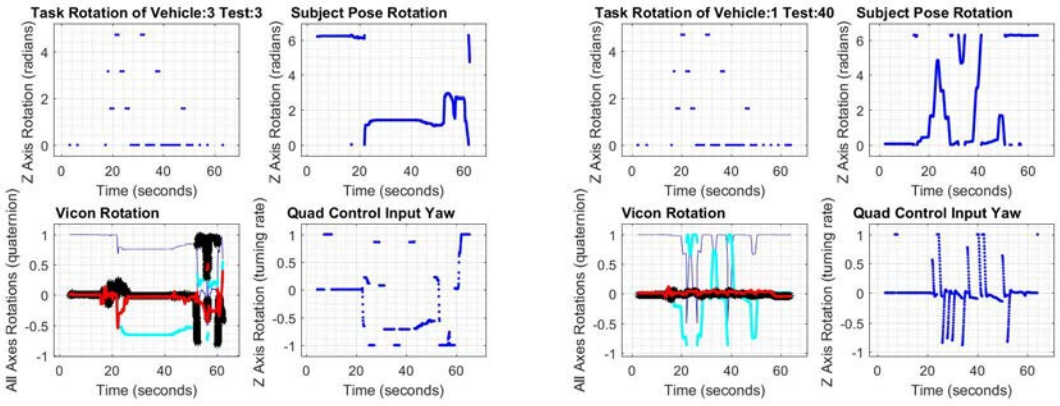


Figure 2.28: Rotation plots for both failed test: (Vehicle 3, Test 3) and successful test: (Vehicle 1, Test 40).

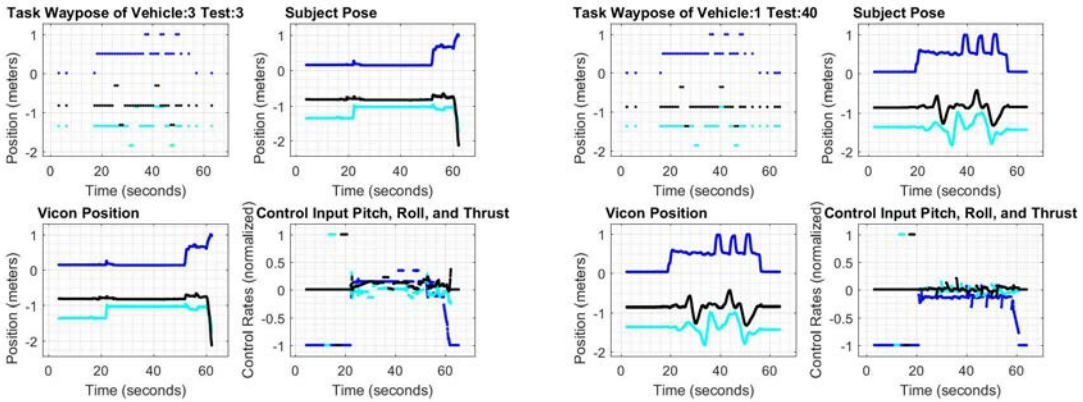


Figure 2.29: Position plots for both failed test: (Vehicle 3, Test 3) and successful test: (Vehicle 1, Test 40).

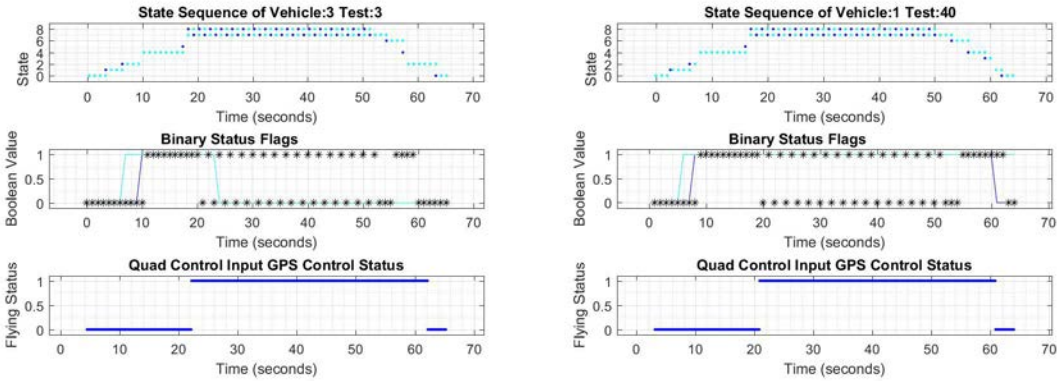


Figure 2.30: State and status plots for both failed test: (Vehicle 3, Test 3) and successful test: (Vehicle 1, Test 40).

An unmanned aerial vehicle flipping around one motor is caused by a damaged motor assembly. A witnessed technique in the laboratory when operating an ARDrone was to nudge the ARDrone while in flight if it went to an undesired waypose. This technique was done once on the first day while adjusting the way the ARDrone responds to the existing state and position controller. This resulted in a bent motor shaft in the motor nearest to the contact point. Prior to discovering this mechanical deficiency the vehicle was flown around 30 times, at least half of which ended with the vehicle flipping over in mid flight due the left back propeller gear disengaging from its respective motor gear mid flight. These failed flights resulted in the right back motor shaft bending as well. Both motors were replaced, no more flipping occurred. The fourth hummingbird tested flipped around a bad motor assembly during the pitch and roll test on test (Vehicle 3, Test 3).

Do not manually interfere with the flight of any vehicle no matter how safe or soft a

contact point may appear. Avoid crashing the unmanned aerial vehicle or performing any other such actions that may cause physical damage or shock to the unmanned aerial vehicle. Minimize exposure to motor controller circuits and rotation mechanisms such as gears or three phase induction motors.

2.7.5 Pitch and Roll Tests

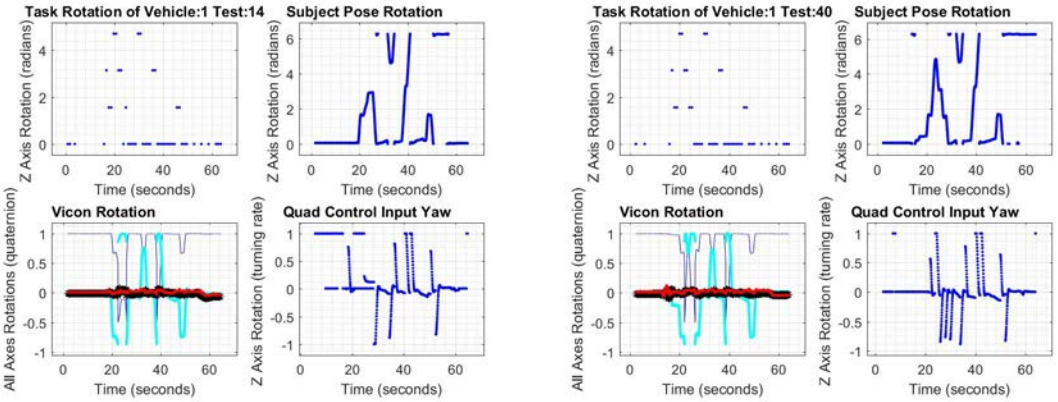


Figure 2.31: Rotation plots for both failed test: (Vehicle 1, Test 14) and successful test: (Vehicle 1, Test 40).

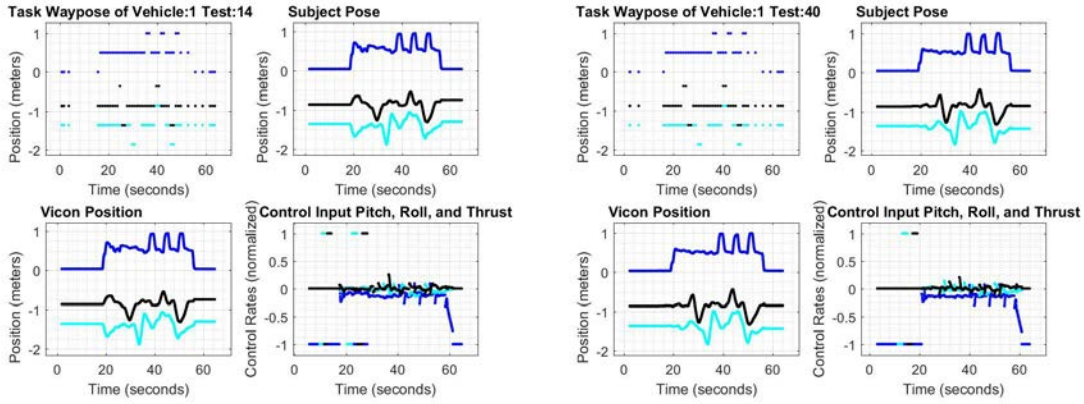


Figure 2.32: Position plots for both failed test: (Vehicle 1, Test 14) and successful test: (Vehicle 1, Test 40).

Symptom 5 is observed. The unmanned aerial vehicle performed additional maneuvers than those tasked: a pitch and roll test performed while airborne immediately after launch. On test (Vehicle 1, Test 14) the vehicle was described as jittering right after launch, which was due to the quick extreme variations in pitch and roll induced by the pitch and roll test while the vehicle was airborne. This failure affected 2/100 (2.000%) of tests while this failure had a non-zero probability of occurring.

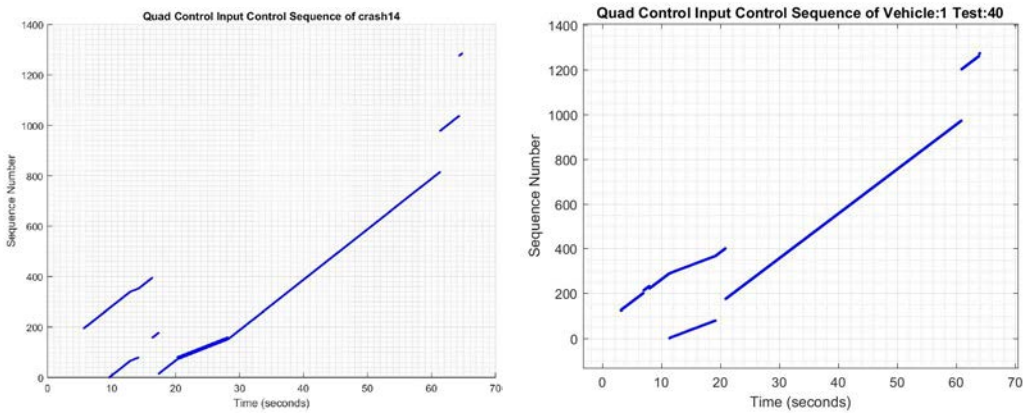


Figure 2.33: Control source plots for both failed test: (Vehicle 1, Test 14) and successful test: (Vehicle 1, Test 40).

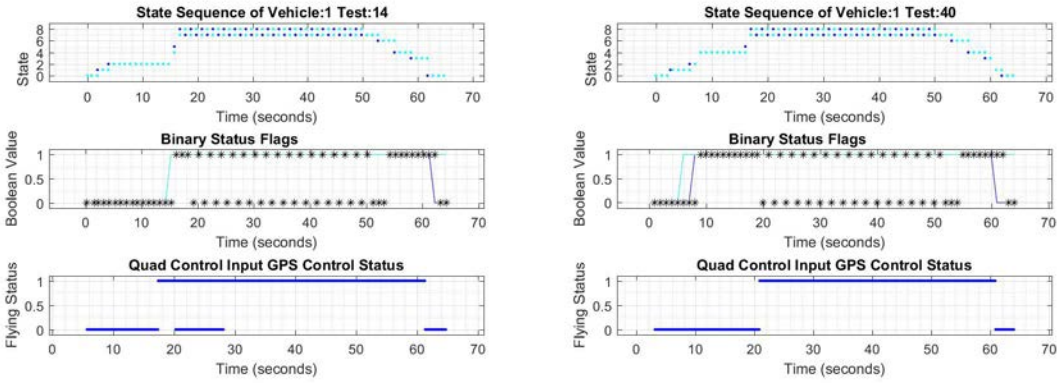


Figure 2.34: State and status plots for both failed test: (Vehicle 1, Test 14) and successful test: (Vehicle 1, Test 40).

The laboratory had a non fully functional procedure for testing maximum and minimum pitch and roll prior to flight. This procedure is implemented as the node uav_test of package collab test with the topic uav_test_input. A call to this node was enabled alongside a function that implements testing of maximum and minimum pitch and roll prior to flight, which is internal to the flight_testing node. Both implementations were enabled while performing the first 100 tests. This node was activated twice during the first 100 tests, which resulted in the vehicle (first Hummingbird with Ascending Technologies safety propellers) testing maximum and minimum pitch and roll while in the air immediately after launch. The uav_test node activates after receiving a robot_imu message. The update frequency of robot_imu messages in the corresponding logs is approximately 2.200 Hz. During the two tests that uav_test worked the publishing of a robot_imu message occurred shortly after the

publishing of an idle subject_ctrl_state while launching. The unmanned aerial vehicle did not enter the idle state until after the launch command state was received. The automatic transition from motors on to the idle state did not happen. One or more bytes were dropped from the applicable packet with the single attempt automatic state transition data and the packet failed the checksum. At this point in time, the testing code did not account for this state transition failure and assumed the automatic state transition did not fail. The internal pitch and roll test occurred while in the motors on state and not in the idle state as desired. While passing through the idle state during launch the uav_test code initialized and waited to receive a robot_imu message and performed the second pitch and roll test while in the air. During the tests that uav_test did not work subject_ctrl_state was updated prior to a delay variable associated with the node and the test did not commence.

After the second occurrence of the mid air jittering the flight testing code was analyzed for anything abnormal and the non functional call to uav_test was identified and removed. Further instances of this failure did not occur, thus, this was assumed to be the cause. This fault was corrected on the second test day by removing the uav_test node. Though no one else attempts to call this node in the laboratory, it is a parasitic node launched whenever hummingbird indoor is launched. This node runs in the background and publishes state updates to stdout. Cyber resources are wasted on this node. Removing the node from the checked in version of the hummingbird indoor launch file on the repository and removing the package as well would be a solution to gain back these wasted resources for other members of the laboratory.

2.7.6 Power Cord Interference

Symptom 5 is observed. Performance of additional maneuvers than those tasked: power cord wrapping around unmanned aerial vehicle caused additional maneuvers. This failure occurred during test (Vehicle 2, Test 10) and manual control was initiated.

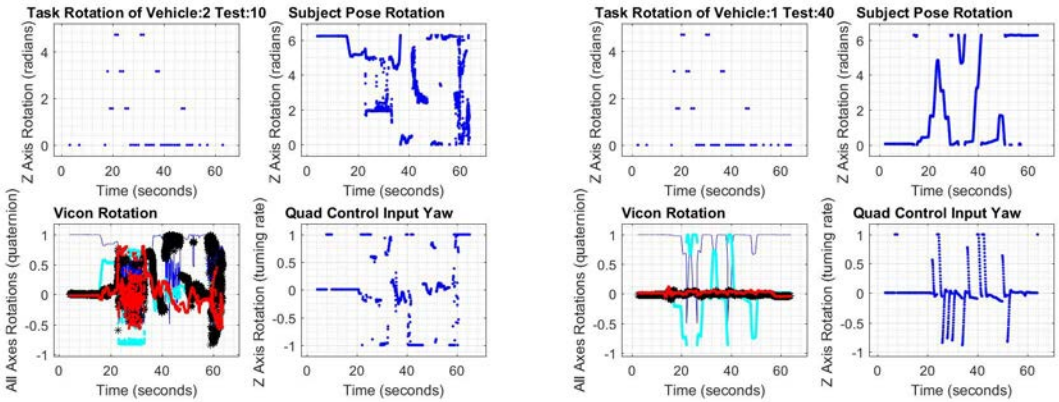


Figure 2.35: Rotation plots for both failed test: (Vehicle 2, Test 10) and successful test: (Vehicle 1, Test 40).

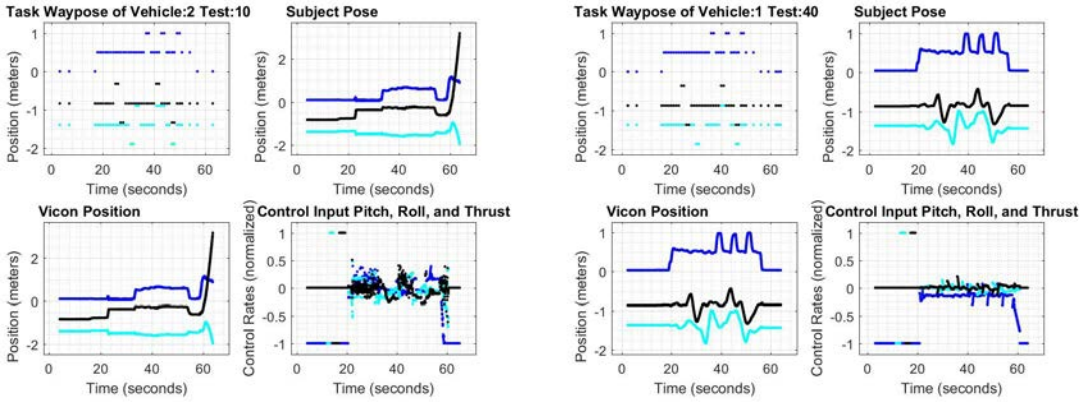


Figure 2.36: Position plots for both failed test: (Vehicle 2, Test 10) and successful test: (Vehicle 1, Test 40).

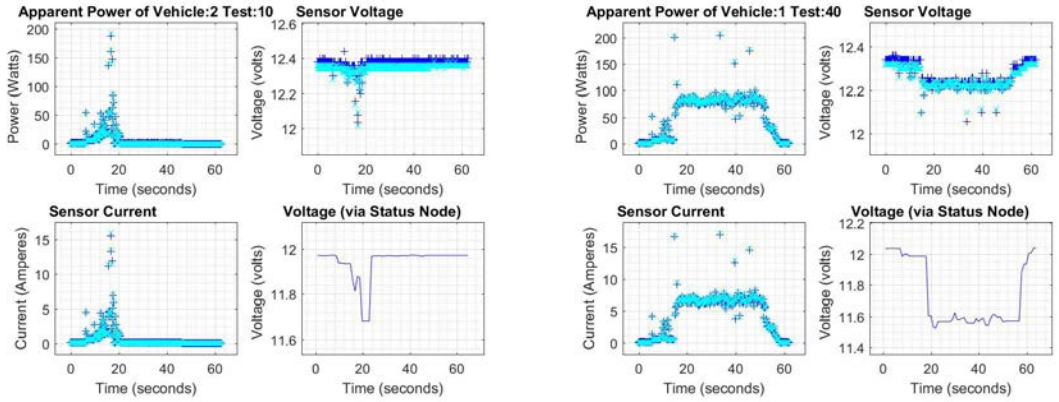


Figure 2.37: Electrical characteristic plots for both failed test: (Vehicle 2, Test 10) and successful test: (Vehicle 1, Test 40).

The power cord wrapped itself around a propeller on test day eight, the vehicle under test was the second Hummingbird with Ascending Technologies safety propellers. For the first vehicle under test, Hummingbird with Ascending Technologies safety propellers, the Deans T Connector on the vehicle required re-soldering at the beginning of each test day.

Successive weak solder jobs were performed causing the recurring need. This vehicle was endurance tested, 510 tests were conducted on this vehicle over a 5 day period. There were a total of 25 recorded instances via the response form of power cord interferences. Most of which, 22 tests, just involved a propeller momentarily impacting the power cord during the pitch and roll test or while landing and shutting off motors.

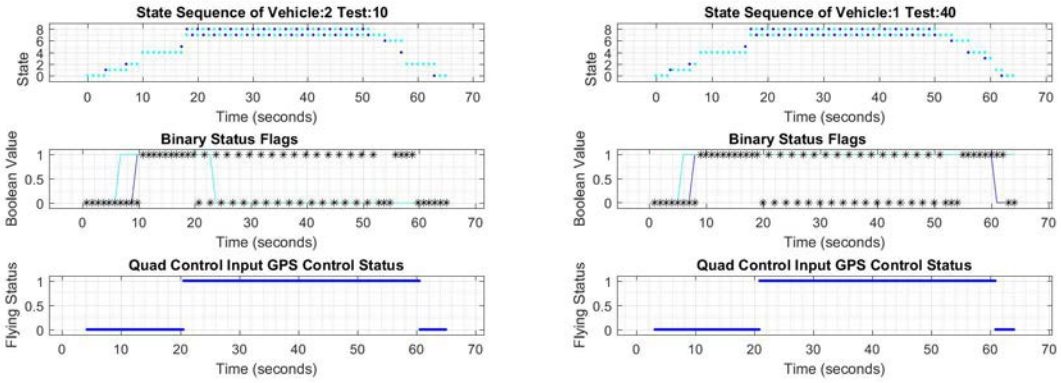


Figure 2.38: State and status plots for both failed test: (Vehicle 2, Test 10) and successful test: (Vehicle 1, Test 40).

Verify prior to starting each test sequence that the power cord is placed to minimize contact with the unmanned aerial vehicle while performing pitch and roll tests. This cause was visually apparent to all observers.

2.7.7 Rotation Failure

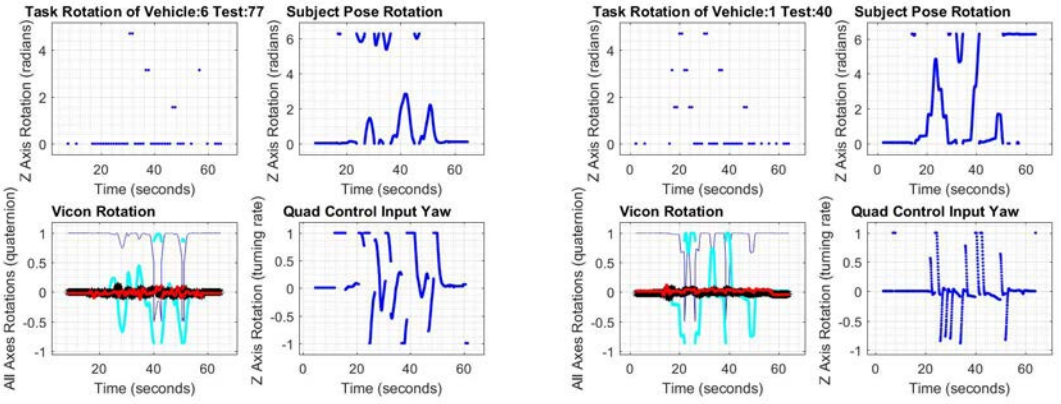


Figure 2.39: Rotation plots for both failed test: (Vehicle 6, Test 77) and successful test: (Vehicle 1, Test 40).

The ARDrone did not complete a full set of rotations during any test run. Therefore, technically there was something wrong with every test. The amount of task related failures indicated in Figure 2.39 for the ARDrone indicates there were additional failures on those specific tests.

2.7.8 Serial Communication Lost

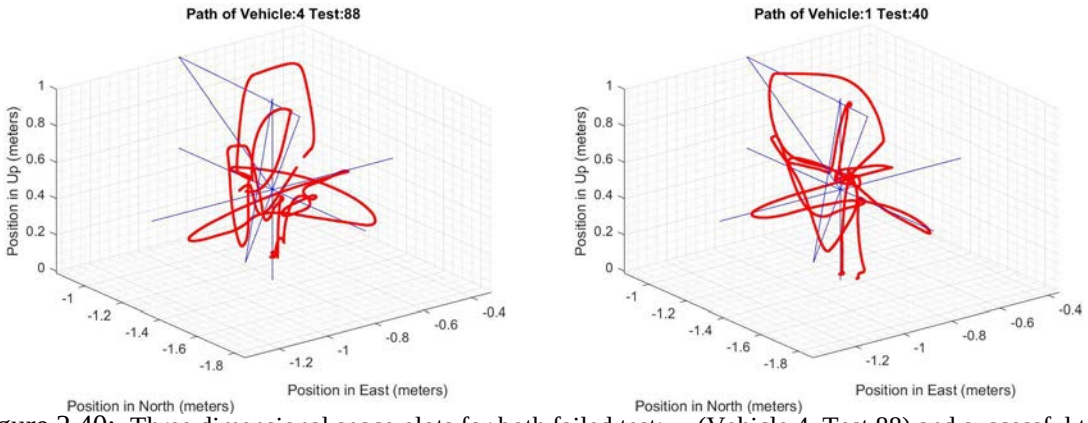


Figure 2.40: Three dimensional space plots for both failed test: (Vehicle 4, Test 88) and successful test: (Vehicle 1, Test 40).

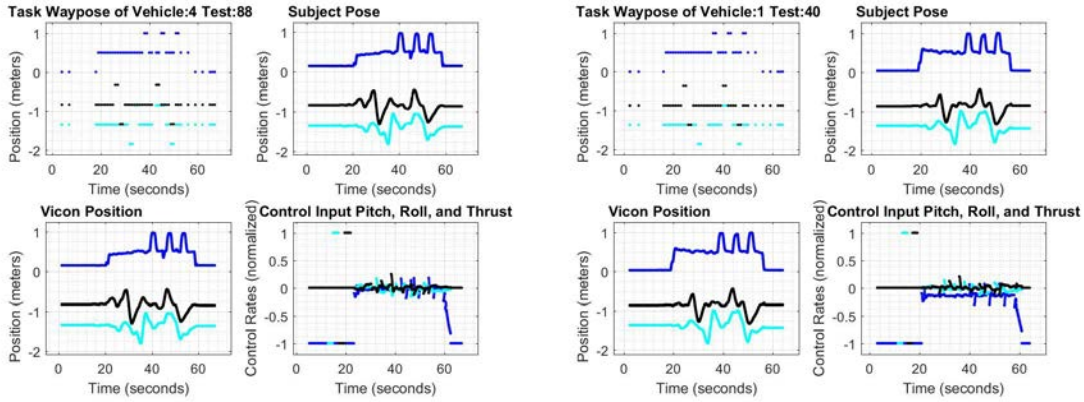


Figure 2.41: Position plots for both failed test: (Vehicle 4, Test 88) and successful test: (Vehicle 1, Test 40).

Symptom 1 is observed. The unmanned aerial vehicle drops to the floor while performing a maneuver then proceeds. This was observed for losses of serial communication during flight. Only losses of serial communication during flight were considered failures although there were twice as many occurrences while grounded. Test (Vehicle 1, Test 155) and test (Vehicle 4, Test 88) suffered a loss of serial communication while in flight while test (Vehicle 1, Test 190) suffered a loss of serial communication while grounded.

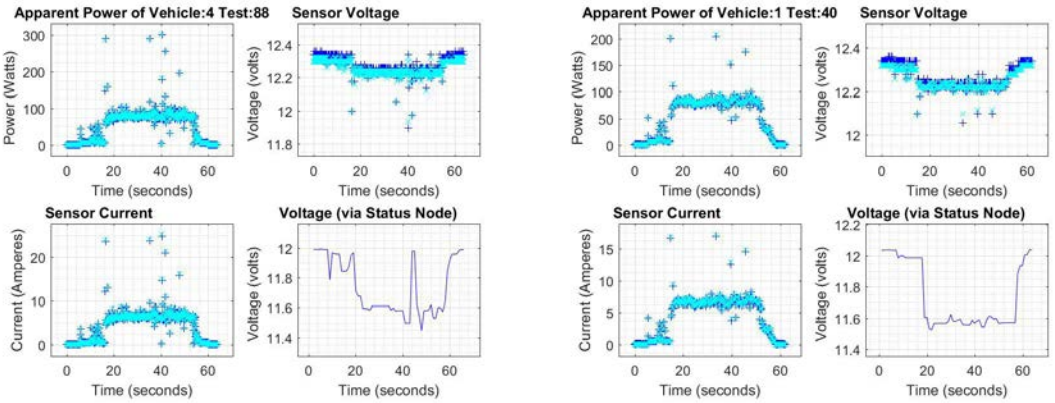


Figure 2.42: Electrical characteristic plots for both failed test: (Vehicle 4, Test 88) and successful test: (Vehicle 1, Test 40).

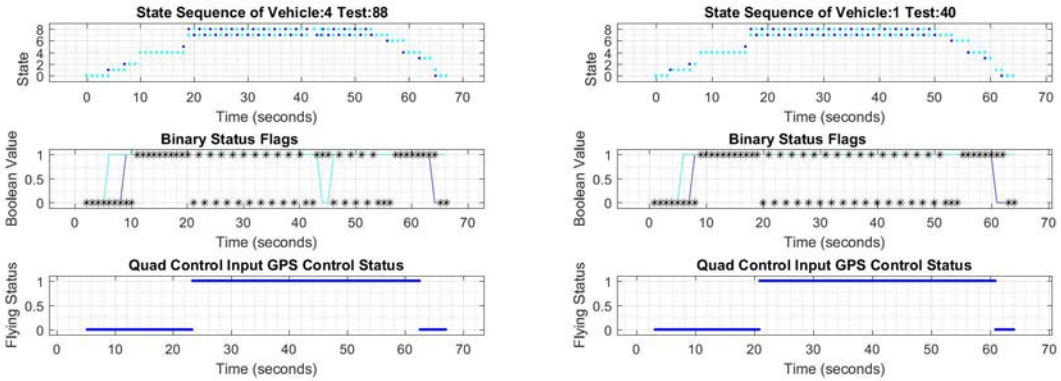


Figure 2.43: State and Status plots for both failed test: (Vehicle 4, Test 88) and successful test: (Vehicle 1, Test 40).

A loss of serial communication will ground a vehicle in flight. Losses of serial communication occur while grounded as well and can occur at any time. These events are automatically recovered and elapsed recovery time is between 3 and 5 seconds. These losses are infrequent, six occurrences in 923 tests, but do occur. Two of these events occurred while in flight, two occurred prior to flight, and two occurred after landing. An external interrupt may cause the serial device to momentarily halt transmission and receiving. Another probable cause may be delays in robot tx data.

If using an USB radio device, actively monitor for pipe halts and/or stalls. Send a warning if this occurs. Since the pipe is halted, a safety action on the controller end is unable to be performed and the autonomous aerial vehicle will fall/land. Therefore, the following solutions are offered.

One solution would be to execute commands from a controlling station that will not be interrupted by the operating system of the controlling station under any circumstance. This may be accomplished by adjusting process priorities, running a native OS instead of a virtual machine, and disabling internet connections, but may be difficult to actually set the test program as a process that cannot be interrupted. Another way to implement this is with a real time operating system connected to enough memory to run Ubuntu and ROS as a controlling station. This proposed solution does not require adding extra weight to the vehicle itself but would be difficult to implement. Obtaining RTOS capable hardware, RTOS software, and attaching this to a relatively large memory storage device might be expensive.

A second solution would be to remove the wireless serial communication altogether. A way to accomplish this would be to provide a rewritable flash storage device on the unmanned aerial vehicle to store controlling procedure while using a wired serial connection between flash storage device and unmanned aerial vehicle processor instead of a wireless connection. In this configuration, the controlling station could establish a remote connection to the rewritable flash storage device to launch all controlling nodes on the device while keeping bag recording on the controlling station. Instead of using a remote connection to relaunch nodes, the program could be started by toggling a switch on a remote control device such as a Futaba Controller. Additional code would have to be written to translate the toggle switch position into a set of commands that starts program execution. Using a rewritable flash storage device would add additional weight, such a device capable of running Ubuntu 14.04 and ROS would require its own fan adding further weight.

2.7.9 Subject Status PID Waypose

No observable symptom. Abnormal waypose status flag propagation in the bagfile. This anomaly is visible in the corresponding data for test (Vehicle 1, Test 106).

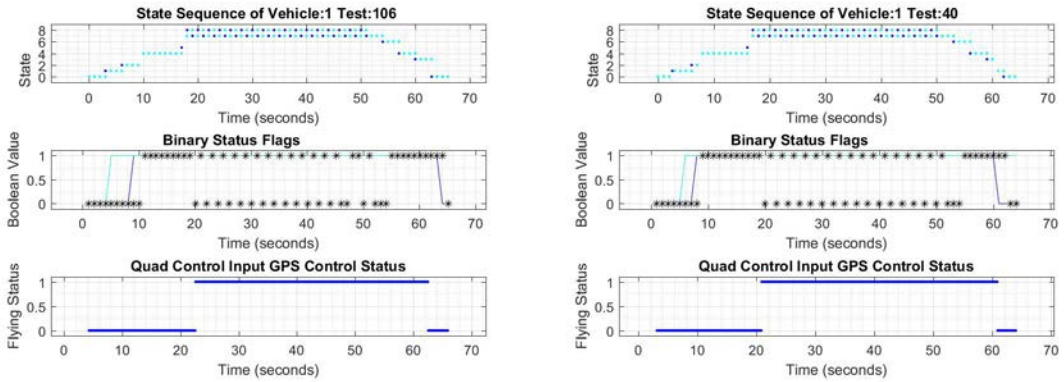


Figure 2.44: State and status plots for both failed test: (Vehicle 1, Test 106) and successful test: (Vehicle 1, Test 40).

An anomaly that does not effect operation of the unmanned aerial vehicles but indicates the presence of a delay is subject status field waypose status flag updates not reflecting actual conditions. This is caused by delays in subject status messages, delays in robot status messages, delays in robot gps messages, and delays in pid waypose messages. These delays affects the value of the waypose status field. The waypose status field is set if a pid waypose message is received. If a pid waypose message is delayed and not received within 1.000 second waypose status is set to idle, otherwise it is set to active. The time difference is determined by subtracting the most recent pid waypose time stamp received from the current subject status time stamp. A new subject status time stamp is set if both a robot status message and a robot gps message are received. Thus, delays in subject status packets as well as delays in pid waypose packets result in this value being incorrectly set. The waypose status field is used only as a launch state guard condition in the robot control flow thus incorrect values of waypose status do not affect robot operation in states other than currently in the idle state and requesting a launch state. There are a total of 56 instances of this anomaly for all Hummingbirds that affected the first four Hummingbirds tested: 40/510 (7.843%), 3/100 (3.000%), 12/110 (10.909%), 1/100 (1.000%). This anomaly is not present in the tests for the fifth Hummingbird due to implementing the synchronization described in the solution section prior to testing the fifth vehicle. All ARDrone tests included this anomaly due to utilizing a different control frequency.

Verify all control rates of arbitrated control inputs in the cascaded launch files are at a similar rate value, ideally 30 Hz. This anomaly was identified when perusing the gathered data. The cause was identified by noting recorded delay times and analyzing the code associated with the subject status node, the code associated with nodes that depend on subject status, and the code associated with nodes that subject status depends upon. Solutions to minimize abnormal delays are discussed in the Control Packet Delay section.

2.7.10 Vicon Communication Delay

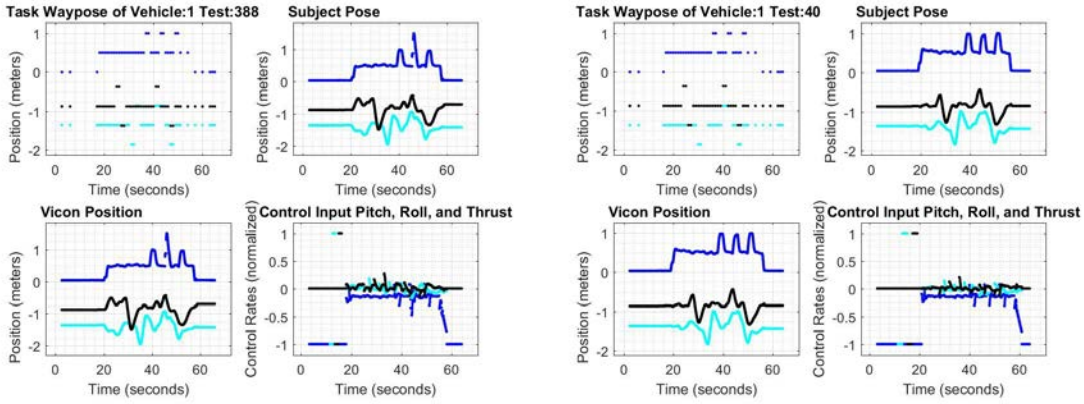


Figure 2.45: Position plots for both failed test: (Vehicle 1, Test 388) and successful test: (Vehicle 1, Test 40).

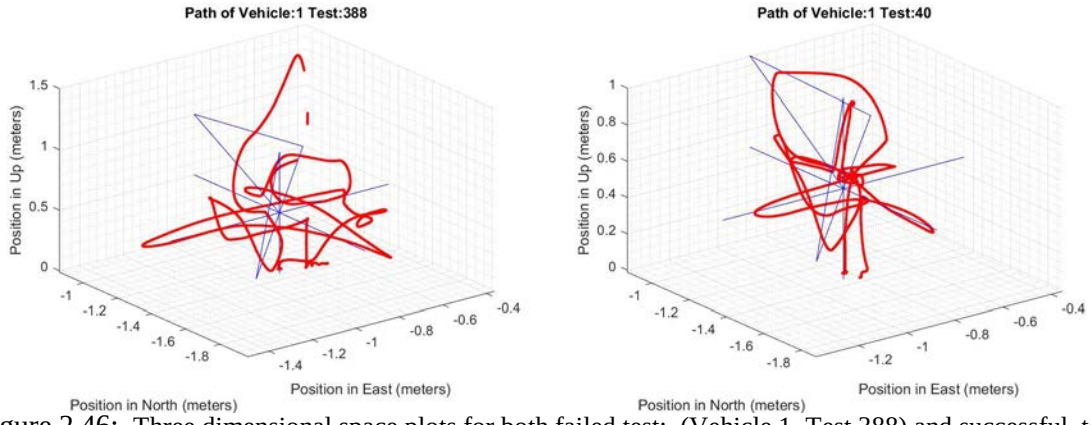


Figure 2.46: Three dimensional space plots for both failed test: (Vehicle 1, Test 388) and successful test: (Vehicle 1, Test 40).

Symptom 2 or 3 is observed. Either additional time taken to perform a maneuver or a quicker than normal transition through a maneuver is observed depending on if the vehicle has started executing the current task, if the vehicle is in the process of executing the current

task, or if the vehicle is near completion of the current task when the Vicon signal is momentarily dropped. A test where this was observed is test (Vehicle 2, Test 59). Symptom 5 is observed, performance of additional maneuvers than those tasked: the vehicle continues to execute a translation movement during a Vicon signal delay with a duration that exceeds the time required to execute the current task such as in test (Vehicle 1, Test 388).

Object is hidden from tracking cameras. Whenever an object is between the target object and a Vicon camera the target object is occluded and no longer able to be tracked while the occlusion remains. The reflective material on one or more Vicon markers has considerable wear. The applicable marker(s) does(do) not contribute to the object identification and the object is considered occluded. External interrupts from operating system running on Vicon station. If the operating system is not scheduling the required run time for Vicon Tracker to run properly, which Windows update sometimes causes, errors associated with Vicon such as occluded objects may occur. On two consecutive days the same vehicle was tested in the same location using the Vicon tracking system. There was no visible noise in the tracking system on either day. The first day multiple losses of Vicon were recorded while on the second day no losses of Vicon occurred. The cause of this was attributed to a present available windows update on the first day and the absence of such on the following day. Automatic prediction, of frequent Vicon communication lost is determined by visualizing noise in the Vicon Tracking window. Vicon communication lost messages are sent in real time but otherwise the receipt of such messages cannot be predicted prior to flight unless there is something visually wrong with the Vicon markers, considerable wear or misalignment. The frequency of occurrence of this failure was: 15/510 (2.941%), 5/100 (5.000%), 5/110 (4.545%), 2/100 (2.000%), 1/3 (33.333%), 9/100 (9.000%).

If there is visible noise in the Vicon Tracking window, recalibrate Vicon. Otherwise, halt tests, disconnect from Vicon and restart host machine and Vicon Tracking System. Run only the minimal amount of programs on both Vicon Tracker computer and host machine while performing mission.

2.7.11 Vicon Lost Object

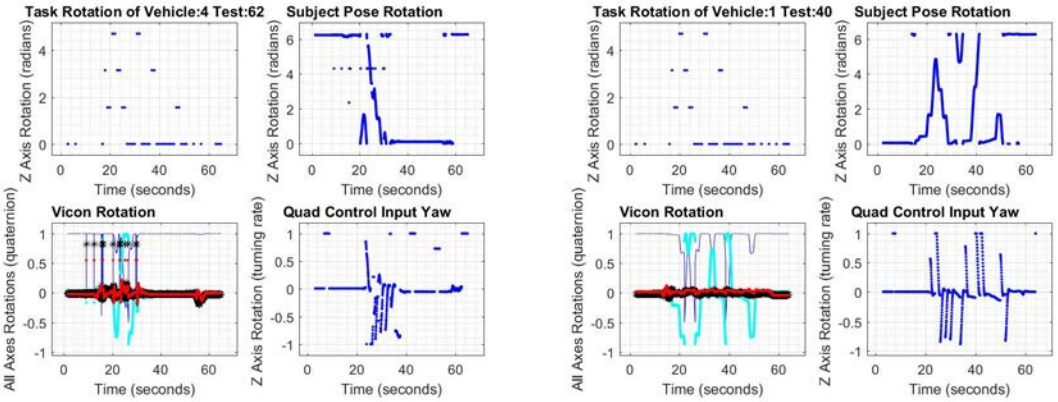


Figure 2.47: Rotation plots for both failed test: (Vehicle 4, Test 62) and successful test: (Vehicle 1, Test 40).

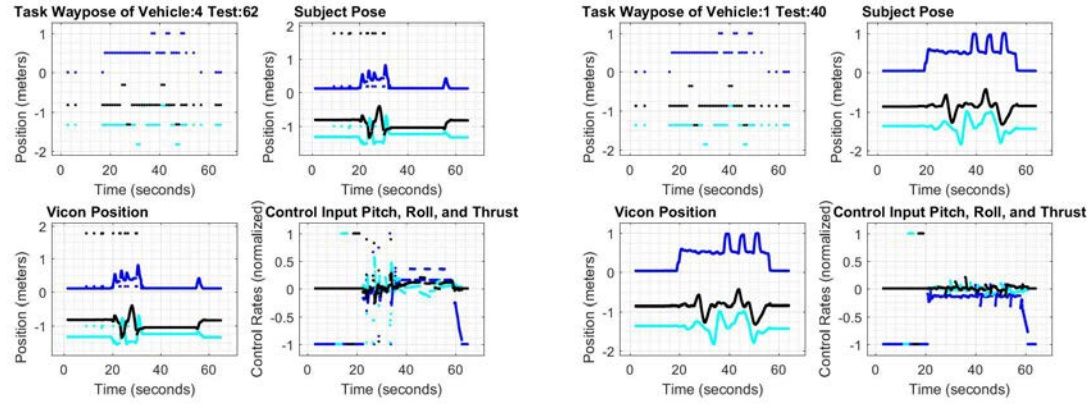


Figure 2.48: Position plots for both failed test: (Vehicle 4, Test 62) and successful test: (Vehicle 1, Test 40).

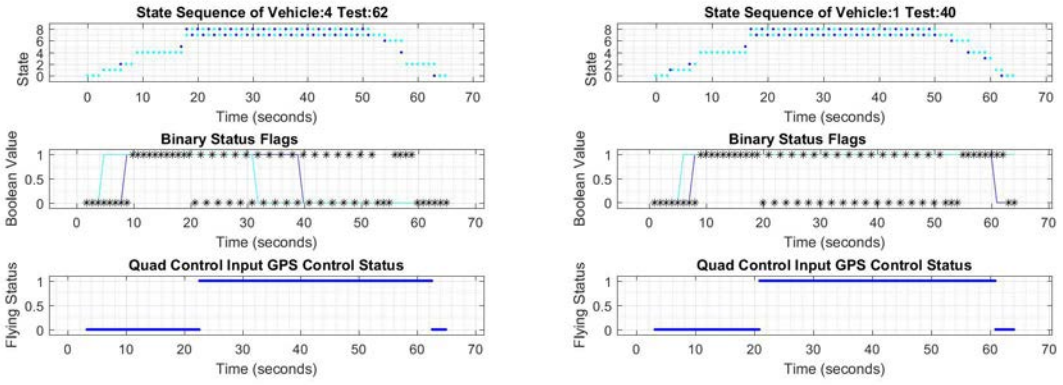


Figure 2.49: State and status plots for both failed test: (Vehicle 4, Test 62) and successful test: (Vehicle 1, Test 40).

Symptom 5 is observed. Performance of additional maneuvers than those tasked: a vehicle may exhibit erratic behavior in flight while using Vicon, if the objects identity in Vicon is lost.

Vicon marker comes loose or moves, during flight, Vicon requires recalibrating, or one or more reflective Vicon markers has considerable wear. A loose marker was identified after a test where multiple Vicon communication lost messages was received. After repositioning the marker, Vicon communication lost messages did not reoccur. Tests on the same vehicle without any other modifications were performed while there was noise in the Vicon Tracking window and immediately after calibrating Vicon. There were frequent Vicon communication lost messages prior to the calibration and none after the calibration. A marker with considerable wear to the reflective material was intentionally placed on a vehicle not implementing the flight testing procedure and numerous Vicon communication lost messages were received while the system was active. The reflective material on the Vicon markers can suffer tears from crash landings, frequent contact with the downward facing side of propellers during flight, or any other such collisions with sharp objects. If there is considerable reflective material wear the object can be mistaken as another object. In several cases, (Vehicle 5, Test 12 and 13), the loss of a Vicon object caused a repeated idle tasked waypose to be sent prior to launch. The most severe case was test (Vehicle 4, Test 62) where the vehicle was repeatedly lost and regained in Vicon. One of the observers remarked that the vehicle looked like it was having a seizure, after observing this behavior for several seconds manual control was initiated. The frequency of occurrence of this failure was: 0/510 (0.000%), 17/100 (17.000%), 6/110 (5.455%), 20/100 (20.000%), 1/3 (33.333%), 1/100 (1.000%).

Verify Vicon markers are correctly positioned, secure, adequately covered with reflective material required for Vicon tracking, and in a unique configuration for the vehicle under test prior to testing the vehicle. If this behavior persists the Vicon Tracking system should be calibrated. Vicon calibration can be implemented whenever there is visible noise in the tracking system or as a precautionary measure on a schedule such as every two weeks or after a number of hours in use since the last calibration.

2.8 Position Error Analysis

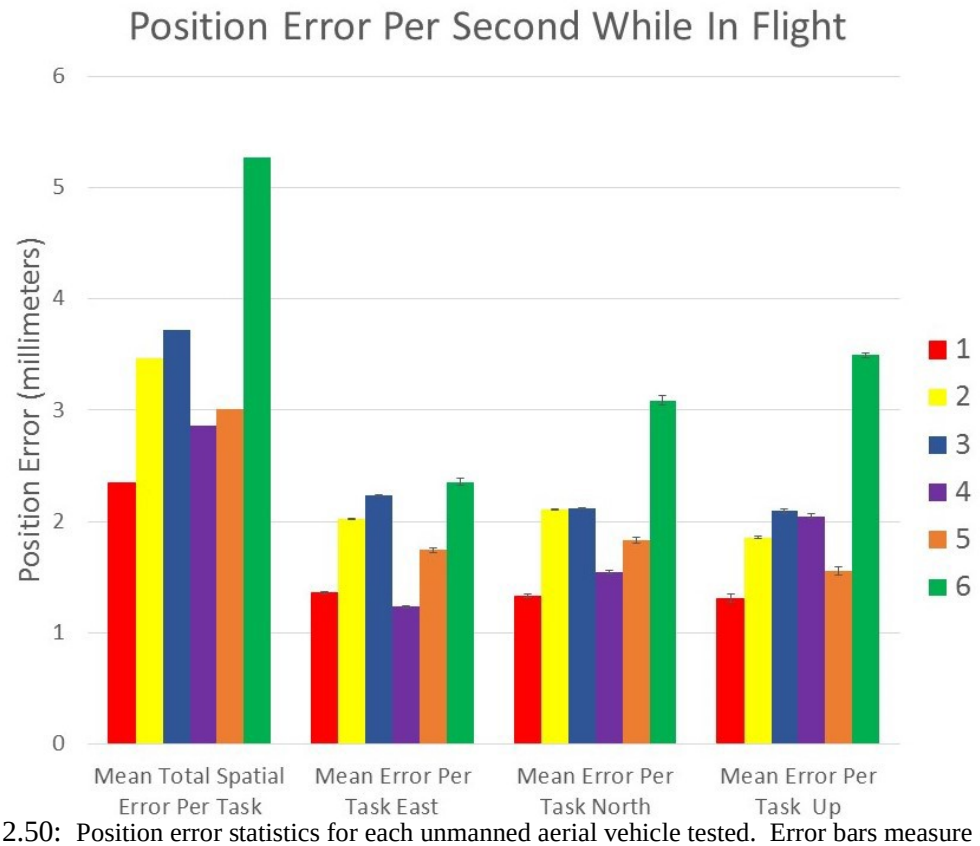


Figure 2.50: Position error statistics for each unmanned aerial vehicle tested. Error bars measure one standard deviation in both directions from the mean.

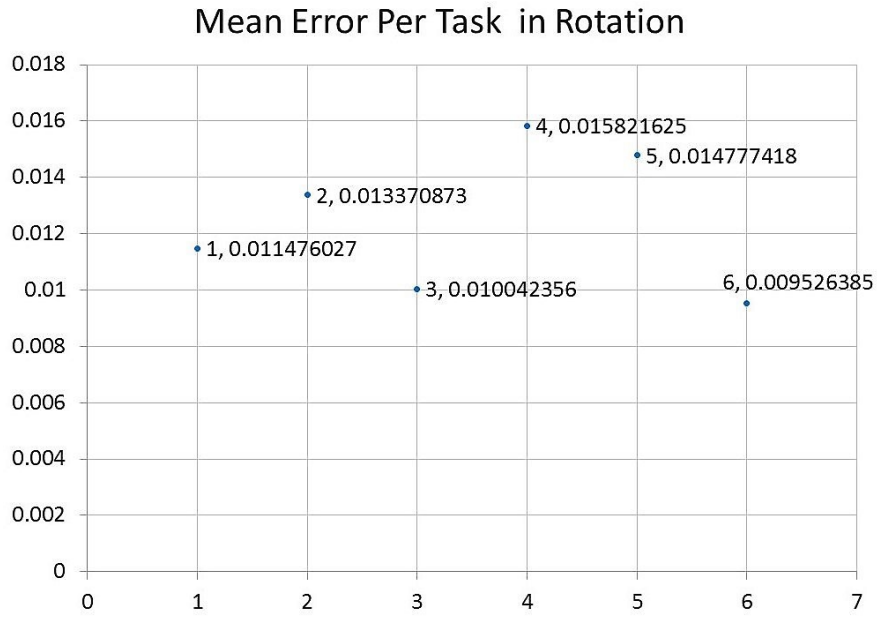


Figure 2.51: Rotation error statistics for each unmanned aerial vehicle tested.

Two different position controller's are available in the NIMBUS Laboratory. One that controls position and velocity and one that controls position. The latter was chosen for these tests, since automatic turning off of the motors was desired and at the time the flight testing source code was written the position and velocity controller was unable to reliably turn off the unmanned aerial vehicle's motors after landing.

A set of 30 tests, 10 tests for a Hummingbird with a different propeller type, was conducted on the first day of testing. From the analysis of these tests it was determined that the value of δ needed to be increased and having a uniform starting location would remove a variable in position analysis.

Throughout the tests of Hummingbirds and the ARDrone initial launch height achieved varied on each launch from barely above the floor to a meter above the floor. Every landing is unique. The set of maneuvers is designed such that the quadcopter ideally returns to its initial position prior to landing. Launch and landing positions differed on every test run by at most 37.160 cm in the x direction and 51.910 cm in the y direction for the Hummingbirds. The maximum difference in the magnitude of launch and landing positions for the ARDrone was 97.660 cm in the x direction and 158.360 cm in the y direction. An imbalance in individual motor strength resulted in each quadcopter being at slightly different rotations than the starting rotation when launching. Therefore, launching and landing rotations differ between 31.510 and 96.250 degrees for vehicles where the difference in motor strength is barely noticeable. In one vehicle, the second Hummingbird with Ascending Technologies safety propellers, there was noticeable differences in motor strength during the pre-flight test of the motors this resulted in a 178.300 degree difference in the worst case. Jenny (Ascending Technologies normal propellers) had the most difficulty landing.

Actual position versus tasked position is calculated by synchronizing the first in flight task with the nearest actual position rostopic time field. A task array of over 12000 entries is generated from the 36 in flight tasks with linear curves connecting previous and current task positions. A parabolic curve connecting previous and current task positions may result

in smaller calculated position errors. Actual position is subtracted from the target position. The similarity between 2*π and 0 is compensated for when calculating the difference between targeted and actual position. The mean absolute errors are calculated for each test flight. The mean for each Cartesian axis error for the vehicle under test, the mean of the total spatial error for the vehicle under test, and the corresponding variances are calculated. Each in flight task takes 1 second to complete, which is equivalent to the poll status update rate. Thus, the entire flight total position errors (centimeters) are divided by the test execution time and multiplied by 10 to get a value in millimeters for the mean position error per task. These values are presented in Figure 2.42. Per task rotation values are not multiplied by 10, both recorded values are in radians. These values are recorded in Figure 2.43.

The Hummingbirds exhibit a mean position error per task between [1.231, 2.236] millimeters in the horizontal plane and a mean position error per task between [1.308, 2.091] millimeters in height. Out of the vehicles tested the ARDrone mean position error for all fields exceeded that of the Hummingbirds. The difference in mean total spatial position error per task between the ARDrone and the Hummingbirds is between [1.543, 2.925] millimeters. Mean total spatial position error per task for the ARDrone is 5.267 millimeters.

For flights with control packet delays the position errors were greater than for flights without control packet delays. This increase in position error is due to the abnormal behaviors discussed in the Control Packet Delay section, abnormal translations and abnormal delays in a given target position. A loss of serial communication during flight as well as a loss of Vicon communication leads to increased position errors. Furthermore, any flight exhibiting symptoms 1-5 added to the total error between target and actual positions.

2.9 Power Analysis

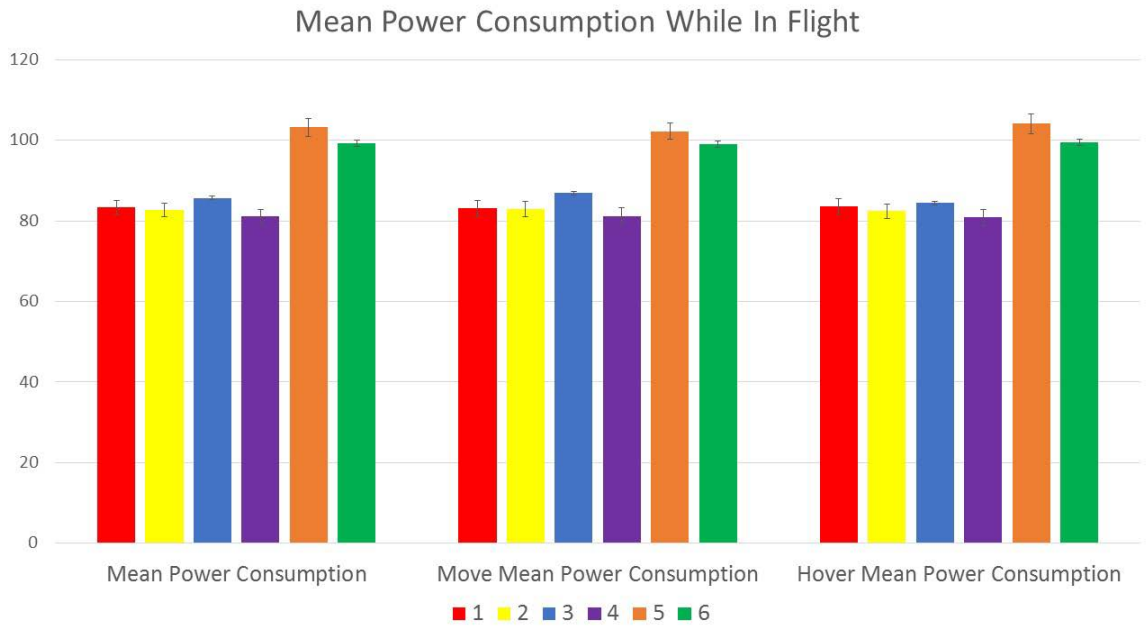


Figure 2.52: Mean power consumed for each vehicle tested. Error bars measure one standard deviation in both directions from the mean.

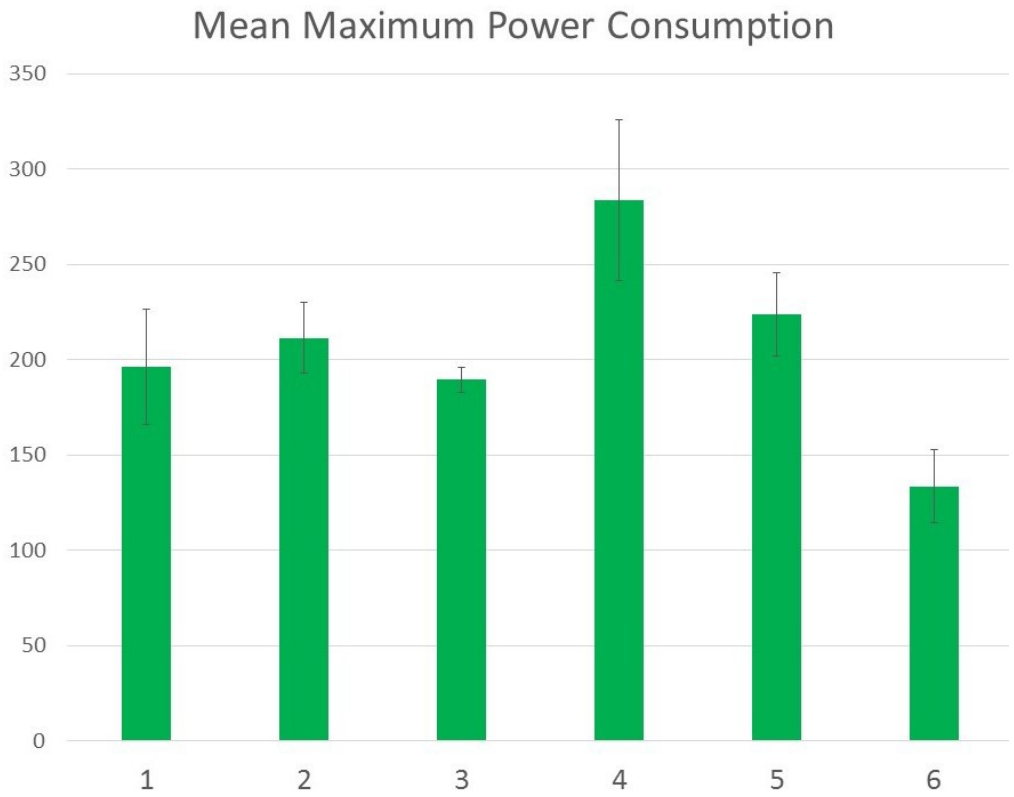


Figure 2.53: Maximum power consumed for each vehicle tested. Error bars measure one standard deviation in both directions from the mean.

2.9.1 Problems Encountered

Even while using the switching DC power supply power spikes were recorded. These noisy values were filtered out prior to calculating mean and maximum power values. Voltage values outside of 12.000 ± 3.000 volts are replaced with the average of the nearest two voltage values as long as those two values are also within band otherwise the value is replaced by the previous value. Current values greater than 30.000 Amperes or less than 0.000 Amperes are replaced by the average of the nearest two current values as long as those two values are also within band otherwise the value is replaced by the previous value. There were no recorded current values less than 0.000 Amperes. The use of two AttoPilot voltage and current sensors provides redundant backup. One of the sensors read consistently higher than the other. One of the sensors is susceptible to recording current spikes while a corresponding abnormal voltage reading is read on the other sensor. Initially, power spikes were assumed to correlate with the abnormal symptoms observed in the Types of Failures and Their Causes section but upon further analysis of the data a causal relation could not be inferred. Thus, the noisy values were replaced with values inside the above bands and later used to calculate in flight mean values if applicable.

2.9.2 Analysis

Mean in flight power values of the ARDrone flights with a bent motor shaft are lower than the mean in flight power values of the ARDrone after both motors were repaired. Mean in flight power values are lower for all failed vehicles tested compared to successful tests except for the Hummingbird with normal propellers and the second Hummingbird with safety propellers. The difference between the two sets for each vehicle is less than 2.000 watts for each vehicle except the Hummingbird with ARDrone propellers where the difference is approximately 5.000 watts for both sensors.

The Hummingbird with ARDrone propellers is the vehicle that consumes the most power while performing all tasks including maintaining a position. The ARDrone itself

consumes less power than the modified Hummingbird. There is not a noticeable difference in the power consumption of Hummingbirds with the Ascending Technology Normal propellers and Ascending Technology Safety propellers. The Hummingbird with ARDrone propellers has the highest variance in mean flight power consumption of the vehicles tested while the ARDrone has the lowest variance in mean flight power consumption of the vehicles tested. The maximum power values are recorded during launch and during positive height translations. Minimum power values during flight are recorded during negative height translations. The maximum power value normally recorded during launch varies from launch to launch.

The range of maximum power consumption across vehicles is [104.472, 357.403] Watts. For the individual vehicles the shortest range spans 105 Watts while the largest range spans 217 Watts. This is illustrated in Figure 2.44. Hummingbird mean power consumption while in flight with Ascending Technologies propellers ranged from 74.601 Watts to 91.188 Watts. The ARDrone mean power consumption varied between 95.790 and 102.702 Watts while the Hummingbird with ARDrone propellers' mean power consumption varied between 93.113 and 108.335 Watts. These values are present in Figure 2.45.

2.10 Analysis of State and Status Flags

2.10.1 Problems Encountered

The native pitch and roll test implementation did not work, thus, a separate routine to implement a pre-flight test of pitch and roll was written. This naive implementation caused a state where two publishers attempted to publish to the same topic simultaneously. In addition to this, a call to the native motors test was still in the test code during the first 100 tests performed. The native motors test actually executed twice during this time resulting in a test of the motors while in flight as well as the pre-flight pitch and roll test. This caused the quad control input GPS control status field to alternate between flying (1) and not flying

(0) while in the situation mentioned above. The multiple publisher issue was not addressed until after test day 9. The ARDrone tests are free of this condition since the pitch and roll test is skipped for ARDrone flights. A modification to the state mappings for the control input in the respective launch file fixed this issue.

2.10.2 State

Each vehicle remains in the off state from 1 to 5 seconds with the mode value of the tests at 3 seconds. Motors on but not ready to be tested state is occupied for 1 to 8 seconds with the mode value of the tests at 2 seconds. Vehicles remain in the pre-flight motors on state from 4 to 11 seconds with a mode of 9 seconds. Time between launching and performing the first in flight task requires 1 to 2 seconds with a mode of 1 second. Landing, staying in post-flight motors on, and turning motors off all have a mode value of 3 seconds and extrema values of [1, 4] seconds for the indicated tasks.

2.10.3 Status Flags

The ARDrone driver used is not configured to properly report all the status flags in the native subject status rostopic or to properly report GPS control status to the control input rostopic. There exists an infrequent anomaly where the vehicle under test performs all required tests, a successful test flight, but the waypose status field of the subject status rostopic is not updated correctly. The status flags annotate the beginning and the ending of serial communication, pid waypose status propagation, and motors turning on or off as well as providing a secondary check of battery voltage to verify that the sensors connected to the Arduino are operating correctly.

2.11 Execution Time Analysis

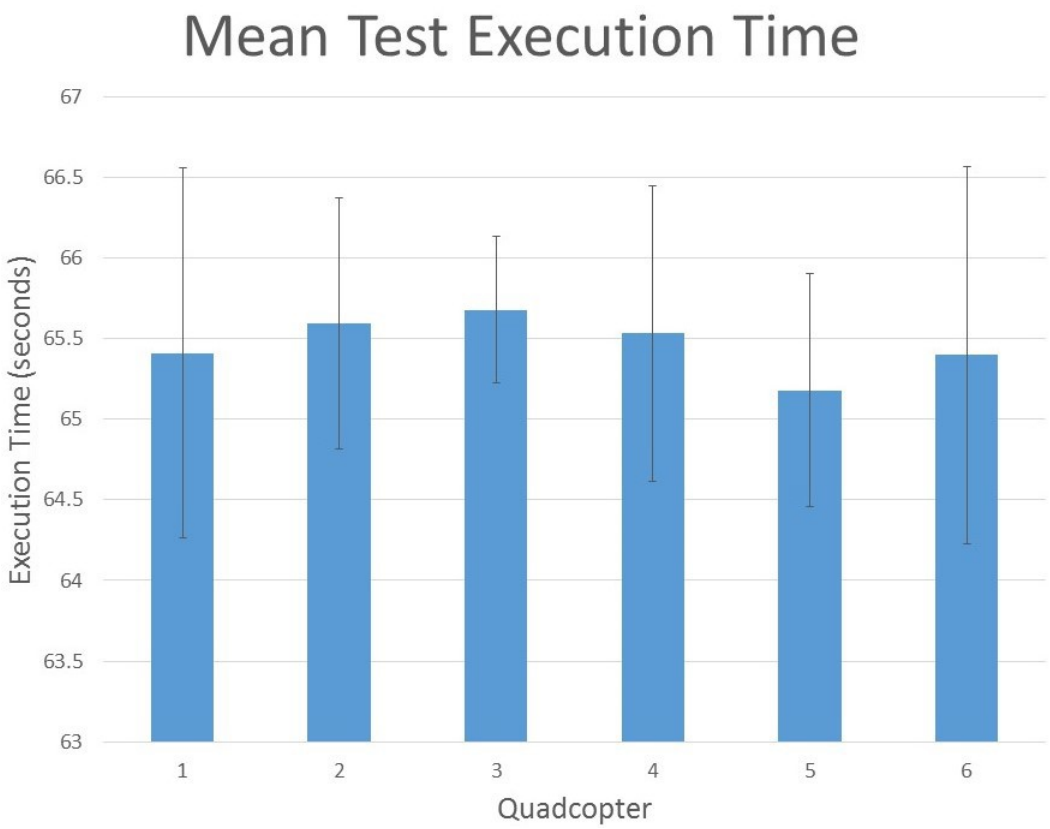


Figure 2.54: Test execution times for each unmanned aerial vehicle tested.

2.11.1 Problems Encountered

The ROS key word ”auto” used in a ROS launch file in conjunction with ”header” generates a single time stamp equivalent to the time when the applicable section of code is processed and assigns this time value to all subsequent calls to the applicable line of code. This resulted in recorded time standing still, progressing backwards, and leaping forward (time travel) in the quad control input node.

While in states 1, 4, and 5 each time stamp was identical and while in states 2 and 3 each time stamp was identical, e. g. while in state 1 and discrete time sample 5 the seconds field is 1433266764 and the nanoseconds field is 332751989, while in state 1 and discrete time sample 6 the seconds field is 1433266764 and the nanoseconds field is 332751989, and while in state 4 and discrete time sample 1163 the seconds field is 1433266764 and the

nanoseconds field is 332751989. These are actual values taken from the quad control input topic of (Vehicle 1, Test 1). One should not use ’header: auto’ in a ROS launch file if one cares about plotting normal time progression.

Each applicable state was given a separate control input rostopic to diffuse this time travel. This effect still occurs for states that are transitioned through more than once but the overall time travel is reduced. An alternative is replacing ’header: auto’ with ’header: stamp: now’ this change was tested, it was hoped that this would cause the time travel to cease. The resulting time stamps exhibited identical behavior. The impact of having non-fluid time in a topic is only visible if that topic’s time stamps are used in calculations by nodes that subscribe to that topic or if one is attempting to visualize changes in the topic’s values with respect to time.

The current pitch and roll test implementation uses the topic motor test input to test pitch and roll, which using the arbitration node is mapped into the quad control input topic. A separate thread of execution is used to publish on this topic. This topic publishes if the pitch and roll test is in progress or if a new quad control input message is generated. If the pitch and roll test is not enabled this thread copies the quad control input values into motor test input and assigns a time stamp based on current time. Doing this allows for a control input topic to exist that mimics quad control input but contains fluid time stamps. Though, this necessarily introduces a slight time skew between motor test input time stamps and correct quad control input time stamps.

2.11.2 Analysis

Time values use unsigned 32-bit integer values for a seconds and a nanoseconds field. System time is used for these values. Due to the time travel anomaly in the quad control input data all plots generated until recently used integer discrete time samples to represent the time axis, plots generated after August 20, 2015 use time from start as the time axis. This was the first approach to representing the progression of time.

The second attempt at calculating total execution time only considered Vicon time stamps. Investigation showed the time fields for the Vicon object rostopic displayed linear time. Thus, the beginning time value was subtracted from the ending time value of the respective fields in the Vicon object rostopic.

The third and final approach to calculating execution time involves multiple topic time stamps. Though Vicon update rate is 200 Hz, the Vicon topic could be delayed and one of the other topics could have an earlier time stamp, likewise the last topic published before the node shutdown sequence begins could be a topic other than Vicon. Thus, beginning time stamps for each topic with a time stamp field per test except quad control input are compared. The earliest time stamp is regarded as time 0.000. The Unix time stamps for all topics are then converted to time from the earliest time stamp. A change included in bag2matlab made the previous step unnecessary after June 2016, instead the earliest time stamp is subtracted from all elements in topic time arrays. The maximum time value from all topics except quad control input is then regarded as the execution time of the test. Though this is still not a completely accurate measure of execution time, increased accuracy is achieved using this method rather than using the Vicon topic beginning and ending time stamp. This result is reported as the total time taken to execute the test.

The mean test execution times for the hummingbirds tested are within 0.500 seconds of each other. Test execution time for all successful Hummingbird tests are within [60.000, 70.016] seconds. A variance of 1.325 was observed for the first Hummingbird tested. This value is higher than the other Hummingbird's due to automatic node shutdown not being implemented until test day 5. Several tests such as test (Vehicle 1, Test 36) were given the manual shutdown signal prior to being in the electrical stop state, which is the state from which automatic shutdown is conducted. While test execution times that exceed the mean are due to delays while starting up and delays while idle, this effected all vehicles tested. Due to the automatic launching, lack of a motors on and not flying state, of the ARDrone, the pre-flight motors test was not executed. Each in flight task takes either 1.000 second for

a Hummingbird to execute the task or 2.000 seconds for an ARDrone to execute the task. The position error values per task were normalized from the total position error values using 1.000 second as the time required to execute a single task. The mean, variance, and range of test execution times for each vehicle tested can be found in Figure 2.46.

2.12 Vehicle Type Analysis

2.12.1 Hummingbird: Safety Propellers

Figure 2.55: Hulk Smash–Ascending Technologies Hummingbird with Safety Propellers

Three vehicles of this type were tested. The first vehicle performed 510 flights. The second vehicle performed 100 flights. The third vehicle performed only 3 flights due to identifying a motor assembly failure on the third flight. Tests did not continue after the vehicle was repaired. The Hummingbird with Ascending Technologies safety propellers operates within a mean power range of [77.015, 91.188] Watts. This vehicle has a maximum power usage

range of [104.472, 259.329]. Position error mean per task is within [0.400, 2.800] millimeters for each Cartesian axis and within [1.600, 2.600] for the second of these vehicles. The second vehicle tested with this type of propeller is the most precise of the vehicles tested. Control packet delays occurred on 13.726%, 2.000%, and 0.000% of flights. Vicon communication was delayed during 2.941%, 2.000%, and 33.333% of flights. Extra position control packets from those tasked were received on 4.706%, 1.000%, and 0.000% of flights. Extra state packets from those tasked were published on 5.686%, 2.000%, and 0.000% of flights. These vehicle's exhibited more resistance to automatically turning off their motors after landing than the other vehicles tested. The second of these vehicles tested failed 4 of 100 tests, two of which exhibited incorrect task propagation and each instance was an extra tasked hover state added after launch. The other two fail cases for the second of these vehicles involved setup failure (power cord and battery cord interfering with propellers). These vehicles are the most reliable of the vehicles tested. Mean tests between failures for these three vehicles are 6.442, 27.333, and 2.000 respectively.

2.12.2 Hummingbird: Normal Propellers

Figure 2.56: Jenny–Ascending Technologies Hummingbird with Normal Propellers

One vehicle of this type performed 110 flights. The Hummingbird with Ascending Technologies normal propellers operates within a mean power range of [74.601, 83.525] Watts. This vehicle has a maximum power usage range of [151.373, 357.403]. Position error mean per task is within [0.900, 2.500] millimeters for each Cartesian axis. Control packet delays occurred on 29.091% of flights. Vicon communication was delayed during 4.545% of flights. Extra position control packets from those tasked were received on 4.545% of flights. Extra state packets from those tasked were published on 9.091% of flights. This Hummingbird had the most difficulty landing but not as much difficulty as the ARDrone had with landing. The highest variance in maximum power usage was exhibited by this vehicle. This vehicle is the least reliable Hummingbird. Mean tests between failures for this vehicle is 3.313.

2.12.3 Hummingbird: ARDrone Propellers

Figure 2.57: Herbie–Ascending Technologies Hummingbird with ARDrone Propellers

One vehicle of this type performed 100 flights. The Hummingbird with ARDrone propellers operates within a mean power range of [93.113, 108.335] Watts. This vehicle has

a maximum power usage range of [132.151, 287.695]. Position error mean per task is within [0.650, 3.200] millimeters for each Cartesian axis. Control packet delays occurred on 11.000% of flights. Vicon communication was delayed during 5.000% of flights. Extra position control packets from those tasked were received on 6.000% of flights. Extra state packets from those tasked were published on 6.000% of flights. This Hummingbird consumed the most power while in flight. The difference in mean power range between this vehicle and the other Hummingbirds is noticeable, [17.481, 22.153] Watts, whereas there is a negligible difference in mean power consumption between the Hummingbirds with the two different Ascending Technologies propellers. The difference in mean power consumption between this vehicle and the ARDrone is 3.927 Watts. Though an enjoyable unmanned aerial vehicle to pilot, the flight time of this vehicle will be shorter per battery than any of the vehicle's tested so far. This Vehicle was the second most reliable vehicle, it is slightly more reliable than the first Hummingbird with safety propellers tested. Though, 12 out of 14 of the failed test occurred within the first 20 tests during which Vicon lost the vehicle during 17 of those 20 tests. Marker orientation was reconfigured and the object was recreated in Vicon between test (Vehicle 5, Test 20) and and test (Vehicle: 5, Test: 21). A calibration of Vicon was conducted between these tests as well. Mean tests between failures for this vehicle is 5.769.

2.12.4 ARDrone

Figure 2.58: Morpheus–ARDrone

One vehicle of this type performed 100 flights. The ARDrone had difficulties performing rotation commands as well as performing landings. The difference between landing and launch locations is proportional to the angle between the fuselage and the vertical axis prior to performing the landing. The ARDrone operates within a mean power range of [95.790, 102.702] Watts. This vehicle has a maximum power usage range of [117.287, 333.990]. Position error mean per task is within [1.300, 5.200] millimeters for each Cartesian axis. Vicon communication was delayed during 9.000% of flights. Extra position control packets from those tasked were received on 9.000% of flights. Extra state packets from those tasked were published on 2.000% of flights. Control packet delays occurred on 9.000% of flights. This vehicle is the least reliable (every flight failed to rotate to one or more rotations) as well as the least precise, it has the highest position error per task. Vicon was lost the least with this vehicle.

Chapter 3

Radio Range Analysis

3.1 Motivation

Precise adaptive autonomous control is necessary to enable effective coordination between unmanned aerial vehicles. Many manufactures of unmanned aerial vehicles use 900 MHz or 2.4 GHz ZigBee radios due to their low cost, low power consumption, and compact form factor. The international XBee-Pros tested have a stated specification indoor range of up to 200 ft (60 m) and an outdoor range of up to 2500 ft (750 m). With a single defective XBee-Pro in a pair of XBee-Pros a range of 140 m outdoors was achieved. A range of 130 m with 70 dBm was achieved with a pair of fully operational international XBee-Pros. Though clearly possible, further range was not achieved due to time and weather constraints. Extending transmission range beyond 140 m is dependent on radio placement with respect to the nearby electromagnetic components and radio antenna orientation. Thus determining the effective range of paired XBee-Pros allows one to design adaptive autonomous control software that adheres to the limits imposed by the XBee-Pros for coordinated flight maneuvers. Two different antenna lengths (1/4 wavelength normal-mode helix antenna for 900 MHz, 1/4 wavelength normal-mode helix antenna for 2.4 GHz) each in a vertical orientation were tested.

Figure 3.1: Various XBee-Pros used during radio range testing.

3.2 Related Work

Grodi et. al. created a simulation of communication range testing, which consisted of transmitting a packet from one controlling station through a network of multiple UAVS at the same altitude, 100 m was the maximum distance between any two nodes in their system [33]. Another relay study recommends 100 m communication links between relay points for a multiple node relay system [57]. Several simulation studies of communication links between multiple UAVs and/or multiple nodes have been conducted [84] [65] [44] [52] [81]. These studies were not experimentally tested to verify effective radio range. Li et al. performed out of line of sight range testing using 2 UAVS as a relay system with a combined distance of 600 m and a maximum single link distance of 400 m [46].

Biaz et. al. investigated radio range irregularity simulations [8]. While Lymberopoulos et. al. conducted radio range irregularity experiments with 9 different transmitters and 5 different receivers [49]. These studies point out that radio range irregularities exist between radios but do not test full radio range.

Chen et. al. studied the use of RSSI as a positioning system [19] as have others [60]. As discovered in this work these methods are not generalizable to any set of radios. This work focuses on radio range and not radio positioning.

Browne et. al. tested a bluetooth radio's range in an urban environment and showed results for a 170 m square area [13]. Bluetooth is regulated by standard IEEE 802.15.1

while ZigBee is regulated by standard IEEE 802.15.4. Bluetooth receivers have shorter specification ranges than ZigBee receivers.

Palmer et al. analyzed air to water communication using an unmanned aerial vehicle and underwater nodes [58]. Along this line of research Carrick Detweiler has thoroughly analyzed communication range metrics of underwater nodes [24] [77] [27] [6] [27] [21]. Detweiler's previous works focus on air-to-water communication or water-to-water communication and not air-to-ground communication as does this work.

Cavka et. al. tested XBee-Pro range indoor through walls in an apartment structure [15]. This work features outdoor tests exclusively. Smith tested XBee-Radio communications with a 90% reliability at 300 m between a controlling station and an unmanned ground vehicle. He noted the Fresnel effect [72]. Shaw and Mohseni demonstrated XBee-Pro communication between fixed-wing UAVs at an altitude of 30 m and a horizontal distance of up to 500 m. The lowest RSSI indicated was -85 dBm, leaving -15 dBm worth of increased range. The effective range of air-to-ground communication was found to be 200 m [71]. There are more electronic components per unit area on the quadcopters tested in this work than there were present on the fixed-wing tested by Shaw and Mohseni, which has an effect on RSSI.

Rusdy et. al. measured the effective range of the national XBee-Pro in an urban environment and found a 73.33% decrease in effective range from specification range. They noted the Fresnel effect [66]. A full range test of the international version (the receivers tested in this work) would probably yield similar results.

3.3 Overview of Tests

A ZyXEL router is connected to a controlling station and the controlling station is connected to the resulting wireless network. A VMWare virtual machine running Ubuntu 14 is started with ROS jade. Meanwhile a distance measurement is marked. The unmanned

aerial vehicle is powered along with an Odroid Xu4 using a power splitter connected to a lithium-ion polymer (LIPO) battery. The vehicle is switched on and the Odroid is logged into via secure shell.

The radio range testing nodes are started on the odroid in a screen session. The screen session is detached and the ssh connection to the odroid is terminated. Radio range testing nodes are started on the controlling station. Battery, GPS, and RSSI (Received Signal Strength Indication) status are checked.

The vehicle is taken to the marked distance and manually piloted to a target of 50 meters altitude, then brought back down. The battery is changed as needed, the above checks are made, the next distance is marked, the vehicle is taken to the next distance, and the same vertical maneuver is performed.

RSSI and packet loss are logged and post-processed to create heatmaps of each indication within a vertical versus horizontal two dimensional map.

3.4 System Components

Figure 3.2: Flight System.

3.4.1 Flight System

An Ascending Technologies Hummingbird with normal propellers is used as the test vehicle. An Odroid XU4 is secured to the bottom the vehicle. Ascending Technologies Firefly landing gear is attached to the vehicle, suspending the Odroid. One XBee-Pro is attached to the Odroid via a SparkFun XBee Explorer Dongle. The Odroid is equipped with an Edimax USB Wi-Fi dongle. An in-house soldered Deans-T connector power splitter is used to power both the Odroid and the vehicle. A Turnigy ubec-5a (switching battery eliminator circuit) is utilized between the Deans-T connector and the Odroid.

Figure 3.3: Radio range testing at Havelock and 84th, Lincoln, Nebraska.

3.4.2 Controlling Station

A Futaba 7C controller is used for manual flight control. Two XBee-Pros are connected to the controlling station. The first XBee-Pro receives polled packets from the vehicle. This data is used to track battery level and to log GPS position estimates. The second XBee-Pro transmits test packets and receives packets with the vehicle RSSI of the last packet the XBee-Pro attached to the Odroid received as well as tracked packet loss as of the last packet successfully processed. The ZyXEL router is powered from the controlling station.

3.4.3 Testing Environment

Tests were conducted near soy bean and corn crops to minimize interference sources and maximize line of sight. A set of tests were conducted at the Oklahoma State University Unmanned Aircraft Flight Station.

3.5 Testing Procedure

3.5.1 Description

The vehicle is piloted up to a target altitude of 50 meters starting at a horizontal distance right next to the control station operator and brought back down. This is repeated in 50 ft horizontal distance increments in front of the control station operator. The device used to measure horizontal distance increments is a Keson RR318N 3-foot RoadRunner Measuring Wheel. Since this device uses units of feet, 50 foot horizontal distance increments were chosen. The horizontal distance increment of 50 feet provides a large enough displacement to visualize variations in RSSI while providing a short enough distance that the variation in RSSI is not large between measured distances.

A ROS launch file is launched that consists of two include statements for two other launch files. The first of these is hummingbird_outdoor.launch. This launch file starts nodes required to receive battery indication, GPS status, and to send control messages. The last of these is not used during the tests.

The second launch file starts nodes in a different namespace than the previous launch file. A node that records all published messages is started. A communication node that handles the interface between ROS and the connected USB device is started. Each instantiation of the communication node type must be created in a different namespace than any other active devices of the same type (e.g. ttyUSB, ttyACM). Simply assigning different topic names within the same namespace may cause data intended for the mapped topic

not to be routed correctly. A node that parses API (Application Programming Interface) ZigBee packets as well as transmits timed API test packets is started. This node accepts the 16-bit source address of the paired receiving XBee-Pro as a parameter. A node that publishes RSSI and packet success rate data is started.

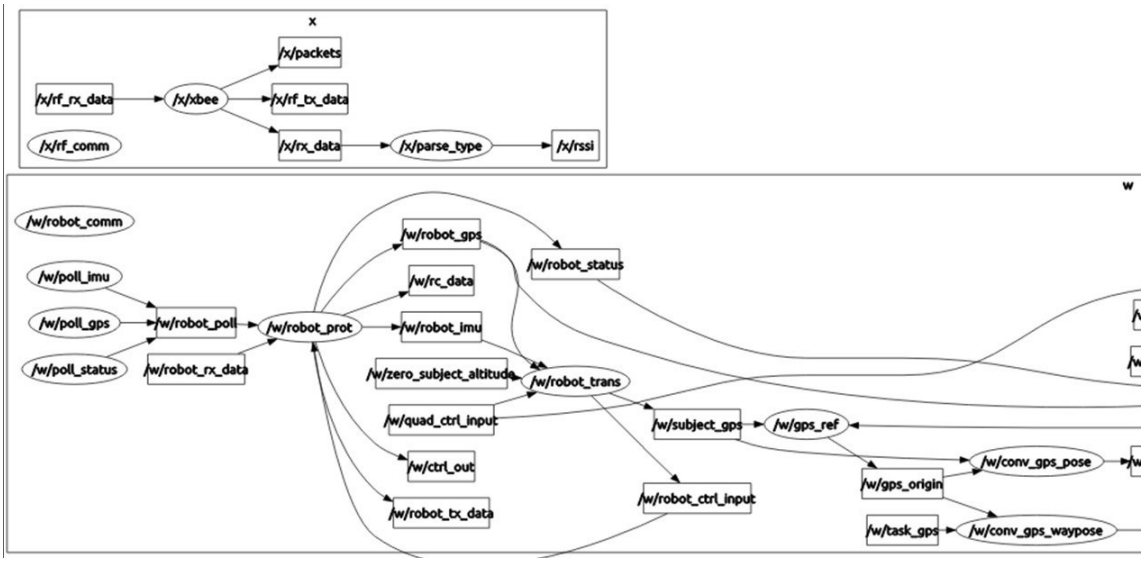


Figure 3.4: Left side of controlling station ROS graph with all nodes and topics except for record.

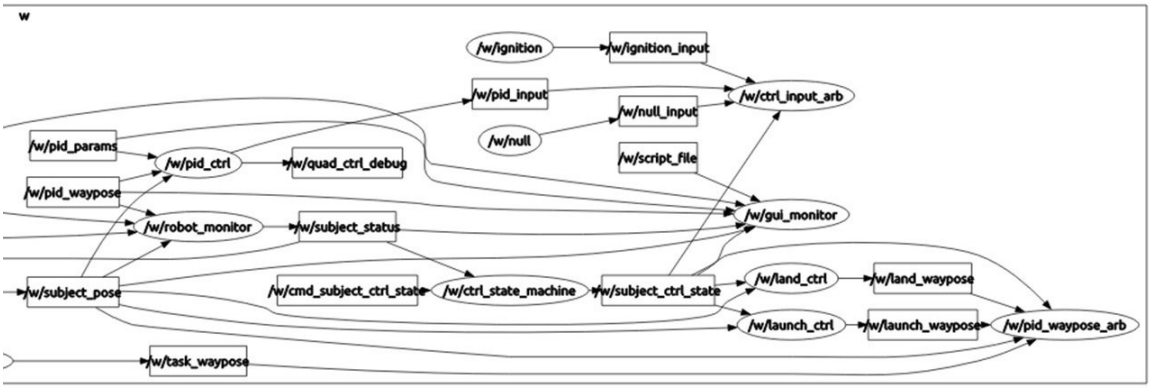


Figure 3.5: Right side of controlling station ROS graph with all nodes and topics except for record.

Test packets are transmitted from the controlling station every 250 milliseconds. Test packets consist of byte values from 0x00 to 0x17. There is an incremented, rolling over byte sequence number associated with each API packet sent. Additionally, transmitted API packets include a start byte, a 16-bit data length, a 8-bit packet type, receiver 16-bit address, a 8-bit options field, and a 8-bit checksum [83].

The transmitted API test packet is parsed by a similar node on the Odroid. This node starts a callback when a message is published on topic rf rx data. The callback starts in the

START state and loops through the received bytes one byte at a time. When a start byte is received the state transitions to state LEN0, otherwise the state remains in START.

A start byte is received and the current state is LEN0. If the byte = 0, then the state transitions to state LEN1. If the byte > MAX DATA LENGTH (128 bytes), then the state transitions to STATE START. If the byte is between these two values, it is assumed that the byte for STATE LEN0 was lost, thus, the current byte belongs to the next state, which is state LEN1. The data length is set to the current byte. A computed checksum variable is reset to 0. A counter is reset to 0. The data structure (a C++ vector of unsigned 8-bit integers) containing data bytes is cleared. The state transitions to state DATA.

Byte = 0 is received and the current state is state LEN1. If the byte = 0 or the byte > MAX DATA LENGTH, then the state transitions to state START. If the byte is between these two values, the data length is set to the current byte, a compute checksum variable is reset to 0, a counter is reset to 0, the vector containing received data bytes is cleared, and then the state transitions to state DATA.

Byte = 29 is received and the current state is state DATA. The byte is added to the 16-bit checksum variable and the 8 most significant bits are discarded. If the counter = 0 and the byte = 129 (the first byte of the data packet is the 8-bit packet type), then a publishing flag is set. If the counter = 0 and the byte does not equal 129 the publishing flag is reset. If the publishing flag is set and the counter = 4 (packet type, source address, and RSSI appended), append previous number of packets processed and previous number of checksum failures to the vector. If the publishing flag is set, the byte is appended to the vector. The counter is iterated and checked against the data length. If the counter ≥ data length, the number of packets processed is iterated, which like the number of checksum failures starts at 0 and is a 32-bit integer. If the counter ≥ data length, then the state transitions to state CHECKSUM.

The first data byte is equal to 129, all 29 bytes in the data packet are received, thus, the state transitions to state CHECKSUM. The final checksum is the difference of 0x00ff and the computed checksum. If the byte does not equal the final checksum, the number

of checksum failures is incremented. If the failed byte is equal to the start byte, there is another byte in the received queue, and the next byte is equal to 0, then the state transitions to state LEN0 (the checksum byte is assumed to be the only lost byte). If the byte = final checksum and the publishing flag is set, the vector is published on topic rx data. The state transitions to state START.

A node on the Odroid starts a callback when a message is published on topic rx data. If the first byte = 129 and the vector size is > 3, an API 16-bit transmit packet is placed in another vector. The data field contains an 8-bit sequence number generated by this node on the Odroid, the number of bytes in the test packet received, the receiver XBee-Pro least significant 8-bit address, the RSSI of the received packet, the 16-bit number of packets processed, and the 16-bit number of failed checksums. This packet is transmitted to the controlling station.

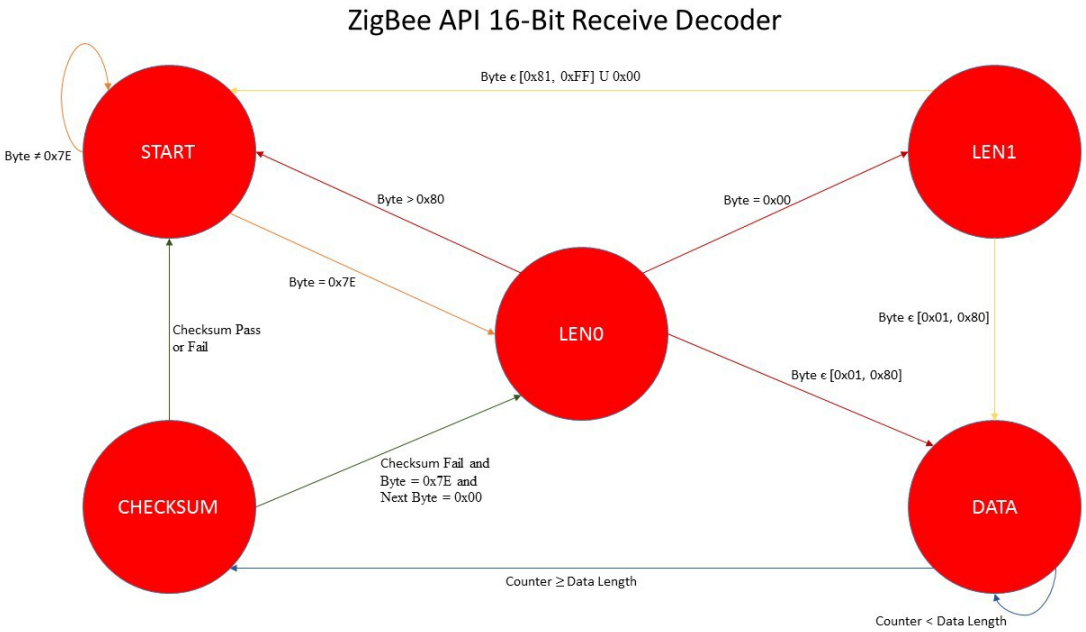


Figure 3.6: ZigBee API 16-bit Receive protocol top layer byte stream decoder.

The received data on the controlling station iterates through the same state machine as described above. The number of packets processed and the number of failed checksums along with a source flag is published directly to the topic packets in the STATE CHECK-SUM prior to setting the next state instead of being appended to the data vector.

A node on the controlling station starts a callback when a message is published on topic rx data. If the first byte = 129 and the vector size is > 3, three messages are published. The controlling station RSSI message is published. The Odroid RSSI message is published. Both of these messages contain a source = the XBee-Pro least significant 8-bit address, the RSSI value, the number of bytes received, and the sequence number. The Odroid packet data message is published.

Steps from sending a test packet until publishing the XBee-Pro attached to the Odroid's packet data is repeated with periodicity of the timer, which is set to 250 milliseconds.

3.5.2 Analysis

One of a set of paired XBee-Pros was found to have limited range nowhere near expected limits. This malfunctioning hardware hindered testing progress. After this discovery, multiple XBee-Pros were prepared to be swapped in case of a recurrence of this malfunction. At least two extra XBee-Pros were configured for both the controlling station 16-bit source address and the vehicle 16-bit source address. These spares were brought to the test site for all subsequent test. A second XBee-Pro, the second receiver that was used while starting to design these tests, was discovered to result in RSSI values of -50 dBm at a distance of a meter while a pair of XBee-Pros purchased in September 2016 resulted in RSSI values of -36 dBm at the same distance.

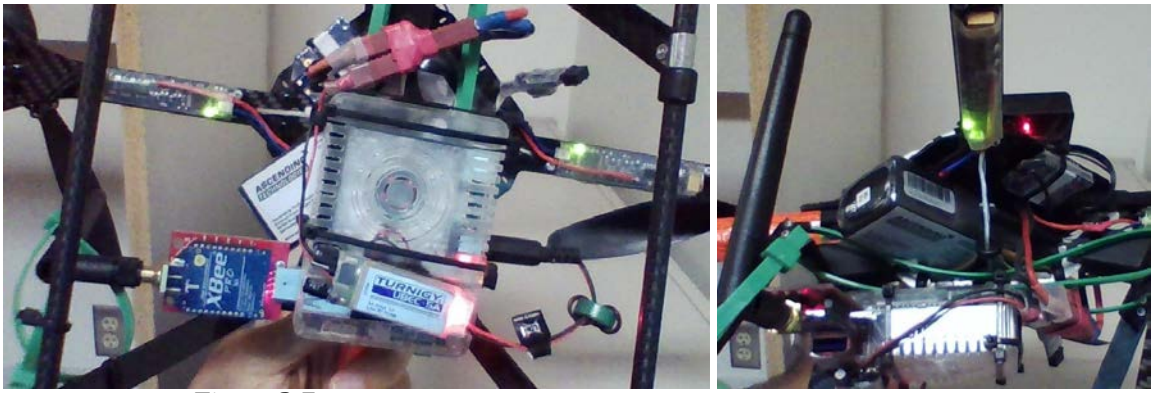


Figure 3.7: Placement of Odroid and electronics near tested XBee-Pro.

Since the RSSI and the source address is part of the API 16-bit receive packet, this

reduces the minimum amount of bytes required to be transmitted from the Odroid to the controlling station to reliably track this data for logging purposes. The use of API mode over AT command mode RSSI request minimizes the amount of bytes required to be transmitted to receive the same statistical data (less overhead) as well as allows for a shorter test period, AT mode command packet acknowledgements and guard times raise the minimum required period of data transfer.

The state machine compensates for a single byte loss of all protocol bytes except the data length. This reduces the amount of failed checksums. Thus, there are more broken and then recovered packets than there are failed checksums. Single and double byte lost are more common than losing an entire packet. This was observed while writing code to translate ArduCopter Mavlink protocol messages into a ROS topic, using paired XBee-Pros as the transmission devices. This pattern of byte loss was also observed while testing earlier versions of the above procedure indoors. The lost of a single byte often causes the resulting checksum of the communication protocol to fail. Thus, a byte lost = a packet lost in simple communication protocol decoder state machines.

The router is required to start nodes on the Odroid. The nodes on the Odroid are started in a screen session, and then the screen session is detached. An automatic starting screen session with the required ROS nodes would save time after replacing LIPO batteries, but starting a screen session that launches ROS nodes was not achieved, though attempts were made.

While below 2 meters signal propagation is quickly effected by the Fresnel Zone [30]. Thus, a XBee-Pro attached to a vehicle on the ground no longer receives transmissions from a XBee-Pro positioned between 1 and 2 meters above the ground after a distance D is reached. This effect was noticed with the malfunctioning XBee-Pro mentioned above as well as with XBee-Pros purchased during September 2016. At a horizontal distance D while below a certain altitude Z transmitted signals from the controlling station XBee-Pro are not received by the vehicle XBee-Pro but are received once the vehicle exceeds Z.

The battery splitter connection worked with a specific subset of the total available batteries. But worked only intermittently with a specific second subset of the total available batteries. This resulted in one crash from an excess of 30 meters. An unused set of Dean's T Connectors were soldered together in a similar battery splitter configuration. The new battery splitter performed as intended every time it was used.

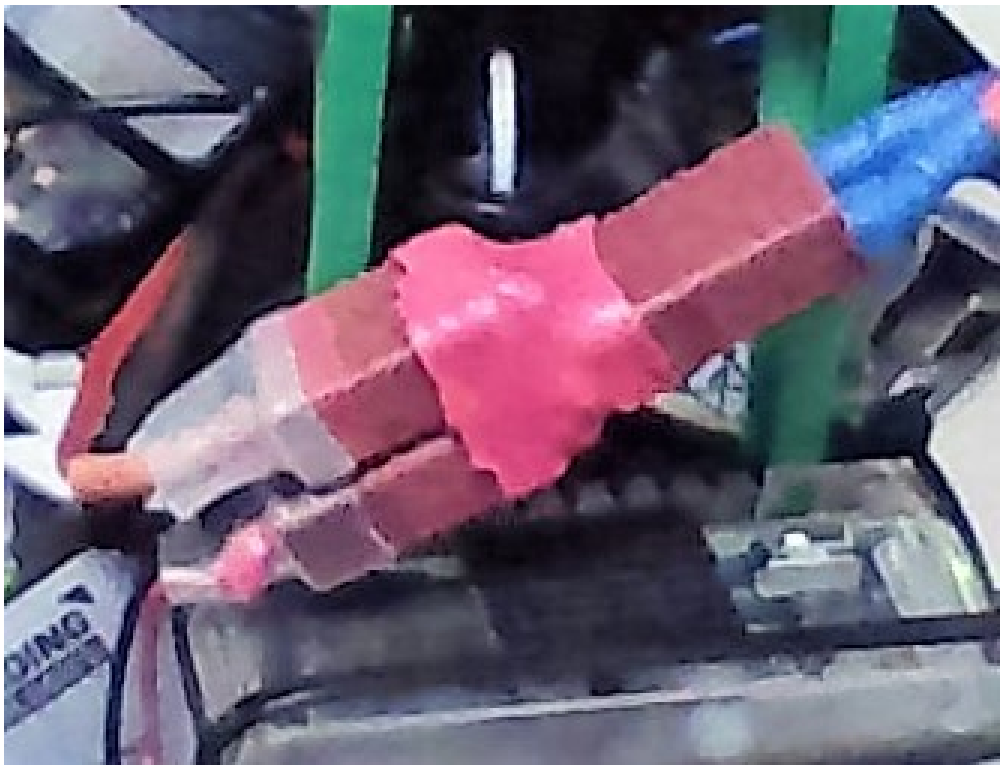


Figure 3.8: Battery Splitter.

3.6 Experimental Results

In all the following heatmaps, the vertical axis is altitude as estimated from GPS positions. The horizontal axis is Euclidean distance in the horizontal plane, also generated from GPS position estimates. An arc of white rectangles marks an equal distance in Euclidean three space from the origin point. The arc serves to artificially guarantee the high end, -32 dBm, of the desired heatmap range while the background enforces the low end, -110 dBm.

The heatmaps of Figure 3.10 display XBee-Pro RSSI values with respect to the controlling station XBee-Pro. The heatmap of Figure 3.10 (a) values were recorded after replacing

the previous Odroid with a similar Odroid, orienting the new Odroid upside down (fan away from the Hummingbird), securing the Odroid on sliding rails rather than a fixed position, and repositioning the power regulator (Turnigy ubec-5a) from the same side as the battery splitter to the fan side of the Odroid.

Odroid Configuration	Position	Fan Facing	Power Regulator
1	Fixed	Towards UAV	Near Power Splitter
2	Rails	Away From UAV	Away From UAV

Figure 3.9: Odroid configurations

Both heatmaps of Figure 3.10 were recorded with the same XBee-Pro pair and the same Sparkfun Explorer Dongles. The major setup differences are those stated above, displayed in Figure 3.9. Both XBee-Pros use the 2.4 GHz 1/4 wavelength (1 inch) antennas in the vertical position, these were purchased in September 2016. The heatmaps of Figure 3.11 are from the same test flights as those of Figure 3.10 with the exception that the RSSI values are the values reported by the XBee-Pro attached to the Odroid. Figure 3.12 displays the difference between the controlling station and the vehicle XBee-Pro RSSI values.

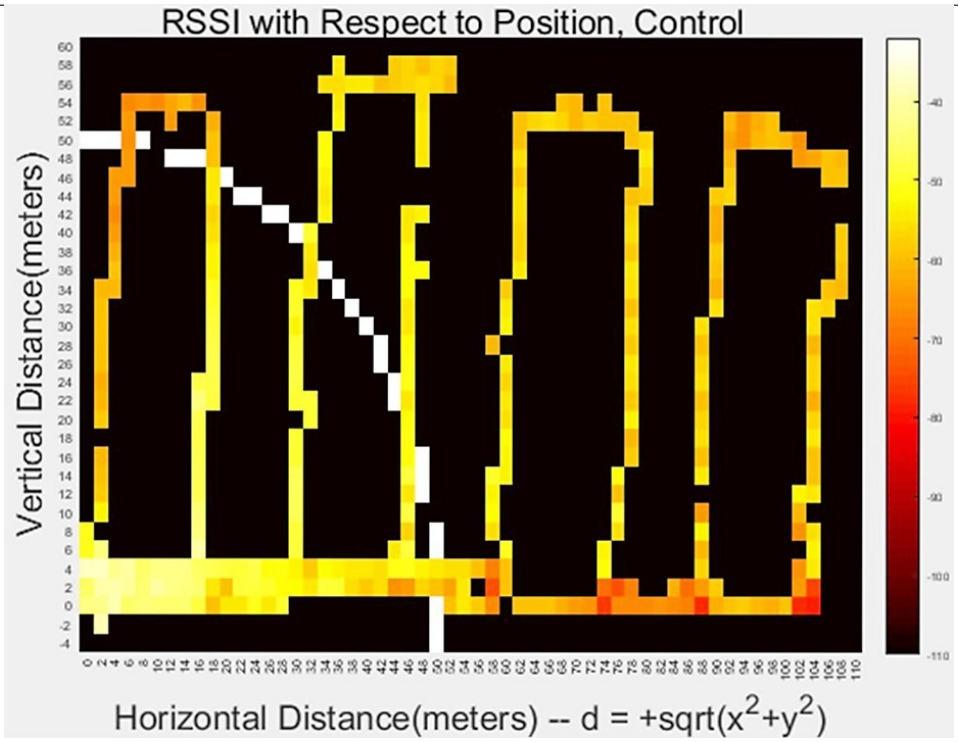
Figures 3.13-15 were generated from a XBee-Pro using the 2.4 GHz 1/4 wavelength (1 inch) antenna in the vertical position on the controlling station and a XBee-Pro using a 900 MHz 1/4 wavelength (3.5 inch) antenna in the vertical position on the Odroid. In addition to the Odroid setup changes mentioned above, the XBee-Pro used on the Odroid was discovered to be degraded when replaced with a spare XBee-Pro, a known good Spark-fun Explorer Dongle, and a tested good 900 MHz 1/4 wavelength antenna. Heatmaps of Figures 3.13-15 (a) are with the good components and heatmaps of Figures 3.13-15 (b) are with the degraded components. The degraded XBee-Pro was used in the earliest version

of these tests, its mate was found to be degraded earlier in the testing progression. It was incorrectly assumed that the hardware was not at fault for packets not being received after exceeding the limited range of the XBee-Pro pair, the software was assumed to be at fault. When the controlling station XBee-Pro module was swapped for another XBee-Pro, RSSI values were received at further distances than the previous attempts. The vehicle XBee-Pro module was not swapped at this point in time although the vehicle XBee-Pro module was degraded as well.

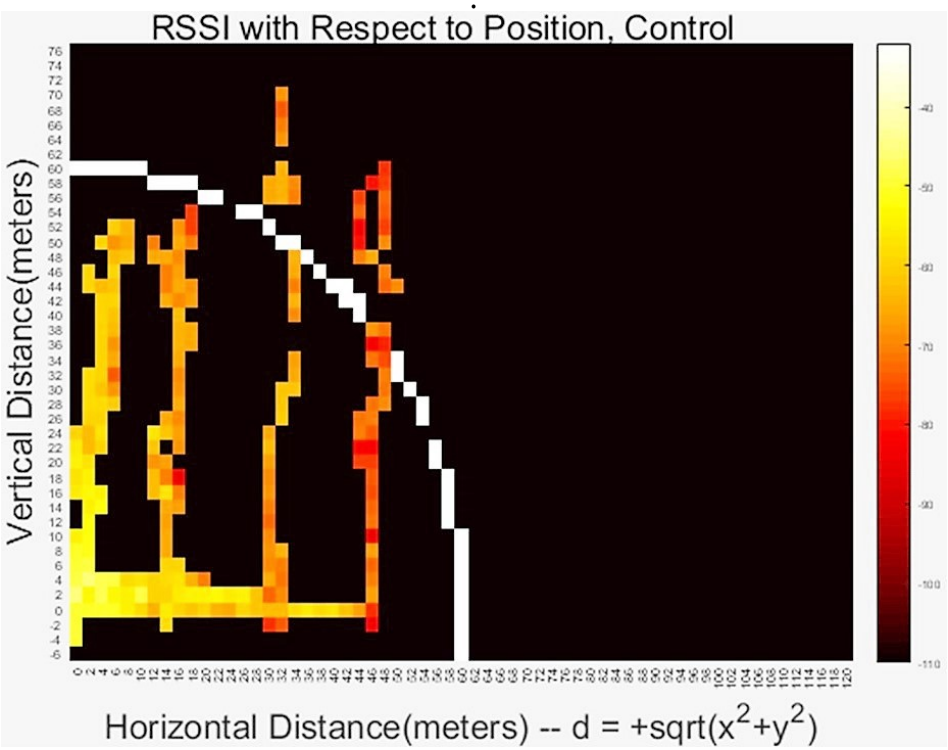
The heatmaps of Figures 3.16-18 exhibit reported data from both XBee-Pros using 900 MHz 1/4 wavelength (3.5 inch) antennas in the vertical position. The six components (2 XBee-Pros, 2 Sparkfun Explorer Dongles, and two 900 MHz 1/4 wavelength antennas) with an unknown time of being in the NIMBUS laboratory were tested and verified to report comparable RSSI values to the components used in the tests of Figures 3.10-12 (a) before performing the tests of Figures 3.16-18 (a). The heatmaps of Figures 3.16-18 (a) are with the described good components and good Odroid setup. Heatmaps of Figures 3.16-18 (b) are with the degraded components and bad Odroid setup.

The final antenna combination tested was the first combination tested with the new Odroid setup. Tests with the degraded XBee-Pro receiver and/or with the bad Odroid setup were not performed with this combination of antennas. Heatmaps of Figures 3.19-21 were generated from a XBee-Pro using the 2.4 GHz 1/4 wavelength (1 inch) antenna in the vertical position on the Odroid and a XBee-Pro using a 900 MHz 1/4 wavelength (3.5 inch) antenna in the vertical position on the controlling station.

The black circles of Figure 3.22 are where test packets were received while the vehicle was at the red triangle position (time stamps of RSSI test packets were checked versus GPS packet time stamps). The plot from Figure 3.22 (a) was generated while using 2 defective XBee-Pros. The plot from Figure 3.22 (b) was generated while using 1 defective XBee-Pro connected to the Odroid connected to the vehicle. The XBee-Pros were not known to be defective prior to conducting these tests.

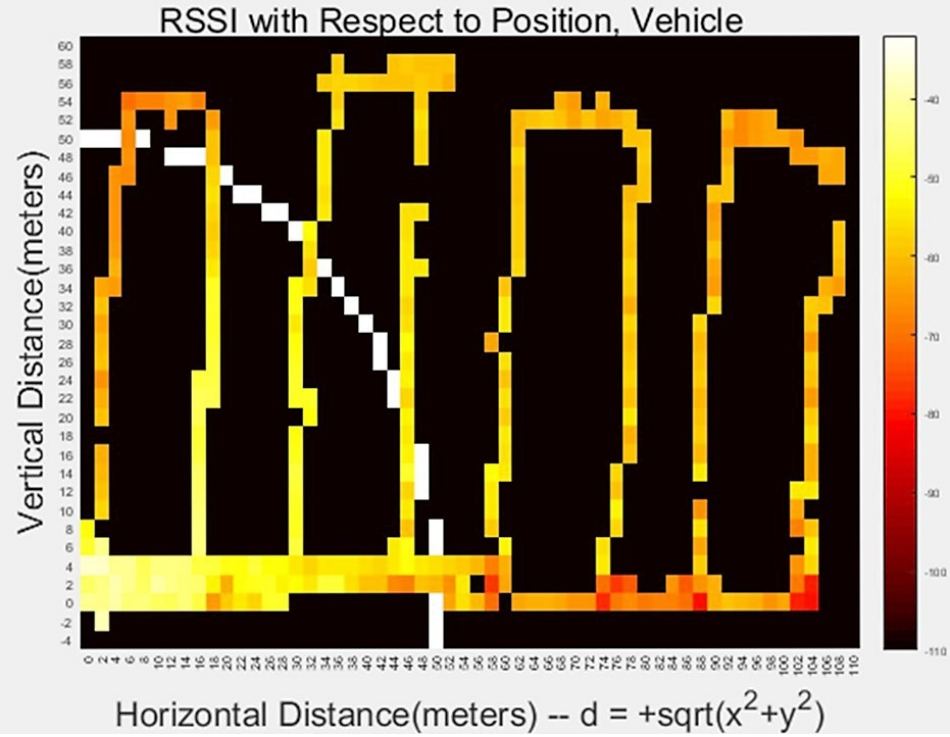


(a)

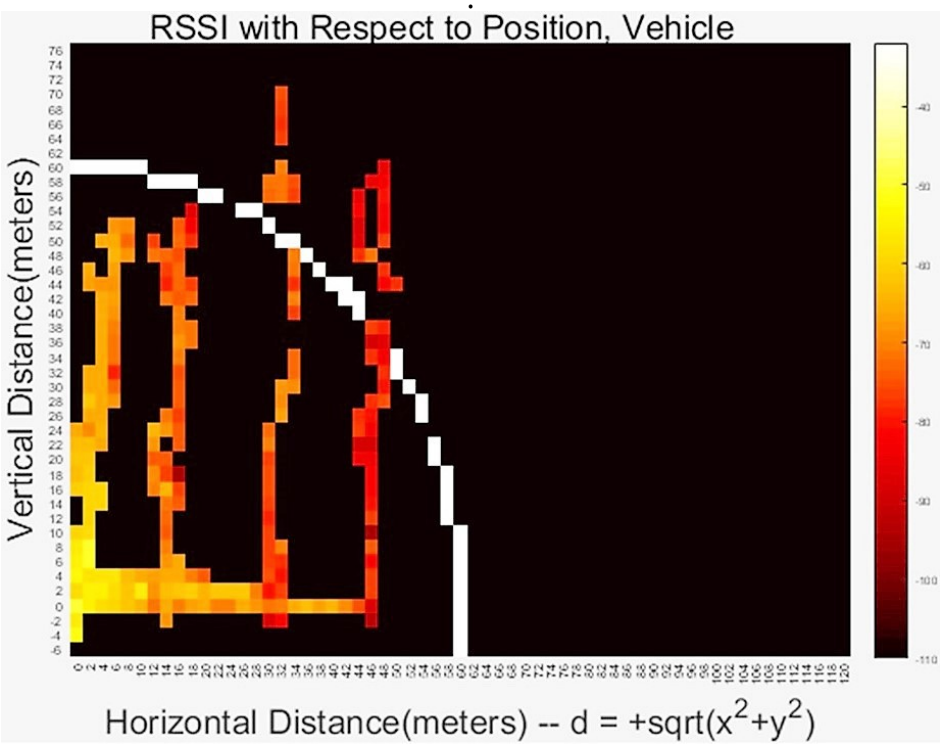


(b)

Figure 3.10: Heatmaps of controlling station XBee-Pro RSSI values in the range [-110, -32] dBm, background values are set to -110 dBm, a: Odroid in good orientation, b: Odroid in bad orientation.

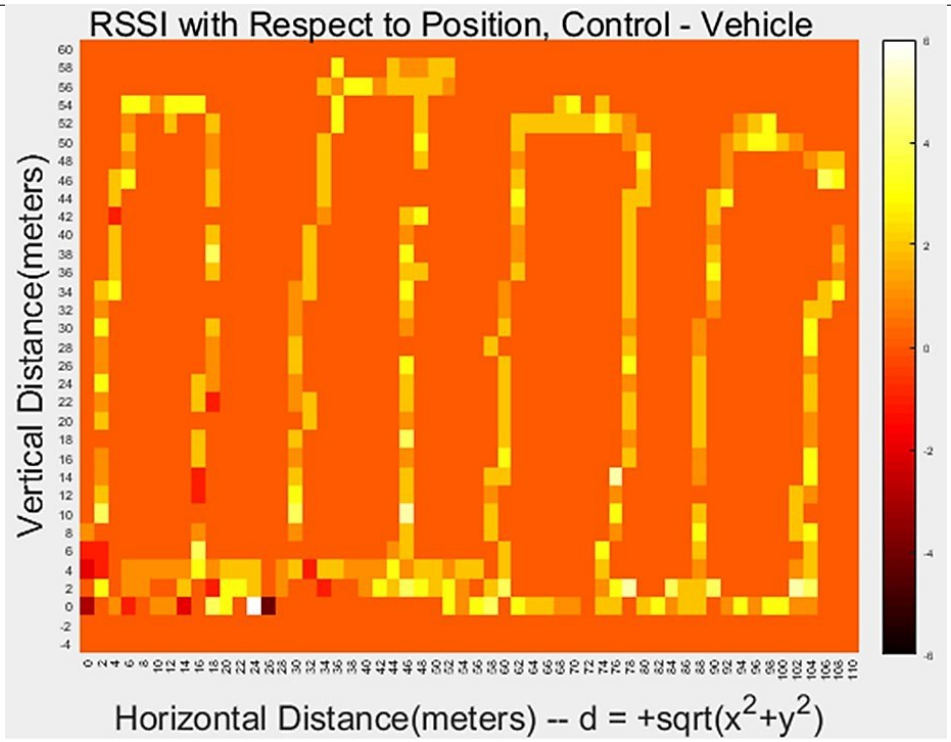


(a)

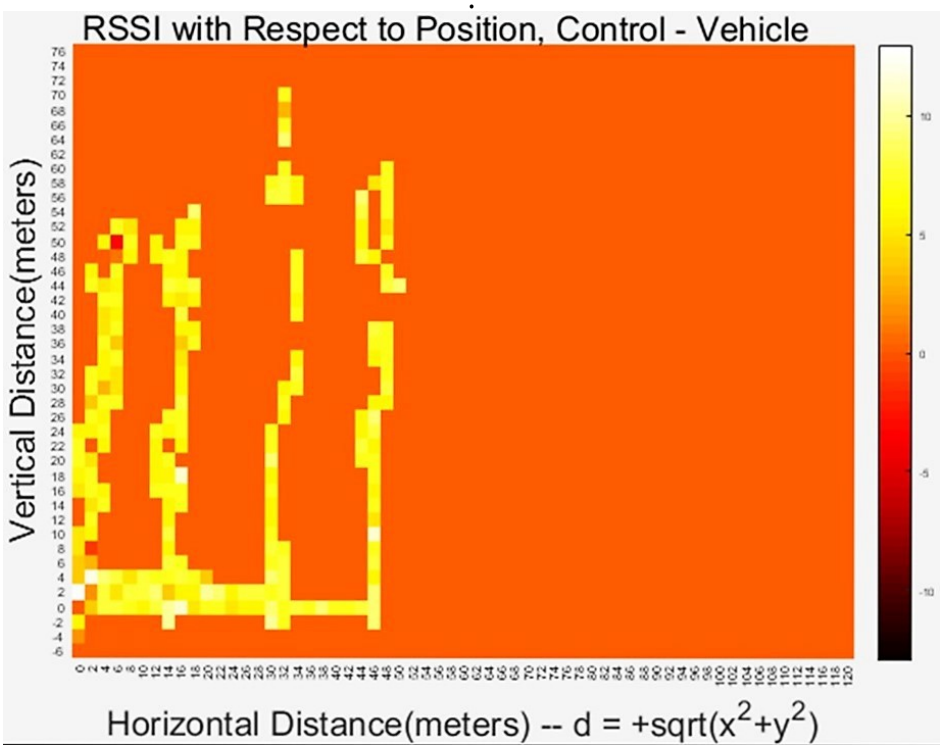


(b)

Figure 3.11: Heatmaps of controlling station XBee-Pro RSSI values in the range [-110, -32] dBm, background values are set to -110 dBm, a: Odroid in good orientation, b: Odroid in bad orientation.

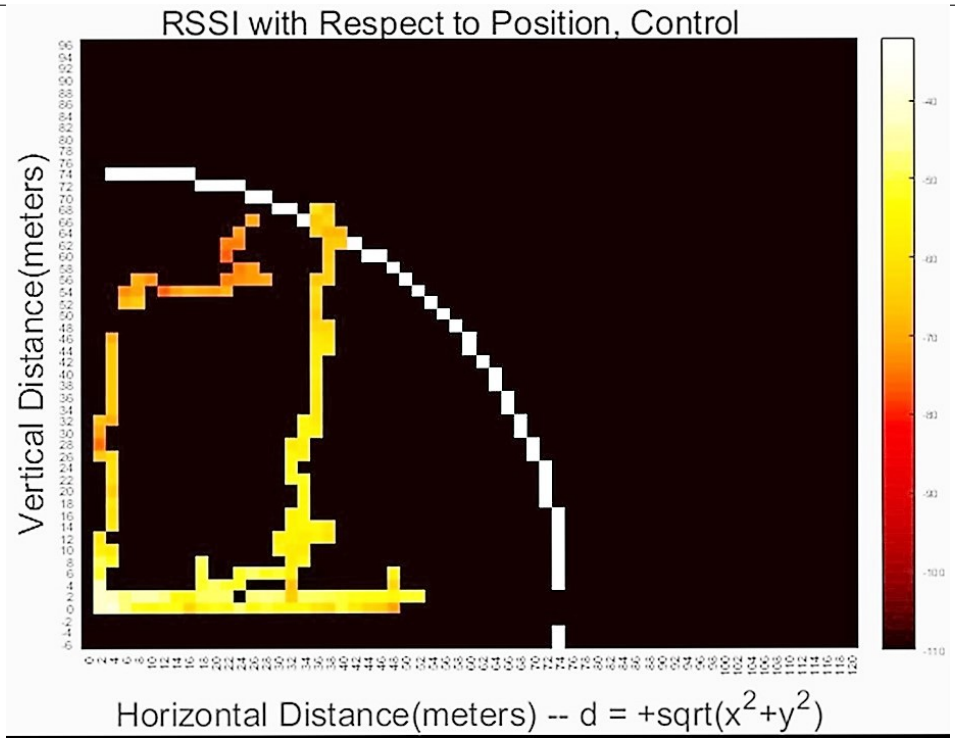


(a)

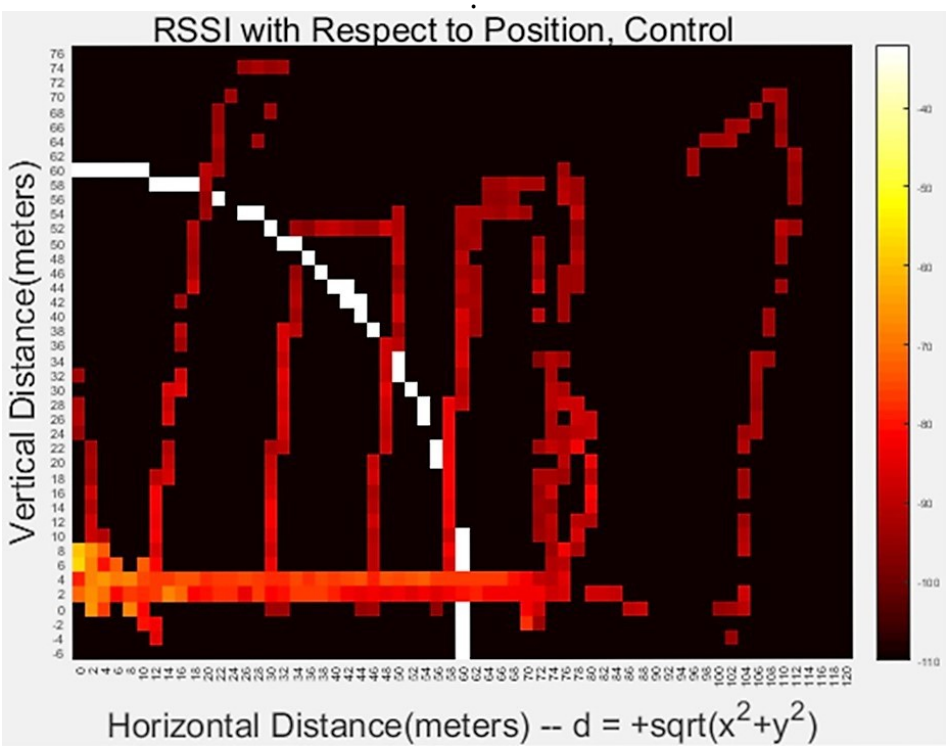


(b)

Figure 3.12: Heatmap of the difference of controlling station XBee-Pro RSSI values and vehicle XBee-Pro RSSI values in the ranges (a) [-6, 6] dBm and (b) [-13, 13] dBm, background values are set to 0 dBm.

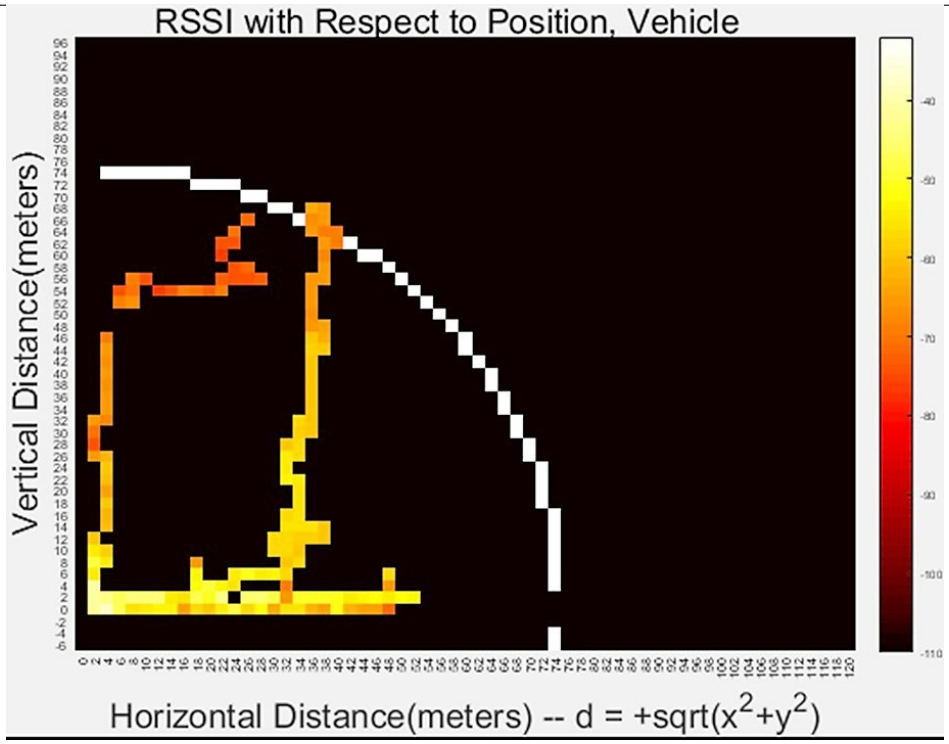


(a)

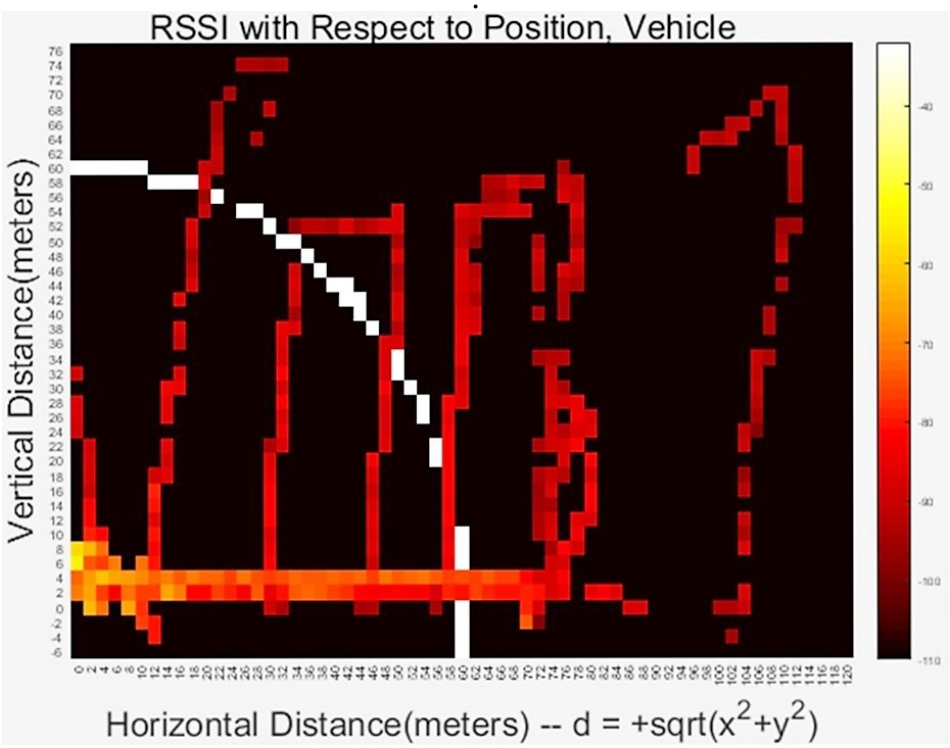


(b)

Figure 3.13: Heatmaps of controlling station XBee-Pro RSSI values in the range [-110, -32] dBm, background values are set to -110 dBm, a: Odroid in good orientation, b: Odroid in bad orientation.

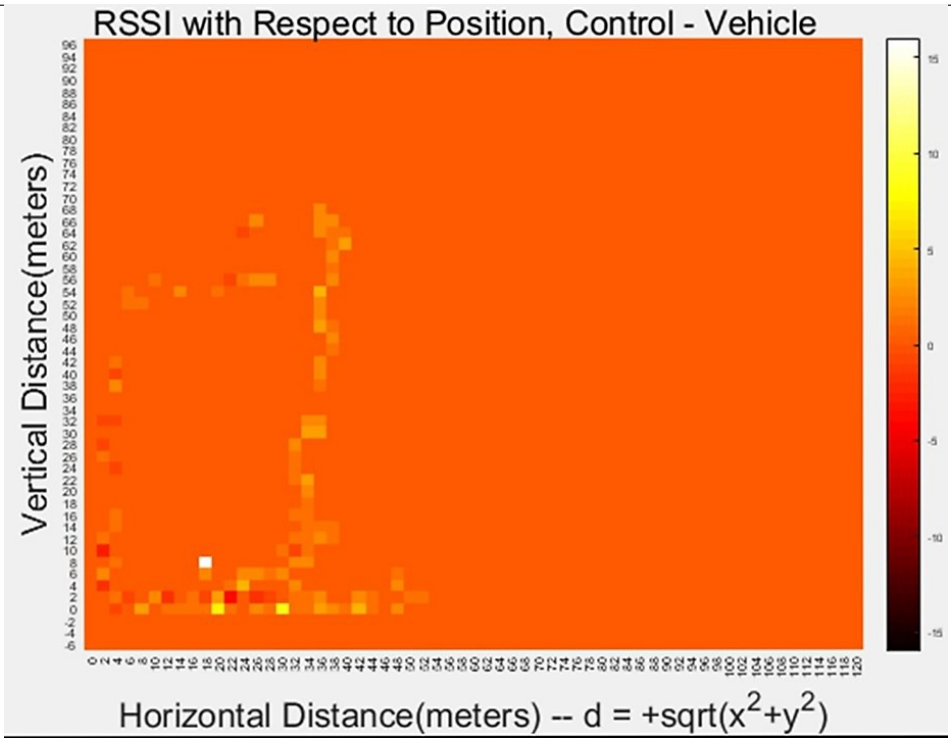


(a)

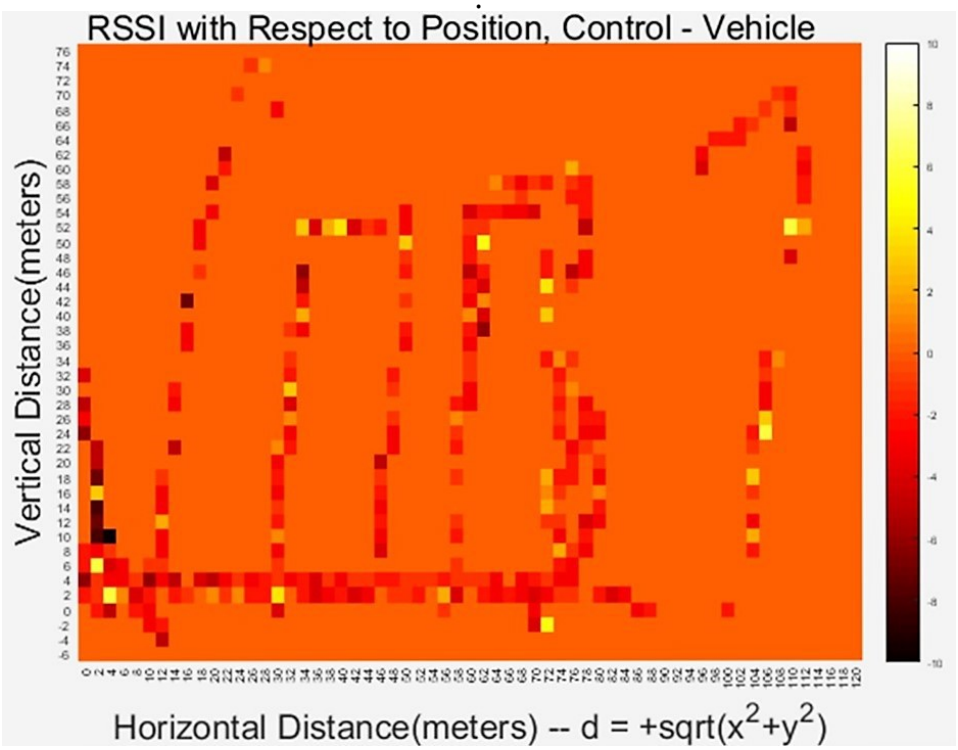


(b)

Figure 3.14: Heatmaps of controlling station XBee-Pro RSSI values in the range [-110, -32] dBm, background values are set to -110 dBm, a: Odroid in good orientation, b: Odroid in bad orientation.

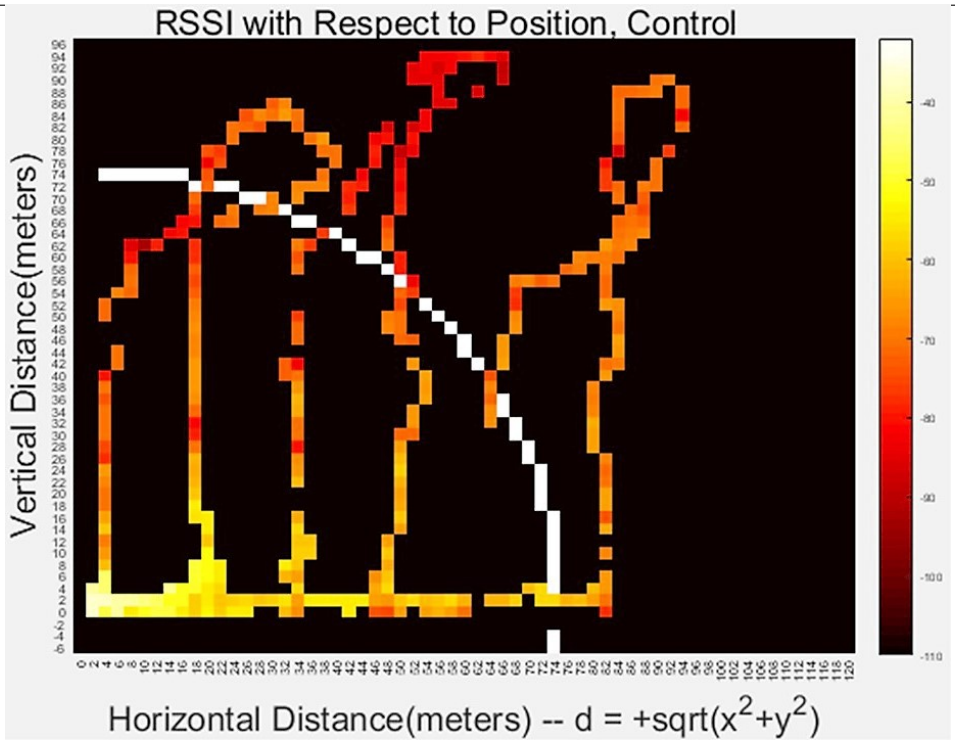


(a)

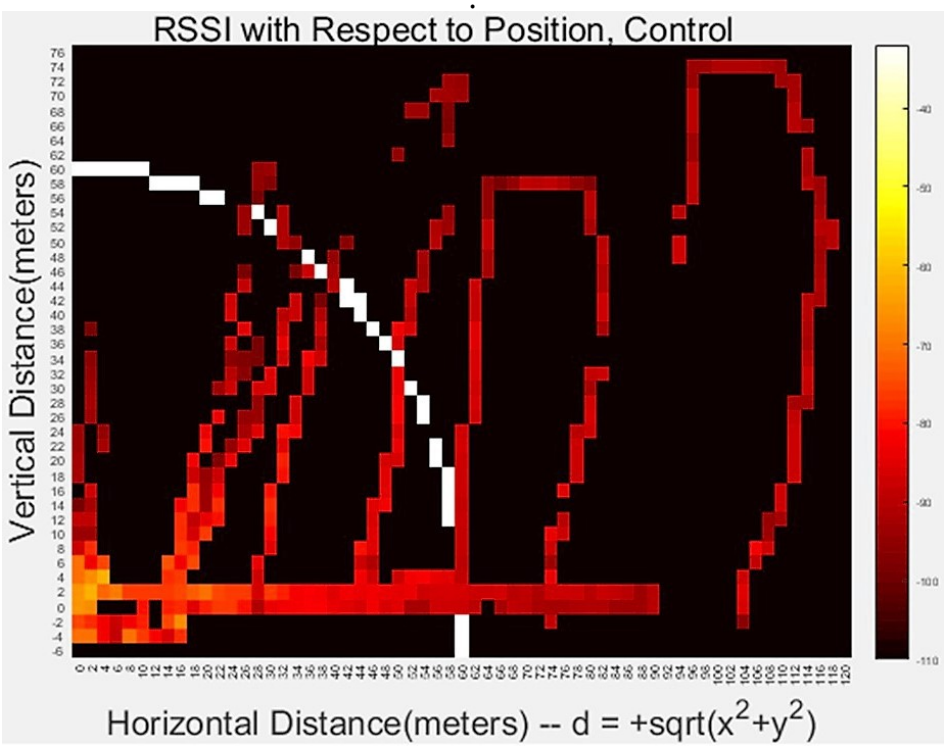


(b)

Figure 3.15: Heatmap of the difference of controlling station XBee-Pro RSSI values and vehicle XBee-Pro RSSI values in the ranges (a) [-16, 16] dBm and (b) [-10, 10] dBm, background values are set to 0 dBm.

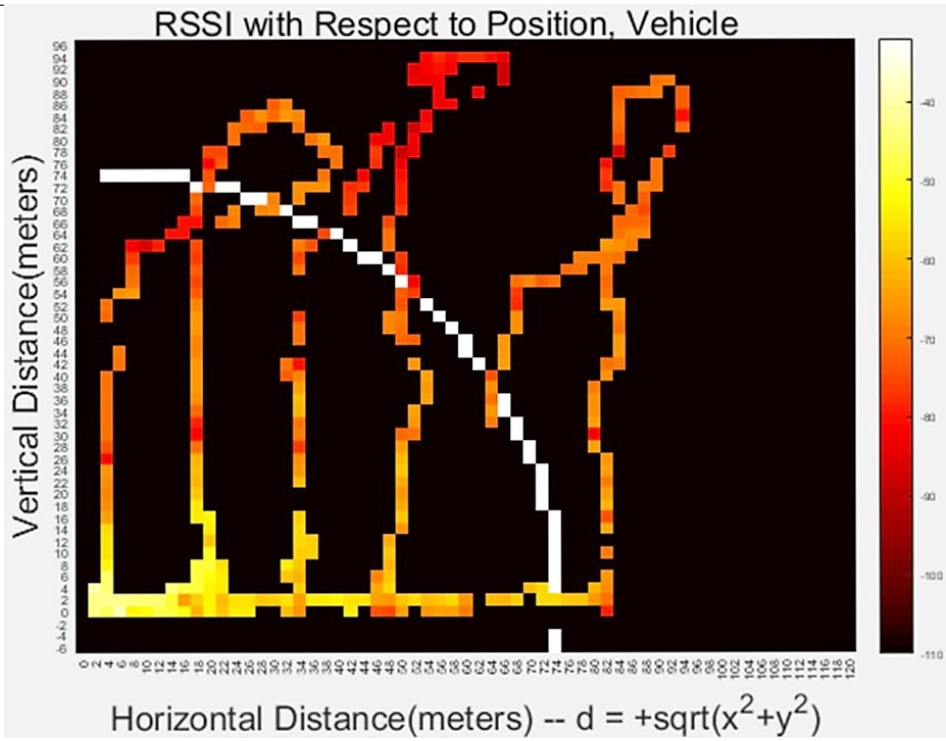


(a)

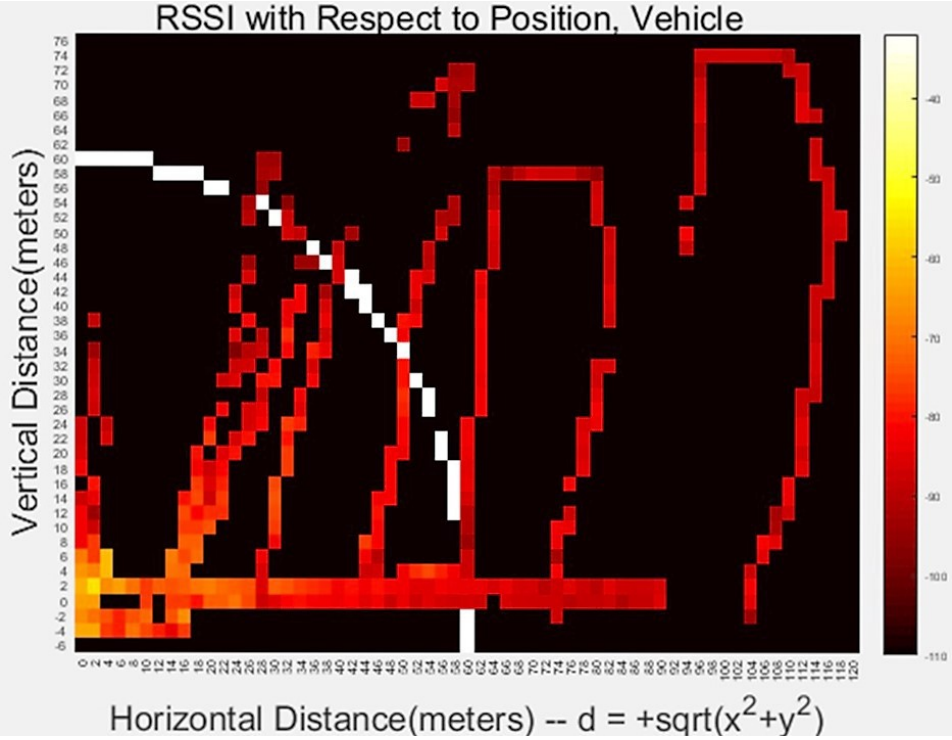


(b)

Figure 3.16: Heatmaps of controlling station XBee-Pro RSSI values in the range [-110, -32] dBm, background values are set to -110 dBm, a: Odroid in good orientation, b: Odroid in bad orientation.

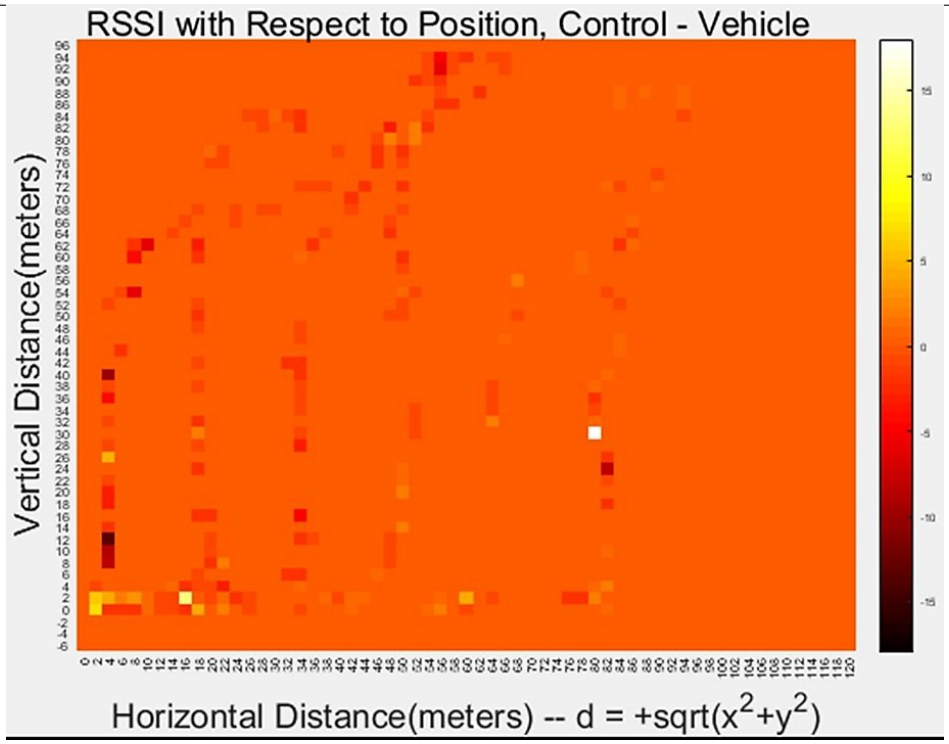


(a)

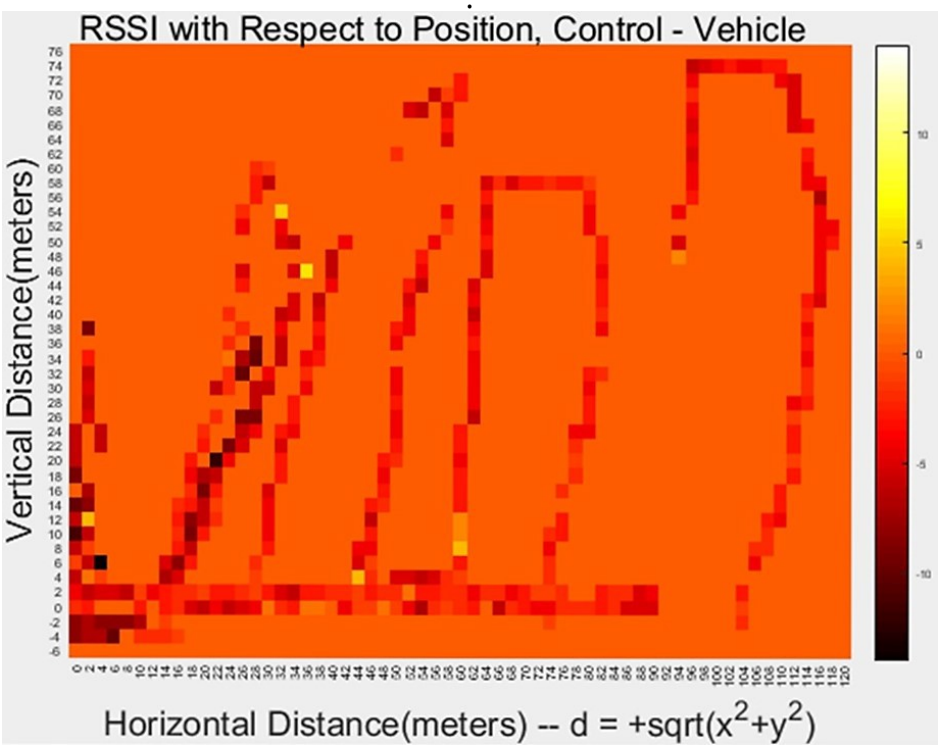


(b)

Figure 3.17: Heatmaps of controlling station XBee-Pro RSSI values in the range [-110, -32] dBm, background values are set to -110 dBm, a: Odroid in good orientation, b: Odroid in bad orientation.



(a)



(b)

Figure 3.18: Heatmap of the difference of controlling station XBee-Pro RSSI values and vehicle XBee-Pro RSSI values in the ranges (a) [-18, 18] dBm and (b) [-14, 14] dBm, background values are set to 0 dBm.

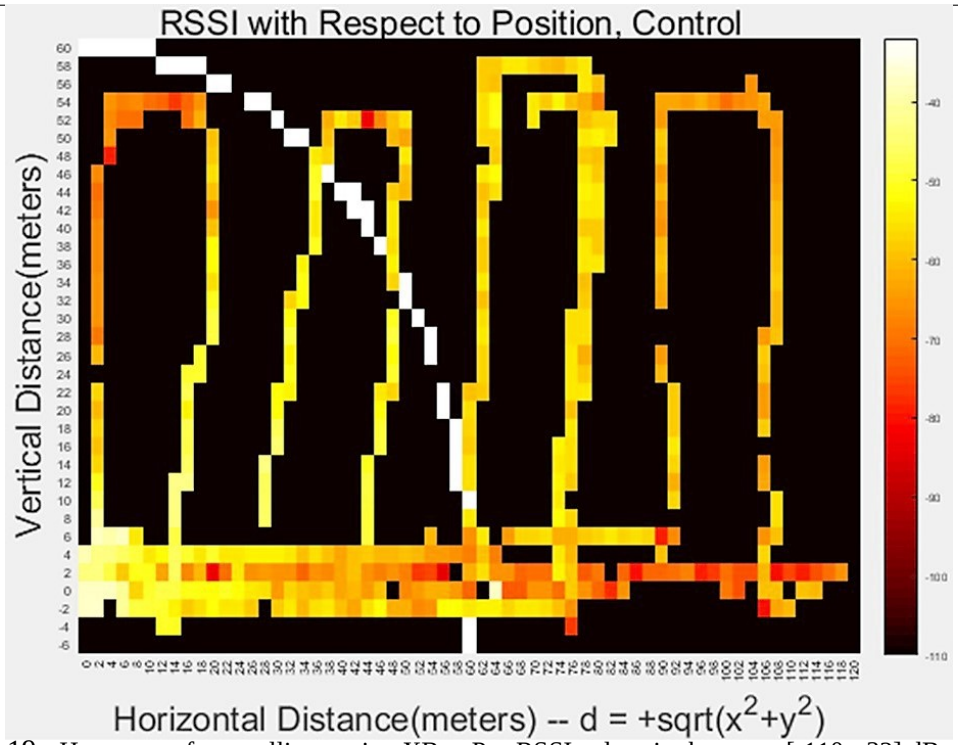


Figure 3.19: Heatmaps of controlling station XBee-Pro RSSI values in the range [-110, -32] dBm, background values are set to -110 dBm.

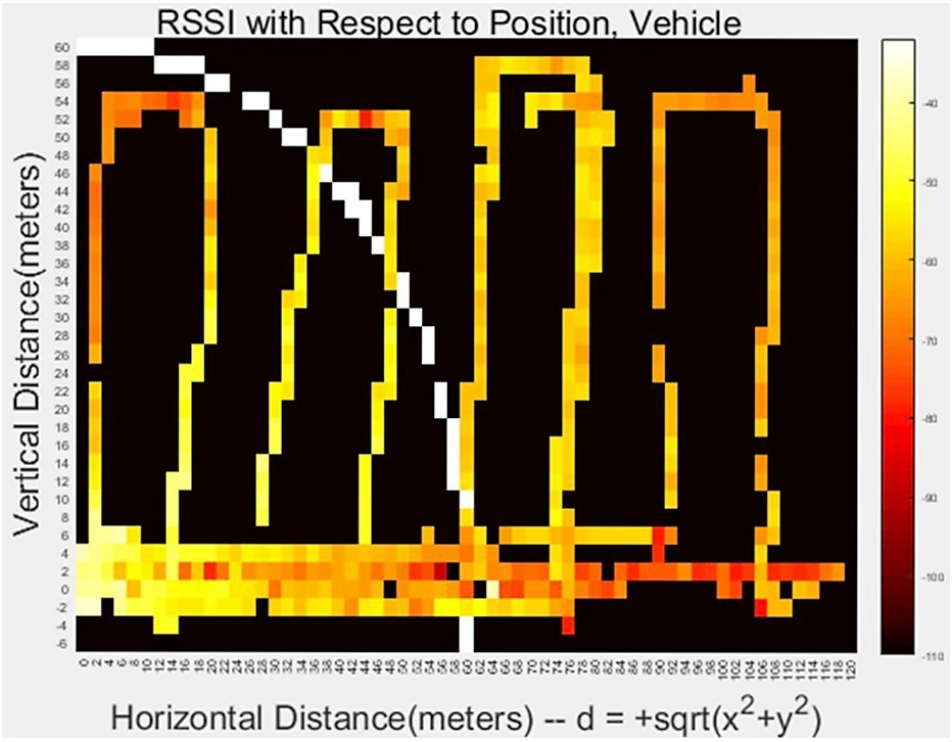


Figure 3.20: Heatmaps of controlling station XBee-Pro RSSI values in the range [-110, -32] dBm, background values are set to -110 dBm.

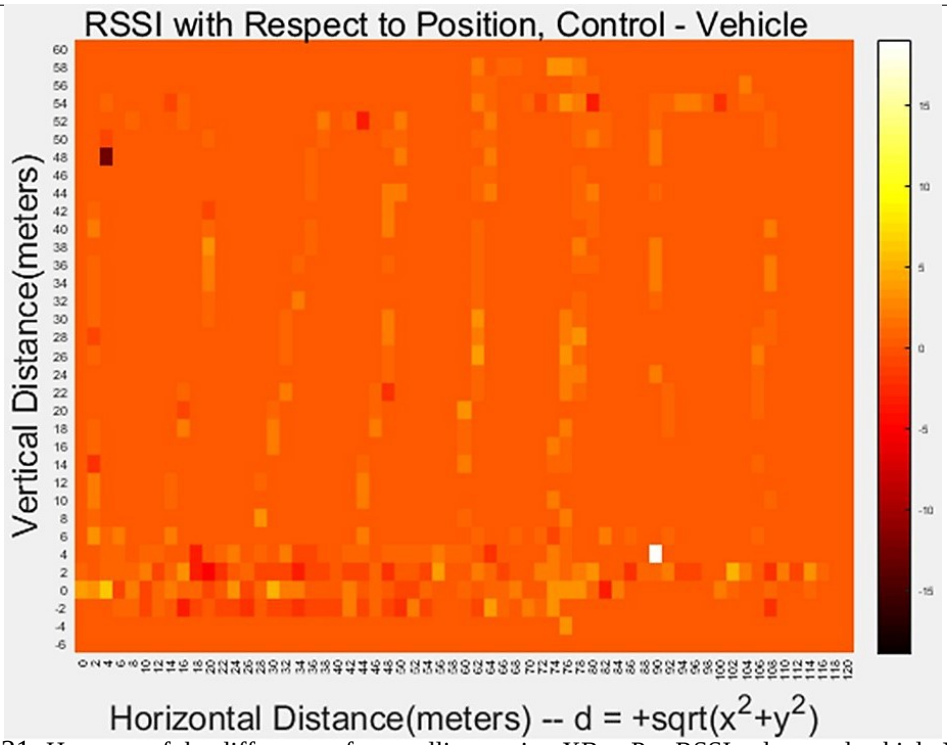


Figure 3.21: Heatmap of the difference of controlling station XBee-Pro RSSI values and vehicle XBee-Pro RSSI values in the range [-18, 18] dBm, background values are set to 0 dBm.

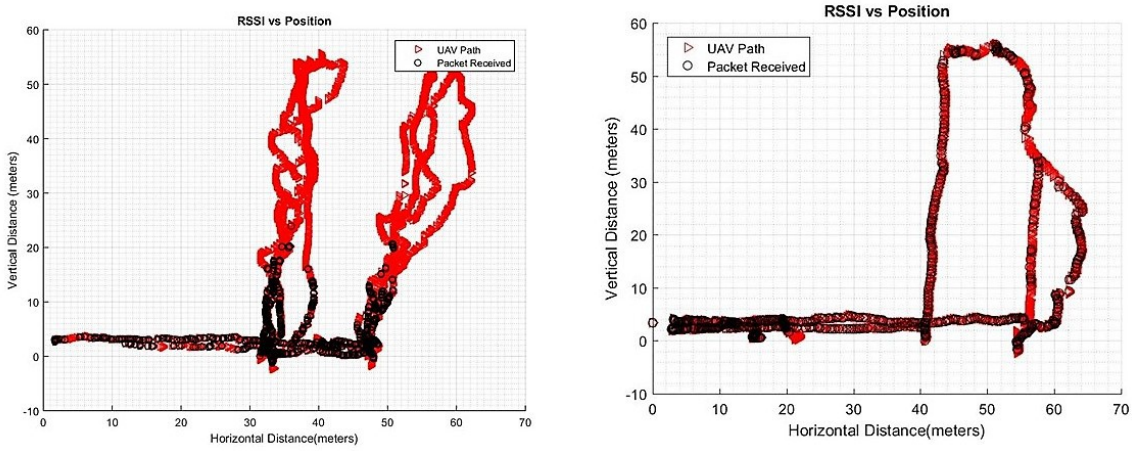


Figure 3.22: RSSI with respect to UAV path using defective XBee-Pros

RSSI values from the first flight used to generate Figures 3.10-12 is displayed with respect to time in Figure 3.23. These values are within expected RSSI values for properly functioning XBee-Pros. In contrast, RSSI values from the flights used to generate Figure 3.22 (b) is displayed with respect to time in Figure 3.24. These values are outside the expected RSSI values for properly functioning XBee-Pros. One defective XBee-Pro, which was unknown at the time of the test, was connected to the Odroid which was connected to

the vehicle. The gap in Figure 3.24 is when the test software on the Odroid was not running.

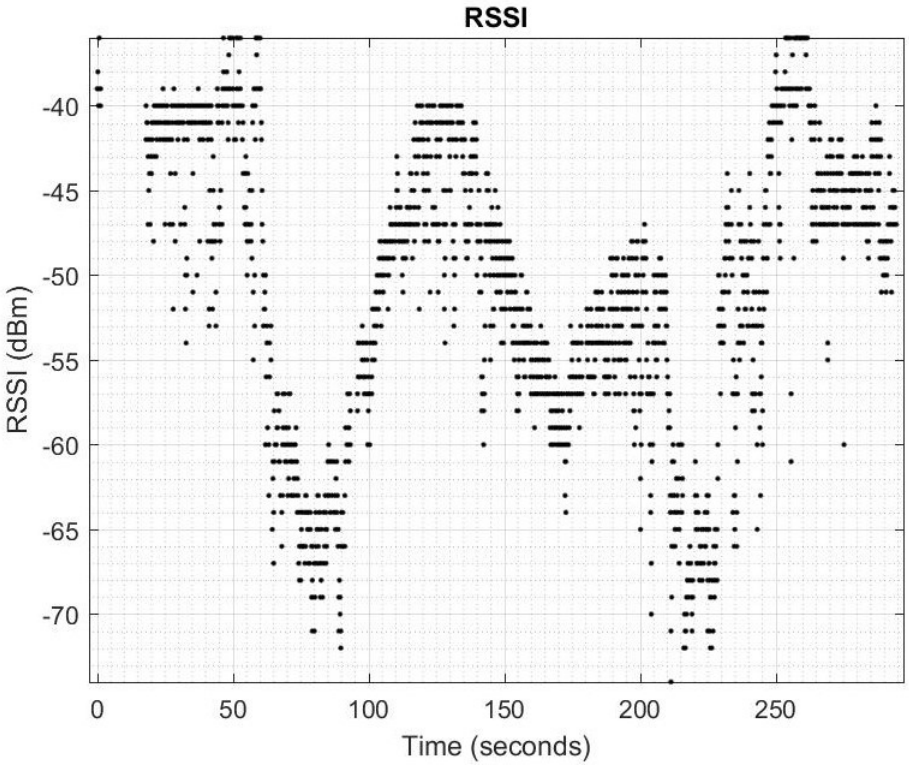


Figure 3.23: Expected RSSI values

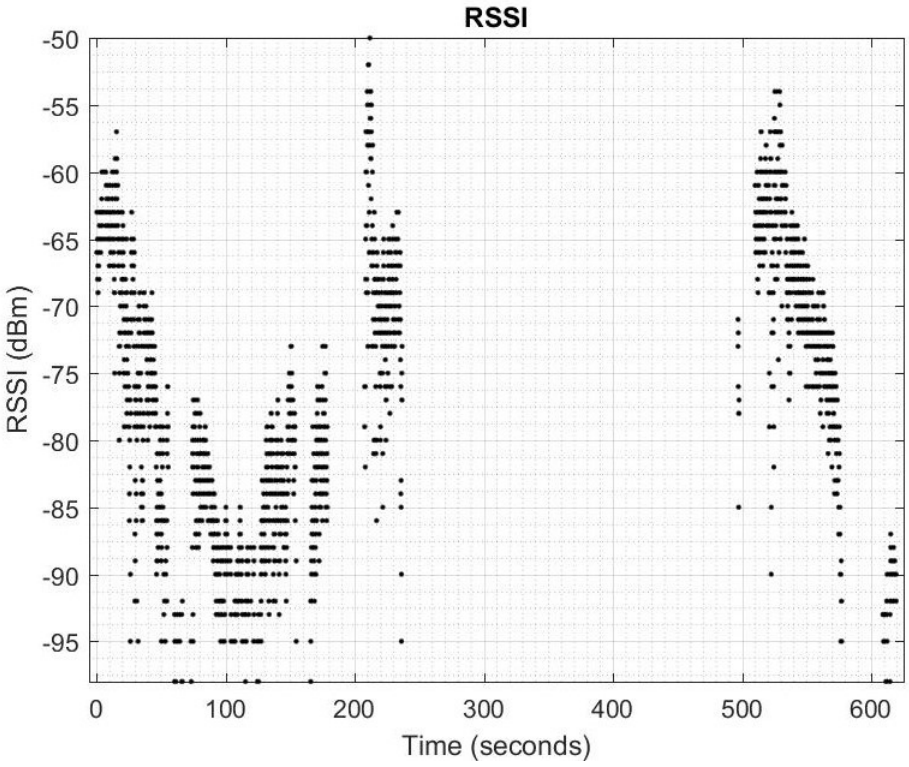


Figure 3.24: RSSI values outside of expected range

Chapter 4

Integrating RTK GPS

4.1 Introduction

4.1.1 Motivation

The viability of utilizing affordable devices with Real Time Kinematic Global Positioning System capabilities to provide position estimates for unmanned aerial vehicles was tested. RTK GPS offers centimeter level precision in position estimates as opposed to normal GPS, which offers meter level precision. The GPS device packaged with a typical micro unmanned aerial vehicle is incapable of delivering cm range precision. Using a micro unmanned aerial vehicle close to the environment to collect samples, monitor crop health, transfer power, ignite a prescribed burn, or any other such use requires a level of precision that is currently unavailable while just using the packaged GPS. Other sensing devices such as Lidar and cameras are commonly used to obtain the level of precision required in the above environments.

4.1.2 Background

Real Time Kinematic (RTK) Global Positioning Systems (GPS) have been around since the late 1990s. [54] RTK GPS' that can reasonably fit on an Unmanned Aerial Vehicle have

been available since late 2013. [79] RTK GPS offers centimeter level precision versus meter level precision of normal GPS. For the systems/kits with passive patch antennas a non-grounded ground plane is required. [76] Manufacturers such as SwiftNav provide ground planes in their kits other manufacturers such as Emlid do not provide ground planes in their kits. The RTK GPS system requires two GPS receivers. As a consequence there is a trade off between momentary high precision followed by normal GPS precision when a packet is delayed or lost and reduced average precision (due to reduced accuracy of ground station position) with perfect communication between units (wired connection between devices on the vehicle). All RTK GPS systems require ground station GPS latitude, longitude, and altitude to be configured in the ground station unit.

4.1.3 Configuration

Various communication paths realized in software were created to facilitate communication between base and rover units. Hardware in the communication paths included at most a laptop, paired XBee-Pros, base and rover units, and an Odroid-xu4/ZyXEL router–post processing only. The ZyXEL router is also used as an IP address distribution device. The Odroid XU4 was physically attached to the bottom of an Ascending Technologies Hummingbird while running ROS nodes to pass data from/to the rover RTK GPS unit. The XBee-Pros are connected to SparkFun USB boards. State machines were created to decode the generated GPS solutions for “real time” position estimate access. Hardware was added onto an Ascending Technologies Hummingbird to construct a stable platform for the base and rover antennas. Cables were constructed to facilitate direct connection between units while providing a serial output path as well.

Additional hardware and software are required to provide basic positions. Including additional hardware and software into a system increases the amount of possible failure points and may effect overall system performance such as signal interference.

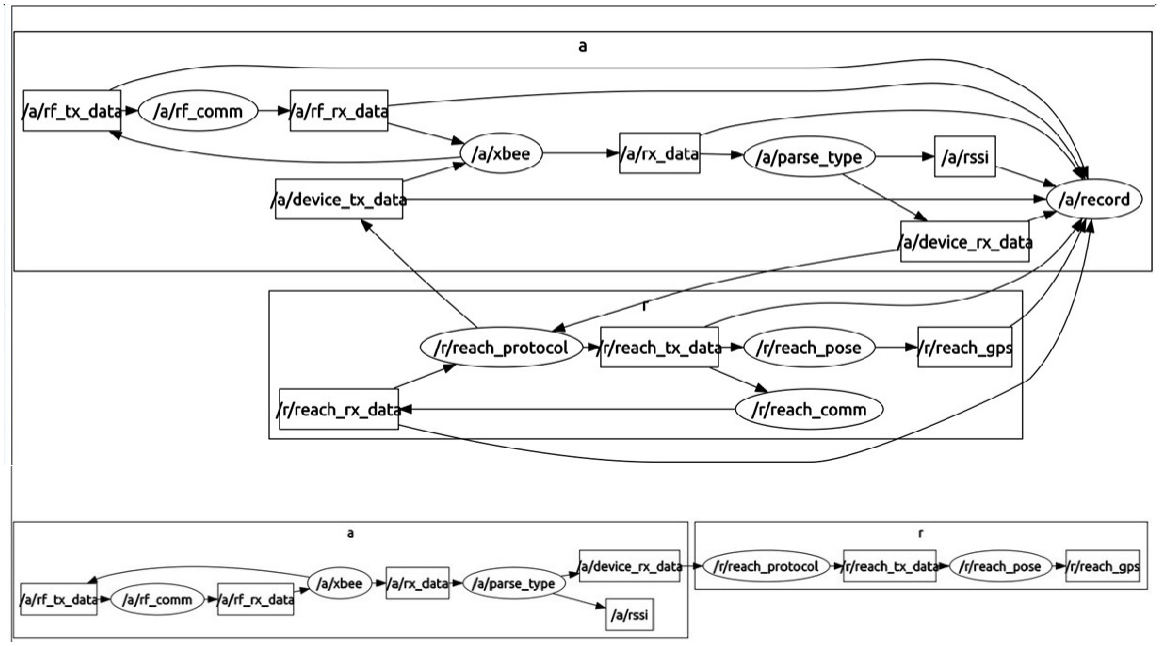


Figure 4.1: This figure displays two different ROS communication path with topics and nodes. The first communication path is with the ground station and rover receivers separated while the second communication path is with both receivers directly connected.

4.2 Challenges

Nimbus Lab purchased two affordable kits, NS-HP and Reach. Neither system is ready to use right out of the box. This offered several challenges. The first challenge encountered was not being able to setup Wi-Fi using Windows 10. A Wi-Fi connection was setup using VMWare and Ubuntu while following steps from an Intel Edison site. It was assumed one could use the micro-USB connection on the RTK GPS receiver to transmit data between the Reach and the Odroid. After discovering that was not the case, a uart-to-USB connection was used to connect the Reach and the Odroid. While the rover RTK GPS unit was powered through the uart-to-USB connection to the Odroid, the device would freeze and become unreachable. The final insurmountable challenge was integrating the RTK GPS receivers with the UAV. Communication between the receivers and the Ascending Technologies Hummingbird GPS uart could not be established.

4.3 Experimental Results

The following figure displays position estimates at three stationary positions with RTK GPS receivers 7.5 m, 4.0 m, and 0.1 m apart from each other from right to left. The RTK GPS receivers were both powered from a laptop. The rover antenna had a 0.07 m squared ground plane while the base antenna rested on a 0.1 m squared ground plane.

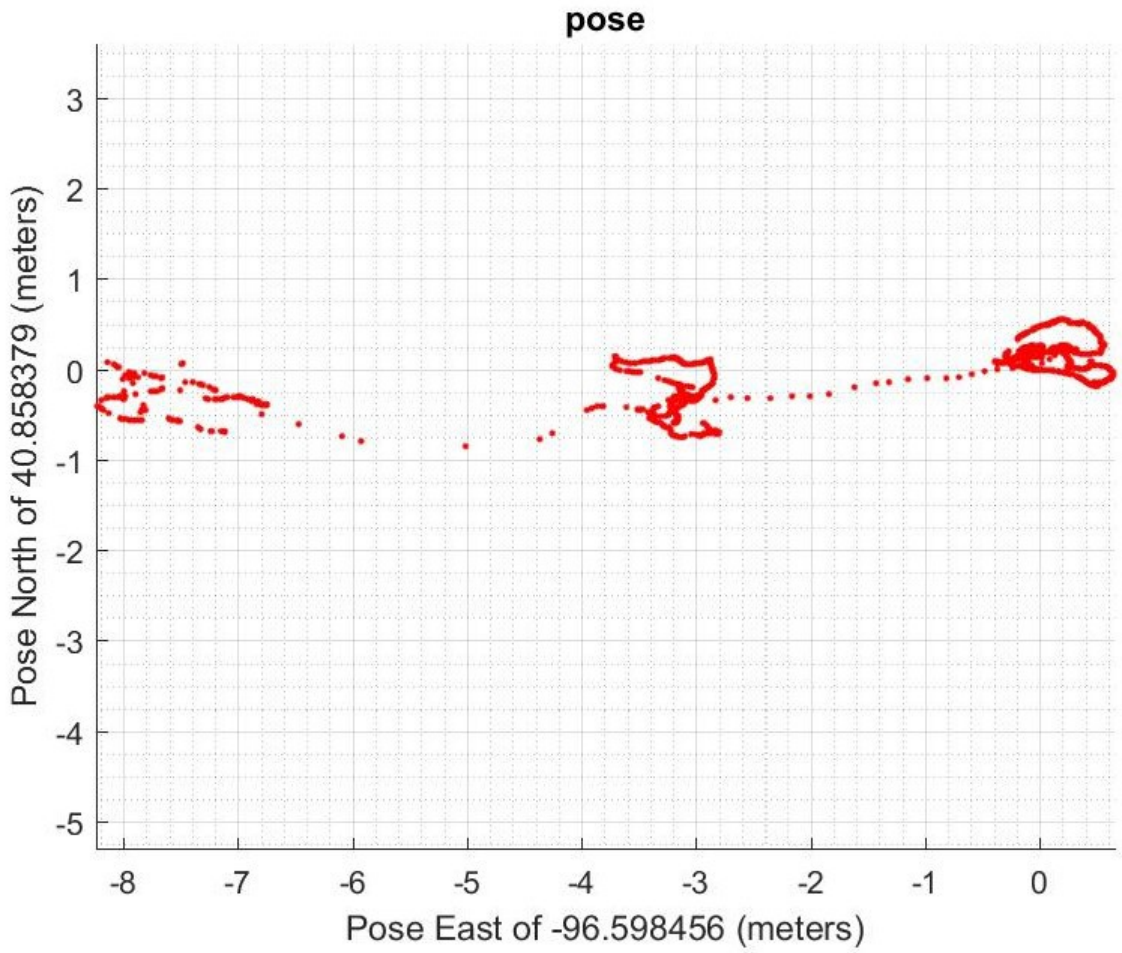


Figure 4.2: Position estimates with RTK GPS receivers 7.5 m, 4.0 m, and 0.1 m apart while off the vehicle. This figure was post processed with MATLAB from a ROStopic.

The top image of the following figure depicts the precision achieved during one of the last flights involving the RTK GPS receivers. While the bottom image is a zoomed in version of the top plot highlighting the stationary precision achieved.

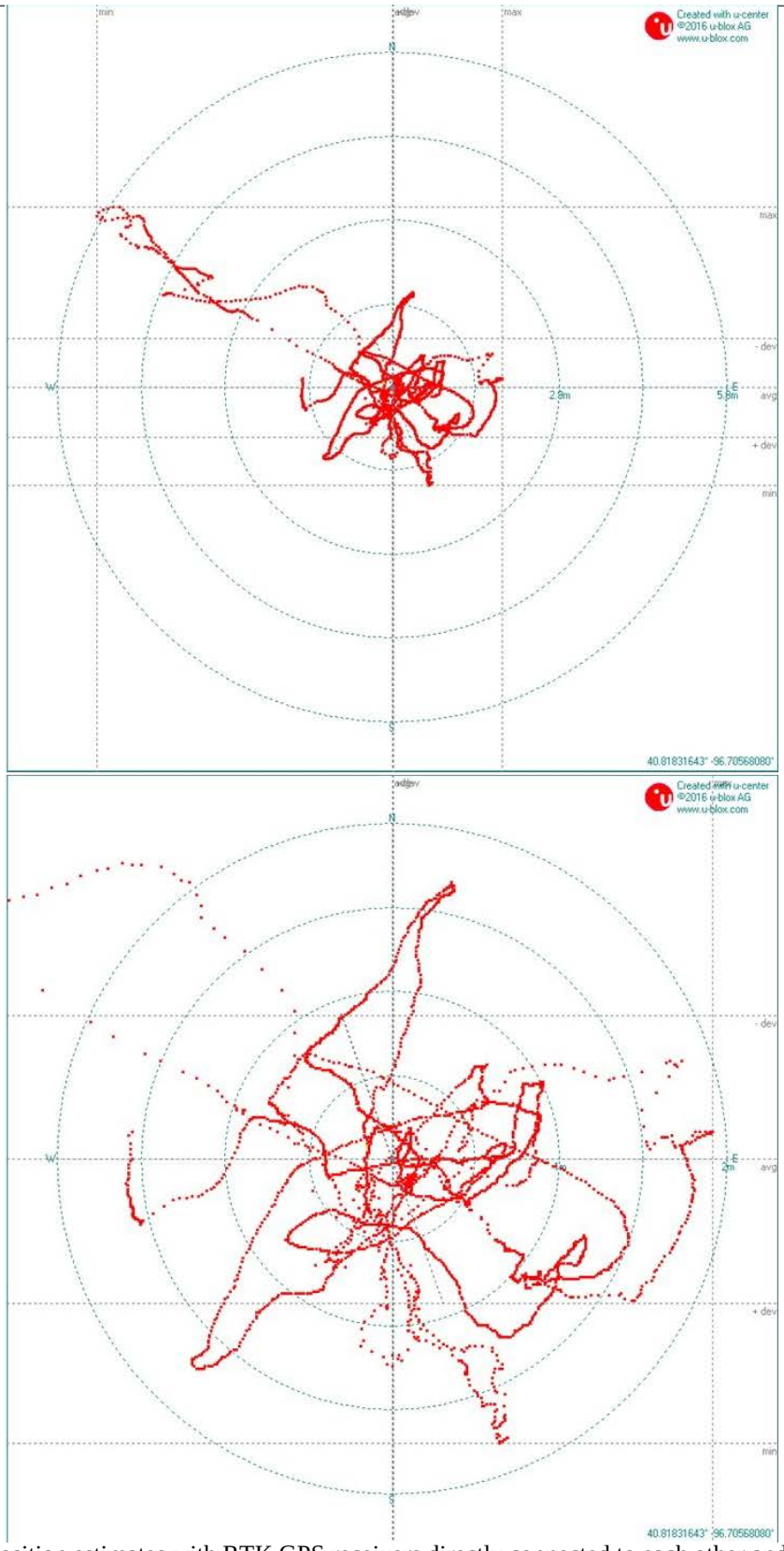


Figure 4.3: Position estimates with RTK GPS receivers directly connected to each other and powered through the vehicle. This figure was post processed with u-Center from a ReachView solution.

Chapter 5

Discussion

5.1 Future Work

Creating a trust model of Ascending Technologies Hummingbird vehicle motor assemblies based on the deviation of the piloted vehicle's indoor position data from the position data of the vehicle that performed 510 flights is a possible next step. Small deviations may require recalibration of the vehicle and/or firmware updates while larger deviations may indicate failing motor assemblies. Successfully integrating NIMBUS Laboratories control software with other popular unmanned aerial vehicles and executing the same tests would be a step forward. This poses challenges, since other popular unmanned aerial vehicles have autopilots that are not easily replaced or modifiable.

Using an algorithm to fill in missing data blocks in the generated heatmaps by using the data from adjacent blocks or linear regression is a next step. Finding a reliable power splitting solution to power the vehicle and an additional computing device such as the Odroid would be beneficial to current and future members of NIMBUS. These tests were performed with an operator manually piloting the vehicle due to inexperience of the test designer with automated outdoor flight. Automating the navigation task in these tests would be a step forward. Automatic launching of ROS nodes in a screen session and detaching

that session when the Odroid boots up would be a time and battery saving feat.

Range testing the two types of XBee-Pro antennas to their range limits in the vertical position is not complete. Full range testing of fully operational XBee-Pro receivers with the above antennas in horizontal North-South and horizontal East-West positions has yet to be conducted. The results of these tests would be valuable when deciding how to orient XBee-Pro antennas and for avoiding bad antenna orientations.

Comparison of multiple RTK GPS receivers, a regular GPS receiver, and a ground truth measurement is a test scenario designed for the purchased UAV RTK GPS receivers. This comparison did not take place due to the problems encountered in getting the units to function. Though this comparison did not occur, it is a test that should be done in the future at some time.

5.2 Conclusion

The reliability of all Hummingbirds would increase if packet delays could be minimized. Implementing a safety node that is activated after an amount of seconds, exact value can be parameterized, has elapsed between control messages. This value should be between [1.000, 1.300] seconds for tasked position and commanded state control messages and between [control interval, (control interval)*(queue size)] seconds for control input. A lower value would provide further safety but would increase overall mission time. A higher value would allow for more fluid operation of the vehicle but would leave the vehicle susceptible to more anomalies. Of the abnormal behaviors observed most lasted several seconds and then recovery occurred. Though packet delays and losses of serial communication occurred, the unmanned aerial vehicles recovered within several seconds. Power supply noise did not have an affect on vehicle performance. The variance in maximum power required during height changes between tests per vehicle was a surprise. Variable launch heights achieved contributed to the wide variance in maximum power usage. Position error

measurements for each axis are within several millimeters for each vehicle. The respective variances for position error measurements are all relatively small values. Implementing a parabolic tasked waypose comparison curve instead of a linear curve may reduce calculated position error measurements.

The effective radio range of XBee-Pros is considerably less than the specification radio range (750 m) of XBee-Pros. A vehicle within an unmanned aerial vehicle fleet using XBee-Pros to communicate to neighboring vehicles should be able to conduct reliable communications to any vehicle within a 100 m radius sphere from the vehicles position. To achieve a maximum vertical distance of approximately 50 m a signal to start descending or to fly to the next marked position was given to the pilot after exceeding 46 m.

During one flight both XBee-Pros were in the same namespace, all pre-flight checks (battery, GPS, RSSI) looked good but communication with the XBee-Pro transmitting GPS data was interrupted and when recovered the vehicle was above 50 m. USB devices on different channels, using the same baud rate, and using different topic names must be placed in different ROS namespaces for reliable communications.

The tests that utilized XBee-Pros with 2.4 GHz 1/4 wavelength antennas in the vertical position displayed the lowest RSSI values per distance. This might be correlated with the fact that they are newer than the XBee-Pros used with the 900 MHz 1/4 wavelength antennas. A full set of tests conducted with equipment (2 XBee-Pros with attached 2.4 GHz 1/4 wavelength antennas, 2 XBee-Pros with attached co-axial connection, 4 Sparkfun Explorer Dongles, 2 co-axial normal-mode helical 900 MHz 1/4 wavelength antennas) purchased at the same time should alleviate this bias. Differences between RSSI values of two paired XBee-Pros were expected to be much lower than the measured differences.

Professional positioning systems with RTK GPS capabilities exist but cost as much or more than the vehicle itself. Though there exist RTK GPS receivers that can reasonably fit on unmanned aerial vehicles, not all systems are black box solutions. One or more of the available systems not tested may offer less setup time and increased ease of system

integration. Maybe in another three years UAV RTK GPS receivers will be both affordable and black box solutions. At the present, the affordable systems are fit for hobbyists to spend their “free time” tuning and not for research or other professional uses while the black box solutions are unaffordable for upgrading a UAV fleet’s GPS.

Bibliography

[1] Addy, H. E., Orchard, D., Wright, W. B., Oleskiw, M. (2016). Altitude Effects on Thermal Ice Protection System Performance; a Study of an Alternative Approach.

[2] Amoozgar, M. H., Chamseddine, A., Zhang, Y. (2012). Fault-Tolerant Fuzzy Gain-Scheduled PID for a Quadrotor Helicopter Testbed in the Presence of Actuator Faults. IFAC Proceedings Volumes, 45(3), 282–287. https://doi.org/10.3182/20120328-3-IT-3014.00048

[3] "Parrot AR.Drone," Wikipedia, the free encyclopedia. 14-Sep-2016.

[4] Ascending Technologies "AscTec Hummingbird with AutoPilot User's Manual - AscTec_AutoPilot_manual_v1.0_small.pdf." [Online]. Available: [Accessed: 05-Oct-2016].

[5] AttoPilot International, "Compact DC Voltage and Current Sense PCB with Analog Output," AttoPilot International, Oklahoma, 2011.

[6] Basha, E., Yuen, N., O'Rourke, M., Detweiler, C. (2014). Analysis of Algorithms for Multi-Modal Communications in Underwater Sensor Networks (p. 17). Presented at the Proceedings of the International Conference on Underwater Networks and Systems, ACM.

[7] Bateman, F., Noura, H., Ouladsine, M. (2011). Fault Diagnosis and Fault-Tolerant Control Strategy for the Aerosonde UAV. IEEE Transactions on Aerospace and Electronic Systems, 47(3), 2119–2137. https://doi.org/10.1109/TAES.2011.5937287

[8] Biaz, S., Ji, Y., Qi, B., Shaoen, W. (2005). Realistic radio range irregularity model and its impact on localization for wireless sensor networks. In Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005. (Vol. 2, pp. 669–673). https://doi.org/10.1109/WCNM.2005.1544143

[9] BK Precision"1693 & 1694 - High Current Switching Mode Power Supply - 1900_1901_1902 manual.pdf." [Online]. Available: [Accessed: 05-Oct-2016].

[10] Boyd, M. A., Bavuso, S. J. (1992). Modeling a highly reliable fault-tolerant guidance, navigation, and control system for long duration manned spacecraft. In Digital Avionics Systems Conference, 1992. Proceedings., IEEE/AIAA 11th (pp. 464–469). https://doi.org/10.1109/DASC.1992.282116

[11] Bradley, J. M., Atkins, E. M. (2014). Cyber–Physical Optimization for Unmanned Aircraft Systems. Journal of Aerospace Information Systems, 11(1), 48-60.

[12] Bradley, J. M., Atkins, E. M. (2012). Toward Continuous State Space Regulation of Coupled Cyber Physical Systems. Proceedings of the IEEE, 100(1), 60–74. https://doi.org/10.1109/JPROC.2011.2161239

[13] Browne, D. W., Loo, C., Ha, J., Borgstrom, H., Fitz, M. P., Kaiser, W., Tabrizi, H. Directional Radio Propagation Measurements for Near-Ground Peer-to-Peer Networks.

[14] Burcham, F., et al. Development and flight. NASA, Technical Report Technical Paper 3627, 1996.

[15] Cavka, M., Krpetic´, R., Vasic´, D., Bilas, V. (2013). TeleAssistant - Assisted living platform applied to fall detection. In 2013 36th International Convention on Information Communication Technology Electronics Microelectronics (MIPRO) (pp. 490–494).

[16] Cen, Z., Noura, H., Younes, Y. A. (2013). Robust Fault Estimation on a real quadrotor UAV using optimized Adaptive Thau Observer. In 2013 International Conference on Unmanned Aircraft Systems (ICUAS) (pp. 550–556). https://doi.org/10.1109/ICUAS.2013.6564732

[17] Chamseddine, A., Theilliol, D., Zhang, Y. m., Join, C., Rabbath, C. a. (2015). Active fault-tolerant control system design with trajectory re-planning against actuator faults and saturation: Application to a quadrotor unmanned aerial vehicle. International Journal of Adaptive Control and Signal Processing, 29(1), 1–23. https://doi.org/10.1002/acs.2451

[18] Chen, F., Lu, F., Jiang, B., Tao, G. (2014). Adaptive compensation control of the quadrotor helicopter using quantum information technology and disturbance observer. Journal of the Franklin Institute, 351(1), 442–455. https://doi.org/10.1016/j.jfranklin.2013.09.009

[19] Chen, Y., Guo, D., Cui, W., Li, J. (2015). Self-adaptive Wi-Fi indoor positioning model. In 2015 23rd International Conference on Geoinformatics (pp. 1–6). https://doi.org/10.1109/GEOINFORMATICS.2015.7378593

[20] Cork, L., Walker, R. (2007). Sensor Fault Detection for UAVs using a Nonlinear Dynamic Model and the IMM-UKF Algorithm. In Information, Decision and Control, 2007. IDC ’07 (pp. 230–235). https://doi.org/10.1109/IDC.2007.374555

[21] Corke, P., Detweiler, C., Dunbabin, M., Hamilton, M., Rus, D., Vasilescu, I. (2007). Experiments with Underwater Robot Localization and Tracking. In Proceedings 2007 IEEE International Conference on Robotics and Automation (pp. 4556–4561). https://doi.org/10.1109/ROBOT.2007.364181

[22] Corps, S. G., “A320 flight controls,” in The Society of Experimental Test Pilots, Twenty-Ninth Symposium Proceedings, September 1985.

[23] Dannenhoffer, J., “Development of hardware for the X-29A flight control system,” in NAECON, vol. 1, pp. 440-445, 1985.

[24] Detweiler, C., Decentralized Sensor Placement and Mobile Localization on an Underwater Sensor Network with Depth Adjustment Capabilities. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, July 2010.

[25] Diamanti, K., Soutis, C. (2010). Structural health monitoring techniques for aircraft composite structures. Progress in Aerospace Sciences, 46(8), 342-352.

[26] Doniec, M., Detweiler, C., Vasilescu, I., Chitre, M., Hoffmann-Kuhnt, M., Rus, D. (2010). AquaOptical: A Lightweight Device for High-rate Long-range Underwater Point-to-Point Communication. Marine Technology Society Journal, 44(4), 55–65. https://doi.org/10.4031/MTSJ.44.4.6

[27] Doniec, M., Detweiler, C., Vasilescu, I., Rus, D. (2010). Using optical communication for remote underwater robot operation. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 4017–4022). https://doi.org/10.1109/IROS.2010.5650224

[28] U.S. Department of Defense Unmanned aerial vehicle road map 2005—2030. Office of the Secretary of Defense, Technical Report, 2005.

[29] Freddi, A., Lanzon, A., Longhi, S. (2011). A Feedback Linearization Approach to Fault Tolerance in Quadrotor Vehicles. IFAC Proceedings Volumes, 44(1), 5413–5418. https://doi.org/10.3182/20110828-6-IT-1002.02016

[30] RF Link Planning - Fresnel Zone Clearance. (n.d.). Retrieved October 17, 2016, from http://www.softwright.com/faq/engineering/Fresnel %20Zone%20Clearance.html

[31] Futaba 8J 8-channel 2.4GHz Computer Radio System. (n.d.). Retrieved October 5, 2016, from http://www.futabarc.com/systems/futk8100-8j/index.html

[32] Goyal, A., Shahabuddin, P., Heidelberger, P., and Nicola, V., “A unified framework for simulating Markovian models of highly dependable systems,” IEEE Transactions on Computers, vol. 41, pp. 36 -51, January 1992.

[33] Grodi, R., Rawat, D. B., Bajracharya, C. (2015). Performance evaluation of Unmanned Aerial Vehicle ad hoc networks. In SoutheastCon 2015 (pp. 1–4). https://doi.org/10.1109/SECON.2015.7133020.

[34] Heredia, G., Caballero, F., Maza, I., Merino, L., Viguria, A., Ollero, A. (2009). Multi-Unmanned Aerial Vehicle (UAV) Cooperative Fault Detection Employing Differential Global Positioning (DGPS), Inertial and Vision Sensors. Sensors, 9(9), 7566–7579. https://doi.org/10.3390/s90907566

[35] Hecht, H., and Florentino, E., “Reliability assessment of spacecraft electronics,” in Proceedings of the Reliability and Maintainability Symposium, pp. 341-346, January 1987

[36] Heinz, A., Haszler, A., Keidel, C., Moldenhauer, S., Benedictus, R., Miller, W. S. (2000). Recent development in aluminium alloys for aerospace applications. Materials Science and Engineering: A, 280(1), 102-107.

[37] Higgins, A. (2000). Adhesive bonding of aircraft structures. International Journal of Adhesion and Adhesives, 20(5), 367-376.

[38] Hopkins, A. L., Smith, T. B., Lala, J. H. (1978). FTMP #8212;A highly reliable fault-tolerant multiprocess for aircraft. Proceedings of the IEEE, 66(10), 1221–1239. https://doi.org/10.1109/PROC.1978.11113

[39] Jiang, J., Yu, X. (2012). Fault-tolerant control systems: A comparative study between active and passive approaches. Annual Reviews in Control, 36(1), 60–72. https://doi.org/10.1016/j.arcontrol.2012.03.005

[40] Jordan, T., Langford, W., Belcastro, C., Foster, J., Shah, G., Howland, G., Kidd, R. (2004). Development of a dynamically scaled generic transport model testbed for flight research experiments.

[41] Kasdaglis, N., Bernard, T., Stephane, L., Boy, G. A. (2016). Affordant Guidance for In-Flight Loss of Control: The Trajectory Recovery System (TRS). Paris, France: HCI-Aero.

[42] Kendoul, F., Yu, Z., Nonami, K. (2010). Guidance and nonlinear control system for autonomous flight of minirotorcraft unmanned aerial vehicles. Journal of Field Robotics, 27(3), 311–334. https://doi.org/10.1002/rob.20327

[43] Kessler, M. R., Sottos, N. R., White, S. R. (2003). Self-healing structural composite materials. Composites Part A: applied science and manufacturing, 34(8), 743-753.

[44] Khare, V. R., Wang, F. Z., Wu, S., Deng, Y., Thompson, C. (2008). Ad-hoc network of unmanned aerial vehicle swarms for search and destroy tasks. In 2008 4th International IEEE Conference Intelligent Systems (Vol. 1, pp. 6–65–6–72). https://doi.org/10.1109/IS.2008.4670440

[45] Lewis, E., E., and Boehm, F., “Monte Carlo simulation of Markov unreliability models,” Nuclear Engineering and Design, vol. 77, pp. 49-62, 1984

[46] Li, B., Jiang, Y., Sun, J., Cai, L., Wen, C.-Y. (2016). Development and Testing of a Two-UAV Communication Relay System. Sensors, 16(10), 1696. https://doi.org/10.3390/s16101696

[47] Liao, F., Wang, J. L., Yang, G. H. (2002). Reliable robust flight tracking control: an LMI approach. IEEE Transactions on Control Systems Technology, 10(1), 76–89. https://doi.org/10.1109/87.974340

[48] Lippiello, V., Ruggiero, F., Serra, D. (2014). Emergency landing for a quadrotor in case of a propeller failure: A backstepping approach. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 4782–4788). https://doi.org/10.1109/IROS.2014.6943242

[49] Lymberopoulos, D., Lindsey, Q., and Savvides, A., An empirical characterization of radio signal strength variability in 3-d ieee 802.15.4 networks using monopole antennas. The European Workshop on Wireless Sensor Networks, 2006.

[50] Ma, L., (2011, January 12). Development of Fault Detection and Diagnosis Techniques with Applications to Fixed-wing and Rotary-wing UAVs (masters). Concordia University. Retrieved from http://spectrum.library.concordia.ca/7466/

[51] Marais, K., Dulac, N., Leveson, N. (2004, March). Beyond normal accidents and high reliability organizations: The need for an alternative approach to safety in complex systems. In Engineering Systems Division Symposium (pp. 29-31).

[52] Messous, M.-A., Senouci, S.-M., Sedjelmaci, H. (2016). Network connectivity and area coverage for UAV fleet mobility model with energy constraint (pp. 1–6). IEEE. https://doi.org/10.1109/WCNC.2016.7565125

[53] Meyer, G., Su, R., Hunt, L. R. (1984). Application of nonlinear transformations to automatic flight control. Automatica, 20(1), 103–107. https://doi.org/10.1016/0005-1098(84)90069-4

[54] Navipedia "RTK Fundamentals - Navipedia." [Online]. Available: [Accessed: 28-Sep-2016].

[55] Nicola, V., F., Nakayama, M., K., Heidelberger, P., Goyal, A., "Fast simulation of dependability models with general failure, repair, and maintenance processes," in Proceedings of the Twentieth International Symposium on Fault Tolerant Computing, pp. 491-498, June, 1990.

[56] Napolitano, M. R., An, Y., Seanor, B. A. (2000). A fault tolerant flight control system for sensor and actuator failures using neural networks. Aircraft Design, 3(2), 103–128. https://doi.org/10.1016/S1369-8869(00)00009-4

[57] Pabst, R., Walke, B. H., Schultz, D. C., Herhold, P., Yanikomeroglu, H., Mukherjee, S., Fettweis, G. P. (2004). Relay-based deployment concepts for wireless and mobile broadband radio. IEEE Communications Magazine, 42(9), 80–89. https://doi.org/10.1109/MCOM.2004.1336724

[58] Palmer, J., Yuen, N., Ore, J. P., Detweiler, C., Basha, E. (2015, May). On air-to-water radio communication between UAVs and water sensor networks. In 2015 IEEE International Conference on Robotics and Automation (ICRA) (pp. 5311-5317). IEEE.

[59] Pappu, V. S. R., Steck, J. E., Balakrishnan, S. (2015). Hardware-In-Loop and Flight Testing of Modified State Observer Based Adaptation for a General Aviation Aircraft.

[60] Park, M. J., Coldwell, C. (2013). Sensor Integrated Navigation for a Target Finding UAV. In A. Natraj, S. Cameron, C. Melhuish, M. Witkowski (Eds.), Towards Autonomous Robotic Systems (pp. 76–89). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-662-43645-5_10

[61] Qi, J., Han, J. (2007). Application of Wavelets Transform to Fault Detection in Rotorcraft UAV Sensor Failure. Journal of Bionic Engineering, 4(4), 265–270. https://doi.org/10.1016/S1672-6529(07)60040-7

[62] Rafaralahy, H., Richard, E., Boutayeb, M., Zasadzinski, M. (2008). Simultaneous observer based sensor diagnosis and speed estimation of Unmanned Aerial Vehicle. In 47th IEEE Conference on Decision and Control, 2008. CDC 2008 (pp. 2938–2943). https://doi.org/10.1109/CDC.2008.4739369

[63] Rago, C., Prasanth, R., Mehra, R. K., Fortenbaugh, R. (1998). Failure detection and identification and fault tolerant control using the IMM-KF with applications to the Eagle-Eye UAV. In Proceedings of the 37th IEEE Conference on Decision and Control, 1998 (Vol. 4, pp. 4208–4213 vol.4). https://doi.org/10.1109/CDC.1998.761963

[64] Ranjbaran, M., Khorasani, K. (2010a). Fault recovery of an under-actuated quadrotor Aerial Vehicle. In 49th IEEE Conference on Decision and Control (CDC) (pp. 4385–4392). https://doi.org/10.1109/CDC.2010.5718140

[65] Rawat, D. B., Grodi, R., Bajracharya, C. (2015). Enhancing connectivity for communication and control in Unmanned Aerial Vehicle networks. In 2015 IEEE Radio and Wireless Symposium (RWS) (pp. 200–202). https://doi.org/10.1109/RWS.2015.7129745

[66] Rusdy, Y. M., Nik Syahrim, N. A., Nazrin, M. M., Nurdiana, N. (2012). Effect of Distance on the Maximum Data Transfer for Different Mounting Elevations of XBee Pro Module in Viral Advertisement System. Presented at the 2012 2nd International Conference on Electrical Engineering and Applications, Bali. Retrieved from http://www.iceea.net/cfp.htm

[67] Rusnak, I., Guez, A., Bar-Kana, I., Steinberg, M. (1992). Online identification and control of linearized aircraft dynamics. IEEE Aerospace and Electronic Systems Magazine, 7(7), 56–60. https://doi.org/10.1109/62.149797

[68] Sadeghzadeh, I., Chamseddine, A., Zhang, Y., Theilliol, D. (2012). Control Allocation and Re-allocation for a Modified Quadrotor Helicopter against Actuator Faults. IFAC Proceedings Volumes, 45(20), 247–252. https://doi.org/10.3182/20120829-3-MX-2028.00291

[69] Schumann, J., Liu, Y. (2007). Tools and Methods for the Verification and Validation of Adaptive Aircraft Control Systems. In 2007 IEEE Aerospace Conference (pp. 1–8). https://doi.org/10.1109/AERO.2007.352766

[70] Sharifi, F., Mirzaei, M., Gordon, B. W., Zhang, Y. (2010). Fault tolerant control of a quadrotor UAV using sliding mode control. In 2010 Conference on Control and Fault-Tolerant Systems (SysTol) (pp. 239–244). https://doi.org/10.1109/SYSTOL.2010.5675979

[71] Shaw, A., Mohseni, K. (2011). A Fluid Dynamic Based Coordination of a Wireless Sensor Network of Unmanned Aerial Vehicles: 3-D Simulation and Wireless Communication Characterization. IEEE Sensors Journal, 11(3), 722–736. https://doi.org/10.1109/JSEN.2010.2064294

[72] Smith, D. D. (2011). Design, development, and testing of a multi-agent autonomous surface fleet for environmental applications (Doctoral dissertation, Faculty of the Louisiana State University and Agricultural and Mechanical College in partial fulfillment of the requirements for the degree of Master of Science in Biological and Agricultural Engineering in The Department of Biological and Agricultural Engineering by Daniel Davis Smith BS, Louisiana State University).

[73] "UAV Roadmap 2005", Office of the Secretary of Defense, Aug. 2005, pp. 213,

[74] Srikanth, M. B., Dydek, Z. T., Annaswamy, A. M., Lavretsky, E. (2009). A robust environment for simulation and testing of adaptive control for mini-UAVs. In 2009 American Control Conference (pp. 5398–5403). https://doi.org/10.1109/ACC.2009.5160468

[75] Steinberg, M. (2005). Historical Overview of Research in Reconfigurable Flight Control. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 219(4), 263–275. https://doi.org/10.1243/095441005X30379

[76] u-Blox "u-blox8-M8_ReceiverDescrProtSpec_(UBX-13003221)_Public.pdf." [Online]. Available: [Accessed: 28-Sep-2016].

[77] Vasilescu, I.; Detweiler, C.; Rus, D. AquaNodes: An Underwater Sensor Network. In Proceedings of the Second Workshop on Underwater Networks (WuWNet '07), Montreal, Quebec, Canada, September 2007.

[78] VICON. (n.d.). What is motion capture. Retrieved October 5, 2016, from http://www.vicon.com/what-is-motion-capture

[79] T. Editor, "Low Cost RTK GPS Receiver Launched on Kickstarter," UAS VISION, 07-Aug-2013.

[80] Walter, C. J., "MAFT: An architecture for reliable fly-by wire flight controls," in Proceedings of the AIAA/IEEE Digital Avionics Systems Conference, San Jose, CA, October 1988.

[81] Wang, Y., Ramamurthy, B. (2007). Layered Clustering Communication Protocol for Wireless Sensor Networks. In Proceedings of 16th International Conference on Computer Communications and Networks, 2007. ICCCN 2007 (pp. 844–849). https://doi.org/10.1109/ICCCN.2007.4317923

[82] Wensley, J. H., Lamport, L., Goldberg, J., Green, M. W., Levitt, K. N., Melliar-Smith, P. M., Weinstock, C. B. (1978). SIFT: Design and analysis of a fault-tolerant computer for aircraft control. Proceedings of the IEEE, 66(10), 1240–1255. https://doi.org/10.1109/PROC.1978.11114

[83] product-manual_XB_802.15.4_OEM_RF-Modules_v1.xAx.book - XBee-Datasheet.pdf. (n.d.). Retrieved October 17, 2016, from https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

[84] Xu, Z., Huo, J., Wang, Y., Yuan, J., Shan, X., Feng, Z. (2011). Analyzing two connectivities in UAV-ground mobile ad hoc networks. In 2011 IEEE International Conference on Computer Science and Automation Engineering (CSAE) (Vol. 2, pp. 158–162). https://doi.org/10.1109/CSAE.2011.5952445

[85] Yang, G.H., Wang, J. L., Soh, Y. C. (2001). Reliable H-infinity controller design for linear systems. Automatica, 37(5), 717–725. https://doi.org/10.1016/S0005-1098(01)00007-3

[86] Zhang, Y. M., Chamseddine, A., Rabbath, C. A., Gordon, B. W., Su, C.-Y., Rakheja, S., Gosselin, P. (2013). Development of advanced FDD and FTC techniques with application to an unmanned quadrotor helicopter testbed. Journal of the Franklin Institute, 350(9), 2396–2422. https://doi.org/10.1016/j.jfranklin.2013.01.009

[87] Zhang, Y., and Jiang, J., "Bibliographical review on reconfigurable fault-tolerant control systems," Annual Reviews in Control, vol. 32, p. 229-252, 2008.

[88] Zhang, Y., and Jiang, J. Bibliographical review on reconfigurable fault-tolerant control systems. In Proceedings of IFAC Safe Process, Washington, D.C.,2003; doi:10.1016/j.arcontrol.2008.03.008.

[89] Zhang, Y., and Li, X., R., "Detection and diagnosis of sensor and actuator failures using interacting multiple model estimator", presented at Decision and Control, 1997. Proceedings of the 37th IEEE Conference on, 1997.

Appendix A

Figure Descriptions

A.1 Control

Plots displaying control sequence numbers. Jumps in the graph represent different quad control input sources. Overlapping lines represent two or more sources controlling quad control input at nearly the same time.

A.2 Position

Position plots for task waypose position, subject pose position, Vicon position, and quad control input pitch, roll, and thrust. The top two plots are measured in meters while the bottom left plot is measured in meters as well and the points in the bottom right plot are unit normalized controller control rates.

A.3 Power

Plots displaying calculated apparent power measured in Watts, sensor voltage measured in volts, and sensor current measured in Amperes from the data recorded from both AttoPilot sensors as well as battery voltage recorded via the subject status node.

A.4 Rotation

Rotation plots for task waypose rotation, subject pose rotation, Vicon rotation, and quad control input rotation. The top two plots are measured in radians while the bottom left plot is measured in unit quaternion and the points in the bottom right plot are unit normalized directional turning rates.

A.5 3-D

Three dimensional plots comparing x, y, and z position from subject pose with the tasked x, y, and z position (rigid blue lines).

A.6 Status

The top plot displays target state (blue +) and current state (black .). The middle plot displays binary motor status in black (lagging line) as a continuous line (on = 1, off = 0), binary serial mode status in cyan (leading line) as a continuous line (enabled = 1, disabled = 0), and binary waypose status (blue x) as discrete points. The bottom plot displays binary GPS control (flight) status as indicated by quad control input.

Appendix B

Documents

B.1 Failure Testing Procedure

Failure Testing Procedure

1. **Introduction**
    a. **Purpose**
        1. The research conducted by the NIMBUS Lab involves operation and use of unmanned aerial vehicles. An area of research is providing a complete understanding of each unmanned aerial vehicles' voltage and amperage consumption characteristics needed to perform basic maneuvers. This series of tests is focused on detailing the voltage and current used by an unmanned aerial vehicle during basic flying operations such as launching, maneuvering in the x, y and z directions with and without rotating, and landing. Testing will be conducted to determine characteristics that cause failure of normal unmanned aerial vehicle operation. Monitoring of parameters is to be conducted such that the cause of any failure will be recorded and can be analyzed.
    b. **System Overview**
        i. **Environment**
            1. An 28' by 24' by 9.5' indoor caged area consisting of PVC pipe, zip ties and wire mesh.
            2. Landing surface consists of carpeted cement.
            3. Twelve cameras, eight cameras mounted near the ceiling and four mounted near the floor, are integrated into Vicon Tracker.
        ii. **Firmware**
            1. ATmega328 on an Arduino Uno.
            2. Manufacturer specific firmware is installed on each unmanned aerial vehicle. AscTec AutoPilot is installed on the Ascending Technology Hummingbird.
                a. AscTec AutoPilot versions installed on AscTec Hummingbird 0033878000 are AutoPilot LowLevel #20636 V3.12 and AutoPilot HighLevel V3.11.
                b. AscTec AutoPilot versions installed on AscTec Hummingbird 0033913900 are AutoPilot LowLevel #20632 V2.14 and AutoPilot HighLevel V3.0.
                c. AscTec AutoPilot versions installed on AscTec Hummingbird 0033913800 are AutoPilot LowLevel #20633 V3.12 and AutoPilot HighLevel V3.0.
                d. AscTec AutoPilot versions installed on AscTec Hummingbird 0033878100 are AutoPilot LowLevel #20637 V3.12 and AutoPilot HighLevel V3.0.
                e. AscTec AutoPilot versions installed on AscTec Hummingbird 0033811700 are AutoPilot LowLevel #20502 V3.12 and AutoPilot HighLevel V3.11.

3. The FASST system is installed on a Futaba 7 channel 2.4 GHz controller.
4. ZigBee Pro feature set version 2x64 is installed on the XBees used.
5. B&K Precision control logic is installed on a switching DC power supply.

iii. Hardware

1. A switching DC power supply powered from an 120 volt AC wall socket to provide power to an unmanned aerial vehicle and to monitor changing voltage and current levels as the connected unmanned aerial vehicle performs basicmaneuvers.
    a. B&K Precision 1900 Switching DC Power Supply 7611-3600-1010
2. Two voltage and current sensing units to provide indication of changing voltage and current as an unmanned aerial vehicle performs basic maneuvers. The usage of two sensors provides redundancy as well as an indication that the individual sensors are performing as theyshould.
    a. SEN - 10643 ROHS
3. An Arduino Uno used to receive and convert the output of the analog voltage and current signals sensed by the above mentioned sensors to a digital signal that is then transmitted over a serial line using a USB A to B cable to alaptop.
    a. Arduino Uno R3
4. A laptop used to start and stop the test runs, record test performance data, and store the results of the test runs.
5. Two paired Digi International XBee Pros used to communicate between the laptop and the unmanned aerial vehicle.
    a. XBee-Pro ZB
6. Cables and wires used to connect the switching DC power supply to 120 volt AC power, to connect the switching DC power supply to two voltage and current sensors as well as to an unmanned aerial vehicle, to connect two voltage and current sensors to an Arduino Uno, to connect an Arduino Uno to a laptop, to connect a laptop to local area network, and to connect a XBee Pro to a laptop.
7. A paired Futaba 7-channel 2.4 GHz controller set in automatic mode but available to take manual control of the unmanned aerial vehicle in event of system failure.
    a. Futaba T7C T282330
8. An unmanned aerial vehicle that is the system being tested.
    a. AscTec Hummingbird

iv. Software

1. The software used on the Arduino Uno was provided by the retailer we bought the AttoPilot sensors from, Sparkfun Electronics. It's language is in C++ and was edited and compiled on the Arduino IDE. Modifications were made to Sparkfun's original program for use over a serial port and output of raw digital data instead of floats.
2. The Indigo Igloo distribution of the Robot Operating System (ROS) running on a VMWare Player 7.1.0 virtual machine using Ubuntu Trusty 14.04 as the operating system.
3. Vicon Tracker is used to create a bounded indoor 3-D position system.
4. Massachusetts Institute of Technology Ascending Technologies suite is used to control the maneuvers of an unmanned aerial vehicle.

2. Testing Setup

a. Test Equipment

i. B&K Precision 1900 Switching DC Power Supply
ii. Arduino Uno
iii. 2 AttoPilot Voltage and Current Sense Breakout - 45A
iv. Black 12 AWG (American Wire Gauge), 10 feet 6 inches
v. Red 12 AWG (American Wire Gauge), 10 feet 6 inches
vi. 2x Black 12 AWG (American Wire Gauge), 5 inches
vii. 2x Red 12 AWG (American Wire Gauge), 5 inches
viii. 2x Black 12 AWG (American Wire Gauge), 4 inches
ix. 2x Red 12 AWG (American Wire Gauge), 4 inches
x. 2x Black 26 AWG (American Wire Gauge), 11 inches, with contact.
xi. 2x Red 26 AWG (American Wire Gauge), 11 inches, with contact.
xii. 2x Green 26 AWG (American Wire Gauge), 11 inches, with contact.
xiii. 3 Black Banana Plugs
xiv. 2 Black Banana Sockets
xv. 3 Red Banana Plugs
xvi. 2 Red Banana Sockets
xvii. E88446 - Ching Cheng - 18/3 SVT Power Cord 7.5 feet
xviii. USB Mini-B Cable
xix. USB Cable A to B
xx. Clear, green, yellow and red heat shrink.
xxi. Electrical Tape
xxii. Male and Female Deans T-connector
xxiii. Ethernet cable
xxiv. Futaba 7-channel 2.4 GHz radio control system
xxv. 2 XBee Pros
xxvi. Laptop
xxvii. Unmanned Aerial Vehicle (AscTec Hummingbird)

b. Test Results Format

i. Select topics are recorded in a bag file using rosbag. The bag file is placed in a test directory on the cse server. The bag file topics are imported into .cvs files that then are imported into MATLAB. Individual fields are parsed and plotted in MATLAB.

3. Testing Procedure

a. System Setup

i. Construct 10.5 foot cable that connects switching DC power supply to voltage and current sensors as well as the Deans T-connector of the unmanned aerial vehicle. Steps to construction:

1. Cut out 10.5' lengths of red and black 12 AWG wire.
2. Solder red and black banana plugs to one end of the red and black wires respectively.
3. Solder red and black wires to their corresponding male plugs of a Dean's T-connector, which matches the GND and Voltage of the unmanned aerial vehicle it'll be plugging into (It is suggested to plug another Dean's T-connector into the female end of the one being soldered to the wire to keep the sockets from misaligning).
4. Heat shrink the Dean's T-connector end to add structure, and then heat shrink small bands about every 8" down the wire to keep the red and black cables together.

ii. Construct AttoPilot 45A sensors cable:

1. Solder 4" Red 12 AWG to In^+ and Out^+ solder pads and 4" Black 12 AWG to GND pad, flooding large via holes with solder.
2. Solder black, red and green 26 AWG into GND, V and I 0.1" spaced plated through holes respectively, with connectors at opposite ends.
3. Solder red and black banana plugs onto red and black 12 AWG respectively on the In^+ side of the sensor.
4. Solder Red and black banana sockets onto red and black 12 AWG respectively on the Out^+ side of the sensor.
5. Heat shrink banana sockets with red and black and the whole sensor with clear heat shrink.

iii. Connect voltage and current sensors to Arduino Uno.

iv. Program Arduino to convert analog signals received from the voltage and current sensors to digital signals transmitted over a serial port.

v. Install VMWare Player, Ubuntu, and ROS Indigo on a laptop. Checkout mit_asctec code from repository and build workspace. Write package to perform failure testing.

vi. Adhere Vicon markers to unmanned aerial vehicle and create object in Vicon.

b. Test Setup

i. Connect the switching DC power supply and the laptop to 120 volts AC using their respective power cords. *This step should take approximately 5 seconds to execute.*

ii. Connect the 10.5' cable with the two AttoPilot sensors that are designed to be in series with the power supply to the switching DC power supply via the black and red banana plugs on one end to the black and red banana sockets on the back of the switching DC power supply. *This step should take approximately 3 seconds to execute.*

iii. Turn on laptop and start VMWare virtual machine. *This step should take approximately 5 minutes to execute.*

iv. Connect Arduino Uno to the laptop via USB A to B cable. *This step should take approximately 5 seconds to execute.*

v. Connect a XBee to laptop via USB mini-B cable. *This step should take approximately 5 seconds toexecute.*

vi. Connect ethernet cable to laptop. *This step should take approximately 5 seconds to execute.*

vii. Turn on machine with Vicon. Start VMWare Workstation and Vicon Tracker. Select unmanned aerial vehicle object in Vicon Tracker. *This step should take approximately 3 minutes to execute.*

viii. Use electrical tape to to create takeoff origin, with indicator of forward facing direction.

ix. Turn on switching DC power supply. Wait for switching DC power supply's self check to finish. *This step should take approximately 20 seconds to execute.*

x. Adjust voltage setting on switching DC power supply to 12_V (only needs to be done once on initial setup) or verify voltage setting on switching DC power supply is set to 12_V. *This step should take approximately 5 seconds to adjust the voltage and approximately 1 second to verify the voltage setting.*

xi. Place a battery inside battery socket of unmanned aerial vehicle. *This step should take approximately 15 seconds to execute.*

xii. Connect unmanned aerial vehicle's Deans T-connector to the Deans T-connector end of the 10.5' cable. *This step should take approximately 7 seconds to execute.*

xiii. Turn on unmanned aerial vehicle. *This step should take approximately 3 seconds to execute.*

xiv. If current setting has not been set, set Futaba controller in manual mode and turn on Futaba controller. Safely secure top of unmanned aerial vehicle and apply maximum thrust to the unmanned aerial vehicle while watching current level on switching DC power supply. Adjust current setting on switching DC power supply to near maximum reading observed. Maximum current observed using the AscTec Hummingbird was 25_A.

xv. If current setting has been set, verify current on switching DC power supply reads a value near zero with unmanned aerial vehicle's motors off. *This step should take approximately 1 second to execute.*

xvi. Verify current setting on switching DC power supply. *This step should take approximately 3 second to execute.*

xvii. Start ROS on laptop by running roscore in a terminal within the virtual machine. *This step should take approximately 5 seconds to execute.*

xviii. In a new terminal within the virtual machine on the laptop, secure shell into virtual machine running on computer with Vicon and launch Vicon interface. *This step should take approximately 20 seconds to execute.*

xix. Set Futaba controller in computer control mode and turn on Futaba controller. *This step should take approximately 1 seconds to execute.*

c. Test Procedure

i. Place drone on takeoff origin, with front facing designated direction, and power cord not parallel with propellers.

ii. In a new terminal within the virtual machine on the laptop, launch the failure testing package. *This step should take approximately 25 seconds to execute.*

iii. Observe these maneuvers: *Each maneuver should take approximately 2 seconds , unless specified.*

1. Motors turn on. *This step should take approximately 4.3 seconds to execute.*
2. Test operation of:
    a. front motor *1 second*
    b. back motor *1 second*
    c. left motor *1 second*
    d. right motor *1 second*
3. Launch (The default launch height can be found in the specifications section and is a parameter that can be modified within the launch file, initial/home x and y position is the 2 D position in which the unmanned aerial vehicle started).
4. Rotate 90º counterclockwise.
5. Rotate 180º
    a. Note: on 180º turns drone rotates the direction of the smallest angle between it's front axis and its next point, whether it be clockwise or counterclockwise.
6. Rotate 90º clockwise.
7. Rotate 90º clockwise and move $\triangle x$ in the positive x direction (amount of movement, $\triangle x$, is recorded in the specifications section and is a parameter that can be modified within the launch file).
8. Rotate 90º clockwise and move $2 * \triangle x$ in the negative x direction.
9. Move $\triangle x$ in the positive x direction.

10. Rotate 90º clockwise and move △y = △x in the negative y direction.
11. Rotate 90º counter clockwise and move 2 * △y in the positive y direction.
12. Move △y in the negative y direction.
13. Rotate 180º and move △z = △x towards the ceiling.
14. Rotate 180º and move △z towards the floor.
15. Move △x in the positive x direction and move △y in the positive y direction.
16. Move △x in the negative x direction and move △z towards the ceiling.
17. Move △y in the negative y direction, and move △z towards the floor.
18. Rotate 90º counter clockwise, move △x in the negative x direction, and move △y in the negative y direction.
19. Rotate 90º clockwise, move △x in the positive x direction, move △y in the positive y direction, and move △z towards the ceiling.
20. Land.
21. Motors turn off. *This step should take approximately 10 seconds to execute.*

iv. Escape running launch file by pressing ctrl and c simultaneously. *This step should take approximately 17 seconds to execute.*

v. Save bag file to cse server. Change target bag file name.

vi. Repeat steps i through v.

d. System Shutdown

i. Turn off Futaba controller. *This step should take approximately 1 seconds to execute.*

ii. Turn off unmanned aerial vehicle. *This step should take approximately 1 seconds to execute.*

iii. Disconnect Dean's T-connector from unmanned aerial vehicle. *This step should take approximately 3 seconds to execute.*

iv. Disconnect Arduino Uno from laptop. *This step should take approximately 7 seconds to execute.*

v. Turn off switching DC power supply. *This step should take approximately 8 seconds to execute.*

vi. Disconnect XBee from laptop. *This step should take approximately 6 seconds to execute.*

vii. Exit Vicon.launch by pressing ctrl and c simultaneously in the respective terminal. *This step should take approximately 3 seconds to execute.*

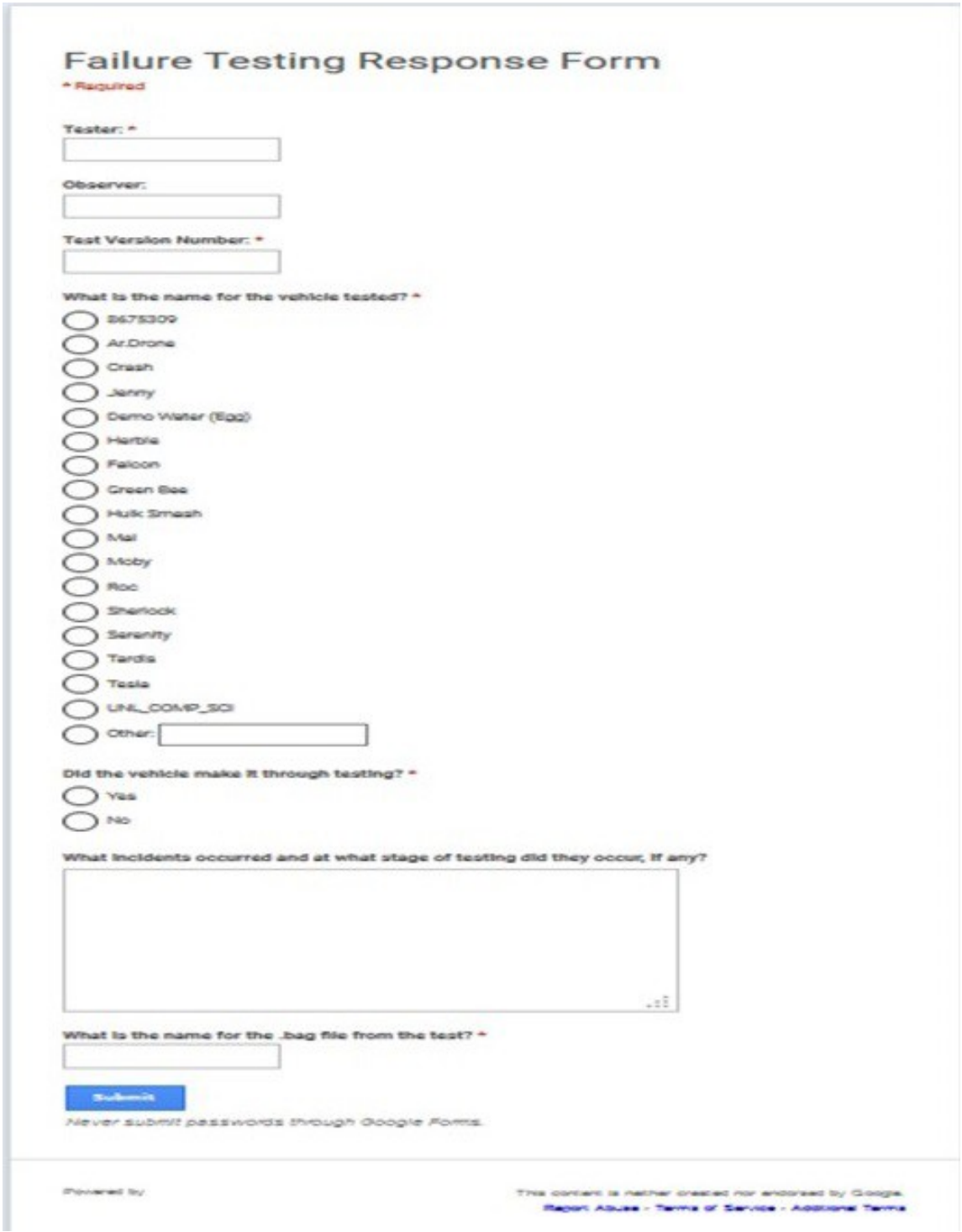
viii. Exit secure shell connection to Vicon machine by typing exit and pressing enter in the same terminal. *This step should take approximately 2 seconds to execute.*

ix. Disconnect ethernet cable from laptop. *This step should take approximately 1 seconds toexecute.*
x. Coil disconnected cables and safely store out of the way along with the switching DC power supply itself. *This step should take approximately 20 seconds to execute.*
xi. Exit roscore by pressing ctrl and c simultaneously in the respective terminal. *This step should take approximately 1 seconds to execute.*
xii. Store Futaba controller and unmanned aerial vehicle in a safe out of the way location. *This step should take approximately 5 seconds to execute.*

4. **Unmanned Aerial Vehicle Specific Hardware:**
   a. AscTec Hummingbird 0033878000 (Crash)
      i. 2 8”x4.5” #10521 propeller pair, 6” Flexible propellers
      ii. 2 8”x4.5” #10521 propeller pair, 6” Flexible propellers
   b. AscTec Hummingbird 0033913900 (Jenny)
      i. 2 8”x4.5” #10969 propeller pair, 8” AscTec L
      ii. 2 8”x4.5” #10969 propeller pair, 8” AscTec”R
   c. AscTec Hummingbird 0033913800 (Herbie)
      i. 2 Parrot AR Drone 2.0 L
      ii. 2 Parrot AR Drone 2.0 R
   d. AscTec Hummingbird 0033878100 (Hulk Smash)
      i. 2 8”x4.5” #10969 propeller pair, 8” AscTec L
      ii. 2 8”x4.5” #10969 propeller pair, 8” AscTec”R
   e. AscTec Hummingbird 0033811700 (UNL_COMP_SCI)
      i. 2 8”x4.5” #10521 propeller pair, 6” Flexible propellers
      ii. 2 8”x4.5” #10521 propeller pair, 6” Flexible propellers
5. **Specifications:**
   a. Arduino UNO Resolution: 0.0049_V
   b. AttoPilot Voltage Current Sense Breakout 45A:
      i. Voltage: $1/49.44_V$
      ii. Current: $1/14.9_A$
   c. Meter type and accuracy of the B&K Precision 1900 is 3-Digit LED Display ± 0.2% + 3 counts.
   d. Default initial launch height is 0.5 meters.
   e. Default position change in x, y and z ($\triangle x$, $\triangle y$, $\triangle z$) is a single value, 0.5 meters.
6. **Resources:**
   a. Arduino Uno
   b. AscTec Hummingbird
   c. AttoPilot Voltage and Current Sense Breakout 45A
   d. B&K Precision
   e. Digi International XBee
   f. Futaba 7-channel 2.4GHz controller
   g. Massachusetts Institute of Technology
   h. Nimbus Lab

i. **Robotic Operating System**
j. Ubuntu
k. Vicon Tracker
l. VMWare Player

B.2 Failure Testing Response Form



Failure Testing Response Form

* Required

Tester: *

Observer:

Test Version Number: *

What is the name for the vehicle tested? *

- 8675309
- Ar.Drone
- Crash
- Jenny
- Demo Water (Egg)
- Herbie
- Falcon
- Green Bee
- Hulk Smash
- Mal
- Moby
- Roc
- Sherlock
- Serenity
- Tardis
- Tesla
- UNL_COMP_SCI
- Other:

Did the vehicle make it through testing? *

- Yes
- No

What incidents occurred and at what stage of testing did they occur, if any?

What is the name for the .bag file from the test? *

Submit

Never submit passwords through Google Forms.

Powered by

This content is neither created nor endorsed by Google.
Report Abuse - Terms of Service - Additional Terms

Appendix C

Tables

C.1 Unmanned Aerial Vehicle Statistics

C.1.1 Failure Statistics

Vehicle	Propeller	Firmware	Failure Rate	Failed Tests	Total Tests	Manual Control	Interrupted Serial Communication	Extra Task	Missing Task	Extra Target State	Missing Target State	Quad Control Input Delay	Target State Delay	Target Position Delay	Subject Pose Delay	Vicon Delay	Vicon Lost Object	Motors Stayed On	Vehicle Flipped	Subject Status Delay
1	AscTec Safety	AutoPilot LowLevel #20636 V3.12, AutoPilot HighLevel V3.11	15.294	78	510	1	5	24	13	29	7	58	61	57	57	15	0	12	0	40
2	AscTec Safety	AutoPilot LowLevel #20637 V3.12, AutoPilot HighLevel V3.0	4	4	100	1	0	1	1	2	0	0	2	0	1	2	20	4	1	1
3	AscTec Safety	AutoPilot LowLevel #20502 V3.12, AutoPilot HighLevel V3.11	33.333	1	3	1	0	0	0	0	0	0	0	0	0	1	1	0	1	0
4	AscTec Normal	AutoPilot LowLevel #20633 V2.14, AutoPilot HighLevel V3.0	30	33	110	3	1	5	8	10	5	29	22	21	27	5	6	1	0	12
5	ARDrone	AutoPilot LowLevel #20632 V3.12, AutoPilot HighLevel V3.0	14	14	100	0	0	6	6	6	0	5	7	9	6	5	17	0	0	3
6	ARDrone	ARDrone	10	10	100	0	0	9	0	2	0	0	1	1	43	9	1	0	0	100

C.1.2 Position Statistics

Vehicle	Total Spatial Error	Total Mean Error East	Total Mean Error North	Total Mean Error Up	Total Mean Error Yaw	Mean Total Spatial Error Per Task	Mean Error Per Task East	Mean Error Per Task North	Mean Error Per Task Up	Mean Error Per Task Yaw	Variance Error Per Task East	Variance Error Per Task North	Variance Error Per Task Up	Variance Error Per Task Yaw
	(centimeters - cm)	(cm)	(cm)	(cm)	(radians)	(millimeters - mm)	(mm)	(mm)	(mm)	(radians)				
1	15.321	8.877	8.704	8.556	0.751	2.342	1.357	1.331	1.308	0.011	1.619E-04	1.746E-04	1.243E-03	9.535E-02
2	22.716	13.257	13.811	12.176	0.877	3.463	2.021	2.106	1.856	0.013	5.589E-05	7.685E-05	7.411E-05	7.273E-02
3	24.461	14.684	13.890	13.733	0.660	3.724	2.236	2.115	2.091	0.010	3.358E-05	1.458E-05	3.205E-04	1.788E-03
4	18.729	8.069	10.110	13.400	1.037	2.858	1.231	1.543	2.045	0.016	1.150E-04	2.379E-04	4.915E-04	3.698E-02
5	19.593	11.344	11.948	10.156	0.963	3.006	1.740	1.833	1.558	0.015	3.952E-04	7.500E-04	1.398E-03	0.237
6	34.444	15.392	20.185	22.848	0.623	5.267	2.354	3.087	3.494	0.010	1.203E-03	1.984E-03	3.958E-04	2.573E-02

C.1.3 Power Statistics

Vehicle	Mean Power Consumption	Move Mean Power Consumption	Hover Mean Power Consumption	Mean Power Range Minimum	Mean Power Range Maximum	Mean Power Variance	Move Power Variance	Hover Power Variance	Mean Maximum Power Consumption	Maximun Power Range Minimum	Maximun Power Range Maximum	Maximum Power Variance
1	83.327	83.096	83.543	77.015	91.188	2.799	4.208	4.152	196.069	104.472	255.872	918.637
2	82.658	82.867	82.397	79.840	88.349	2.868	3.609	3.388	211.508	154.238	259.329	347.898
3	85.697	86.889	84.408	84.826	86.601	0.215	0.241	0.198	189.580	182.424	195.152	43.328
4	81.026	81.176	80.818	74.601	83.525	2.883	3.979	4.119	283.700	151.373	357.403	1779.082
5	103.179	102.242	104.070	93.113	108.335	5.113	4.224	6.472	223.693	132.151	287.695	478.713
6	99.252	99.066	99.414	95.790	102.702	0.478	0.724	0.638	133.664	117.287	333.990	376.284

C.1.4 Time Statistics

Vehicle	Mean Test Execution Time	Minimum Time	Maximum Time	Time Variance
	(seconds - s)	(s)	(s)	
1	65.40931163	59.99978342	70.01559134	1.318436401
2	65.59476577	63.34760297	67.66205411	0.603104957
3	65.67897066	65.35812973	65.99981159	0.205877806
4	65.53223731	64.00359256	68.0169998	0.837317237
5	65.1793944	63.99887465	67.50047947	0.520099925
6	65.3962184	62.84271722	70.24518828	1.36675083