Flight Testing

W. Willie Wells

August 2015

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 interference from individuals that feel threatened, occupying illegal air space, and vehicle malfunction. Vehicle malfunction can be costly. Testing was conducted on multicopters to determine characteristics that cause failure of normal unmanned aerial vehicle operation. Testing to date has only involved quadcopters. Monitoring of parameters was conducted such that the cause of any failure could be recorded and could be analyzed.

1.2 Overview of Procedure

Figure 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 1 shows the physical test configuration after the completion of these steps.

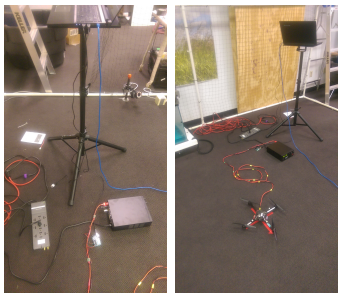


Figure 2: Physical test setup.

Required software, Vicon Tracker and 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 shows the physical test configuration after the completion of these steps.

Figure 3: Herbie–Ascending Technologies Hummingbird with ARDrone propellers progressing through the maneuvers of the flight test.

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 3 shows four snapshots of the test procedure in progress as the unmanned aerial vehicle performs the predefined maneuvers.

1.3 Components and Configuration

1.3.1 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. 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.

1.3.2 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.

1.3.3 Switching Power Supply

A switching DC power supply 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.

1.3.4 Controlling Station

A Futaba controller is used for all Ascending Technologies vehicles to set the vehicle's controlling station. 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.

1.3.5 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. 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).

1.3.6 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. 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.

1.3.7 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 vicon position to subject pose is started. A node that connects to the XBee device 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.

1.4 Testing Methods

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 $\pi -- > \pi/2 -- > 3*\pi/2 -- > 0(2*\pi)$; 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 rotations 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, negative x, positive x, 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. 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 found here bag2csv. The .csv files are then imported into MATLAB for analysis.

1.5 Analysis of Procedure

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 native Proportion Integral Derivative settings may further minimize position errors. A counter keeps track of the current in flight maneuver. Each in flight maneuver has a move to position then hover component. After seeing 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.

2 Experimental Results

	(type)	(type)	(count)	(count)	(%)
Vehicle Name	**Vehicle Type**	**Propeller**	**Failed Tests**	**Total Tests**	**Failure Rate**
CRASH	AscTec Hummingbird	AscTec Safety Propellers	78	510	15.294
HERBIE	AscTec Hummingbird	ARDrone Propellers	14	100	14.000
JENNY	AscTec Hummingbird	AscTec Normal Propellers	33	110	30.000
UNL_COMP_SCI	AscTec Hummingbird	AscTec Safety Propellers	4	100	4.000
HULK_SMASH	AscTec Hummingbird	AscTec Safety Propellers	1	3	33.333
MORPHEUS	ARDrone	ARDrone Propellers	10	100	10.000

Figure 4: Experimental results and the respective failure rate of each vehicle.

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 have been tested so far. 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. This vehicle was repaired and additional tests will be performed to increase the total to 100 tests performed. An Arducopter and a DJI Phantom 3 are in the pipeline to be tested.

3 Types of Failures and Their Causes

	CRASH	HERBIE	JENNY	UNL COMP SCI	HULK SMASH	MORPHEUS
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 Lost (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 1: Failure types per vehicle with amount of occurrences per failure and the associated percentage of failure type per vehicle.

3.1 Symptoms

Failure symptoms observed during the tests were:

1. the unmanned aerial vehicle 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.

3.2 External Polling Device

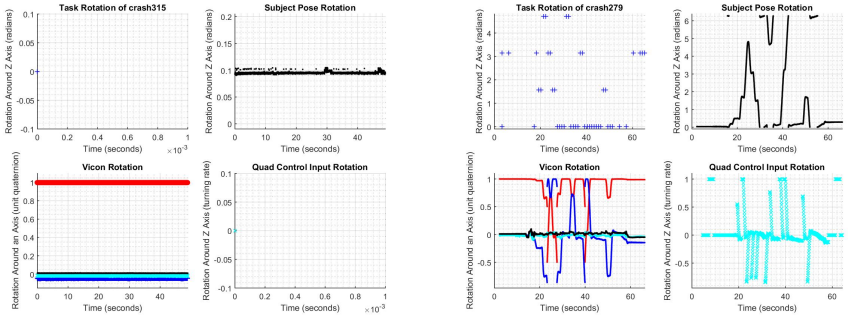


Figure 5: Rotation plots for task waypose rotation, subject pose rotation, Vicon rotation, and quad control input rotation. The top two plots are measured in radians while the bottom left plot is measured in unit quaternion and the points in the bottom right plot are unit normalized directional turning rates.

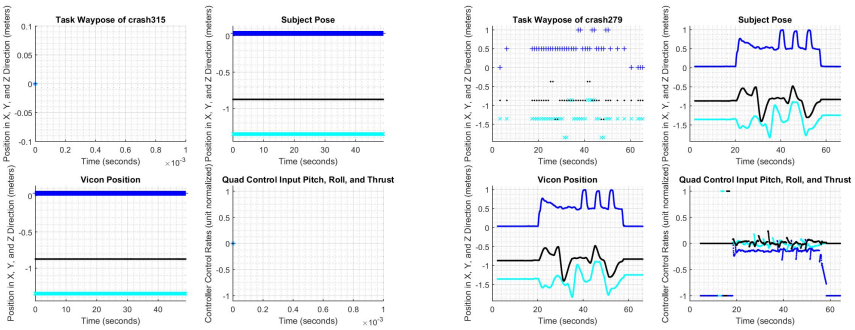


Figure 6: Position plots for task waypose position, subject pose position, Vicon position, and quad control input pitch, roll, and thrust. The top two plots are measured in meters while the bottom left plot is measured in meters as well and the points in the bottom right plot are unit normalized controller control rates.

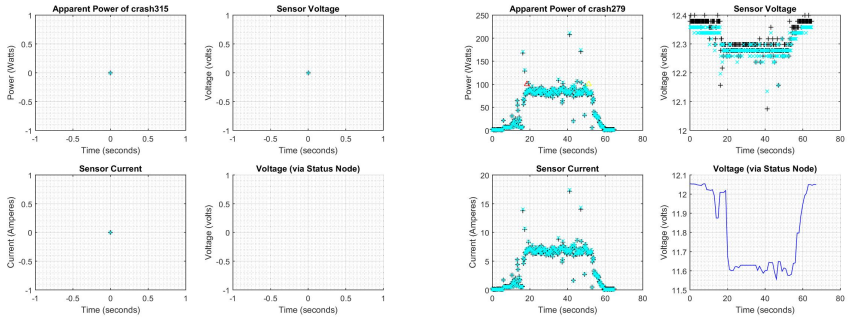


Figure 7: Plots displaying calculated apparent power measured in Watts, sensor voltage measured in volts, and sensor current measured in Amperes from the data recorded from both AttoPilot sensors as well as battery voltage recorded via the subject status node.

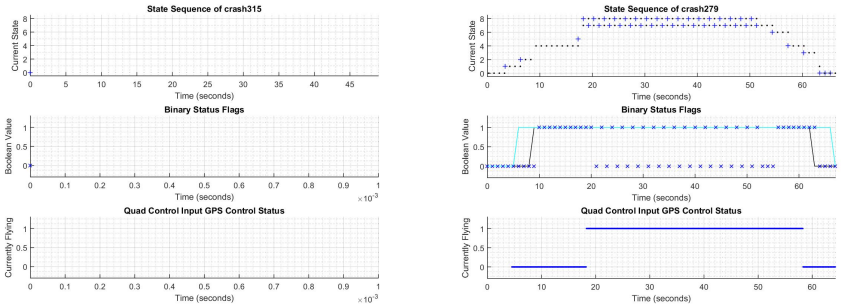


Figure 8: The top plot displays commanded state (blue +) and current state (black .). The middle plot displays binary motor status in black as a continuous line (on = 1, off = 0), binary serial mode status in cyan as a continuous line (enabled = 1, disabled = 0), and binary waypose status in blue as discrete points. The bottom plot displays binary GPS control (flight) status as indicated by quad control input.

3.2.1 Event

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 hummingbird_crash315. There were 4/339 (1.180%) tests affected while this failure had a non-zero probability of occurring.

3.2.2 Cause

A device in the laboratory was polling for an administration device and blocked communication from the controlling computer to the XBee 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 device. Communication setup was verified as connected correctly and additionally it was verified that no other changes to the test environment were introduced between test hummingbird_crash314 and test hummingbird_crash315. 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 ran, thus this was assumed to be the cause.

3.2.3 Solution

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.

3.3 Pitch and Roll Tests

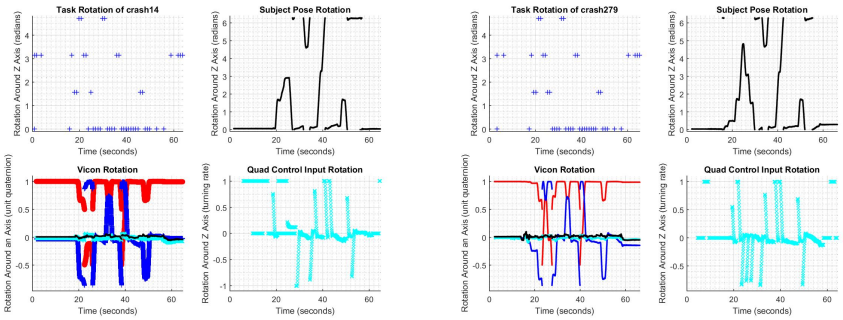


Figure 9: Rotation plots for task waypose rotation, subject pose rotation, Vicon rotation, and quad control input rotation. The top two plots are measured in radians while the bottom left plot is measured in unit quaternion and the points in the bottom right plot are unit normalized directional turning rates.

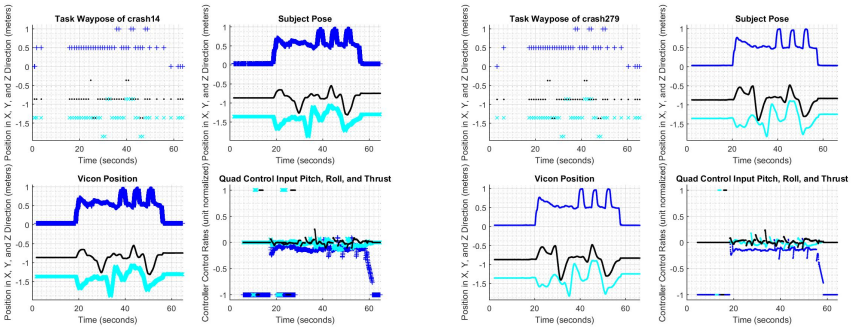


Figure 10: Position plots for task waypose position, subject pose position, Vicon position, and quad control input pitch, roll, and thrust. The top two plots are measured in meters while the bottom left plot is measured in meters as well and the points in the bottom right plot are unit normalized controller control rates.

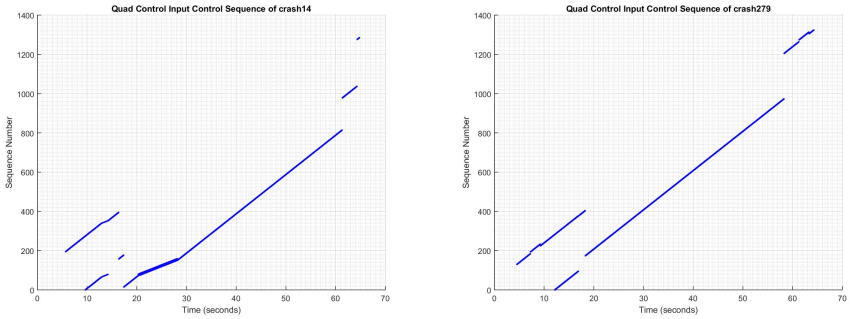


Figure 11: Plots displaying control sequence numbers. Jumps in the graph represent different quad control input sources. Overlapping lines represent two or more sources controlling quad control input at nearly the same time.

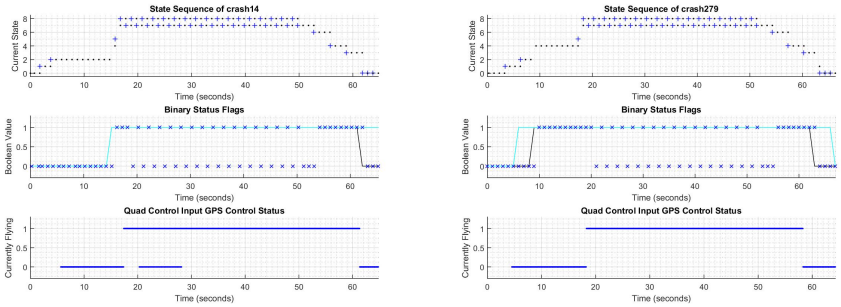


Figure 12: The top plot displays commanded state (blue +) and current state (black .). The middle plot displays binary motor status in black as a continuous line (on = 1, off = 0), binary serial mode status in cyan as a continuous line (enabled = 1, disabled = 0), and binary waypose status in blue as discrete points. The bottom plot displays binary GPS control (flight) status as indicated by quad control input.

3.3.1 Event

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 hummingbird_crash14 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 nonzero probability of occurring.

3.3.2 Cause

The lab has 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, 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 is approximately 2.200 hz. During the two tests that uav_test worked the receipt of a robot_imu message occurred shortly after the receipt 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. At this point in time, the testing code did not account for this 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.

3.3.3 Solution

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.

3.4 Subject Status Tasked Waypose

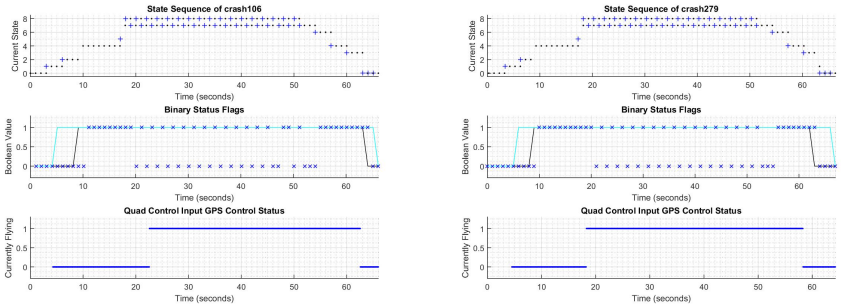


Figure 13: The top plot displays commanded state (blue +) and current state (black .). The middle plot displays binary motor status in black as a continuous line (on = 1, off = 0), binary serial mode status in cyan as a continuous line (enabled = 1, disabled = 0), and binary waypose status in blue as discrete points. The bottom plot displays binary GPS control (flight) status as indicated by quad control input.

3.4.1 Event

No observable symptom. Abnormal waypose status flag propagation in the bagfile. This anomaly is visible in the corresponding data for test hummingbird_crash106.

3.4.2 Cause

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 packets and delays in tasked waypose packets which affects the value of the waypose status field. The waypose status field is set if a task waypose message is received. If a task 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 last tasked waypose time stamp received from the current subject status time stamp. A new subject status time stamp is set if both a robot status and a robot gps packet is received. Thus, delays in subject status packets as well as delays in tasked waypose packets result in this value being incorrectly set. The waypose status field is not used in the robot control flow thus incorrect values of waypose status do not affect overall robot operation. 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.

3.4.3 Solution

Verify all control rates 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.

3.5 Power Cord Interference

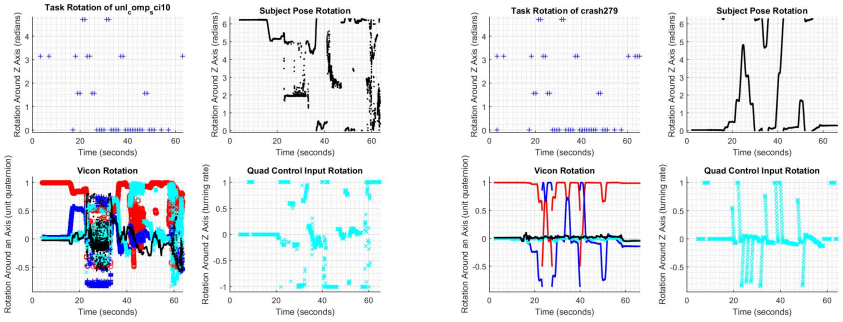


Figure 14: Rotation plots for task waypose rotation, subject pose rotation, Vicon rotation, and quad control input rotation. The top two plots are measured in radians while the bottom left plot is measured in unit quaternion and the points in the bottom right plot are unit normalized directional turning rates.

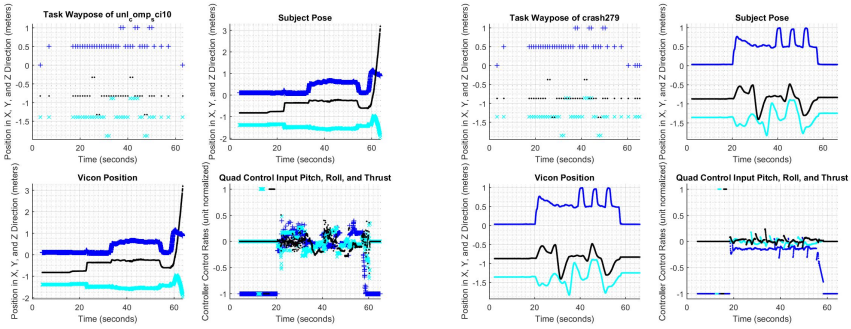


Figure 15: Position plots for task waypose position, subject pose position, Vicon position, and quad control input pitch, roll, and thrust. The top two plots are measured in meters while the bottom left plot is measured in meters as well and the points in the bottom right plot are unit normalized controller control rates.

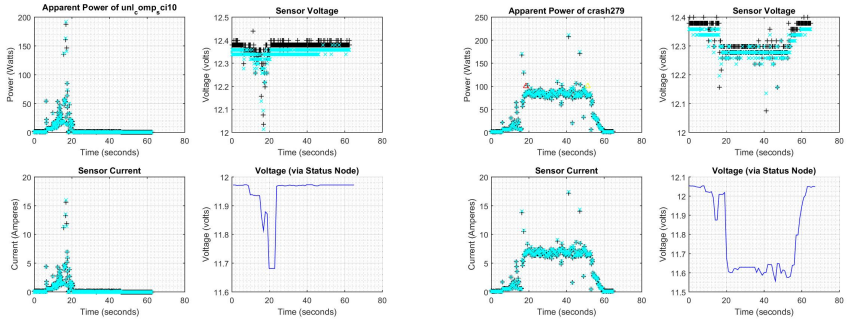


Figure 16: Plots displaying calculated apparent power measured in Watts, sensor voltage measured in volts, and sensor current measured in Amperes from the data recorded from both AttoPilot sensors as well as battery voltage recorded via the subject status node.

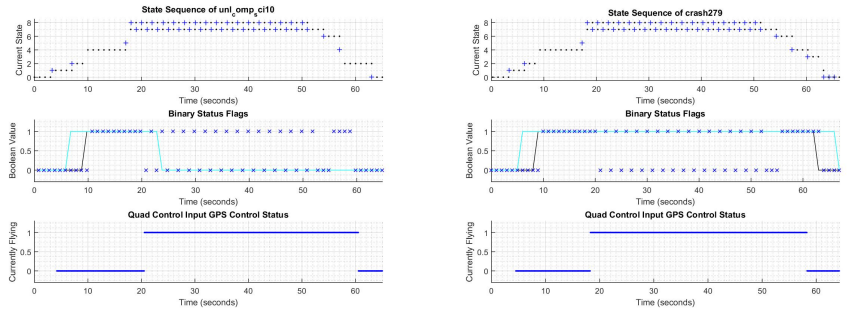


Figure 17: The top plot displays commanded state (blue +) and current state (black .). The middle plot displays binary motor status in black as a continuous line (on = 1, off = 0), binary serial mode status in cyan as a continuous line (enabled = 1, disabled = 0), and binary waypose status in blue as discrete points. The bottom plot displays binary GPS control (flight) status as indicated by quad control input.

3.5.1 Event

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 hummingbird_unl_comp_sci10 and manual control was initiated.

3.5.2 Cause

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.

3.5.3 Solution

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.

3.6 Vicon Lost Object

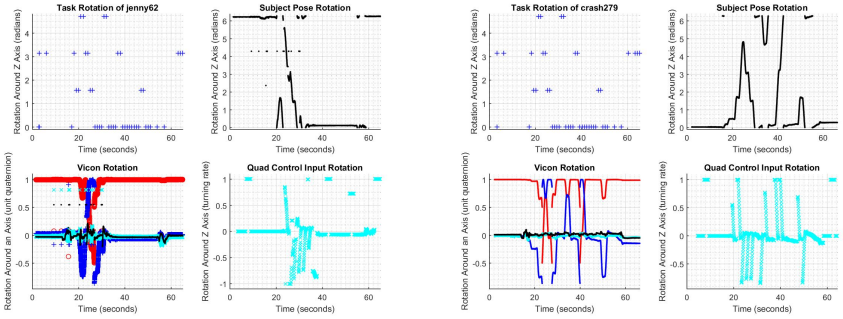


Figure 18: Rotation plots for task waypose rotation, subject pose rotation, Vicon rotation, and quad control input rotation. The top two plots are measured in radians while the bottom left plot is measured in unit quaternion and the points in the bottom right plot are unit normalized directional turning rates.

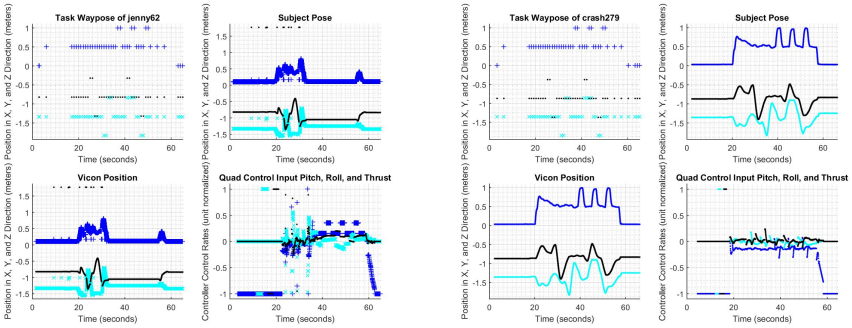


Figure 19: Position plots for task waypose position, subject pose position, Vicon position, and quad control input pitch, roll, and thrust. The top two plots are measured in meters while the bottom left plot is measured in meters as well and the points in the bottom right plot are unit normalized controller control rates.

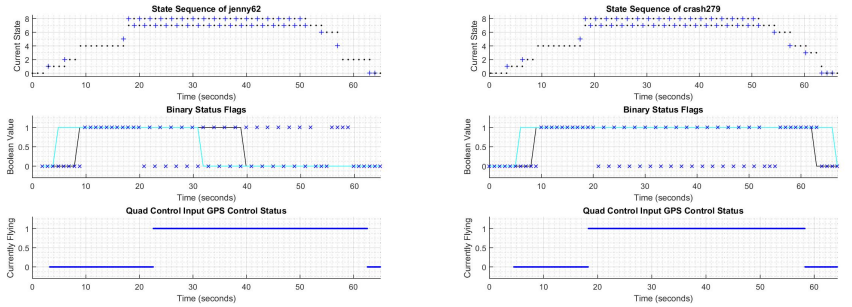


Figure 20: The top plot displays commanded state (blue +) and current state (black .). The middle plot displays binary motor status in black as a continuous line (on = 1, off = 0), binary serial mode status in cyan as a continuous line (enabled = 1, disabled = 0), and binary waypose status in blue as discrete points. The bottom plot displays binary GPS control (flight) status as indicated by quad control input.

3.6.1 Event

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.

3.6.2 Cause

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 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, (hummingbird_herbie12,13), the loss of a Vicon object caused a repeated idle tasked waypose to be sent prior to launch. The most severe case was test hummingbird_jenny62 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%).

3.6.3 Solution

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.

3.7 Vicon Communication Delay

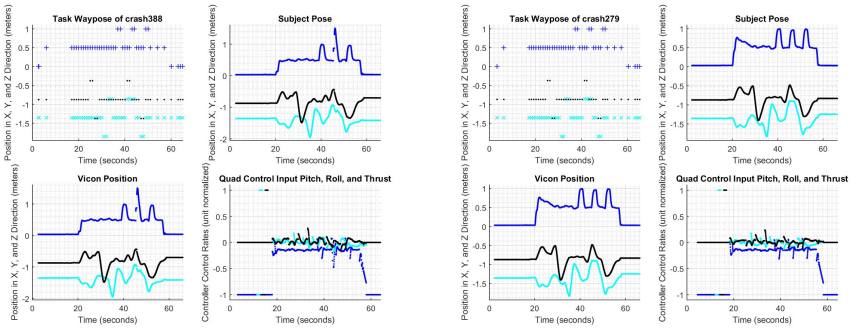


Figure 21: Position plots for task waypose position, subject pose position, Vicon position, and quad control input pitch, roll, and thrust. The top two plots are measured in meters while the bottom left plot is measured in meters as well and the points in the bottom right plot are unit normalized controller control rates.

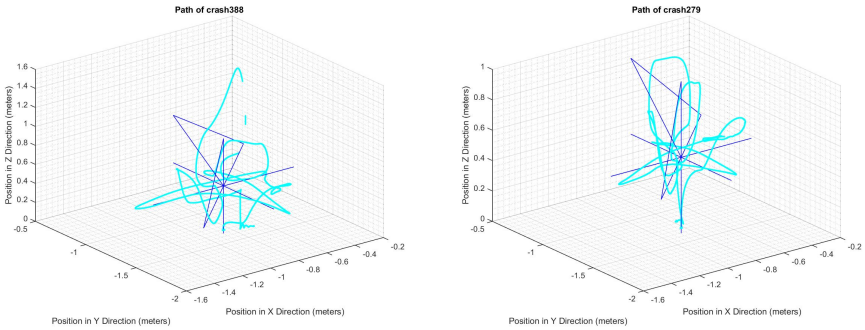


Figure 22: A three dimensional plot comparing x, y, and z position from subject pose with the tasked x, y, and z position.

3.7.1 Event

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 hummingbird_unl_comp_sci59. 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 hummingbird_crash388.

3.7.2 Cause

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%).

3.7.3 Solution

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.

3.8 Control Packet Delay

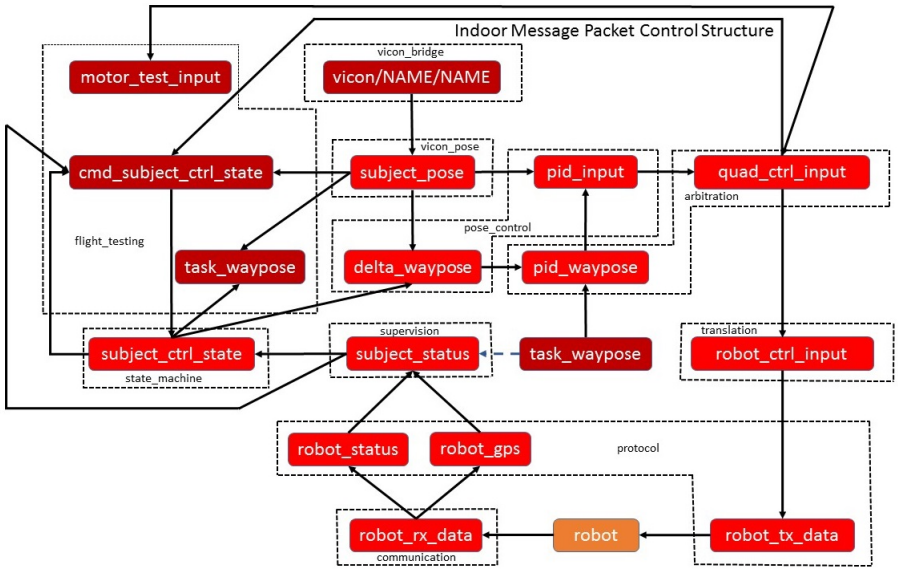


Figure 23: Control message flow loop, solid lines from topic A to topic B indicate topic A is a publishing dependency for topic B while dashed lines from topic A to topic B indicate that topic A is subscribed to but not required for publishing of topic B.

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, 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 multicopter. 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 multicopter. 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 4, specific nodes are not identified in the figure.

3.8.1 Event

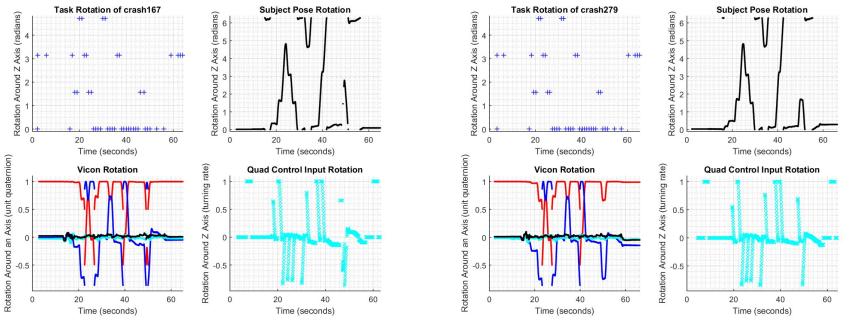


Figure 24: Rotation plots for task waypose rotation, subject pose rotation, Vicon rotation, and quad control input rotation. The top two plots are measured in radians while the bottom left plot is measured in unit quaternion and the points in the bottom right plot are unit normalized directional turning rates.

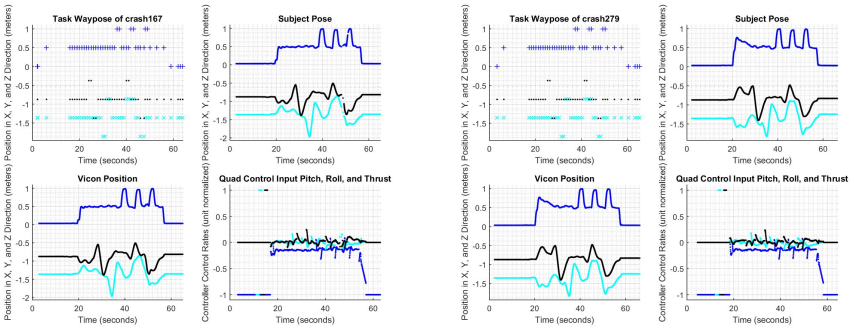


Figure 25: Position plots for task waypose position, subject pose position, Vicon position, and quad control input pitch, roll, and thrust. The top two plots are measured in meters while the bottom left plot is measured in meters as well and the points in the bottom right plot are unit normalized controller control rates.

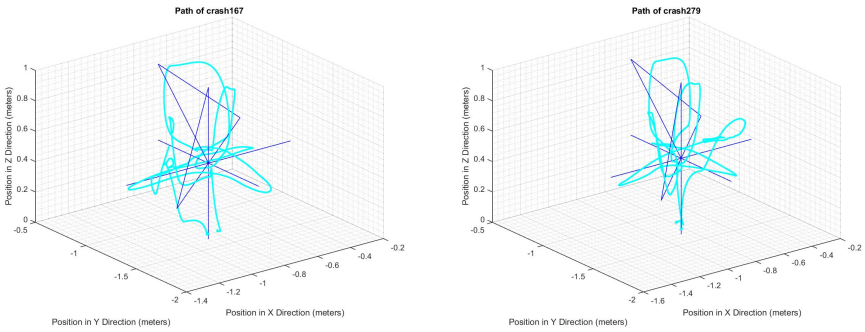


Figure 26: A three dimensional plot comparing x, y, and z position from subject pose with the tasked x, y, and z position.

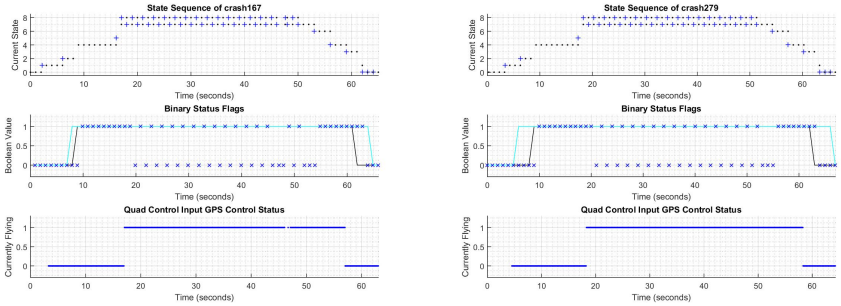


Figure 27: The top plot displays commanded state (blue +) and current state (black .). The middle plot displays binary motor status in black as a continuous line (on = 1, off = 0), binary serial mode status in cyan as a continuous line (enabled = 1, disabled = 0), and binary waypose status in blue as discrete points. The bottom plot displays binary GPS control (flight) status as indicated by quad control input.

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, or performance of additional maneuvers than those tasked is witnessed.

3.8.2 Cause

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 previously noted 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 hummingbird_crash111 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 seconds at approximately 51 seconds into test hummingbird_herbie1 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 hummingbird_jenny77, which physically exhibited symptom 2, additional time than normal to perform a maneuver.

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 hummingbird_crash264, 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 hummingbird_crash167. 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 hummingbird_jenny61. 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 hummingbird_crash296 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.

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 hummingbird_crash399 and hummingbird_jenny19 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 hummingbird_crash107. 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%).

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.

3.8.3 Solution

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 $\pm$ 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.

3.9 Serial Communication Lost

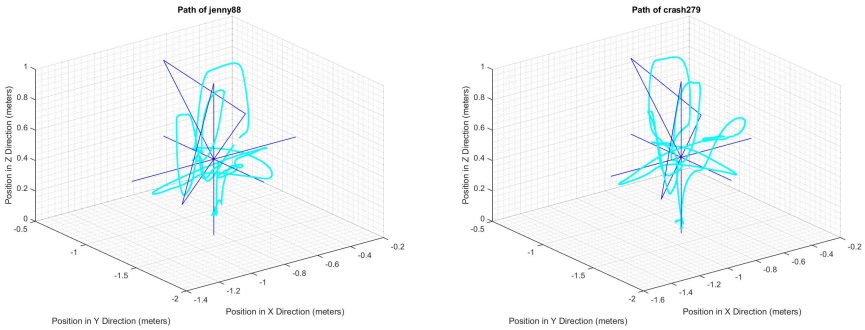


Figure 28: A three dimensional plot comparing x, y, and z position from subject pose with the tasked x, y, and z position.

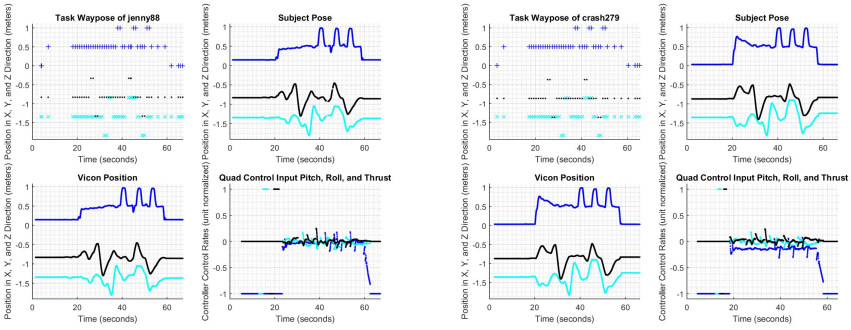


Figure 29: Position plots for task waypose position, subject pose position, Vicon position, and quad control input pitch, roll, and thrust. The top two plots are measured in meters while the bottom left plot is measured in meters as well and the points in the bottom right plot are unit normalized controller control rates.

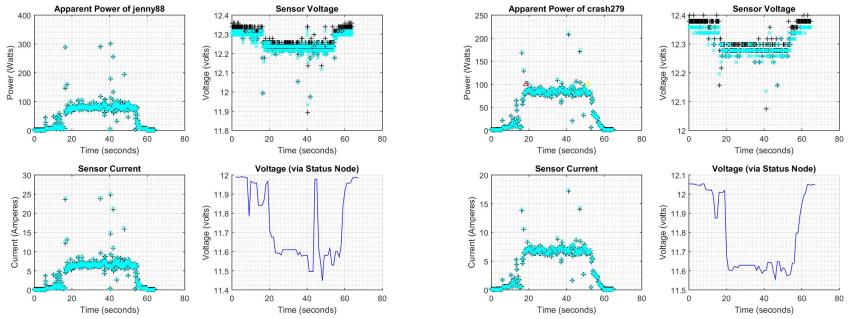


Figure 30: Plots displaying calculated apparent power measured in Watts, sensor voltage measured in volts, and sensor current measured in Amperes from the data recorded from both AttoPilot sensors as well as battery voltage recorded via the subject status node.

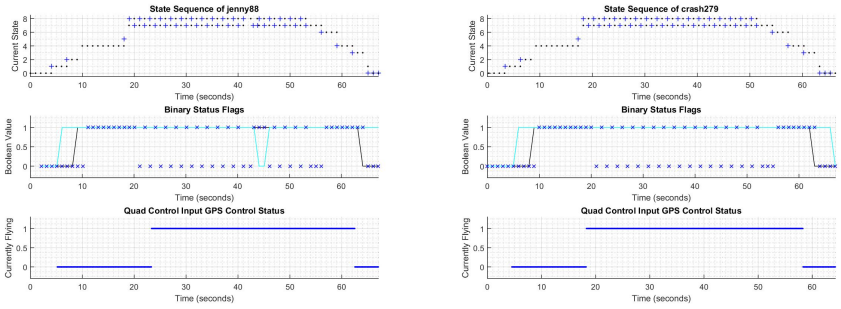


Figure 31: The top plot displays commanded state (blue +) and current state (black .). The middle plot displays binary motor status in black as a continuous line (on = 1, off = 0), binary serial mode status in cyan as a continuous line (enabled = 1, disabled = 0), and binary waypose status in blue as discrete points. The bottom plot displays binary GPS control (flight) status as indicated by quad control input.

3.9.1 Event

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 hummingbird_crash155 and test hummingbird_jenny88 suffered a loss of serial communication while in flight while test hummingbird_crash190 suffered a loss of serial communication while grounded.

3.9.2 Cause

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.

3.9.3 Solution

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. 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. The first proposed solution does not require adding extra weight to the vehicle but would be difficult to implement.

3.10 Motor Failure

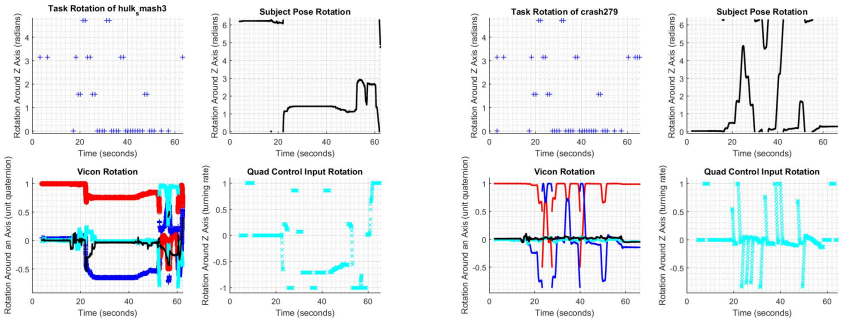


Figure 32: Rotation plots for task waypose rotation, subject pose rotation, Vicon rotation, and quad control input rotation. The top two plots are measured in radians while the bottom left plot is measured in unit quaternion and the points in the bottom right plot are unit normalized directional turning rates.

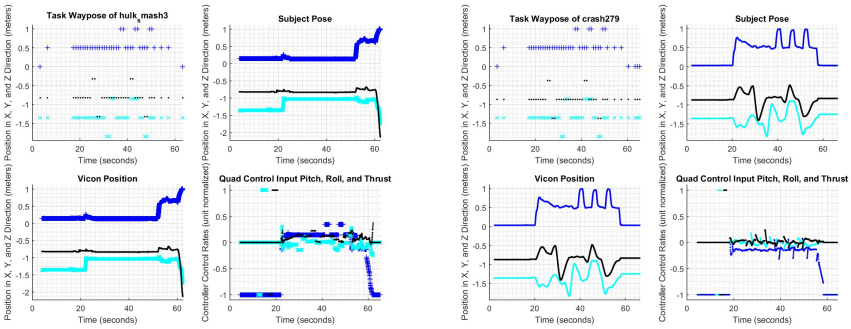


Figure 33: Position plots for task waypose position, subject pose position, Vicon position, and quad control input pitch, roll, and thrust. The top two plots are measured in meters while the bottom left plot is measured in meters as well and the points in the bottom right plot are unit normalized controller control rates.

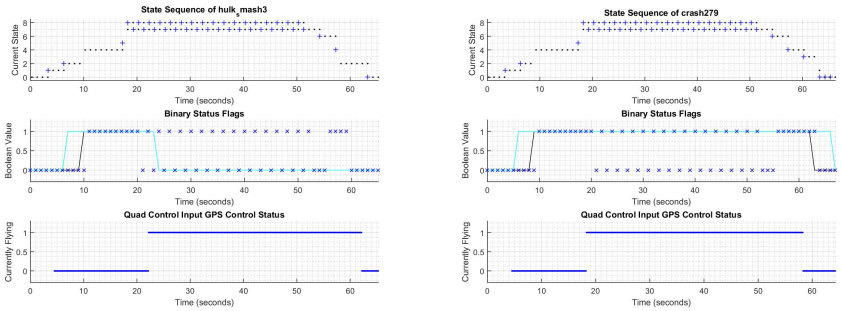


Figure 34: The top plot displays commanded state (blue +) and current state (black .). The middle plot displays binary motor status in black as a continuous line (on = 1, off = 0), binary serial mode status in cyan as a continuous line (enabled = 1, disabled = 0), and binary waypose status in blue as discrete points. The bottom plot displays binary GPS control (flight) status as indicated by quad control input.

3.10.1 Event

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

3.10.2 Cause

This 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 hummingbird_hulk_smash3.

3.10.3 Solution

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.

3.11 Rotation Failure

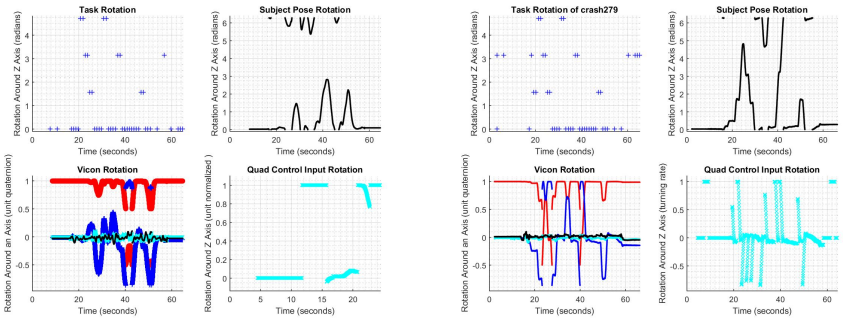


Figure 35: Rotation plots for task waypose rotation, subject pose rotation, Vicon rotation, and quad control input rotation. The top two plots are measured in radians while the bottom left plot is measured in unit quaternion and the points in the bottom right plot are unit normalized directional turning rates.

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 5 for the ARDrone indicates there were additional failures on those specific tests.

4 Power Analysis

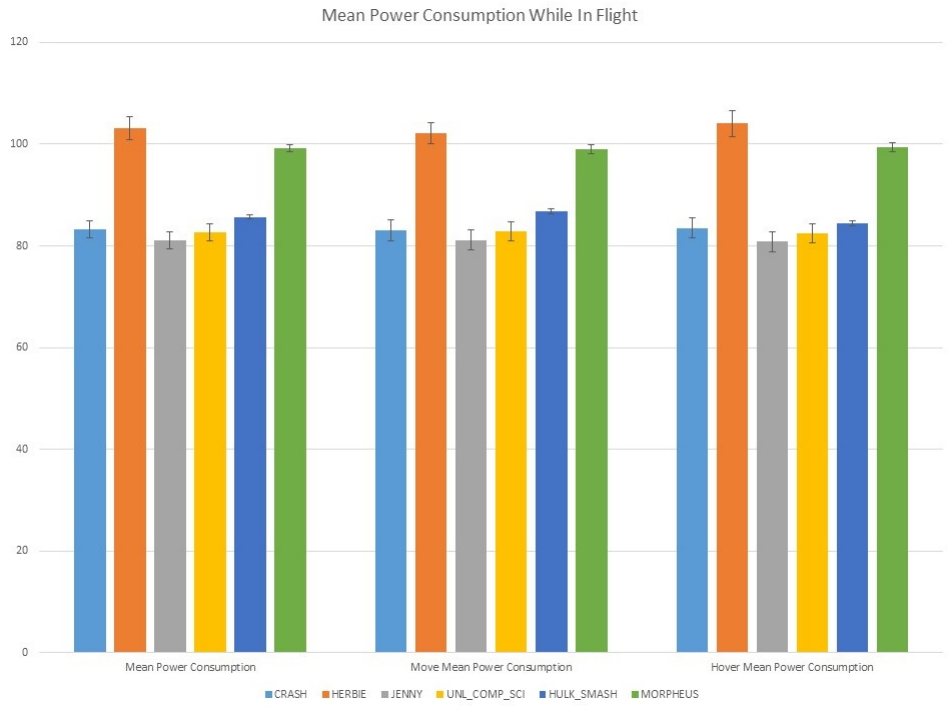


Figure 36: Mean power consumed for each vehicle tested. Error bars measure one standard deviation in both directions from the mean.

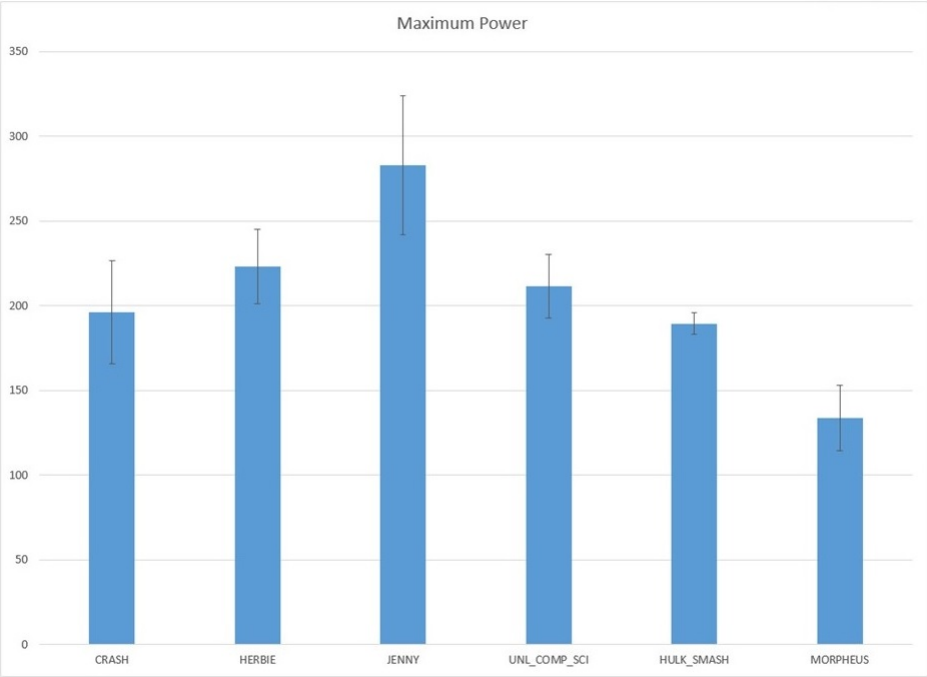


Figure 37: Maximum power consumed for each vehicle tested. Error bars measure one standard deviation in both directions from the mean.

4.1 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 previous 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.

4.2 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 ARDrone 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 7. 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 6.

5 Position Error Analysis

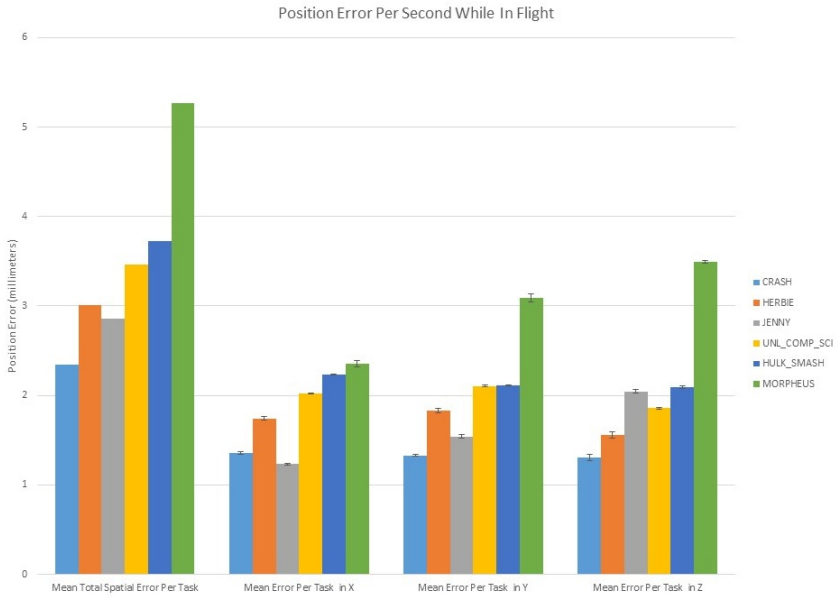


Figure 38: Position error statistics for each unmanned aerial vehicle tested. Error bars measure one standard deviation in both directions from the mean.

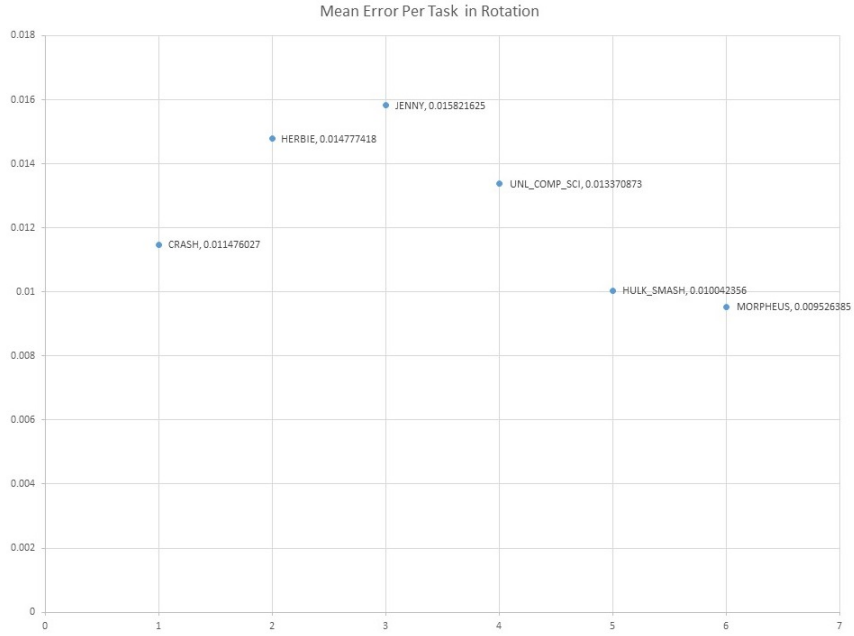


Figure 39: Rotation error statistics for each unmanned aerial vehicle tested.

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 multicopter 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 dif-

ference 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. 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 8. Per task rotation values are not multiplied by 10, both recorded values are in radians. These values are recorded in figure 9.

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.

6 Execution Time Analysis

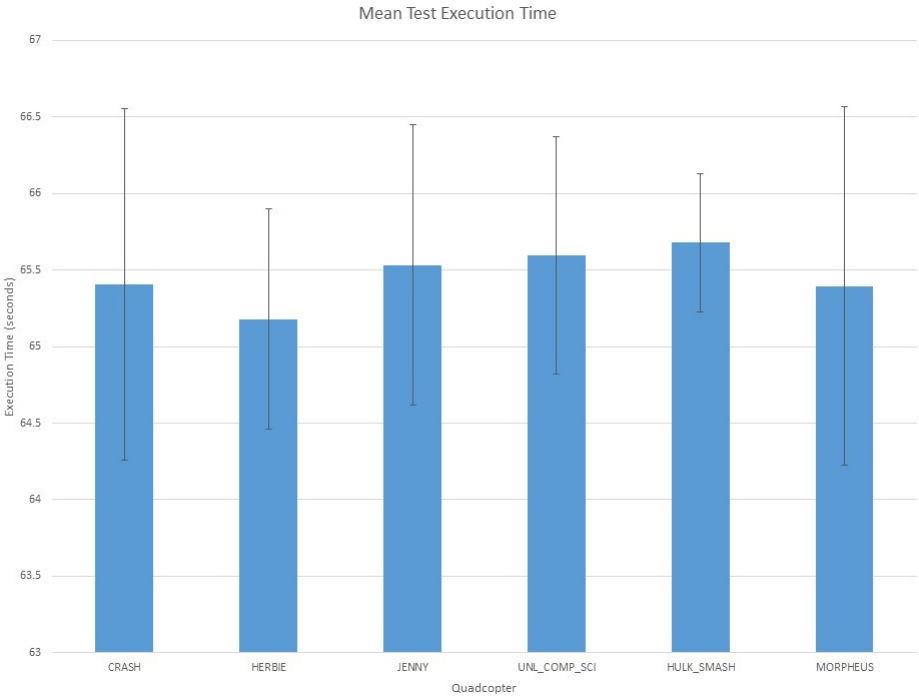


Figure 40: Test execution times for each unmanned aerial vehicle tested.

6.1 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 hummingbird_crash1.bag. 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' with the hope that this will cause the time travel to cease. This change was implemented. 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.

6.2 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. 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. Beginning time stamps for each topic per test except quad control input are compared and 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. 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 hummingbird_crash36 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 10.

7 Analysis of State and Status Flags

7.1 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.

7.2 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.

7.3 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, 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.

8 Vehicle Type Analysis

8.1 Hummingbird: Safety Propellers

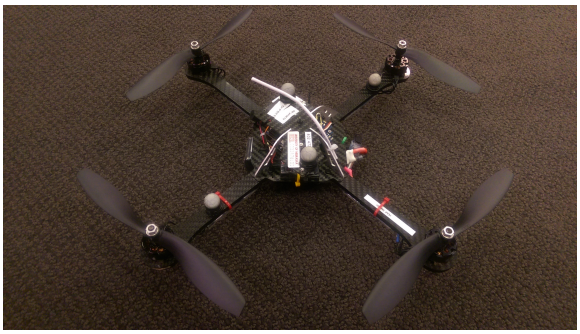


Figure 41: Hulk_Smash–Ascending Technologies Hummingbird with Safety Propellers

Three vehicles of this type were tested. 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 propellers is the most precise of the vehicles tested so far. 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.

8.2 Hummingbird: Normal Propellers

Figure 42: Jenny–Ascending Technologies Hummingbird with Normal Propellers

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.

8.3 Hummingbird: ARDrone Propellers

Figure 43: Herbie–Ascending Technologies Hummingbird with ARDrone Propellers

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 hummingbird_herbie20 and and test hummingbird_herbie21 as well as a calibration of Vicon. Mean tests between failures for this vehicle is 5.769.

8.4 ARDrone

Figure 44: Morpheus–ARDrone

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.

9 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.

A Documents

A.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 basic maneuvers.
    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 they should.
    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 a laptop.
    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 to execute.*
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 counter clockwise.
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

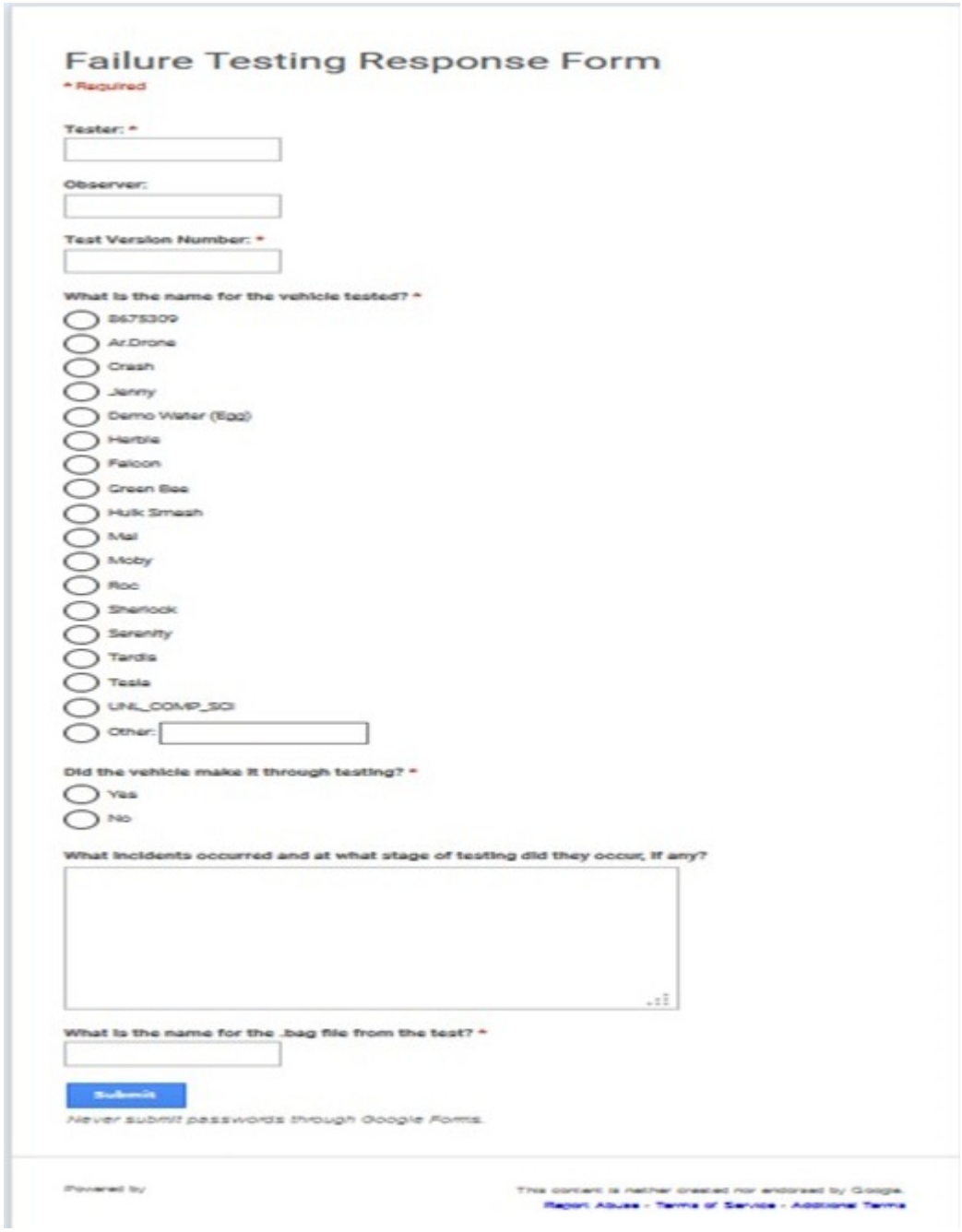
i. Turn off Futaba controller. *This step should take approximately 1 seconds to execute.*
ii. Turn off unmanned aerial vehicle. *This step should take approximately 1 seconds to execute.*
iii. Disconnect Dean's T-connector from unmanned aerial vehicle. *This step should take approximately 3 seconds 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 to execute.*
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

A.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

B Tables

B.1 uav stats

Vehicle Type and Failure Statistics

	(type)	(version)	(U.S. Doll	(%)	(count)	(count)	(count)	(count)	(count)	(count)	(count)	(count)	(count)	(count)	(count)	(count)	(count)	(count)	(count)	(count)
vehicle	propeller	firmware	cost	reliability	failedTest	totalTests	manualCo	serialCom	extraTask	missingTask	extraCom	missingCo	quadCont	command	taskedPos	subjectPo	viconDela	viconLost	motorsSta	vehicleFlipped
CRASH	AscTec Safety Propellers	AutoPilot LowLe	65535	14.11764706	72	510	1	5	24	13	24	11	53	59	56	61	15	0	12	0
HERBIE	ARDrone Propellers	AutoPilot LowLe	65535	14	14	100	0	1	6	6	3	3	5	7	9	6	5	17	0	0
JENNY	AscTec Normal Propellers	AutoPilot LowLe	65535	28.18181818	31	110	3	1	5	8	2	13	25	22	21	27	5	6	1	0
UNL_COM	AscTec Safety Propellers	AutoPilot LowLe	65535	4	4	100	1	0	1	1	0	2	0	2	0	1	2	20	4	1
HULK_SM	AscTec Safety Propellers	AutoPilot LowLe	65535	33.33333333	1	3	1	0	0	0	0	0	0	0	0	0	1	1	0	1
MORPHEU	ARDrone Propellers	ARDrone	65535	9	9	100	0	0	9	0	2	0	0	0	0	17	9	1	0	0

Power Statistics

	(Watts)	(Watts)	(Watts)	(Watts)	(Watts)				(Watts)	(Watts)	(Watts)	
vehicle	meanPower	moveMean	hoverMea	meanPowerRange_1	meanPow	meanPow	movePow	hoverPow	maximum	maximunPowe	maximun	maximumPowerVariance
CRASH	82.66084804	83.0821598	82.20829	58.40651941	90.9698	4.616241	6.185373	5.378042	196.0708	104.472101	255.8725	918.7963
HERBIE	100.5672491	99.36671631	101.7115	92.54076019	105.2594	3.336164	2.663839	4.623415	223.1796	132.1512348	287.6947	486.6856
JENNY	79.91797924	80.06422154	79.71828	74.21481585	82.8202	2.936749	3.433515	4.830631	283.412	151.3732326	357.4029	1749.056
UNL_COM	81.85654031	81.87821055	81.79171	78.28965495	87.76105	2.992963	3.754608	3.660042	211.5079	154.2375274	259.3286	347.8977
HULK_SM	84.78624535	83.89321665	85.64407	83.69525612	85.93508	0.632137	0.416943	0.90724	189.58	182.4243609	195.1521	43.32764
MORPHEU	99.31706498	99.33892008	99.29148	95.70190064	102.6881	0.496051	0.560547	0.751986	133.59	117.2867661	333.9904	372.6219

Position Error Statistics

	(centimeters - cm)	(cm)	(cm)	(cm)	(radians)	(millimet	(mm)	(mm)	(mm)	(radians)					
vehicle	totalSpatialError	totalMeanError	totalMea	totalMeanError_z	totalMea	spatialErr	meanErro	meanErro	meanErro	meanErrorPerT	varianceE	varianceE	varianceE	varianceE	rrorPerTask_rotation
CRASH	15.70689745	9.341345227	9.09747	8.237354659	0.746038	2.401467	1.428222	1.390935	1.25943	0.011406356	0.033678	0.03102	0.126689	0.093697	
HERBIE	19.50730404	11.39126166	11.98633	9.845999386	0.959975	2.992865	1.747679	1.838976	1.510601	0.014728196	0.039878	0.068036	0.137573	0.237242	
JENNY	17.54845046	7.988946309	9.692595	11.96350246	1.028808	2.675283	1.217925	1.477648	1.823851	0.015684311	0.024304	0.024904	0.076659	0.036595	
UNL_COM	23.29917198	13.48277931	13.9778	12.79001239	0.876375	3.551986	2.055466	2.130933	1.949853	0.013360436	0.007095	0.008439	0.027414	0.072426	
HULK_SM	26.05786859	15.31130154	14.74695	15.0447155	0.657777	3.96746	2.331233	2.245308	2.290644	0.010015035	5.45E-05	0.017952	0.001017	0.002035	
MORPHEU	33.40751877	14.24773811	18.54764	23.39562546	0.61617	5.106095	2.177663	2.834872	3.575851	0.009417708	0.127719	0.192669	0.04564	0.025068	

Test Execution Time Statistics

	(seconds - s)	(s)	(s)	
vehicle	timeMean	minimumTime	maximum	timeVariance
CRASH	65.40542527	59.99978342	70.01559	1.329628029
HERBIE	65.17937105	63.99887465	67.50048	0.520107666
JENNY	65.59475374	64.00359256	69.00325	0.993970426
UNL_COM	65.59476577	63.34760297	67.66205	0.603104957
HULK_SM	65.67897066	65.35812973	65.99981	0.205877806
MORPHEU	65.42674219	62.84271722	70.24519	1.436349567

B.2 Failure Testing Response Form (Responses)

C Code

C.1 ROS c++ code

```
cmake_minimum_required(VERSION 2.8.3)
project(flight_testing)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  collab_msgs
  collab_test
  #mavros
  message_generation
  roscpp
  rospy
  std_msgs
)

## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)


## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
# catkin_python_setup()

################################################
## Declare ROS messages, services and actions ##
################################################

## To declare and build messages, services or actions from within this
## package, follow these steps:
## * Let MSG_DEP_SET be the set of packages whose message types you use in
##   your messages/services/actions (e.g. std_msgs, actionlib_msgs, ...).
## * In the file package.xml:
##   * add a build_depend and a run_depend tag for each package in MSG_DEP_SET
##   * If MSG_DEP_SET isn't empty the following dependencies might have been
##     pulled in transitively but can be declared for certainty nonetheless:
##     * add a build_depend tag for "message_generation"
##     * add a run_depend tag for "message_runtime"
## * In this file (CMakeLists.txt):
```

Table 1: Failure Testing Response Form (Responses)

Timestamp	Test Version Number:	What is the name for the vehicle tested?	Did the vehicle make it through testing?	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?	Tester:	Observer:
5/27/2015 9:12:55	1	Crash	Yes	Current and voltage spikes received on sensor 1 as well as a low voltage on sensor 2 at launch.	hummingbird_crash_2015-05-26-13-59-35.bag	Willie	Levi
5/27/2015 9:13:58	1	Crash	Yes	Current and voltage spikes received on opposite sensors as well as a low voltage on sensor 1 during motors test.	hummingbird_crash_2015-05-26-14-01-09.bag	Willie	Levi
5/27/2015 9:14:37	1	Crash	Yes		hummingbird_crash_2015-05-26-14-02-52.bag	Willie	Levi
5/27/2015 9:15:11	1	Crash	Yes		hummingbird_crash_2015-05-26-14-04-54.bag	Willie	Levi
5/27/2015 9:15:41	1	Crash	Yes	Extra tasked waypose for first rotation received.	hummingbird_crash_2015-05-26-14-07-03.bag	Willie	Levi
5/27/2015 9:16:18	1	Crash	Yes	Current spike received during launch on sensor 1 as well as a voltage spike on sensor 2 during multiple position change. Target negative x and y position waypose not received.	hummingbird_crash_2015-05-26-14-08-40.bag	Willie	Levi
5/27/2015 9:16:54	1	Crash	Yes	Current spike prior to motors on received on sensor 1 as well as a voltage spike on sensor 2.	hummingbird_crash_2015-05-26-14-10-44.bag	Willie	Levi
5/27/2015 9:17:27	1	Crash	Yes		hummingbird_crash_2015-05-26-14-12-32.bag	Willie	Levi
5/27/2015 9:17:58	1	Crash	Yes	Extra tasked waypose for third rotation received.	hummingbird_crash_2015-05-26-14-14-47.bag	Willie	Levi
5/27/2015 9:18:30	1	Crash	Yes		hummingbird_crash_2015-05-26-14-16-23.bag	Willie	Levi
5/27/2015 9:19:59	1	Jenny	Yes		hummingbird_jenny_2015-05-26-14-25-45.bag	Willie	Levi
5/27/2015 9:20:52	1	Jenny	Yes		hummingbird_jenny_2015-05-26-14-28-34.bag	Willie	Levi
5/27/2015 9:21:25	1	Jenny	Yes		hummingbird_jenny_2015-05-26-14-30-17.bag	Willie	Levi
5/27/2015 9:21:59	1	Jenny	Yes		hummingbird_jenny_2015-05-26-14-31-49.bag	Willie	Levi
5/27/2015 9:22:32	1	Jenny	Yes		hummingbird_jenny_2015-05-26-14-33-20.bag	Willie	Levi
5/27/2015 9:23:00	1	Jenny	Yes		hummingbird_jenny_2015-05-26-14-35-12.bag	Willie	Levi
5/27/2015 9:23:45	1	Jenny	Yes	Extra tasked waypose for +y change and rotation received.	hummingbird_jenny_2015-05-26-14-37-00.bag	Willie	Levi
5/27/2015 9:24:24	1	Jenny	Yes	Missing move tasked waypose for second rotation.	hummingbird_jenny_2015-05-26-14-38-33.bag	Willie	Levi
5/27/2015 9:24:59	1	Jenny	Yes		hummingbird_jenny_2015-05-26-14-40-09.bag	Willie	Levi
5/27/2015 9:25:40	1	Jenny	Yes		hummingbird_jenny_2015-05-26-14-41-43.bag	Willie	Levi
5/27/2015 9:26:54	1	Herbie	Yes	Missing hover tasked waypose for +y change and rotation.	hummingbird_herbie_2015-05-26-14-47-11.bag	Willie	Levi
5/27/2015 9:27:21	1	Herbie	Yes		hummingbird_herbie_2015-05-26-14-48-48.bag	Willie	Levi
5/27/2015 9:27:55	1	Herbie	Yes	Current spike received on sensor 1 during launch as well as a voltage spike on sensor 2 during multiple position change. Extra tasked waypose for -y change and rotation received.	hummingbird_herbie_2015-05-26-14-50-32.bag	Willie	Levi
5/27/2015 9:28:20	1	Herbie	Yes		hummingbird_herbie_2015-05-26-14-52-12.bag	Willie	Levi
5/27/2015 9:28:46	1	Herbie	Yes		hummingbird_herbie_2015-05-26-14-53-57.bag	Willie	Levi
5/27/2015 9:29:15	1	Herbie	Yes		hummingbird_herbie_2015-05-26-14-55-34.bag	Willie	Levi
5/27/2015 9:29:53	1	Herbie	Yes	Missing hover tasked waypose for -x change, +z change, and rotation manuever.	hummingbird_herbie_2015-05-26-14-57-06.bag	Willie	Levi
5/27/2015 9:30:23	1	Herbie	Yes	Missing move tasked waypose for second rotation manuever.	hummingbird_herbie_2015-05-26-14-58-40.bag	Willie	Levi
5/27/2015 9:30:53	1	Herbie	Yes	Current spike received on sensor 1 as well as a voltage spike on sensor 2 during motors test. Missing move tasked waypose for second rotation manuever.	hummingbird_herbie_2015-05-26-15-00-14.bag	Willie	Levi
5/27/2015 9:31:18	1	Herbie	Yes	Extra hover state prior to performing flight maneuvers.	hummingbird_herbie_2015-05-26-15-02-12.bag	Willie	Levi
6/2/2015 13:22:10	1.1	Crash	Yes	The ground wire of the dean connector for the power to crash disconnected before test. A new male dean connector was soldered on. Current spikes in sensor 1 during launch and during multiple position change with corresponding voltage spikes in sensor 2.	hummingbird_crash1.bag	Willie	Levi
6/2/2015 13:22:57	1.1	Crash	Yes		hummingbird_crash2.bag	Willie	Levi
6/2/2015 13:23:11	1.1	Crash	Yes		hummingbird_crash3.bag	Willie	Levi
6/2/2015 13:23:28	1.1	Crash	Yes	Excess voltage received while turning off motors on sensor 2.	hummingbird_crash4.bag	Willie	Levi
6/2/2015 13:23:58	1.1	Crash	Yes	Excess voltage received while landing on sensor 2 and zero current received on sensor 1.	hummingbird_crash5.bag	Willie	Levi
6/2/2015 13:24:14	1.1	Crash	Yes		hummingbird_crash6.bag	Willie	Levi
6/2/2015 13:24:32	1.1	Crash	Yes		hummingbird_crash7.bag	Willie	Levi
6/2/2015 13:25:55	1.1	Crash	Yes		hummingbird_crash8.bag	Willie	Levi
6/2/2015 13:26:12	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while maneuvering. Current spike received during motors test on sensor 1.	hummingbird_crash9.bag	Willie	Levi
6/2/2015 13:26:25	1.1	Crash	Yes	Voltage spike on sensor 2, low voltage on sensor 1, and current spike on sensor 1 received after motors turned off.	hummingbird_crash10.bag	Willie	Levi
6/2/2015 13:29:30	1.1	Crash	Yes		hummingbird_crash11.bag	Willie	Levi
6/2/2015 13:31:32	1.1	Crash	Yes		hummingbird_crash12.bag	Willie	Levi
6/2/2015 13:38:09	1.1	Crash	Yes	Two motors test performed. Loss of serial communication after first motors test. Two additional tasked rotations during second rotation received.	hummingbird_crash13.bag	Willie	Levi
6/2/2015 13:46:04	1.1	Crash	Yes	Two motors test and loss of GPS control. There was odd jittering of the drone during the three rotation tests. Second motors test performed in the air during rotation step. Missed hover tasked waypose for +x position change.	hummingbird_crash14.bag	Willie	Levi
6/2/2015 13:46:31	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while performing motors test.	hummingbird_crash15.bag	Willie	Levi
6/2/2015 13:47:18	1.1	Crash	Yes	Excess voltage on both sensors and current spike on sensor 1 while manuevering.	hummingbird_crash16.bag	Willie	Levi
6/2/2015 13:49:03	1.1	Crash	Yes	Excess voltage on sensor 2 while maneuvering and current spike on sensor 1 while motors off after landing.	hummingbird_crash17.bag	Willie	Levi
6/2/2015 13:51:45	1.1	Crash	Yes		hummingbird_crash18.bag	Willie	Levi
6/2/2015 13:54:42	1.1	Crash	Yes		hummingbird_crash19.bag	Willie	Levi
6/2/2015 13:56:48	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while launching. Voltage spike received while motors off after landing on sensor 1.	hummingbird_crash20.bag	Willie	Levi
6/2/2015 14:00:27	1.1	Crash	Yes		hummingbird_crash21.bag	Willie	Levi
6/2/2015 14:02:56	1.1	Crash	Yes		hummingbird_crash22.bag	Willie	Levi
6/2/2015 14:09:39	1.1	Crash	Yes		hummingbird_crash23.bag	Willie	Levi
6/2/2015 14:10:18	1.1	Crash	Yes	Excessively low voltage on sensor 2 and uncharacteristically low current on sensor 1 while performing rotation test.	hummingbird_crash24.bag	Willie	Levi
6/2/2015 14:10:38	1.1	Crash	Yes		hummingbird_crash25.bag	Willie	Levi
6/2/2015 14:11:44	1.1	Crash	Yes	Low current received on sensor 2 during maneuvering. Prop hit power cord when the drone landed.	hummingbird_crash26.bag	Willie	Levi
6/2/2015 14:13:17	1.1	Crash	Yes	Excess voltage on sensor 1 prior to turning on motors and voltage spike on sensor 2 while launching.	hummingbird_crash27.bag	Willie	Levi
6/2/2015 14:15:10	1.1	Crash	Yes		hummingbird_crash28.bag	Willie	Levi
6/2/2015 14:16:55	1.1	Crash	Yes	Props hit power cord on landing.	hummingbird_crash29.bag	Willie	Levi
6/2/2015 14:20:14	1.1	Crash	Yes		hummingbird_crash30.bag	Willie	Levi
6/2/2015 14:22:48	1.1	Crash	Yes	Excess voltage on sensor 1 prior to turning on motors.	hummingbird_crash31.bag	Willie	Levi
6/2/2015 14:23:39	1.1	Crash	Yes		hummingbird_crash32.bag	Willie	Levi
6/2/2015 14:25:30	1.1	Crash	Yes		hummingbird_crash33.bag	Willie	Levi
6/2/2015 14:27:06	1.1	Crash	Yes		hummingbird_crash34.bag	Willie	Levi
6/2/2015 14:28:41	1.1	Crash	Yes		hummingbird_crash35.bag	Willie	Levi
6/2/2015 14:30:16	1.1	Crash	Yes		hummingbird_crash36.bag	Willie	Levi
6/2/2015 14:50:04	1.1	Crash	Yes		hummingbird_crash37.bag	Willie	Levi
6/2/2015 14:50:29	1.1	Crash	Yes	Excess voltage on sensor 2 while maneuvering.	hummingbird_crash38.bag	Willie	Levi
6/2/2015 14:50:46	1.1	Crash	Yes	Excess current on sensor 2 while maneuvering.	hummingbird_crash39.bag	Willie	Levi
6/2/2015 14:51:10	1.1	Crash	Yes		hummingbird_crash40.bag	Willie	Levi
6/2/2015 14:51:28	1.1	Crash	Yes		hummingbird_crash41.bag	Willie	Levi
6/2/2015 14:52:23	1.1	Crash	Yes	Near zero voltage on sensor 2 and current spike on sensor 1 while performing motors test.	hummingbird_crash42.bag	Willie	Levi
6/2/2015 14:54:22	1.1	Crash	Yes		hummingbird_crash43.bag	Willie	Levi
6/2/2015 14:56:00	1.1	Crash	Yes		hummingbird_crash44.bag	Willie	Levi
6/2/2015 14:57:41	1.1	Crash	Yes		hummingbird_crash45.bag	Willie	Levi
6/2/2015 14:59:30	1.1	Crash	Yes	Props hit power cord on landing.	hummingbird_crash46.bag	Willie	Levi
6/2/2015 15:01:32	1.1	Crash	Yes		hummingbird_crash47.bag	Willie	Levi
6/2/2015 15:03:20	1.1	Crash	Yes	Task waypose delayed one time stamp then two different task in rapid succession during last sequence. (subject_status not state)	hummingbird_crash48.bag	Willie	Levi
6/2/2015 15:05:03	1.1	Crash	Yes	Excess voltage on sensor 2 while performing motors test.	hummingbird_crash49.bag	Willie	Levi
6/2/2015 15:05:26	1.1	Crash	Yes	Three voltage spikes on sensor 2 and three corresponding current spikes on sensor 1 while performing motors test.	hummingbird_crash50.bag	Willie	Levi
6/2/2015 15:08:52	1.1	Crash	Yes		hummingbird_crash51.bag	Willie	Levi
6/2/2015 15:10:39	1.1	Crash	Yes		hummingbird_crash52.bag	Willie	Levi
6/2/2015 15:12:22	1.1	Crash	Yes		hummingbird_crash53.bag	Willie	Levi
6/2/2015 15:14:23	1.1	Crash	Yes		hummingbird_crash54.bag	Willie	Levi

Table 2: Failure Testing Response Form (Responses) Continued

6/2/2015 15:16:26	1.1	Crash	Yes		hummingbird_crash55.bag	Willie	Levi
6/2/2015 15:18:15	1.1	Crash	Yes		hummingbird_crash56.bag	Willie	Levi
6/2/2015 15:20:04	1.1	Crash	Yes		hummingbird_crash57.bag	Willie	Levi
6/2/2015 15:21:56	1.1	Crash	Yes		hummingbird_crash58.bag	Willie	Levi
6/2/2015 15:23:41	1.1	Crash	Yes		hummingbird_crash59.bag	Willie	Levi
6/2/2015 15:26:40	1.1	Crash	Yes		hummingbird_crash60.bag	Willie	Levi
6/2/2015 15:27:54	1.1	Crash	Yes	Voltage spike on sensor 2, low voltage on sensor 1, and current spike on sensor 1 after motors turned off. Prop hit power line during motor test.	hummingbird_crash61.bag	Willie	Levi
6/2/2015 15:29:35	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while performing rotation test.	hummingbird_crash62.bag	Willie	Levi
6/2/2015 15:31:26	1.1	Crash	Yes		hummingbird_crash63.bag	Willie	Levi
6/2/2015 15:37:15	1.1	Crash	Yes	Two motors test and loss of GPS control. Launched prior to performing motors test. Performed second motors test at ~0.7 m. Missing third move tasked rotation.	hummingbird_crash64.bag	Willie	Levi
6/2/2015 15:38:33	1.1	Crash	Yes	Excess voltage on sensor 2 while maneuvering and current spike on sensor 1 prior to turning motors on.	hummingbird_crash65.bag	Willie	Levi
6/2/2015 15:40:26	1.1	Crash	Yes	Excess voltage on sensor 2 while performing motors test. Zero current on sensor 2 during maneuvering.	hummingbird_crash66.bag	Willie	Levi
6/2/2015 15:42:13	1.1	Crash	Yes		hummingbird_crash67.bag	Willie	Levi
6/2/2015 15:44:06	1.1	Crash	Yes		hummingbird_crash68.bag	Willie	Levi
6/2/2015 15:45:53	1.1	Crash	Yes	Excess voltage on sensor 2 while performing motors test.	hummingbird_crash69.bag	Willie	Levi
6/2/2015 15:47:45	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while landing.	hummingbird_crash70.bag	Willie	Levi
6/2/2015 15:49:28	1.1	Crash	Yes		hummingbird_crash71.bag	Willie	Levi
6/2/2015 15:51:06	1.1	Crash	Yes		hummingbird_crash72.bag	Willie	Levi
6/2/2015 15:52:45	1.1	Crash	Yes	Excess voltage on sensor 2 while launching.	hummingbird_crash73.bag	Willie	Levi
6/2/2015 15:54:14	1.1	Crash	Yes	Current spike on sensor 2 while maneuvering.	hummingbird_crash74.bag	Willie	Levi
6/2/2015 15:56:11	1.1	Crash	Yes		hummingbird_crash75.bag	Willie	Levi
6/2/2015 15:58:38	1.1	Crash	Yes		hummingbird_crash76.bag	Willie	Levi
6/2/2015 15:59:50	1.1	Crash	Yes		hummingbird_crash77.bag	Willie	Levi
6/2/2015 16:01:40	1.1	Crash	Yes		hummingbird_crash78.bag	Willie	Levi
6/2/2015 16:03:23	1.1	Crash	Yes		hummingbird_crash79.bag	Willie	Levi
6/2/2015 16:04:59	1.1	Crash	Yes		hummingbird_crash80.bag	Willie	Levi
6/2/2015 16:07:29	1.1	Crash	Yes		hummingbird_crash81.bag	Willie	Levi
6/2/2015 16:09:45	1.1	Crash	Yes		hummingbird_crash82.bag	Willie	Levi
6/2/2015 16:10:25	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while landing.	hummingbird_crash83.bag	Willie	Levi
6/2/2015 16:11:24	1.1	Crash	Yes		hummingbird_crash84.bag	Willie	Levi
6/2/2015 16:13:02	1.1	Crash	Yes	Voltage spike on sensor 2, low voltage on sensor 1, and current spike on sensor 1 received while launching.	hummingbird_crash85.bag	Willie	Levi
6/2/2015 16:14:34	1.1	Crash	Yes		hummingbird_crash86.bag	Willie	Levi
6/2/2015 16:16:24	1.1	Crash	Yes		hummingbird_crash87.bag	Willie	Levi
6/2/2015 16:18:11	1.1	Crash	Yes		hummingbird_crash88.bag	Willie	Levi
6/2/2015 16:19:44	1.1	Crash	Yes		hummingbird_crash89.bag	Willie	Levi
6/2/2015 16:21:35	1.1	Crash	Yes		hummingbird_crash90.bag	Willie	Levi
6/2/2015 16:23:46	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while maneuvering.	hummingbird_crash91.bag	Willie	Levi
6/2/2015 16:25:50	1.1	Crash	Yes		hummingbird_crash92.bag	Willie	Levi
6/2/2015 16:27:09	1.1	Crash	Yes		hummingbird_crash93.bag	Willie	Levi
6/2/2015 16:27:59	1.1	Crash	Yes		hummingbird_crash94.bag	Willie	Levi
6/2/2015 16:29:36	1.1	Crash	Yes		hummingbird_crash95.bag	Willie	Levi
6/2/2015 16:31:15	1.1	Crash	Yes		hummingbird_crash96.bag	Willie	Levi
6/2/2015 16:32:49	1.1	Crash	Yes		hummingbird_crash97.bag	Willie	Levi
6/2/2015 16:34:30	1.1	Crash	Yes		hummingbird_crash98.bag	Willie	Levi
6/2/2015 16:36:11	1.1	Crash	Yes		hummingbird_crash99.bag	Willie	Levi
6/2/2015 16:37:58	1.1	Crash	Yes		hummingbird_crash100.bag	Willie	Levi
6/3/2015 11:05:23	1.1	Crash	Yes		hummingbird_crash101.bag	Willie	Levi
6/3/2015 11:07:05	1.1	Crash	Yes		hummingbird_crash102.bag	Willie	Levi
6/3/2015 11:09:31	1.1	Crash	Yes	Low voltage on sensor 2 and current spike on sensor 1 received while performing motors test.	hummingbird_crash103.bag	Willie	Levi
6/3/2015 11:11:09	1.1	Crash	Yes		hummingbird_crash104.bag	Willie	Levi
6/3/2015 11:12:44	1.1	Crash	Yes	Loss of Vicon	hummingbird_crash105.bag	Willie	Levi
6/3/2015 11:14:22	1.1	Crash	Yes	Excess voltage on sensor 2 received while launching. Delayed and rushed receipt of tasked waypose during last maneuvering sequence. (subject_status not state)	hummingbird_crash106.bag	Willie	Levi
6/3/2015 11:14:41	1.1	Crash	No	During height change maneuver, drone lost control and shot high into air. Manual was used to recover.	hummingbird_crash107.bag	Willie	Levi
6/3/2015 11:19:51	1.1	Crash	Yes	Late start, operator error, resulted in loss of GPS control and 4 motors test.	hummingbird_crash108.bag	Willie	Levi
6/3/2015 11:21:46	1.1	Crash	Yes	Nearly zero voltage on sensor 2 and current spike on sensor 1 received while launching.	hummingbird_crash109.bag	Willie	Levi
6/3/2015 11:23:32	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while maneuvering.	hummingbird_crash110.bag	Willie	Levi
6/3/2015 11:25:23	1.1	Crash	Yes	Excess voltage on sensor 2 while launching. Extra hover state at launch , missing task on first rotation.	hummingbird_crash111.bag	Willie	Levi
6/3/2015 11:27:05	1.1	Crash	Yes		hummingbird_crash112.bag	Willie	Levi
6/3/2015 11:29:11	1.1	Crash	Yes		hummingbird_crash113.bag	Willie	Levi
6/3/2015 11:30:47	1.1	Crash	Yes		hummingbird_crash114.bag	Willie	Levi
6/3/2015 11:32:30	1.1	Crash	Yes	A duplicate -x position change and rotation received. Three delays and rapid succession of tasked waypose received prior to -y change, -z change, and -x and +z change. Tasked waypose for last 2 maneuvers not received.	hummingbird_crash115.bag	Willie	Levi
6/3/2015 11:34:18	1.1	Crash	Yes		hummingbird_crash116.bag	Willie	Levi
6/3/2015 11:36:05	1.1	Crash	Yes		hummingbird_crash117.bag	Willie	Levi
6/3/2015 11:37:55	1.1	Crash	Yes		hummingbird_crash118.bag	Willie	Levi
6/3/2015 11:41:41	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while performing motors test.	hummingbird_crash119.bag	Willie	Levi
6/3/2015 11:42:50	1.1	Crash	Yes	Two current spikes on sensor 1 and a voltage spike on sensor 2 received during maneuvering. Voltage spike received on sensor 2 after motors turned off. Missing a move tasked waypose. Missed rotation during rotation test, and during the last part of the test it missed a commanded vicon position.	hummingbird_crash120.bag	Willie	Levi
6/3/2015 11:46:24	1.1	Crash	Yes		hummingbird_crash121.bag	Willie	Levi
6/3/2015 11:47:12	1.1	Crash	Yes	Excess voltage on sensor 2 while launching.	hummingbird_crash122.bag	Willie	Levi
6/3/2015 11:49:00	1.1	Crash	Yes	Voltage spike on sensor 2, low voltage on sensor 1, and current spike on sensor 1 received while launching. An uncommanded move tasked waypose received during rotation test.	hummingbird_crash123.bag	Willie	Levi
6/3/2015 11:50:49	1.1	Crash	Yes	Low voltage on sensor 2 and current spike on sensor 1 received while performing motors test.	hummingbird_crash124.bag	Willie	Levi
6/3/2015 11:52:48	1.1	Crash	Yes		hummingbird_crash125.bag	Willie	Levi
6/3/2015 11:54:20	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 received while launching.	hummingbird_crash126.bag	Willie	Levi
6/3/2015 11:55:49	1.1	Crash	Yes		hummingbird_crash127.bag	Willie	Levi

Table 3: Failure Testing Response Form (Responses) Continued

6/3/2015 11:58:35	1.1	Crash	Yes	Voltage spike on sensor 2, low voltage on sensor 1, and current spike on sensor 1 received while performing motors test. Missing tasked move +z change waypose. Missed middle set of rotations, the height rotations.	hummingbird_crash128.bag	Willie	Levi
6/3/2015 11:59:51	1.1	Crash	Yes		hummingbird_crash129.bag	Willie	Levi
6/3/2015 12:01:34	1.1	Crash	Yes		hummingbird_crash130.bag	Willie	Levi
6/3/2015 12:03:14	1.1	Crash	Yes		hummingbird_crash131.bag	Willie	Levi
6/3/2015 12:06:46	1.1	Crash	Yes	Excess voltage on sensor 2 received while maneuvering. Vehicle remained in hover for an additional time stamp and then proceeded normally.	hummingbird_crash132.bag	Willie	Levi
6/3/2015 12:07:31	1.1	Crash	Yes	Loss of connection during change of y axis direction and rotation. Missed a tasked move waypose during indicated maneuver (subject status not state).	hummingbird_crash133.bag	Willie	Levi
6/3/2015 12:11:41	1.1	Crash	Yes		hummingbird_crash134.bag	Willie	Levi
6/3/2015 12:11:59	1.1	Crash	Yes	Excess voltage on sensor 2 received prior to turning on motors.	hummingbird_crash135.bag	Willie	Levi
6/3/2015 12:13:12	1.1	Crash	Yes		hummingbird_crash136.bag	Willie	Levi
6/3/2015 12:15:04	1.1	Crash	Yes	Excess voltage on sensor 2 received prior to turning on motors as well as while performing motors test.	hummingbird_crash137.bag	Willie	Levi
6/3/2015 12:27:55	1.1	Crash	Yes		hummingbird_crash138.bag	Willie	Levi
6/3/2015 12:28:15	1.1	Crash	Yes		hummingbird_crash139.bag	Willie	Levi
6/3/2015 12:28:40	1.1	Crash	Yes		hummingbird_crash140.bag	Willie	Levi
6/3/2015 12:32:18	1.1	Crash	Yes	Duplicate uncommanded waypose received during first multiple position change.	hummingbird_crash141.bag	Willie	Levi
6/3/2015 12:32:39	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while maneuvering. Tasked waypose delayed one time stamp, and an additional waypose (uncommanded change in x) received later.	hummingbird_crash142.bag	Willie	Levi
6/3/2015 12:32:55	1.1	Crash	Yes		hummingbird_crash143.bag	Willie	Levi
6/3/2015 12:33:33	1.1	Crash	Yes		hummingbird_crash144.bag	Willie	Levi
6/3/2015 12:34:05	1.1	Crash	Yes		hummingbird_crash145.bag	Willie	Levi
6/3/2015 12:34:35	1.1	Crash	Yes		hummingbird_crash146.bag	Willie	Levi
6/3/2015 12:35:00	1.1	Crash	Yes		hummingbird_crash147.bag	Willie	Levi
6/3/2015 12:35:29	1.1	Crash	Yes		hummingbird_crash148.bag	Willie	Levi
6/3/2015 12:35:52	1.1	Crash	Yes		hummingbird_crash149.bag	Willie	Levi
6/3/2015 12:40:36	1.1	Crash	Yes		hummingbird_crash150.bag	Willie	Levi
6/3/2015 12:40:55	1.1	Crash	Yes		hummingbird_crash151.bag	Willie	Levi
6/3/2015 12:42:40	1.1	Crash	Yes		hummingbird_crash152.bag	Willie	Levi
6/3/2015 12:44:23	1.1	Crash	Yes		hummingbird_crash153.bag	Willie	Levi
6/3/2015 12:45:46	1.1	Crash	Yes	Duplicate move tasked waypose during -y change.	hummingbird_crash154.bag	Willie	Levi
6/3/2015 12:47:33	1.1	Crash	Yes	Lost control during rotation tests. Dropped serial communication and UAV dropped (0.5 m).	hummingbird_crash155.bag	Willie	Levi
6/3/2015 12:49:33	1.1	Crash	Yes		hummingbird_crash156.bag	Willie	Levi
6/3/2015 12:51:29	1.1	Crash	Yes		hummingbird_crash157.bag	Willie	Levi
6/3/2015 12:52:53	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 recorded while maneuvering.	hummingbird_crash158.bag	Willie	Levi
6/3/2015 12:55:50	1.1	Crash	Yes	Duplicate waypose sent during first set of multiple position changes. Twice the commanded position change acheived in x, y, and z directions. Excess voltage recorded while motors off. Went to limit of out-of-bound position during multiple position change.	hummingbird_crash159.bag	Willie	Levi
6/3/2015 12:57:17	1.1	Crash	Yes	Excess voltage on sensor 2 while maneuvering and current spike on sensor 1 prior to turning motors on.	hummingbird_crash160.bag	Willie	Levi
6/3/2015 12:59:17	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 while performing motors test. Voltage spike on sensor 2, low voltage on sensor 1, and current spike on sensor 1 received while launching.	hummingbird_crash161.bag	Willie	Levi
6/3/2015 13:01:04	1.1	Crash	Yes		hummingbird_crash162.bag	Willie	Levi
6/3/2015 13:02:53	1.1	Crash	Yes		hummingbird_crash163.bag	Willie	Levi
6/3/2015 13:04:51	1.1	Crash	Yes	Duplicate hover waypose received while rotating and -y change. Overshot rotation, likely due to moving before rotating which is visa versa of its correct behavior.	hummingbird_crash164.bag	Willie	Levi
6/3/2015 13:05:30	1.1	Crash	Yes		hummingbird_crash165.bag	Willie	Levi
6/3/2015 13:06:29	1.1	Crash	Yes		hummingbird_crash166.bag	Willie	Levi
6/3/2015 13:10:57	1.1	Crash	Yes	Missed a tasked waypose during multiple position change. Went to a intermediary state before last rotation (subject status not state).	hummingbird_crash167.bag	Willie	Levi
6/3/2015 13:12:14	1.1	Crash	Yes	Dropped serial communication for 2 time stamps and a tasked waypose prior to turning off motors.	hummingbird_crash168.bag	Willie	Levi
6/3/2015 13:14:53	1.1	Crash	Yes		hummingbird_crash169.bag	Willie	Levi
6/3/2015 13:16:26	1.1	Crash	Yes	During motors turn on, left motor stalled for some time before starting up.	hummingbird_crash170.bag	Willie	Levi
6/3/2015 13:17:53	1.1	Crash	Yes	Excess voltage recorded on sensor 2 while maneuvering.	hummingbird_crash171.bag	Willie	Levi
6/3/2015 13:19:35	1.1	Crash	Yes	One time stamp stall during first rotation and prior to landing.	hummingbird_crash172.bag	Willie	Levi
6/3/2015 13:21:16	1.1	Crash	Yes	Serial communication lost for 3 time stamps prior to launch.	hummingbird_crash173.bag	Willie	Levi
6/3/2015 13:22:50	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 recorded while maneuvering.	hummingbird_crash174.bag	Willie	Levi
6/3/2015 13:24:35	1.1	Crash	Yes	Excess voltage ron sensor 2 and low current on sensor 1 while launching recorded.	hummingbird_crash175.bag	Willie	Levi
6/3/2015 13:26:53	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 while maneuvering received. Nearly zero current recorded on both sensors several time stamps later.Two commanded waypose not received. Lost control during height change.	hummingbird_crash176.bag	Willie	Levi
6/3/2015 13:28:51	1.1	Crash	Yes	Excess voltage recorded on sensor 1 prior to landing. Lost control during landing.	hummingbird_crash177.bag	Willie	Levi
6/3/2015 13:30:32	1.1	Crash	Yes		hummingbird_crash178.bag	Willie	Levi
6/3/2015 13:32:54	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 recorded while maneuvering. Receiving of target waypose delayed by one time stamp during rotation test. (subject_status and not state)	hummingbird_crash179.bag	Willie	Levi
6/3/2015 13:35:05	1.1	Crash	Yes	Did not receive -y change and rotation target waypose. Lost commanded waypose.	hummingbird_crash180.bag	Willie	Levi
6/3/2015 13:36:44	1.1	Crash	Yes		hummingbird_crash181.bag	Willie	Levi
6/3/2015 13:38:30	1.1	Crash	Yes		hummingbird_crash182.bag	Willie	Levi
6/3/2015 13:40:19	1.1	Crash	Yes		hummingbird_crash183.bag	Willie	Levi
6/3/2015 13:42:01	1.1	Crash	Yes		hummingbird_crash184.bag	Willie	Levi
6/3/2015 13:44:04	1.1	Crash	Yes	Overshot rotation by 90 degrees on -y change. Duplicate task waypose received delayed receipt of target waypose by two time stamps.	hummingbird_crash185.bag	Willie	Levi
6/3/2015 13:46:16	1.1	Crash	Yes	Zero voltage on sensor 2 and current spike on sensor 1 received prior to turning on motors. Delayed target waypose update 3 times. (subject_status and not state)	hummingbird_crash186.bag	Willie	Levi
6/3/2015 13:47:55	1.1	Crash	Yes	Voltage spike on sensor 2, low voltage on sensor 1, and current spike on sensor 1 received prior to turning on motors.	hummingbird_crash187.bag	Willie	Levi

Table 4: Failure Testing Response Form (Responses) Continued

6/3/2015 13:49:39	1.1	Crash	Yes		hummingbird_crash188.bag	Willie	Levi
6/3/2015 13:51:16	1.1	Crash	Yes	Zero voltage on sensor 2 and current spike on sensor 1 received while maneuvering.	hummingbird_crash189.bag	Willie	Levi
6/3/2015 13:53:50	1.1	Crash	Yes	While turning on motors, motors momentarily turned off and turned back on again. Excess voltage recorded on sensor 2 while maneuvering. Serial communication lost for 3 time stamps during motors turning off.	hummingbird_crash190.bag	Willie	Levi
6/3/2015 13:55:19	1.1	Crash	Yes		hummingbird_crash191.bag	Willie	Levi
6/3/2015 13:57:25	1.1	Crash	Yes	Prop hit power line on landing.	hummingbird_crash192.bag	Willie	Levi
6/3/2015 13:59:27	1.1	Crash	Yes	Lost control during negative height change, prop hit power cord during landing.	hummingbird_crash193.bag	Willie	Levi
6/3/2015 14:01:33	1.1	Crash	Yes	Excess voltage on sensor 2 recorded when motors turned off and move state maintained for an additional time stamp during first multiple position change.	hummingbird_crash194.bag	Willie	Levi
6/3/2015 14:03:23	1.1	Crash	Yes		hummingbird_crash195.bag	Willie	Levi
6/3/2015 14:05:32	1.1	Crash	Yes	Excess voltage recorded on sensor 1 while turning motors off.	hummingbird_crash196.bag	Willie	Levi
6/3/2015 14:06:54	1.1	Crash	Yes		hummingbird_crash197.bag	Willie	Levi
6/3/2015 14:08:38	1.1	Crash	Yes	Excess voltage on sensor 2 and current spike on sensor 1 recorded while maneuvering. Did not receive a tasked waypose during multiple position change.	hummingbird_crash198.bag	Willie	Levi
6/3/2015 14:10:23	1.1	Crash	Yes		hummingbird_crash199.bag	Willie	Levi
6/3/2015 14:12:06	1.1	Crash	Yes		hummingbird_crash200.bag	Willie	Levi
6/4/2015 9:38:15	1.1	Crash	Yes	Low voltage on sensor 2 and current spike on sensor 1 received while testing motors.	hummingbird_crash201.bag	Willie	Levi
6/4/2015 9:40:34	1.1	Crash	Yes		hummingbird_crash202.bag	Willie	Levi
6/4/2015 9:42:25	1.1	Crash	Yes		hummingbird_crash203.bag	Willie	Levi
6/4/2015 9:44:17	1.1	Crash	Yes		hummingbird_crash204.bag	Willie	Levi
6/4/2015 9:46:01	1.1	Crash	Yes		hummingbird_crash205.bag	Willie	Levi
6/4/2015 9:47:29	1.1	Crash	Yes		hummingbird_crash206.bag	Willie	Levi
6/4/2015 9:49:16	1.1	Crash	Yes		hummingbird_crash207.bag	Willie	Levi
6/4/2015 9:51:34	1.1	Crash	Yes	Low voltage on sensor 2 and current spike on sensor 1 recorded while launching.	hummingbird_crash208.bag	Willie	Levi
6/4/2015 9:53:07	1.1	Crash	Yes		hummingbird_crash209.bag	Willie	Levi
6/4/2015 9:54:44	1.1	Crash	Yes		hummingbird_crash210.bag	Willie	Levi
6/4/2015 9:56:26	1.1	Crash	Yes		hummingbird_crash211.bag	Willie	Levi
6/4/2015 9:58:05	1.1	Crash	Yes		hummingbird_crash212.bag	Willie	Levi
6/4/2015 10:00:21	1.1	Crash	Yes		hummingbird_crash213.bag	Willie	Levi
6/4/2015 10:01:39	1.1	Crash	Yes	Prior to turning on motors current spike on sensor 1 and voltage spike on sensor 2 recorded.	hummingbird_crash214.bag	Willie	Levi
6/4/2015 10:03:17	1.1	Crash	Yes		hummingbird_crash215.bag	Willie	Levi
6/4/2015 10:05:01	1.1	Crash	Yes		hummingbird_crash216.bag	Willie	Levi
6/4/2015 10:05:27	1.1	Crash	Yes		hummingbird_crash217.bag	Willie	Levi
6/4/2015 10:06:13	1.1	Crash	Yes		hummingbird_crash218.bag	Willie	Levi
6/4/2015 10:10:09	1.1	Crash	Yes	Voltage spike on sensor 2 received while launching.	hummingbird_crash219.bag	Willie	Levi
6/4/2015 10:11:48	1.1	Crash	Yes		hummingbird_crash220.bag	Willie	Levi
6/4/2015 10:13:29	1.1	Crash	Yes		hummingbird_crash221.bag	Willie	Levi
6/4/2015 10:15:35	1.1	Crash	Yes		hummingbird_crash222.bag	Willie	Levi
6/4/2015 10:17:00	1.1	Crash	Yes		hummingbird_crash223.bag	Willie	Levi
6/4/2015 10:18:27	1.1	Crash	Yes		hummingbird_crash224.bag	Willie	Levi
6/4/2015 10:20:09	1.1	Crash	Yes		hummingbird_crash225.bag	Willie	Levi
6/4/2015 10:21:53	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 received while launching.	hummingbird_crash226.bag	Willie	Levi
6/4/2015 10:23:38	1.1	Crash	Yes	Current spike on sensor 1, voltage spike on sensor 2 and zero voltage on sensor 1 recorded while testing motors.	hummingbird_crash227.bag	Willie	Levi
6/4/2015 10:25:18	1.1	Crash	Yes		hummingbird_crash228.bag	Willie	Levi
6/4/2015 10:26:49	1.1	Crash	Yes		hummingbird_crash229.bag	Willie	Levi
6/4/2015 10:30:31	1.1	Crash	Yes		hummingbird_crash230.bag	Willie	Levi
6/4/2015 10:31:25	1.1	Crash	Yes		hummingbird_crash231.bag	Willie	Levi
6/4/2015 10:32:32	1.1	Crash	Yes		hummingbird_crash232.bag	Willie	Levi
6/4/2015 10:34:08	1.1	Crash	Yes		hummingbird_crash233.bag	Willie	Levi
6/4/2015 10:37:00	1.1	Crash	Yes	Prop hit power line when landing.	hummingbird_crash234.bag	Willie	Levi
6/4/2015 10:38:03	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while turning motors off.	hummingbird_crash235.bag	Willie	Levi
6/4/2015 10:39:35	1.1	Crash	Yes		hummingbird_crash236.bag	Willie	Levi
6/4/2015 10:41:17	1.1	Crash	Yes		hummingbird_crash237.bag	Willie	Levi
6/4/2015 10:43:09	1.1	Crash	Yes		hummingbird_crash238.bag	Willie	Levi
6/4/2015 10:45:18	1.1	Crash	Yes	Prop hit power cord on motors test. Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching.	hummingbird_crash239.bag	Willie	Levi
6/4/2015 10:46:48	1.1	Crash	Yes		hummingbird_crash240.bag	Willie	Levi
6/4/2015 10:48:40	1.1	Crash	Yes		hummingbird_crash241bag	Willie	Levi
6/4/2015 10:50:16	1.1	Crash	Yes		hummingbird_crash242.bag	Willie	Levi
6/4/2015 10:51:59	1.1	Crash	Yes		hummingbird_crash243.bag	Willie	Levi
6/4/2015 10:53:41	1.1	Crash	Yes		hummingbird_crash244.bag	Willie	Levi
6/4/2015 10:55:23	1.1	Crash	Yes		hummingbird_crash245.bag	Willie	Levi
6/4/2015 10:57:04	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while testing motors.	hummingbird_crash246.bag	Willie	Levi
6/4/2015 11:06:35	1.1	Crash	Yes		hummingbird_crash247.bag	Willie	Levi
6/4/2015 11:06:56	1.1	Crash	Yes		hummingbird_crash248.bag	Willie	Levi
6/4/2015 11:07:24	1.1	Crash	Yes	Went to intermediate rotation state during first commanded rotation.	hummingbird_crash249.bag	Willie	Levi
6/4/2015 11:08:10	1.1	Crash	Yes	Went to intermediate rotation state during first commanded rotation.	hummingbird_crash250.bag	Willie	Levi
6/4/2015 11:08:41	1.1	Crash	Yes		hummingbird_crash251.bag	Willie	Levi
6/4/2015 11:09:10	1.1	Crash	Yes		hummingbird_crash252.bag	Willie	Levi
6/4/2015 11:09:41	1.1	Crash	Yes		hummingbird_crash253.bag	Willie	Levi
6/4/2015 11:10:44	1.1	Crash	Yes		hummingbird_crash254.bag	Willie	Levi
6/4/2015 11:12:27	1.1	Crash	Yes		hummingbird_crash255.bag	Willie	Levi
6/4/2015 11:15:30	1.1	Crash	Yes	Voltage spike on both sensors and current spike on sensor 1 recorded while launching.	hummingbird_crash256.bag	Willie	Levi
6/4/2015 11:18:11	1.1	Crash	Yes	Zero voltage on sensor 2 and current spike on sensor 1 recorded prior to turning on motors.	hummingbird_crash257.bag	Willie	Levi
6/4/2015 11:19:27	1.1	Crash	Yes		hummingbird_crash258.bag	Willie	Levi
6/4/2015 11:21:02	1.1	Crash	Yes		hummingbird_crash259.bag	Willie	Levi
6/4/2015 11:24:41	1.1	Crash	Yes		hummingbird_crash260.bag	Willie	Levi
6/4/2015 11:26:24	1.1	Crash	Yes		hummingbird_crash261.bag	Willie	Levi
6/4/2015 11:27:53	1.1	Crash	Yes		hummingbird_crash262.bag	Willie	Levi
6/4/2015 11:29:23	1.1	Crash	Yes	Zero voltage on sensor 2 and current spike on sensor 1 recorded while testing motors.	hummingbird_crash263.bag	Willie	Levi
6/4/2015 11:30:58	1.1	Crash	Yes	Did not perform -y position change. Current spike on sensor 1, voltage spike on sensor 2 and zero voltage on sensor 1 recorded prior to turning on motors. Did not receive +x position change move waypose during multiple position change maneuver.	hummingbird_crash264.bag	Willie	Levi
6/4/2015 11:32:35	1.1	Crash	Yes		hummingbird_crash265.bag	Willie	Levi
6/4/2015 11:34:01	1.1	Crash	Yes		hummingbird_crash266.bag	Willie	Levi
6/4/2015 11:35:39	1.1	Crash	Yes		hummingbird_crash267.bag	Willie	Levi
6/4/2015 11:37:07	1.1	Crash	Yes		hummingbird_crash268.bag	Willie	Levi
6/4/2015 11:39:10	1.1	Crash	Yes	Abnormally quick decent on negative height change.	hummingbird_crash269.bag	Willie	Levi
6/4/2015 11:40:13	1.1	Crash	Yes		hummingbird_crash270.bag	Willie	Levi
6/4/2015 11:41:39	1.1	Crash	Yes		hummingbird_crash271.bag	Willie	Levi
6/4/2015 11:43:06	1.1	Crash	Yes		hummingbird_crash272.bag	Willie	Levi
6/4/2015 11:44:49	1.1	Crash	Yes	Current spike on sensor 1 recorded prior to turning on motors.	hummingbird_crash273.bag	Willie	Levi
6/4/2015 11:46:19	1.1	Crash	Yes		hummingbird_crash274.bag	Willie	Levi

Table 5: Failure Testing Response Form (Responses) Continued

6/4/2015 11:47:47	1.1	Crash	Yes		hummingbird_crash275.bag	Willie	Levi
6/4/2015 11:49:14	1.1	Crash	Yes		hummingbird_crash276.bag	Willie	Levi
6/4/2015 11:51:02	1.1	Crash	Yes		hummingbird_crash277.bag	Willie	Levi
6/4/2015 11:52:49	1.1	Crash	Yes	Left motor stalled on motors turning on.	hummingbird_crash278.bag	Willie	Levi
6/4/2015 11:54:31	1.1	Crash	Yes		hummingbird_crash279.bag	Willie	Levi
6/4/2015 11:55:59	1.1	Crash	Yes		hummingbird_crash280.bag	Willie	Levi
6/4/2015 11:57:28	1.1	Crash	Yes		hummingbird_crash281.bag	Willie	Levi
6/4/2015 11:59:03	1.1	Crash	Yes		hummingbird_crash282.bag	Willie	Levi
6/4/2015 12:00:31	1.1	Crash	Yes		hummingbird_crash283.bag	Willie	Levi
6/4/2015 12:02:04	1.1	Crash	Yes	Zero voltage on sensor 2 and current spike on sensor 1 recorded prior to turning on motors.	hummingbird_crash284.bag	Willie	Levi
6/4/2015 12:03:25	1.1	Crash	Yes		hummingbird_crash285.bag	Willie	Levi
6/4/2015 12:05:00	1.1	Crash	Yes		hummingbird_crash286.bag	Willie	Levi
6/4/2015 12:06:35	1.1	Crash	Yes		hummingbird_crash287.bag	Willie	Levi
6/16/2015 16:58:51	1.1	Crash	Yes	Voltage spike on sensor 2 current spiike sensor 1 while launching.	hummingbird_crash288.bag	Willie	Levi
6/4/2015 12:08:41	1.1	Crash	Yes		hummingbird_crash289.bag	Willie	Levi
6/4/2015 12:09:36	1.1	Crash	Yes		hummingbird_crash290.bag	Willie	Levi
6/4/2015 12:11:01	1.1	Crash	Yes		hummingbird_crash291.bag	Willie	Levi
6/4/2015 12:12:25	1.1	Crash	Yes		hummingbird_crash292.bag	Willie	Levi
6/4/2015 12:14:09	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while testing motors.	hummingbird_crash293.bag	Willie	Levi
6/4/2015 12:29:25	1.1	Crash	Yes		hummingbird_crash294.bag	Willie	Levi
6/4/2015 12:29:47	1.1	Crash	Yes		hummingbird_crash295.bag	Willie	Levi
6/4/2015 12:30:05	1.1	Crash	Yes	Received repeated move task waypose for first rotation.	hummingbird_crash296.bag	Willie	Levi
6/4/2015 12:30:30	1.1	Crash	Yes		hummingbird_crash297.bag	Willie	Levi
6/4/2015 12:30:53	1.1	Crash	Yes	Current spike on sensor 1, voltage spike on sensor 2 and low voltage on sensor 1 recorded while turning on motors.	hummingbird_crash298.bag	Willie	Levi
6/4/2015 12:31:17	1.1	Crash	Yes		hummingbird_crash299.bag	Willie	Levi
6/4/2015 12:31:50	1.1	Crash	Yes		hummingbird_crash300.bag	Willie	Levi
6/4/2015 12:32:37	1.1	Crash	Yes	Left motor stalled between motors on and motor test.	hummingbird_crash301.bag	Willie	Levi
6/4/2015 12:33:53	1.1	Crash	Yes	Current spike on sensor 1, voltage spike on sensor 2 and low voltage on sensor 1 recorded while testing motors.	hummingbird_crash302.bag	Willie	Levi
6/4/2015 12:35:28	1.1	Crash	Yes		hummingbird_crash303.bag	Willie	Levi
6/4/2015 12:36:41	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while testing motors.	hummingbird_crash304.bag	Willie	Levi
6/4/2015 12:38:00	1.1	Crash	Yes	Current spike on sensor 1, voltage spike on sensor 2 and low voltage on sensor 1 recorded while turning off motors.	hummingbird_crash305.bag	Willie	Levi
6/4/2015 12:39:47	1.1	Crash	Yes	Voltage spike recorded on sensor 2 prior to turnring on motors.	hummingbird_crash306.bag	Willie	Levi
6/4/2015 12:42:27	1.1	Crash	Yes		hummingbird_crash307.bag	Willie	Levi
6/4/2015 12:46:01	1.1	Crash	Yes		hummingbird_crash308.bag	Willie	Levi
6/4/2015 12:48:24	1.1	Crash	Yes		hummingbird_crash309.bag	Willie	Levi
6/4/2015 12:50:02	1.1	Crash	Yes		hummingbird_crash310.bag	Willie	Levi
6/4/2015 12:51:39	1.1	Crash	Yes		hummingbird_crash311.bag	Willie	Levi
6/4/2015 12:54:32	1.1	Crash	Yes		hummingbird_crash312.bag	Willie	Levi
6/4/2015 12:55:47	1.1	Crash	Yes		hummingbird_crash313.bag	Willie	Levi
6/4/2015 12:57:23	1.1	Crash	Yes		hummingbird_crash314.bag	Willie	Levi
6/4/2015 13:00:54	1.1	Crash	No	Did not progress through states, did not launch.	hummingbird_crash315.bag	Willie	Levi
6/4/2015 13:04:49	1.1	Crash	No	Communication failure.	hummingbird_crash316.bag	Willie	Levi
6/4/2015 13:09:10	1.1	Crash	No	Did not launch.	hummingbird_crash317.bag	Willie	Levi
6/4/2015 13:09:41	1.1	Crash	No	Failure to launch.	hummingbird_crash318.bag	Willie	Levi
6/4/2015 14:20:43	1.1	Crash	Yes		hummingbird_crash319.bag	Willie	Levi
6/4/2015 14:22:57	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded at different time stamps prior to turning on motors.	hummingbird_crash320.bag	Willie	Levi
6/4/2015 14:23:59	1.1	Crash	Yes		hummingbird_crash321.bag	Willie	Levi
6/4/2015 14:25:54	1.1	Crash	Yes		hummingbird_crash322.bag	Willie	Levi
6/4/2015 14:27:03	1.1	Crash	Yes		hummingbird_crash323.bag	Willie	Levi
6/4/2015 14:28:20	1.1	Crash	Yes	Receiving of tasked waypose delayed then rushed for a set of move-hover tasks after fixing excessive rotation on last rotation test.	hummingbird_crash324.bag	Willie	Levi
6/4/2015 14:29:43	1.1	Crash	Yes		hummingbird_crash325.bag	Willie	Levi
6/4/2015 14:31:54	1.1	Crash	Yes		hummingbird_crash326.bag	Willie	Levi
6/4/2015 14:33:19	1.1	Crash	Yes		hummingbird_crash327.bag	Willie	Levi
6/4/2015 14:36:12	1.1	Crash	Yes	Back motor stalled on motors turn on.	hummingbird_crash328.bag	Willie	Levi
6/4/2015 14:36:30	1.1	Crash	Yes		hummingbird_crash329.bag	Willie	Levi
6/4/2015 14:37:58	1.1	Crash	Yes		hummingbird_crash330.bag	Willie	Levi
6/4/2015 14:39:49	1.1	Crash	Yes		hummingbird_crash331.bag	Willie	Levi
6/4/2015 14:41:12	1.1	Crash	Yes	Current spike on sensor 1, voltage spike on both sensors recorded while launching. Subject_status' waypose received field failed to update for the majority of the flight. Task_waypose was updated and flight appeared normal to the observers.	hummingbird_crash332.bag	Willie	Levi
6/4/2015 14:42:21	1.1	Crash	Yes	Excess voltage recorded on sensor 1 prior to turning on motors.	hummingbird_crash333.bag	Willie	Levi
6/4/2015 14:43:42	1.1	Crash	Yes		hummingbird_crash334.bag	Willie	Levi
6/4/2015 14:44:54	1.1	Crash	Yes		hummingbird_crash335.bag	Willie	Levi
6/4/2015 14:46:26	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while turning on motors.	hummingbird_crash336.bag	Willie	Levi
6/4/2015 14:47:56	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while launching.	hummingbird_crash337.bag	Willie	Levi
6/4/2015 14:49:18	1.1	Crash	Yes		hummingbird_crash338.bag	Willie	Levi
6/4/2015 14:50:46	1.1	Crash	Yes		hummingbird_crash339.bag	Willie	Levi
6/4/2015 14:51:11	1.1	Crash	Yes		hummingbird_crash340.bag	Willie	Levi
6/4/2015 14:53:45	1.1	Crash	Yes		hummingbird_crash341.bag	Willie	Levi
6/4/2015 14:55:16	1.1	Crash	Yes		hummingbird_crash342.bag	Willie	Levi
6/4/2015 14:56:45	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while landing.	hummingbird_crash343.bag	Willie	Levi
6/4/2015 14:58:14	1.1	Crash	Yes	Current and voltage spikes on sensor 1 recorded while maneuvering as well as low voltage on sensor 2.	hummingbird_crash344.bag	Willie	Levi
6/4/2015 14:59:59	1.1	Crash	Yes		hummingbird_crash345.bag	Willie	Levi
6/4/2015 15:01:35	1.1	Crash	Yes		hummingbird_crash346.bag	Willie	Levi
6/4/2015 15:02:55	1.1	Crash	Yes		hummingbird_crash347.bag	Willie	Levi
6/4/2015 15:04:23	1.1	Crash	Yes		hummingbird_crash348.bag	Willie	Levi
6/4/2015 15:06:02	1.1	Crash	Yes		hummingbird_crash349.bag	Willie	Levi
6/4/2015 15:07:15	1.1	Crash	Yes		hummingbird_crash350.bag	Willie	Levi
6/4/2015 15:08:55	1.1	Crash	Yes		hummingbird_crash351.bag	Willie	Levi
6/4/2015 15:10:40	1.1	Crash	Yes	Voltage spike recorded on sensor 1 while launching.	hummingbird_crash352.bag	Willie	Levi
6/4/2015 15:11:58	1.1	Crash	Yes		hummingbird_crash353.bag	Willie	Levi
6/4/2015 15:13:24	1.1	Crash	Yes		hummingbird_crash354.bag	Willie	Levi
6/4/2015 15:14:56	1.1	Crash	Yes		hummingbird_crash355.bag	Willie	Levi
6/4/2015 15:16:30	1.1	Crash	Yes		hummingbird_crash356.bag	Willie	Levi
6/4/2015 15:17:50	1.1	Crash	Yes		hummingbird_crash357.bag	Willie	Levi
6/4/2015 15:25:27	1.1	Crash	Yes		hummingbird_crash358.bag	Willie	Levi
6/4/2015 15:25:45	1.1	Crash	Yes		hummingbird_crash359.bag	Willie	Levi
6/4/2015 15:26:13	1.1	Crash	Yes	Excess voltage on sensor 2 while testing motors and current spike on sensor 1 prior to turning on motors recorded.	hummingbird_crash360.bag	Willie	Levi
6/4/2015 15:26:45	1.1	Crash	Yes		hummingbird_crash361.bag	Willie	Levi
6/4/2015 15:28:31	1.1	Crash	Yes		hummingbird_crash362.bag	Willie	Levi
6/4/2015 15:29:36	1.1	Crash	Yes		hummingbird_crash363.bag	Willie	Levi
6/4/2015 15:31:10	1.1	Crash	Yes		hummingbird_crash364.bag	Willie	Levi
6/4/2015 15:32:56	1.1	Crash	Yes		hummingbird_crash365.bag	Willie	Levi

Table 6: Failure Testing Response Form (Responses) Continued

6/4/2015 15:34:17	1.1	Crash	Yes	Received 2 -x change move task waypose in rapid succession, did not move to +x position or rotate to pi/2.	hummingbird_crash366.bag	Willie	Levi
6/4/2015 15:35:49	1.1	Crash	Yes		hummingbird_crash367.bag	Willie	Levi
6/4/2015 15:37:07	1.1	Crash	Yes		hummingbird_crash368.bag	Willie	Levi
6/4/2015 15:38:42	1.1	Crash	Yes	Task waypose starts at launch height and not the ground.	hummingbird_crash369.bag	Willie	Levi
6/4/2015 15:40:14	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded after turning off motors.	hummingbird_crash370.bag	Willie	Levi
6/4/2015 15:43:01	1.1	Crash	Yes		hummingbird_crash371.bag	Wilie	Levi
6/4/2015 15:43:41	1.1	Crash	Yes		hummingbird_crash372.bag	Willie	Levi
6/4/2015 15:45:11	1.1	Crash	Yes		hummingbird_crash373.bag	Willie	Levi
6/4/2015 15:46:45	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while launching.	hummingbird_crash374.bag	Willie	Levi
6/4/2015 15:51:31	1.1	Crash	Yes		hummingbird_crash375.bag	Willie	Levi
6/4/2015 15:53:01	1.1	Crash	Yes		hummingbird_crash376.bag	Willie	Levi
6/4/2015 15:54:36	1.1	Crash	Yes	Voltage spike recorded on sensor 2 before turning on motors.	hummingbird_crash377.bag	Willie	Levi
6/4/2015 15:56:03	1.1	Crash	Yes		hummingbird_crash378.bag	Willie	Levi
6/4/2015 15:57:33	1.1	Crash	Yes	Subject_status' waypose received field failed to update for the majority of the flight. Task_waypose was updated and flight appeared normal to the observers.	hummingbird_crash379.bag	Willie	Levi
6/4/2015 15:59:06	1.1	Crash	Yes		hummingbird_crash380.bag	Willie	Levi
6/4/2015 16:00:42	1.1	Crash	Yes		hummingbird_crash381.bag	Willie	Levi
6/4/2015 16:02:16	1.1	Crash	Yes		hummingbird_crash382.bag	Willie	Levi
6/4/2015 16:03:45	1.1	Crash	Yes		hummingbird_crash383.bag	Willie	Levi
6/4/2015 16:05:22	1.1	Crash	Yes		hummingbird_crash384.bag	Willie	Levi
6/4/2015 16:06:49	1.1	Crash	Yes	During launch, Crash went to uncommanded waypos. Voltage spikes recorded on sensor 2 while launching and while maneuvering.	hummingbird_crash385.bag	Willie	Levi
6/4/2015 16:09:40	1.1	Crash	Yes	Voltage spike recorded on sensor 2 before turning on motors.	hummingbird_crash386.bag	Willie	Levi
6/4/2015 16:17:03	1.1	Crash	Yes		hummingbird_crash387.bag	Willie	Levi
6/4/2015 16:19:09	1.1	Crash	Yes	Second height change, doubled normal change in height.	hummingbird_crash388.bag	Willie	Levi
6/4/2015 16:20:59	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while maneuvering.	hummingbird_crash389.bag	Willie	Levi
6/4/2015 16:22:32	1.1	Crash	Yes	Current spikes on sensor 1 and voltage spikes on sensor 2 recorded while launching and while landing.	hummingbird_crash390.bag	Willie	Levi
6/4/2015 16:24:01	1.1	Crash	Yes		hummingbird_crash391.bag	Willie	Levi
6/4/2015 16:26:02	1.1	Crash	Yes		hummingbird_crash392.bag	Willie	Levi
6/4/2015 16:27:29	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while launching.	hummingbird_crash393.bag	Willie	Levi
6/4/2015 16:29:00	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while landing.	hummingbird_crash394.bag	Willie	Levi
6/4/2015 16:30:29	1.1	Crash	Yes		hummingbird_crash395.bag	Willie	Levi
6/4/2015 16:32:08	1.1	Crash	Yes		hummingbird_crash396.bag	Willie	Levi
6/4/2015 16:34:38	1.1	Crash	Yes		hummingbird_crash397.bag	Willie	Levi
6/4/2015 16:36:09	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering.	hummingbird_crash398.bag	Willie	Levi
6/4/2015 16:38:12	1.1	Crash	Yes	Lost control on + X change. Received 2 +x position changes and did not rotate to pi/2.	hummingbird_crash399.bag	Willie	Levi
6/4/2015 16:40:40	1.1	Crash	Yes		hummingbird_crash400.bag	Willie	Levi
6/10/2015 12:36:52	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching. Voltage spike on sensor 2 recorded while testing motors.	hummingbird_crash401.bag	Willie	Levi
6/10/2015 12:39:04	1.1	Crash	Yes		hummingbird_crash402.bag	Willie	Levi
6/10/2015 12:40:25	1.1	Crash	Yes		hummingbird_crash403.bag	Willie	Levi
6/10/2015 12:41:32	1.1	Crash	Yes	Did not preform the height change (what a coincidence!). Voltage spike recorded on sensor 2 while maneuvering with nearly zero currents for both sensors. Did not receive +z position change move waypose.	hummingbird_crash404.bag	Willie	Levi
6/10/2015 12:43:08	1.1	Crash	Yes		hummingbird_crash405.bag	Willie	Levi
6/10/2015 12:44:31	1.1	Crash	Yes	Current spikes on sensor 1 and voltage spike on sensor 2 recorded while testing motors and while maneuvering.	hummingbird_crash406.bag	Willie	Levi
6/10/2015 12:46:13	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering. Voltage spike on sensor 2 recorded while testing motors.	hummingbird_crash407.bag	Willie	Levi
6/10/2015 12:47:39	1.1	Crash	Yes		hummingbird_crash408.bag	Willie	Levi
6/10/2015 12:49:06	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while maneuvering. Extra +z position change move waypose received.	hummingbird_crash409.bag	Willie	Levi
6/10/2015 12:51:02	1.1	Crash	Yes	Voltage spike recorded on sensor 1 while launching.	hummingbird_crash410.bag	Willie	Levi
6/10/2015 12:52:29	1.1	Crash	Yes		hummingbird_crash411.bag	Willie	Levi
6/10/2015 12:53:58	1.1	Crash	Yes		hummingbird_crash412.bag	Willie	Levi
6/10/2015 12:55:55	1.1	Crash	Yes		hummingbird_crash413.bag	Willie	Levi
6/10/2015 12:57:36	1.1	Crash	Yes	Slow on implementing the height change. Current spikes on sensor 1 and voltage spike on sensor 2 recorded while testing motors and while maneuvering.	hummingbird_crash414.bag	Willie	Levi
6/10/2015 13:00:18	1.1	Crash	Yes	Twice +y change distance acheived during + y change. Two move wayposes received in a row.	hummingbird_crash415.bag	Willie	Levi
6/10/2015 13:01:10	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching.	hummingbird_crash416.bag	Willie	Levi
6/10/2015 13:02:36	1.1	Crash	Yes		hummingbird_crash417.bag	Willie	Levi
6/10/2015 13:04:04	1.1	Crash	Yes		hummingbird_crash418.bag	Willie	Levi
6/10/2015 13:05:36	1.1	Crash	Yes		hummingbird_crash419.bag	Willie	Levi
6/10/2015 13:08:16	1.1	Crash	Yes	Prop got caught on power cord and spun the drone around on landing.	hummingbird_crash420.bag	Willie	Levi
6/10/2015 13:10:36	1.1	Crash	Yes		hummingbird_crash421.bag	Willie	Levi
6/10/2015 13:12:04	1.1	Crash	Yes		hummingbird_crash422.bag	Willie	Levi
6/10/2015 13:14:08	1.1	Crash	Yes	Possible loss of waypose around height change stage. Delayed update of subject_status prior to +z position change.	hummingbird_crash423.bag	Willie	Levi
6/10/2015 13:15:29	1.1	Crash	Yes	Current spike on sensor 1, voltage spike on sensor 2 and low voltage on sensor 1 recorded while turning on motors.	hummingbird_crash424.bag	Willie	Levi
6/10/2015 13:20:31	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while launching.	hummingbird_crash425.bag	Willie	Levi
6/10/2015 13:21:00	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded before turning on motors. Voltage spike on sensor 2 recorded while maneuvering. Prop hit power cable on landing.	hummingbird_crash426.bag	Willie	Levi
6/10/2015 13:21:30	1.1	Crash	Yes	Current spike on sensor 1 recorded while testing motors. Current spike on sensor 2 recorded while landing. Voltage spike on sensor 2 recorded while launching.	hummingbird_crash427.bag	Willie	Levi
6/10/2015 13:23:46	1.1	Crash	Yes		hummingbird_crash428.bag	Willie	Levi
6/10/2015 13:25:08	1.1	Crash	Yes		hummingbird_crash429.bag	Willie	Levi
6/10/2015 13:26:39	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while turning off motors.	hummingbird_crash430.bag	Willie	Levi
6/10/2015 13:28:04	1.1	Crash	Yes	Current spike on sensor 1 recorded while maneuvering. Voltage spikes on sensor 2 recorded while testing motors and maneuvering.	hummingbird_crash431.bag	Willie	Levi
6/10/2015 13:29:35	1.1	Crash	Yes	Current spike recorded on sensor 2 while launching.	hummingbird_crash432.bag	Willie	Levi
6/10/2015 13:31:15	1.1	Crash	Yes	Voltage spike recorded on sensor 1 while turning off motors.	hummingbird_crash433.bag	Willie	Levi
6/10/2015 13:32:47	1.1	Crash	Yes		hummingbird_crash434.bag	Willie	Levi
6/10/2015 13:34:43	1.1	Crash	Yes		hummingbird_crash435.bag	Willie	Levi
6/10/2015 13:36:12	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while testing motors.	hummingbird_crash436.bag	Willie	Levi

Table 7: Failure Testing Response Form (Responses) Continued

6/10/2015 13:37:38	1.1	Crash	Yes	Failed to turn motors off after landing. Motors turned off and turned back on.	hummingbird_crash437.bag	Willie	Levi
6/10/2015 13:40:59	1.1	Crash	Yes		hummingbird_crash438.bag	Willie	Levi
6/10/2015 13:45:11	1.1	Crash	Yes	Extra move waypose for +x and +y position change maneuver received.	hummingbird_crash439.bag	Willie	Levi
6/10/2015 13:45:39	1.1	Crash	Yes		hummingbird_crash440.bag	Willie	Levi
6/10/2015 13:46:20	1.1	Crash	Yes	Did not complete target rotation during rotation test. Voltage spike recorded on sensor 2 while launching. Third rotation not executed though tasked waypose received.	hummingbird_crash441.bag	Willie	Levi
6/10/2015 13:47:20	1.1	Crash	Yes	Extra move waypose for +x and rotate to pi/2 position change maneuver received.	hummingbird_crash442.bag	Willie	Levi
6/10/2015 13:48:57	1.1	Crash	Yes		hummingbird_crash443.bag	Willie	Levi
6/10/2015 13:53:38	1.1	Crash	Yes		hummingbird_crash444.bag	Willie	Levi
6/10/2015 13:54:02	1.1	Crash	Yes		hummingbird_crash445.bag	Willie	Levi
6/10/2015 13:54:21	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching.	hummingbird_crash446.bag	Willie	Levi
6/10/2015 13:55:17	1.1	Crash	Yes	Extra task added after launch.	hummingbird_crash447.bag	Willie	Levi
6/10/2015 13:56:26	1.1	Crash	Yes	Current spike on sensor 1 and low voltage on sensor 2 recorded while launching.	hummingbird_crash448.bag	Willie	Levi
6/10/2015 13:57:59	1.1	Crash	Yes		hummingbird_crash449.bag	Willie	Levi
6/10/2015 13:59:24	1.1	Crash	Yes		hummingbird_crash450.bag	Willie	Levi
6/10/2015 14:00:44	1.1	Crash	Yes		hummingbird_crash451.bag	Willie	Levi
6/10/2015 14:02:25	1.1	Crash	Yes	Abnormal landing, Crash bounced.	hummingbird_crash452.bag	Willie	Levi
6/10/2015 14:04:46	1.1	Crash	Yes	Lost control during final waypose, didn't make it back to home position, hit power line on loss of control.	hummingbird_crash453.bag	Willie	Levi
6/10/2015 14:05:41	1.1	Crash	Yes		hummingbird_crash454.bag	Willie	Levi
6/10/2015 14:08:03	1.1	Crash	Yes		hummingbird_crash455.bag	Willie	Levi
6/10/2015 14:08:56	1.1	Crash	Yes		hummingbird_crash456.bag	Willie	Levi
6/10/2015 14:10:41	1.1	Crash	Yes	UAV dropped. Immediately landed after launch, and relaunched. Voltage spike recorded on sensor 2 while turning on motors. Extra move task waypose for first rotation but did not rotate.	hummingbird_crash457.bag	Willie	Levi
6/10/2015 14:13:09	1.1	Crash	Yes	Immediately landed after launch, and relaunched. Extra move task waypose for first rotation.	hummingbird_crash458.bag	Willie	Levi
6/10/2015 14:15:01	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while launching. Prop grazed power line on landing.	hummingbird_crash459.bag	Willie	Levi
6/10/2015 14:16:29	1.1	Crash	Yes		hummingbird_crash460.bag	Willie	Levi
6/10/2015 14:18:03	1.1	Crash	Yes	Possible delay in second to last maneuver. Voltage spike recorded on sensor 2 while maneuvering.	hummingbird_crash461.bag	Willie	Levi
6/10/2015 14:20:18	1.1	Crash	Yes	Lost control during -y change without rotation. Extra move waypose for +y change position received.	hummingbird_crash462.bag	Willie	Levi
6/10/2015 14:21:38	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while launching.	hummingbird_crash463.bag	Willie	Levi
6/10/2015 14:22:57	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching.	hummingbird_crash464.bag	Willie	Levi
6/10/2015 14:24:32	1.1	Crash	Yes		hummingbird_crash465.bag	Willie	Levi
6/10/2015 14:26:14	1.1	Crash	Yes	Extra move waypose for -z and -y position change maneuver received. Voltage spike recorded on sensor 2 while maneuvering during indicated maneuver.	hummingbird_crash466.bag	Willie	Levi
6/10/2015 14:28:53	1.1	Crash	Yes	Duplicate waypose sent during second set of multiple position changes. Extra move waypose for +z, +x, and +y position change maneuver received.	hummingbird_crash467.bag	Willie	Levi
6/10/2015 14:30:16	1.1	Crash	Yes	Current spike on sensor 1 and low voltage on sensor 2 recorded before turning on motors.	hummingbird_crash468.bag	Willie	Levi
6/10/2015 14:31:48	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while launching. Battery power cord hit prop on landing.	hummingbird_crash469.bag	Willie	Levi
6/10/2015 14:33:33	1.1	Crash	Yes		hummingbird_crash470.bag	Willie	Levi
6/10/2015 14:35:44	1.1	Crash	Yes	Duplicate waypose on multiple position change. Extra move waypose for +z, +x, and +y position change maneuver received. Voltage spike on sensor 1 recorded while maneuvering during indicated maneuver. Voltage spike on sensor 2 recorded while launching.	hummingbird_crash471.bag	Willie	Levi
6/10/2015 14:37:05	1.1	Crash	Yes	Lost vicon on first multiple change sequence, UAV dropped.	hummingbird_crash472.bag	Willie	Levi
6/10/2015 14:38:09	1.1	Crash	Yes		hummingbird_crash473.bag	Willie	Levi
6/10/2015 14:39:48	1.1	Crash	Yes		hummingbird_crash474.bag	Willie	Levi
6/10/2015 14:41:15	1.1	Crash	Yes	Current spike on sensor 1, low voltage on sensor 1, and voltage spike on sensor 2 recorded before turning on motors.	hummingbird_crash475.bag	Willie	Levi
6/10/2015 14:42:55	1.1	Crash	Yes	Loss of Vicon during +x position change and rotate to pi/2 maneuver occurred. Voltage spike recorded on sensor 2 while performing indicated maneuver.	hummingbird_crash476.bag	Willie	Levi
6/10/2015 14:44:21	1.1	Crash	Yes	Current spike on sensor 1 and low voltage on sensor 2 recorded while launching. Extra move waypose for -x position change maneuver received.	hummingbird_crash477.bag	Willie	Levi
6/10/2015 14:48:14	1.1	Crash	Yes		hummingbird_crash478.bag	Willie	Levi
6/10/2015 14:48:30	1.1	Crash	Yes		hummingbird_crash479.bag	Willie	Levi
6/10/2015 14:49:04	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching.	hummingbird_crash480.bag	Willie	Levi
6/10/2015 14:50:28	1.1	Crash	Yes	Loss of Vicon before landing.	hummingbird_crash481.bag	Willie	Levi
6/10/2015 14:52:39	1.1	Crash	Yes	Motors turned back on after turning off. Voltage spike recorded on sensor 2 while launching.	hummingbird_crash482.bag	Willie	Levi
6/10/2015 14:54:21	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while launching.	hummingbird_crash483.bag	Willie	Levi
6/10/2015 14:56:10	1.1	Crash	Yes		hummingbird_crash484.bag	Willie	Levi
6/10/2015 14:57:32	1.1	Crash	Yes		hummingbird_crash485.bag	Willie	Levi
6/10/2015 14:59:23	1.1	Crash	Yes	Lost Vicon while performing -y change and rotate to 3*pi/2.	hummingbird_crash486.bag	Willie	Levi
6/10/2015 15:00:59	1.1	Crash	Yes	Failed to turn off motors.	hummingbird_crash487.bag	Willie	Levi
6/10/2015 15:02:56	1.1	Crash	Yes	Voltage spike recorded on sensor 2 while testing motors.	hummingbird_crash488.bag	Willie	Levi
6/10/2015 15:04:56	1.1	Crash	Yes	Stall before +z change and rotate to pi maneuver. Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering.	hummingbird_crash489.bag	Willie	Levi
6/10/2015 15:06:53	1.1	Crash	Yes		hummingbird_crash490.bag	Willie	Levi
6/10/2015 15:07:56	1.1	Crash	Yes	Voltage spike recorded on sensor 2 before turning on motors. Extra move waypose for +x and +y position change maneuver received.	hummingbird_crash491.bag	Willie	Levi
6/10/2015 15:10:09	1.1	Crash	Yes	Abnormal behavior on -x, -y, and rotate to pi/2 maneuver.	hummingbird_crash492.bag	Willie	Levi
6/10/2015 15:11:36	1.1	Crash	Yes		hummingbird_crash493.bag	Willie	Levi
6/10/2015 15:13:23	1.1	Crash	Yes	Current spike on sensor 1 recorded before turning on motors. Voltage spike on sensor 2 recorded before turning on motors (different time from current spike).	hummingbird_crash494.bag	Willie	Levi
6/10/2015 15:14:48	1.1	Crash	Yes		hummingbird_crash495.bag	Willie	Levi
6/10/2015 15:16:39	1.1	Crash	Yes	Possibly didn't preform the final rotation test. Loss of vicon while performing rotate to pi maneuver.	hummingbird_crash496.bag	Willie	Levi
6/10/2015 15:18:12	1.1	Crash	Yes	Lost vicon during -x change. Voltage spikes recorded on sensor 2 while turning on motors and maneuvering.	hummingbird_crash497.bag	Willie	Levi
6/10/2015 15:19:29	1.1	Crash	Yes		hummingbird_crash498.bag	Willie	Levi
6/10/2015 15:20:55	1.1	Crash	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while testing motors.	hummingbird_crash499.bag	Willie	Levi
6/10/2015 15:22:56	1.1	Crash	Yes		hummingbird_crash500.bag	Willie	Levi
6/11/2015 11:11:59	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while testing motors.	hummingbird_jenny1.bag	Willie	Levi
6/11/2015 11:12:26	1.1	Jenny	Yes	Lost Vicon on return to home. Current spike on sensor 1 and voltage spike on sensor 2 recorded while performing -z, +x, +y, and rotate to 0.0 change maneuver.	hummingbird_jenny2.bag	Willie	Levi
6/11/2015 11:12:42	1.1	Jenny	Yes		hummingbird_jenny3.bag	Willie	Levi
6/11/2015 11:13:15	1.1	Jenny	Yes		hummingbird_jenny4.bag	Willie	Levi
6/11/2015 11:13:55	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching. Props hit power cord on landing.	hummingbird_jenny5.bag	Willie	Levi

Table 8: Failure Testing Response Form (Responses) Continued

6/11/2015 11:15:14	1.1	Jenny	Yes	Lost Vicon on 2nd rotation, and hit power cord on landing. Current spike on sensor 2 and zero voltage on sensor 2 recorded while maneuvering.	hummingbird_jenny6.bag	Willie	Levi
6/11/2015 11:17:00	1.1	Jenny	Yes	Voltage spikes recorded on sensor 2 while launching and maneuvering. Hit power cord on landing.	hummingbird_jenny7.bag	Willie	Levi
6/11/2015 11:20:15	1.1	Jenny	Yes	Hit power cord on landing.	hummingbird_jenny8.bag	Willie	Levi
6/11/2015 11:20:45	1.1	Jenny	Yes	Right motor stall on motors on. Current spike on sensor 1, low voltage on sensor 1, and voltage spike on sensor 2 recorded before turning on motors.	hummingbird_jenny9.bag	Willie	Levi
6/11/2015 11:22:55	1.1	Jenny	Yes	Lost VICON while performing -x, -y, and rotate to pi/2 position change maneuver. Voltage spike recorded on sensor 2 while performing indicated maneuver.	hummingbird_jenny10.bag	Willie	Levi
6/11/2015 11:25:34	1.1	Jenny	Yes	Current spike on sensor 1, low voltage on sensor 1, and voltage spike on sensor 2 recorded while testing motors.	hummingbird_jenny11.bag	Willie	Levi
6/11/2015 11:28:49	1.1	Jenny	Yes	Current and voltage spikes on sensor 1 as well as low voltage on sensor 2 recorded while maneuvering. Voltage spike on sensor 2 recorded while launching.	hummingbird_jenny12.bag	Willie	Levi
6/11/2015 11:29:03	1.1	Jenny	Yes	Voltage spike recorded on sensor 2 before turning on motors.	hummingbird_jenny13.bag	Willie	Levi
6/11/2015 11:30:45	1.1	Jenny	Yes	Lost waypose during x position change. Did not receive move waypose for -x and rotate to 0.0 position change. Voltage spike recorded on sensor 2 after turning off motors.	hummingbird_jenny14.bag	Willie	Levi
6/11/2015 11:32:34	1.1	Jenny	Yes		hummingbird_jenny15.bag	Willie	Levi
6/11/2015 11:33:42	1.1	Jenny	Yes	Current spike on sensor 1, low voltage on sensor 1, and voltage spike on sensor 2 recorded before turning on motors.	hummingbird_jenny16.bag	Willie	Levi
6/11/2015 11:35:20	1.1	Jenny	Yes	Tapped power cord on landing.	hummingbird_jenny17.bag	Willie	Levi
6/11/2015 11:36:58	1.1	Jenny	Yes	Voltage spike recorded on sensor 2 while turning on motors. Tapped power cord on landing.	hummingbird_jenny18.bag	Willie	Levi
6/11/2015 11:39:11	1.1	Jenny	Yes	Lost several waypose's during x position change, seemed to skip the y position change, and tapped power line on landing. Lost Vicon while performing -x and rotate to pi/2 position change. Current spike on sensor 1 and voltage spike on sensor 2 recorded while performing indicated maneuver.	hummingbird_jenny19.bag	Willie	Levi
6/11/2015 11:40:48	1.1	Jenny	Yes	Lost Vicon while peforming +x, +y, and rotate to pi/2 position change maneuver.	hummingbird_jenny20.bag	Willie	Levi
6/11/2015 11:42:25	1.1	Jenny	Yes	Prop hit power cord on motors off.	hummingbird_jenny21.bag	Willie	Levi
6/11/2015 11:44:05	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching. Prop hit power cord on landing.	hummingbird_jenny22.bag	Willie	Levi
6/11/2015 11:45:22	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while testing motors.	hummingbird_jenny23.bag	Willie	Levi
6/11/2015 11:47:19	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded while testing motors.	hummingbird_jenny24.bag	Willie	Levi
6/11/2015 11:48:57	1.1	Jenny	Yes	Hit power cord on landing.	hummingbird_jenny25.bag	Willie	Levi
6/11/2015 11:50:29	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded while launching.	hummingbird_jenny26.bag	Willie	Levi
6/11/2015 11:52:04	1.1	Jenny	Yes		hummingbird_jenny27.bag	Willie	Levi
6/11/2015 11:55:58	1.1	Jenny	Yes		hummingbird_jenny28.bag	Willie	Levi
6/11/2015 11:56:31	1.1	Jenny	Yes	Lost Vicon while performing +x position change and rotate to pi/2. Current spike on sensor 1 recorded while maneuvering. Voltage spikes on sensor 2 recorded while maneuvering (different time then current spike).	hummingbird_jenny29.bag	Willie	Levi
6/11/2015 11:57:29	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering.	hummingbird_jenny30.bag	Willie	Levi
6/11/2015 11:58:54	1.1	Jenny	Yes		hummingbird_jenny31.bag	Willie	Levi
6/11/2015 12:00:31	1.1	Jenny	Yes		hummingbird_jenny32.bag	Willie	Levi
6/11/2015 12:02:25	1.1	Jenny	Yes	Hit power cord on landing.	hummingbird_jenny33.bag	Willie	Levi
6/11/2015 12:04:14	1.1	Jenny	Yes	Lost Vicon while performing -z position change.	hummingbird_jenny34.bag	Willie	Levi
6/11/2015 12:06:26	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering. Current spikes on sensor 1 recorded before and while turning on motors.	hummingbird_jenny35.bag	Willie	Levi
6/11/2015 12:07:15	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded while launching.	hummingbird_jenny36.bag	Willie	Levi
6/11/2015 12:08:48	1.1	Jenny	Yes		hummingbird_jenny37.bag	Willie	Levi
6/11/2015 12:10:09	1.1	Jenny	Yes		hummingbird_jenny38.bag	Willie	Levi
6/11/2015 12:11:40	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering and while testing motors.	hummingbird_jenny39.bag	Willie	Levi
6/11/2015 12:13:33	1.1	Jenny	Yes	Lost Vicon while performing +y and rotate to 0.0. Zero current on sensor 1 and voltage spike on sensor 2 recorded while maneuvering.	hummingbird_jenny40.bag	Willie	Levi
6/11/2015 12:14:49	1.1	Jenny	Yes		hummingbird_jenny41.bag	Willie	Levi
6/11/2015 12:16:34	1.1	Jenny	Yes		hummingbird_jenny42.bag	Willie	Levi
6/11/2015 12:17:53	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering. Voltage spike on sensor 2 recorded while launching.	hummingbird_jenny43.bag	Willie	Levi
6/11/2015 12:19:47	1.1	Jenny	Yes	Landed during second rotation, then flew again to third rotation. Lost Vicon while performing rotate to pi/2. Did not receive rotate to 3*pi/2 move waypose. Current and voltage spikes on sensor 1 and low voltage on sensor 2 recorded while maneuvering.	hummingbird_jenny44.bag	Willie	Levi
6/11/2015 12:21:09	1.1	Jenny	Yes	Current spikes on sensor 1 and voltage spikes on sensor 2 recorded while launching.	hummingbird_jenny45.bag	Willie	Levi
6/11/2015 12:22:50	1.1	Jenny	Yes	Lost Vicon while performing rotate to pi/2 maneuver. Prop hit power cord during motors test.	hummingbird_jenny46.bag	Willie	Levi
6/11/2015 12:24:12	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded after turning off motors.	hummingbird_jenny47.bag	Willie	Levi
6/11/2015 12:25:31	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching.	hummingbird_jenny48.bag	Willie	Levi
6/11/2015 12:32:18	1.1	Jenny	Yes	Right motor stalled during motors on, Hit power cord on motor's test	hummingbird_jenny49.bag	Willie	Levi
6/11/2015 12:34:12	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering.	hummingbird_jenny50.bag	Willie	Levi
6/11/2015 12:35:37	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded after turning off motors.	hummingbird_jenny51.bag	Willie	Levi
6/11/2015 12:38:09	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded while launching. Hit power cord on motors test.	hummingbird_jenny52.bag	Willie	Levi
6/11/2015 12:38:56	1.1	Jenny	Yes		hummingbird_jenny53.bag	Willie	Levi
6/11/2015 12:40:26	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching. Voltage spike on sensor 2 recorded while maneuvering. Current spike on sensor 1 recorded after turning off motors.	hummingbird_jenny54.bag	Willie	Levi
6/11/2015 12:44:29	1.1	Jenny	No	Lost control during multiple position change, manual control taken. Lost Vicon while performing +z, +x,+y, and rotate to pi/2 maneuver. Current spike on both sensors prior to taking manual control	hummingbird_jenny55.bag	Willie	Levi
6/11/2015 12:46:14	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while turning on motors.	hummingbird_jenny56.bag	Willie	Levi
6/11/2015 12:46:49	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded before turning on motors.	hummingbird_jenny57.bag	Willie	Levi
6/11/2015 12:49:18	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching. Voltage spike on sensor 2 recorded before turning on motors.	hummingbird_jenny58.bag	Willie	Levi
6/11/2015 12:50:42	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded while testing motors.	hummingbird_jenny59.bag	Willie	Levi
6/11/2015 12:53:05	1.1	Jenny	Yes	Did not preform first multiple position change. Did not receive move waypose for +x and +y position change. Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching.	hummingbird_jenny60.bag	Willie	Levi
6/11/2015 12:54:10	1.1	Jenny	Yes	Lost task waypose of -x. Lost Vicon while performing +x position change and rotate to pi/2. Did not receive move waypose for -x position change and rotate to 0.0. Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching.	hummingbird_jenny61.bag	Willie	Levi
6/11/2015 12:55:50	1.1	Jenny	No	Lost VICON, took manual control (seizure).	hummingbird_jenny62.bag	Willie	Levi
6/11/2015 12:58:09	1.1	Jenny	Yes		hummingbird_jenny63.bag	Willie	Levi
6/11/2015 13:00:20	1.1	Jenny	Yes	Lost control during height change. Lost Vicon while performing +z and rotate to pi maneuver. Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering at different times.	hummingbird_jenny64.bag	Willie	Levi
6/11/2015 13:01:46	1.1	Jenny	No	Battery cord came loose and hit props, manual control taken.	hummingbird_jenny65.bag	Willie	Levi

Table 9: Failure Testing Response Form (Responses) Continued

6/11/2015 13:03:19	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded while maneuvering.	hummingbird_jenny66.bag	Willie	Levi
6/11/2015 13:05:36	1.1	Jenny	Yes	Current spike on sensor 1, low voltage on sensor 1, and voltage spike on sensor 2 recorded while testing motors.	hummingbird_jenny67.bag	Willie	Levi
6/11/2015 13:17:49	1.1	Jenny	Yes	Voltage spikes on sensor 2 recorded while turning off motors, while launching, and before turning on motors. One time stamp delay before performing -z position change.	hummingbird_jenny68.bag	Willie	Levi
6/11/2015 13:19:00	1.1	Jenny	Yes		hummingbird_jenny69.bag	Willie	Levi
6/11/2015 13:20:31	1.1	Jenny	Yes	Current spikes on sensor 1 and voltage spikes on sensor 2 recorded while turning off motors and while maneuvering.	hummingbird_jenny70.bag	Willie	Levi
6/11/2015 13:21:57	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded while testing motors.	hummingbird_jenny71.bag	Willie	Levi
6/11/2015 13:23:41	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching.	hummingbird_jenny72.bag	Willie	Levi
6/11/2015 13:26:26	1.1	Jenny	Yes		hummingbird_jenny73.bag	Willie	Levi
6/11/2015 13:27:49	1.1	Jenny	Yes	Voltage spikes on sensor 2 recorded while launching.	hummingbird_jenny74.bag	Willie	Levi
6/11/2015 13:29:27	1.1	Jenny	Yes		hummingbird_jenny75.bag	Willie	Levi
6/11/2015 13:30:59	1.1	Jenny	Yes		hummingbird_jenny76.bag	Willie	Levi
6/11/2015 13:33:02	1.1	Jenny	Yes	Lost task waypose on third rotation, and hit power cord on landing. Received an extra move waypose for third rotation.	hummingbird_jenny77.bag	Willie	Levi
6/11/2015 13:34:27	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spikes on sensor 2 recorded while maneuvering. Current spike on sensor 1 after turning off motors	hummingbird_jenny78.bag	Willie	Levi
6/11/2015 13:35:51	1.1	Jenny	Yes		hummingbird_jenny79.bag	Willie	Levi
6/11/2015 13:37:33	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded after turning off motors.	hummingbird_jenny80.bag	Willie	Levi
6/11/2015 13:39:02	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded while launching.	hummingbird_jenny81.bag	Willie	Levi
6/11/2015 13:40:36	1.1	Jenny	Yes		hummingbird_jenny82.bag	Willie	Levi
6/11/2015 13:42:25	1.1	Jenny	Yes	Lost task waypose on -y change. Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering. Current spike on sensor 1 while testing motors.	hummingbird_jenny83.bag	Willie	Levi
6/11/2015 13:43:48	1.1	Jenny	Yes	Voltage spikes on sensor 2 recorded while launching.	hummingbird_jenny84.bag	Willie	Levi
6/11/2015 13:45:16	1.1	Jenny	Yes	Current spikes on sensor 1, low voltages on sensor 1, and voltage spikes on sensor 2 recorded before testing motors.	hummingbird_jenny85.bag	Willie	Levi
6/11/2015 13:46:49	1.1	Jenny	Yes		hummingbird_jenny86.bag	Willie	Levi
6/11/2015 13:48:13	1.1	Jenny	Yes	Lost control prior to landing, lost Vicon. Current spike on sensor 1 while landing. Voltage spike on sensor 2 recorded while maneuvering.	hummingbird_jenny87.bag	Willie	Levi
6/11/2015 13:51:06	1.1	Jenny	Yes	Lost serial connection before first multiple position change. Voltage spike on sensor 2 recorded while turning on motors. Zero current on both sensors while serial communication lost.	hummingbird_jenny88.bag	Willie	Levi
6/11/2015 13:51:53	1.1	Jenny	Yes	Current spike on sensor 1, low voltage on sensor 1, and voltage spike on sensor 2 recorded while testing motors.	hummingbird_jenny89.bag	Willie	Levi
6/11/2015 13:58:38	1.1	Jenny	Yes		hummingbird_jenny90.bag	Willie	Levi
6/11/2015 13:58:54	1.1	Jenny	Yes		hummingbird_jenny91.bag	Willie	Levi
6/11/2015 13:59:08	1.1	Jenny	Yes		hummingbird_jenny92.bag	Willie	Levi
6/11/2015 13:59:22	1.1	Jenny	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while launching. Voltage spike on sensor 2 while maneuvering.	hummingbird_jenny93.bag	Willie	Levi
6/11/2015 13:59:49	1.1	Jenny	Yes	Lost control on -x change. Voltage spike on sensor 2 while maneuvering. Current spike on sensor 1 and voltage spike on sensor 2 recorded while landing.	hummingbird_jenny94.bag	Willie	Levi
6/11/2015 14:01:13	1.1	Jenny	Yes		hummingbird_jenny95.bag	Willie	Levi
6/11/2015 14:02:33	1.1	Jenny	Yes		hummingbird_jenny96.bag	Willie	Levi
6/11/2015 14:04:12	1.1	Jenny	Yes		hummingbird_jenny97.bag	Willie	Levi
6/11/2015 14:05:16	1.1	Jenny	Yes	Voltage spike on sensor 2 recorded after turning off motors.	hummingbird_jenny98.bag	Willie	Levi
6/11/2015 14:06:52	1.1	Jenny	Yes		hummingbird_jenny99.bag	Willie	Levi
6/11/2015 14:08:32	1.1	Jenny	Yes		hummingbird_jenny100.bag	Willie	Levi
6/11/2015 14:32:27	1.1	Herbie	Yes	Lost control on second maneuver of second sequence of multiple position change. Low voltage on sensor 2 recorded before turning on motors. Lost Vicon several times during flight. Move waypose for indicated maneuver not received.	hummingbird_herbie1.bag	Willie	Levi
6/11/2015 14:35:40	1.1	Herbie	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering. Lost Vicon several times during flight.	hummingbird_herbie2.bag	Willie	Levi
6/11/2015 14:36:49	1.1	Herbie	Yes	Lost Vicon several times during flight.	hummingbird_herbie3.bag	Willie	Levi
6/11/2015 14:38:52	1.1	Herbie	Yes	Right before -y change, lost connection. Lost Vicon several times during flight. Voltage spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering at different times.	hummingbird_herbie4.bag	Willie	Levi
6/11/2015 14:41:02	1.1	Herbie	Yes	Loss of VICON several times during flight. Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering. Voltage spike on sensor 2 while maneuvering. Move Task waypose for rotate to 3*pi/2 not received.	hummingbird_herbie5.bag	Willie	Levi
6/11/2015 14:45:16	1.1	Herbie	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering.	hummingbird_herbie6.bag	Willie	Levi
6/11/2015 14:46:56	1.1	Herbie	Yes	Loss of control after -x change. Voltage spikes on sensor 2 recorded while maneuvering. Lost Vicon twice the first loss resulted in task failure after -x position change and rotate to 0.0.	hummingbird_herbie7.bag	Willie	Levi
6/11/2015 14:48:40	1.1	Herbie	Yes	Loss of control during rotation change. Loss of VICON. Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering.	hummingbird_herbie8.bag	Willie	Levi
6/11/2015 14:50:16	1.1	Herbie	Yes	Loss of VICON while testing motors.	hummingbird_herbie9.bag	Willie	Levi
6/11/2015 14:52:45	1.1	Herbie	Yes	Loss of VICON while testing motors.	hummingbird_herbie10.bag	Willie	Levi
6/17/2015 9:48:37	1.1	Hulk Smash	Yes		hummingbird_hulk_smash1.bag	Willie	N/A
6/17/2015 9:49:34	1.1	Hulk Smash	Yes	Front right motor stalled during motors on. Voltage spike on sensor 2 recorded while testing motors.	hummingbird_hulk_smash2.bag	Willie	Levi
6/17/2015 9:50:46	1.1	Hulk Smash	No	Front motors stalled during motors on, drone nearly flipped on take off, manual control taken.	hummingbird_hulk_smash3.bag	Willie	Levi
6/17/2015 10:12:31	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci1.bag	Willie	Levi
6/17/2015 10:12:49	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci2.bag	Willie	Levi
6/17/2015 10:15:52	1.1	UNL_COMP_SCI	Yes	Voltage spike on sensor 1 recorded while maneuvering, abnormal path.	hummingbird_unl_comp_sci3.bag	Willie	Levi
6/17/2015 10:16:54	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci4.bag	Willie	Levi
6/17/2015 10:17:22	1.1	UNL_COMP_SCI	Yes	Current spike on sensor 1, senosr 2 dropped to zero voltage while turning on motors.	hummingbird_unl_comp_sci5.bag	Willie	Levi
6/17/2015 10:19:13	1.1	UNL_COMP_SCI	Yes	Voltage spike on sensor 2 recorded while launching.	hummingbird_unl_comp_sci6.bag	Willie	Levi
6/17/2015 10:20:18	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci7.bag	Willie	Levi
6/17/2015 10:22:47	1.1	UNL_COMP_SCI	Yes	Voltage spike on sensor 2 recorded while maneuvering. Lost Vicon but completed all tasks.	hummingbird_unl_comp_sci8.bag	Willie	Levi
6/17/2015 10:25:03	1.1	UNL_COMP_SCI	Yes	Current spike on sensor 1 and voltage spike on sensor 2 recorded while maneuvering.	hummingbird_unl_comp_sci9.bag	Willie	Levi
6/17/2015 10:26:08	1.1	UNL_COMP_SCI	No	Prop caught got in power cord during motors test, drone flipped on take off, manual control taken.	hummingbird_unl_comp_sci10.bag	Willie	Levi
6/17/2015 10:42:48	1.1	UNL_COMP_SCI	Yes	Voltage spike on sensor 2 recorded before turning on motors. Lost Vicon but completed all tasks.	hummingbird_unl_comp_sci11.bag	Willie	Levi
6/17/2015 10:43:10	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci12.bag	Willie	Levi
6/17/2015 10:44:05	1.1	UNL_COMP_SCI	Yes	Lost Vicon but completed all tasks.	hummingbird_unl_comp_sci13.bag	Willie	Levi
6/17/2015 10:45:41	1.1	UNL_COMP_SCI	Yes	Lost Vicon but completed all tasks.	hummingbird_unl_comp_sci14.bag	Willie	Levi
6/17/2015 10:47:11	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci15.bag	Willie	Levi
6/17/2015 10:48:43	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci16.bag	Willie	Levi
6/17/2015 10:50:24	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci17.bag	Willie	Levi
6/17/2015 10:51:49	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci18.bag	Willie	Levi
6/17/2015 10:53:25	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci19.bag	Willie	Levi
6/17/2015 10:55:11	1.1	UNL_COMP_SCI	Yes	Small voltage spike sensor 2.	hummingbird_unl_comp_sci20.bag	Willie	Levi
6/17/2015 10:57:07	1.1	UNL_COMP_SCI	Yes	Loss of VICON.	hummingbird_unl_comp_sci21.bag	Willie	Levi
6/17/2015 10:58:36	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci22.bag	Willie	Levi
6/17/2015 11:00:03	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci23.bag	Willie	Levi
6/17/2015 11:03:03	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci24.bag	Willie	Levi
6/17/2015 11:04:34	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci25.bag	Willie	Levi
6/17/2015 11:06:30	1.1	UNL_COMP_SCI	Yes	Loss of VICON.	hummingbird_unl_comp_sci26.bag	Willie	Levi

Table 10: Failure Testing Response Form (Responses) Continued

6/17/2015 11:07:54	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci27.bag	Willie	Levi
6/17/2015 11:08:53	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci28.bag	Willie	Levi
6/17/2015 11:11:07	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci29.bag	Willie	Levi
6/17/2015 11:12:45	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2.	hummingbird_unl_comp_sci30.bag	Willie	Levi
6/17/2015 11:15:19	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2.	hummingbird_unl_comp_sci31.bag	Willie	Levi
6/17/2015 11:16:25	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci32.bag	Willie	Levi
6/17/2015 11:18:27	1.1	UNL_COMP_SCI	Yes	HUGE spike in sensor 2 voltage and HUGE spike in sensor 1 current.	hummingbird_unl_comp_sci33.bag	Willie	Levi
6/17/2015 11:20:26	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2.	hummingbird_unl_comp_sci34.bag	Willie	Levi
6/17/2015 11:22:48	1.1	UNL_COMP_SCI	Yes	Loss of VICON.	hummingbird_unl_comp_sci35.bag	Willie	Levi
6/17/2015 11:25:42	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2.	hummingbird_unl_comp_sci36.bag	Willie	Levi
6/17/2015 11:27:50	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci37.bag	Willie	Levi
6/17/2015 11:29:06	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci38.bag	Willie	Levi
6/17/2015 11:30:37	1.1	UNL_COMP_SCI	Yes	Momentary loss of VICON.	hummingbird_unl_comp_sci39.bag	Willie	Levi
6/17/2015 11:32:36	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci40.bag	Willie	Levi
6/17/2015 11:34:15	1.1	UNL_COMP_SCI	No	HUGE spike in sensor 2 voltage.	hummingbird_unl_comp_sci41.bag	Willie	Levi
6/17/2015 11:36:27	1.1	UNL_COMP_SCI	Yes	Loss of VICON.	hummingbird_unl_comp_sci42.bag	Willie	Levi
6/17/2015 11:44:39	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci43.bag	Willie	Levi
6/17/2015 11:47:31	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci44.bag	Willie	Levi
6/17/2015 11:47:47	1.1	UNL_COMP_SCI	Yes	Zero voltage sensor 2.	hummingbird_unl_comp_sci45.bag	Willie	Levi
6/17/2015 11:48:46	1.1	UNL_COMP_SCI	Yes	Low voltage sensor 1, current spike sensor 1.	hummingbird_unl_comp_sci46.bag	Willie	Levi
6/17/2015 11:50:42	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci47.bag	Willie	Levi
6/17/2015 11:52:20	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci48.bag	Willie	Levi
6/17/2015 11:53:15	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci49.bag	Willie	Levi
6/17/2015 11:54:45	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci50.bag	Willie	Levi
6/17/2015 11:57:15	1.1	UNL_COMP_SCI	Yes	Low voltage sensor 1, current spike sensor 1.	hummingbird_unl_comp_sci51.bag	Willie	Levi
6/17/2015 11:57:56	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci52.bag	Willie	Levi
6/17/2015 11:59:15	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci53.bag	Willie	Levi
6/17/2015 12:00:13	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci54.bag	Willie	Levi
6/17/2015 12:01:13	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci55.bag	Willie	Levi
6/17/2015 12:07:32	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci56.bag	Willie	Levi
6/17/2015 12:08:54	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci57.bag	Willie	Levi
6/17/2015 12:10:26	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci58.bag	Willie	Levi
6/17/2015 12:11:51	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2.	hummingbird_unl_comp_sci59.bag	Willie	Levi
6/17/2015 12:13:17	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci60.bag	Willie	Levi
6/17/2015 12:14:56	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci61.bag	Willie	Levi
6/17/2015 12:16:33	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2.	hummingbird_unl_comp_sci62.bag	Willie	Levi
6/17/2015 12:17:37	1.1	UNL_COMP_SCI	Yes	Loss of VICON.	hummingbird_unl_comp_sci63.bag	Willie	Levi
6/17/2015 12:19:25	1.1	UNL_COMP_SCI	Yes	Loss of VICON.	hummingbird_unl_comp_sci64.bag	Willie	Levi
6/17/2015 12:20:47	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci65.bag	Willie	Levi
6/17/2015 12:22:15	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci66.bag	Willie	Levi
6/17/2015 12:23:43	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci67.bag	Willie	Levi
6/17/2015 12:25:26	1.1	UNL_COMP_SCI	Yes	Voltage spike.	hummingbird_unl_comp_sci68.bag	Willlie	Levi
6/17/2015 12:27:05	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci69.bag	Willie	Levi
6/17/2015 12:28:13	1.1	UNL_COMP_SCI	Yes	Loss of VICON.	hummingbird_unl_comp_sci70.bag	Willie	Levi
6/17/2015 12:30:37	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2, low voltage sensor 1, current spike sensor 1.	hummingbird_unl_comp_sci71.bag	Willie	Levi
6/17/2015 12:31:11	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2.	hummingbird_unl_comp_sci72.bag	Willie	Levi
6/17/2015 12:32:48	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci73.bag	Willie	Levi
6/17/2015 12:34:10	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2, HUGE current spike sensor 1.	hummingbird_unl_comp_sci74.bag	Willie	Levi
6/17/2015 12:35:49	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci75.bag	Willie	Levi
6/17/2015 12:37:14	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci76.bag	Willie	Levi
6/17/2015 12:38:45	1.1	UNL_COMP_SCI	Yes	Loss of VICON, voltage spike sensor 2.	hummingbird_unl_comp_sci77.bag	Willie	Levi
6/17/2015 12:40:04	1.1	UNL_COMP_SCI	Yes	Loss of VICON	hummingbird_unl_comp_sci78.bag	Willie	Levi
6/17/2015 12:41:30	1.1	UNL_COMP_SCI	Yes	Loss of VICON	hummingbird_unl_comp_sci79.bag	Willie	Levi
6/17/2015 12:42:57	1.1	UNL_COMP_SCI	Yes	Loss of VICON momentarily before motors test, voltage spike senosr 2, current spike sensor 1.	hummingbird_unl_comp_sci80.bag	Willie	Levi
6/17/2015 12:44:22	1.1	UNL_COMP_SCI	Yes	Loss of VICON @ very beginning of test.	hummingbird_unl_comp_sci81.bag	Willie	Levi
6/17/2015 12:46:23	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci82.bag	Willie	Levi
6/17/2015 12:48:26	1.1	UNL_COMP_SCI	Yes	Loss of VICON.	hummingbird_unl_comp_sci83.bag	Willie	Levi
6/17/2015 12:51:19	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci84.bag	Willie	Levi
6/17/2015 12:53:15	1.1	UNL_COMP_SCI	Yes	End of test: current spike sensor 1, zero voltage sensor 2.	hummingbird_unl_comp_sci85.bag	Willie	Levi
6/17/2015 12:55:06	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, zero voltage sensor 2, zero voltage sensor 1, current spike sensor 1.	hummingbird_unl_comp_sci86.bag	Willie	Levi
6/17/2015 12:58:19	1.1	UNL_COMP_SCI	No	Power cable got tangled around back prop strut and it flew for sometime, rather poorly, then lost control, flipped and flew way out of bounds pulling off power cable.	hummingbird_unl_comp_sci87.bag	Willie	Levi
6/17/2015 13:02:30	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2.	hummingbird_unl_comp_sci88.bag	Willie	Levi
6/17/2015 13:07:01	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, zero voltage sensor 1, current spike sensor 1.	hummingbird_unl_comp_sci89.bag	Willie	Levi
6/17/2015 13:07:22	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci90.bag	Willie	Levi
6/17/2015 13:07:37	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci91.bag	Willie	Levi
6/17/2015 13:08:30	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci92.bag	Willie	Levi
6/17/2015 13:10:18	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci93.bag	Willie	Levi
6/17/2015 13:12:44	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, zero voltage sensor 1, current spike sensor 1.	hummingbird_unl_comp_sci94.bag	Willie	Levi
6/17/2015 13:14:12	1.1	UNL_COMP_SCI	Yes	Zero voltage sensor 2.	hummingbird_unl_comp_sci95.bag	Willie	Levi
6/17/2015 13:15:52	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2.	hummingbird_unl_comp_sci96.bag	Willie	Levi
6/17/2015 13:17:22	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci97.bag	Willie	Levi
6/17/2015 13:18:49	1.1	UNL_COMP_SCI	Yes	Voltage spike sensor 2, current spike sensor 1.	hummingbird_unl_comp_sci98.bag	Willie	Levi
6/17/2015 13:20:36	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci99.bag	Willie	Levi
6/17/2015 13:22:52	1.1	UNL_COMP_SCI	Yes		hummingbird_unl_comp_sci100.bag	Willie	Levi
7/2/2015 16:06:30	1.2	Herbie	Yes	5 losses of VICON.	hummingbird_herbie21.bag	Willie	Levi
7/2/2015 16:11:10	1.2	Herbie	Yes		hummingbird_herbie22.bag	Willie	Levi
7/2/2015 16:13:12	1.2	Herbie	Yes		hummingbird_herbie23.bag	Willie	Levi
7/2/2015 16:14:25	1.2	Herbie	Yes	Loss of VICON	hummingbird_herbie24.bag	Willie	Levi
7/2/2015 16:16:14	1.2	Herbie	Yes		hummingbird_herbie25.bag	Willie	Levi
7/2/2015 16:18:40	1.2	Herbie	Yes		hummingbird_herbie26.bag	Willie	Levi
7/2/2015 16:20:37	1.2	Herbie	Yes		hummingbird_herbie27.bag	Willie	Levi
7/2/2015 16:25:53	1.2	Herbie	Yes		hummingbird_herbie28.bag	Willie	Levi
7/2/2015 16:26:12	1.2	Herbie	Yes		hummingbird_herbie29.bag	Willie	Levi
7/2/2015 16:28:33	1.2	Herbie	Yes		hummingbird_herbie30.bag	Willie	Levi
7/2/2015 16:28:49	1.2	Herbie	Yes		hummingbird_herbie31.bag	Willie	Levi
7/2/2015 16:29:17	1.2	Herbie	Yes		hummingbird_herbie32.bag	Willie	Levi
7/2/2015 16:30:51	1.2	Herbie	Yes		hummingbird_herbie33.bag	Willie	Levi
7/2/2015 16:34:02	1.2	Herbie	Yes		hummingbird_herbie34.bag	Willie	Levi
7/2/2015 16:35:52	1.2	Herbie	Yes		hummingbird_herbie35.bag	Willie	Levi
7/2/2015 16:39:13	1.2	Herbie	Yes		hummingbird_herbie36.bag	Willie	Levi
7/2/2015 16:40:56	1.2	Herbie	Yes		hummingbird_herbie37.bag	Willie	Levi
7/2/2015 16:42:15	1.2	Herbie	Yes		hummingbird_herbie38.bag	Willie	Levi
7/2/2015 16:45:35	1.2	Herbie	Yes		hummingbird_herbie39.bag	Willie	Levi
7/2/2015 16:47:10	1.2	Herbie	Yes		hummingbird_herbie40.bag	Willie	Levi
7/2/2015 16:48:50	1.2	Herbie	Yes		hummingbird_herbie41.bag	Willie	Levi
7/2/2015 16:50:15	1.2	Herbie	Yes		hummingbird_herbie42.bag	Willie	Levi
7/2/2015 16:53:20	1.2	Herbie	Yes		hummingbird_herbie43.bag	Willie	Levi
7/2/2015 16:54:59	1.2	Herbie	Yes		hummingbird_herbie44.bag	Willie	Levi
7/2/2015 16:56:33	1.2	Herbie	Yes		hummingbird_herbie45.bag	Willie	Levi
7/2/2015 16:58:00	1.2	Herbie	Yes		hummingbird_herbie46.bag	Willie	Levi
7/2/2015 17:00:14	1.2	Herbie	Yes		hummingbird_herbie47.bag	Willie	Levi
7/2/2015 17:00:54	1.2	Herbie	Yes		hummingbird_herbie48.bag	Willie	Levi
7/2/2015 17:02:20	1.2	Herbie	Yes		hummingbird_herbie49.bag	Willie	Levi
7/2/2015 17:03:44	1.2	Herbie	Yes		hummingbird_herbie50.bag	Willie	Levi
7/15/2015 11:17:19	1.2	Ar.Drone	Yes	Two back motors replaced, damaged while tuning PID controller.	ardrone_morpheus1.bag	Willie	Levi
7/15/2015 11:17:36	1.2	Ar.Drone	Yes		ardrone_morpheus2.bag	Willie	Levi
7/15/2015 11:19:15	1.2	Ar.Drone	Yes		ardrone_morpheus3.bag	Willie	Levi
7/15/2015 11:26:55	1.2	Ar.Drone	Yes		ardrone_morpheus4.bag	Willie	Levi
7/15/2015 11:27:14	1.2	Ar.Drone	Yes		ardrone_morpheus5.bag	Willie	Levi
7/15/2015 11:27:36	1.2	Ar.Drone	Yes		ardrone_morpheus6.bag	Willie	Levi
7/15/2015 11:29:45	1.2	Ar.Drone	Yes		ardrone_morpheus7.bag	Willie	Levi
7/15/2015 11:31:15	1.2	Ar.Drone	Yes		ardrone_morpheus8.bag	Willie	Levi

Table 11: Failure Testing Response Form (Responses) Continued

7/15/2015 11:32:55	1.2	Ar.Drone	Yes	ardrone_morpheus9.bag	Willie	Levi
7/15/2015 11:34:32	1.2	Ar.Drone	Yes	ardrone_morpheus10.bag	Willie	Levi
7/15/2015 11:36:06	1.2	Ar.Drone	Yes	ardrone_morpheus11.bag	Willie	Levi
7/15/2015 11:38:02	1.2	Ar.Drone	Yes	ardrone_morpheus12.bag	Willie	Levi
7/15/2015 11:39:27	1.2	Ar.Drone	Yes	ardrone_morpheus13.bag	Willie	Levi
7/15/2015 11:41:04	1.2	Ar.Drone	Yes	ardrone_morpheus14.bag	Willie	Levi
7/15/2015 11:42:31	1.2	Ar.Drone	Yes	ardrone_morpheus15.bag	Willie	Levi
7/15/2015 11:44:07	1.2	Ar.Drone	Yes	ardrone_morpheus16.bag	Willie	Levi
7/15/2015 11:46:10	1.2	Ar.Drone	Yes	ardrone_morpheus17.bag	Willie	Levi
7/15/2015 11:47:47	1.2	Ar.Drone	Yes	ardrone_morpheus18.bag	Willie	Levi
7/15/2015 11:49:27	1.2	Ar.Drone	Yes	ardrone_morpheus19.bag	Willie	Levi
7/15/2015 11:51:13	1.2	Ar.Drone	Yes	ardrone_morpheus20.bag	Willie	Levi
7/15/2015 11:53:02	1.2	Ar.Drone	Yes	ardrone_morpheus21.bag	Willie	Levi
7/15/2015 11:54:51	1.2	Ar.Drone	Yes	ardrone_morpheus22.bag	Willie	Levi
7/15/2015 11:56:23	1.2	Ar.Drone	Yes	ardrone_morpheus23.bag	Willie	Levi
7/15/2015 11:58:03	1.2	Ar.Drone	Yes	ardrone_morpheus24.bag	Willie	Levi
7/15/2015 11:59:41	1.2	Ar.Drone	Yes	ardrone_morpheus25.bag	Willie	Levi
7/15/2015 12:01:19	1.2	Ar.Drone	Yes	ardrone_morpheus26.bag	Willie	Levi
7/15/2015 12:02:57	1.2	Ar.Drone	Yes	ardrone_morpheus27.bag	Willie	Levi
7/15/2015 12:04:50	1.2	Ar.Drone	Yes	ardrone_morpheus28.bag	Willie	Levi
7/15/2015 12:06:29	1.2	Ar.Drone	Yes	ardrone_morpheus29.bag	Willie	Levi
7/15/2015 12:08:21	1.2	Ar.Drone	Yes	ardrone_morpheus30.bag	Willie	Levi
7/15/2015 12:09:46	1.2	Ar.Drone	Yes	ardrone_morpheus31.bag	Willie	Levi
7/15/2015 12:11:25	1.2	Ar.Drone	Yes	ardrone_morpheus32.bag	Willie	Levi
7/15/2015 12:18:16	1.2	Ar.Drone	Yes	ardrone_morpheus33.bag	Willie	Levi
7/15/2015 12:18:35	1.2	Ar.Drone	Yes	ardrone_morpheus34.bag	Willie	Levi
7/15/2015 12:18:50	1.2	Ar.Drone	Yes	ardrone_morpheus35.bag	Willie	Levi
7/15/2015 12:19:28	1.2	Ar.Drone	Yes	ardrone_morpheus36.bag	Willie	Levi
7/15/2015 12:20:54	1.2	Ar.Drone	Yes	ardrone_morpheus37.bag	Willie	Levi
7/15/2015 12:22:26	1.2	Ar.Drone	Yes	ardrone_morpheus38.bag	Willie	Levi
7/15/2015 12:24:01	1.2	Ar.Drone	No	ardrone_morpheus39.bag	Willie	Levi
7/15/2015 12:48:26	1.2	Ar.Drone	Yes	ardrone_morpheus40.bag	Willie	Levi
7/15/2015 12:50:11	1.2	Ar.Drone	Yes	ardrone_morpheus41.bag	Willie	Levi
7/15/2015 12:51:48	1.2	Ar.Drone	Yes	ardrone_morpheus42.bag	Willie	Levi
7/15/2015 12:53:26	1.2	Ar.Drone	Yes	ardrone_morpheus43.bag	Willie	Levi
7/15/2015 12:55:02	1.2	Ar.Drone	Yes	ardrone_morpheus44.bag	Willie	Levi
7/15/2015 12:56:40	1.2	Ar.Drone	Yes	ardrone_morpheus45.bag	Willie	Levi
7/15/2015 12:58:22	1.2	Ar.Drone	Yes	ardrone_morpheus46.bag	Willie	Levi
7/15/2015 12:59:57	1.2	Ar.Drone	Yes	ardrone_morpheus47.bag	Willie	Levi
7/15/2015 13:01:39	1.2	Ar.Drone	Yes	ardrone_morpheus48.bag	Willie	Levi
7/15/2015 13:03:11	1.2	Ar.Drone	Yes	ardrone_morpheus49.bag	Willie	Levi
7/15/2015 13:05:21	1.2	Ar.Drone	Yes	ardrone_morpheus50.bag	Willie	Levi
7/15/2015 13:07:28	1.2	Ar.Drone	Yes	ardrone_morpheus51.bag	Willie	Levi
7/15/2015 13:09:29	1.2	Ar.Drone	Yes	ardrone_morpheus52.bag	Willie	Levi
7/15/2015 13:11:14	1.2	Ar.Drone	Yes	ardrone_morpheus53.bag	Willie	Levi
7/15/2015 13:12:51	1.2	Ar.Drone	Yes	ardrone_morpheus54.bag	Willie	Levi
7/15/2015 13:14:32	1.2	Ar.Drone	Yes	ardrone_morpheus55.bag	Willie	Levi
7/15/2015 13:16:14	1.2	Ar.Drone	Yes	ardrone_morpheus56.bag	Willie	Levi
7/15/2015 13:17:59	1.2	Ar.Drone	Yes	ardrone_morpheus57.bag	Willie	Levi
7/15/2015 13:19:40	1.2	Ar.Drone	Yes	ardrone_morpheus58.bag	Willie	Levi
7/15/2015 13:21:15	1.2	Ar.Drone	Yes	ardrone_morpheus59.bag	Willie	Levi

Table 12: Failure Testing Response Form (Responses) Continued

7/15/2015 13:23:05	1.2	Ar.Drone	Yes	ardrone_morpheus60.bag	Willie	Levi
7/15/2015 13:25:21	1.2	Ar.Drone	Yes	ardrone_morpheus61.bag	Willie	Levi
7/15/2015 13:27:04	1.2	Ar.Drone	Yes	ardrone_morpheus62.bag	Willie	Levi
7/15/2015 13:28:50	1.2	Ar.Drone	Yes	ardrone_morpheus63.bag	Willie	Levi
7/15/2015 13:30:54	1.2	Ar.Drone	Yes	ardrone_morpheus64.bag	Willie	Levi
7/15/2015 13:32:27	1.2	Ar.Drone	Yes	ardrone_morpheus65.bag	Willie	Levi
7/15/2015 13:34:09	1.2	Ar.Drone	Yes	ardrone_morpheus66.bag	Willie	Levi
7/15/2015 13:35:55	1.2	Ar.Drone	Yes	ardrone_morpheus67.bag	Willie	Levi
7/15/2015 13:38:22	1.2	Ar.Drone	Yes	ardrone_morpheus68.bag	Willie	Levi
7/15/2015 13:40:03	1.2	Ar.Drone	Yes	ardrone_morpheus69.bag	Willie	Levi
7/15/2015 13:44:20	1.2	Ar.Drone	Yes	ardrone_morpheus70.bag	Willie	Levi
7/16/2015 13:44:22	1.2	Ar.Drone	Yes	ardrone_morpheus71.bag	Willie	Levi
7/15/2015 13:44:36	1.2	Ar.Drone	Yes	ardrone_morpheus72.bag	Willie	Levi
7/15/2015 13:46:12	1.2	Ar.Drone	Yes	ardrone_morpheus73.bag	Willie	Levi
7/15/2015 13:47:48	1.2	Ar.Drone	Yes	ardrone_morpheus74.bag	Willie	Levi
7/15/2015 13:51:01	1.2	Ar.Drone	Yes	ardrone_morpheus75.bag	Willie	Levi
7/15/2015 13:54:15	1.2	Ar.Drone	Yes	ardrone_morpheus76.bag	Willie	Levi
7/15/2015 13:55:52	1.2	Ar.Drone	Yes	ardrone_morpheus77.bag	Willie	Levi
7/15/2015 13:58:00	1.2	Ar.Drone	Yes	ardrone_morpheus78.bag	Willie	Levi
7/15/2015 13:59:47	1.2	Ar.Drone	Yes	ardrone_morpheus79.bag	Willie	Levi
7/15/2015 14:01:29	1.2	Ar.Drone	Yes	ardrone_morpheus80.bag	Willie	Levi
7/15/2015 14:05:55	1.2	Ar.Drone	Yes	ardrone_morpheus81.bag	Willie	Levi
7/15/2015 14:08:42	1.2	Ar.Drone	Yes	ardrone_morpheus82.bag	Willie	Levi
7/15/2015 14:10:41	1.2	Ar.Drone	No	ardrone_morpheus83.bag	Willie	Levi
7/15/2015 14:16:30	1.2	Ar.Drone	No	ardrone_morpheus84.bag	Willie	Levi
7/15/2015 14:56:30	1.2	Ar.Drone	Yes	ardrone_morpheus85.bag	Willie	Levi
7/15/2015 14:58:06	1.2	Ar.Drone	Yes	ardrone_morpheus86.bag	Willie	Levi
7/15/2015 14:59:55	1.2	Ar.Drone	No	ardrone_morpheus87.bag	Willie	Levi
7/15/2015 15:06:18	1.2	Ar.Drone	No	ardrone_morpheus88.bag	Willie	Levi
7/15/2015 15:11:59	1.2	Ar.Drone	Yes	ardrone_morpheus89.bag	Willie	Levi
7/15/2015 15:13:29	1.2	Ar.Drone	Yes	ardrone_morpheus90.bag	Willie	Levi
7/15/2015 15:15:21	1.2	Ar.Drone	Yes	ardrone_morpheus91.bag	Willie	Levi
7/15/2015 15:22:40	1.2	Ar.Drone	No	ardrone_morpheus92.bag	Willie	Levi
7/15/2015 15:23:15	1.2	Ar.Drone	Yes	ardrone_morpheus93.bag	Willie	Levi
7/15/2015 15:25:58	1.2	Ar.Drone	No	ardrone_morpheus94.bag	Willie	Levi
7/15/2015 15:28:39	1.2	Ar.Drone	No	ardrone_morpheus95.bag	Willie	Levi
7/15/2015 15:34:34	1.2	Ar.Drone	No	ardrone_morpheus96.bag	Willie	Levi
7/15/2015 15:46:19	1.2	Ar.Drone	Yes	ardrone_morpheus97.bag	Willie	Levi
7/15/2015 15:48:03	1.2	Ar.Drone	Yes	ardrone_morpheus98.bag	Willie	Levi
7/15/2015 15:49:07	1.2	Ar.Drone	No	ardrone_morpheus99.bag	Willie	Levi
7/15/2015 15:51:06	1.2	Ar.Drone	Yes	ardrone_morpheus100.bag	Willie	Levi

```
##    * add "message_generation" and every package in MSG_DEP_SET to
##      find_package(catkin REQUIRED COMPONENTS ...)
##    * add "message_runtime" and every package in MSG_DEP_SET to
##      catkin_package(CATKIN_DEPENDS ...)
##    * uncomment the add_*_files sections below as needed
##      and list every .msg/.srv/.action file to be processed
##    * uncomment the generate_messages entry below
##    * add every package in MSG_DEP_SET to generate_messages(DEPENDENCIES ...)

## Generate messages in the 'msg' folder
add_message_files(
  DIRECTORY msg
  FILES
   Sensor.msg
   Mavlink.msg
)

## Generate services in the 'srv' folder
# add_service_files(
#   FILES
#   Service1.srv
#   Service2.srv
# )

## Generate actions in the 'action' folder
# add_action_files(
#   FILES
#   Action1.action
#   Action2.action
# )

## Generate added messages and services with any dependencies listed here
generate_messages(
   DEPENDENCIES
   collab_msgs std_msgs geometry_msgs
 )

###################################
## catkin specific configuration ##
###################################
## The catkin_package macro generates cmake config files for your package
## Declare things to be passed to dependent projects
## INCLUDE_DIRS: uncomment this if you package contains header files
## LIBRARIES: libraries you create in this project that dependent projects also need
## CATKIN_DEPENDS: catkin_packages dependent projects also need
## DEPENDS: system dependencies of this project that dependent projects also need
catkin_package(
  INCLUDE_DIRS include
#  LIBRARIES flight_testing
  CATKIN_DEPENDS collab_msgs collab_test message_runtime roscpp rospy std_msgs
#  DEPENDS system_lib
)

###########
## Build ##
###########

## Specify additional locations of header files
## Your package locations should be listed before other locations
include_directories(
  include
  include/flight_testing
  include/parse_serial
  include/communication
  #/opt/ros/jade/share/mavros
  ${catkin_INCLUDE_DIRS}
)

set(flight_testing_headers
  include/flight_testing/flight_testing.h
  include/parse_serial/parse_serial.h
  include/communication/mavlink2MIT_AscTec.h
)

set(flight_testing_srcs
  src/flight_testing/flight_testing.cpp
  src/flight_testing/flight_testing_init.cpp
  src/flight_testing/flight_testing_callback.cpp
  src/flight_testing/flight_testing_thread.cpp
)

## Declare a cpp library
set(parse_serial_srcs
   src/parse_serial/parse_serial.cpp
   src/parse_serial/parse_serial_init.cpp
   src/parse_serial/parse_serial_callback.cpp
)

set(mavlink2mit_srcs
    src/communication/mavlink2mit_asctec.cpp
    src/communication/mavlink2mit_asctec_init.cpp
```

```
    src/communication/mavlink2mit_asctec_callback.cpp
    #src/communication/mavlink2mit_asctec_thread.cpp
)

## Declare a cpp executable
 add_executable(flight_testing ${flight_testing_srcs})
 add_executable(parse_serial ${parse_serial_srcs})
 add_executable(communication ${mavlink2mit_srcs})

## Add cmake target dependencies of the executable/library
## as an example, message headers may need to be generated before nodes
add_dependencies(flight_testing collab_msgs_gencpp control_msgs_gencpp flight_testing_gencpp)
add_dependencies(communication collab_msgs_gencpp control_msgs_gencpp flight_testing_gencpp)

# roslint?
#roslint_cpp(${flight_testing_headers} ${flight_testing_srcs})

## Specify libraries to link a library or executable target against
target_link_libraries(flight_testing ${catkin_LIBRARIES})

target_link_libraries(parse_serial ${catkin_LIBRARIES})

target_link_libraries(communication ${catkin_LIBRARIES})

#############
## Install ##
#############

# all install targets should use catkin DESTINATION variables
# See http://ros.org/doc/api/catkin/html/adv_user_guide/variables.html

## Mark executable scripts (Python etc.) for installation
## in contrast to setup.py, you can choose the destination
# install(PROGRAMS
#   scripts/my_python_script
#   DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
# )

## Mark executables and/or libraries for installation
# install(TARGETS flight_testing flight_testing_node
#   ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
# )

## Mark cpp header files for installation
# install(DIRECTORY include/${PROJECT_NAME}/
#   DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
#   FILES_MATCHING PATTERN "*.h"
#   PATTERN ".svn" EXCLUDE
# )

## Mark other files for installation (e.g. launch and bag files, etc.)
# install(FILES
#   # myfile1
#   # myfile2
#   DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
# )

#############
## Testing ##
#############

## Add gtest based cpp test target and link libraries
# catkin_add_gtest(${PROJECT_NAME}-test test/test_flight_testing.cpp)
# if(TARGET ${PROJECT_NAME}-test)
#   target_link_libraries(${PROJECT_NAME}-test ${PROJECT_NAME})
# endif()

## Add folders to be run by python nosetests
# catkin_add_nosetests(test)


<?xml version="1.0"?>
<package>
  <name>flight_testing</name>
  <version>0.0.0</version>
  <description>The flight_testing package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="wwilliewells@todo.todo">wwilliewells</maintainer>


  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>BSD</license>
```

```
  <!-- Url tags are optional, but mutiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/flight_testing</url> -->


  <!-- Author tags are optional, mutiple are allowed, one per tag -->
  <!-- Authors do not have to be maintianers, but could be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->


  <!-- The *_depend tags are used to specify dependencies -->
  <!-- Dependencies can be catkin packages or system dependencies -->
  <!-- Examples: -->
  <!-- Use build_depend for packages you need at compile time: -->
  <build_depend>message_generation</build_depend>
  <!-- Use run_depend for packages you need at runtime: -->
  <run_depend>message_runtime</run_depend>
  <!-- Use test_depend for packages you need only for testing: -->
  <!--   <test_depend>gtest</test_depend> -->
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>collab_msgs</build_depend>
  <build_depend>collab_test</build_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>mavros</build_depend>

  <run_depend>collab_msgs</run_depend>
  <run_depend>collab_test</run_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>mavros</run_depend>

  <!-- The export tag contains other, unspecified, tags -->
  <export>
    <!-- Other tools can request additional information be placed here -->

  </export>
</package>


Header header
float64 voltage_one
float64 voltage_two
float64 current_one
float64 current_two
int8 flying
int8 task


/*
 * FILE: flight_testing.h
 * AUTHOR: William Willie Wells
 * DATE: May 2015
 */

#ifndef _FLIGHT_TESTING_H_
#define _FLIGHT_TESTING_H_

// ros includes
#include "ros/ros.h"
#include "rosbag/bag.h"

// mit_asctec includes
#include <collab_msgs/SubjectPose.h>
#include <collab_msgs/SubjectCtrlState.h>
#include <collab_msgs/QuadCtrlInput.h>
#include <collab_msgs/SubjectStatus.h>

// package message includes
//#include <flight_testing/PoseCompare.h>

// c++ includes
#include <boost/thread/thread.hpp>

// namespace
using namespace std;

// defines
#define PI 3.1415926535

// class
class FlightTest{
    public:
        // con(de)structors
        FlightTest();
        ~FlightTest();
```

```
        bool getLast();
    private:
        // callbacks
        void callbackPose(const boost::shared_ptr<collab_msgs::SubjectPose const>&);
        void callbackState(const boost::shared_ptr<collab_msgs::SubjectCtrlState const>&);
        void callbackStatus(const boost::shared_ptr<collab_msgs::SubjectStatus const>&);

        // initializers
        void initParams();
        void initPublishers();
        void initSubscribers();
        //void initServices();

        // start thread
        void startThread(int);

        // stop thread
        void stopThread(int);

        // active threads
        //void compareThread();
        void flightTestThread();
        void poseStateThread();

        // state change
        void startup();
        void forceIdle();
        void initialQCI();
        void publishQCI(int);
        void test();
        void ignition(int);
        void launch();
        void land();
        void sleep();

        // ros objects
        ros::NodeHandle nh, pnh;

        // ros publisher
        ros::Publisher pose_pub, state_pub;
        ros::Publisher test_pub, ignition_pub, off_pub, height_pub;

        // ros subscribers
        ros::Subscriber pose_sub, state_sub, status_sub;

        // thread objects
        boost::mutex mel;
        boost::shared_ptr<boost::thread> poseState_thread;
        boost::shared_ptr<boost::thread> /*compare_thread,*/ flight_thread;

        // position and state messages
        collab_msgs::SubjectPose current_pose, target_pose;
        collab_msgs::SubjectCtrlState command_state, previous_state;
        collab_msgs::QuadCtrlInput target_qci;
        collab_msgs::SubjectStatus status;
        //flight_testing::PoseCompare compare_pose;

        // flags
        volatile bool change_state;
        volatile bool change_pose;
        volatile bool publish_change;
        volatile bool launch_request;
        volatile bool land_request;
        volatile bool first;
        volatile bool last;
        volatile bool partial;
        volatile bool test_enabled;

        // private variables
        volatile int count;
        double initialX, initialY;

        // launch file parameters
        double launchZ;
        double poseChange;
        int test_length;
        int ctrl_rate;
};
#endif // flight test


/*
 * FILE: flight_testing.cpp
 * AUTHOR: William Willie Wells
 * DATE: May 2015 ...
 */

// class include
#include "../../include/flight_testing/flight_testing.h"

// defines
```

```
#define ROS_PKG "flightTest"
#define ROS_NODE "flightTest"

// constructor
FlightTest::FlightTest():nh(),pnh("~"){
    // initialize parameters
    initParams();ROS_INFO("parameteers initialized");

    // initialize publishers
    initPublishers();ROS_INFO("publishers initialized");

    // initialize subscribers
    initSubscribers();ROS_INFO("subscribers initialized");

    // initialize services

    // start main thread
    startThread(0);
}

// destructor
FlightTest::~FlightTest(){ stopThread(0); }

// return true if landing is a success
bool FlightTest::getLast(){ return last; }

// c++ Main
int main(int argc, char **argv){
    // initialize ros node handle
    ros::init(argc,argv,ROS_NODE);

    // log start
    ROS_INFO("Started node %s",ROS_NODE);

    // create and start flight test instance
    FlightTest flytetest;

    // keep spinning until landing acheived
    while(!flytetest.getLast()){ ros::spinOnce(); }

    // log stop and shutdown
    ROS_INFO("Stopped node %s", ROS_NODE);
    ros::shutdown();

    return 0;
}


/*
 * FILE: flight_testing_callback.cpp
 * AUTHOR: William Willie Wells
 * DATE: May 2015
 */

// class include
#include "../../include/flight_testing/flight_testing.h"

// callback for state change
void FlightTest::callbackState(const boost::shared_ptr<collab_msgs::SubjectCtrlState const> &msg){
    //ROS_INFO("why");
    //mel.lock();
    //ROS_INFO("pub: %d, state: %d, pose: %d",publish_change,change_state,change_pose);
    previous_state = (*msg);
    // set target state if a target position is available and not finished
    if(!publish_change && !change_state && !last){
        // between moving and hovering
        //ROS_INFO("state: %d",previous_state.state);
        if(!land_request && count < 18){
            if(previous_state.state == 7){
                /*if(hover_count == 1){*/ command_state.state = 8; change_state = true;/*hover_count = 0; }
                else{ command_state.state = 7; hover_count++; }*/
            }else if(previous_state.state == 8){ command_state.state = 7; change_state = true;}
        }

        // change state
        if(command_state.state == 0){ // ESTOP
            change_state = true;
            land_request = false;
        }else if(command_state.state == 1){ // OFF
            if(previous_state.state == 0){ change_state = true; }
        }else if(command_state.state == 2){ // STARTUP
            if(previous_state.state == 1){ change_state = true; }
        }else if(command_state.state == 3){ // SHUTDOWN
            if(previous_state.state == 4){ change_state = true; }
        }else if(command_state.state == 4){ // IDLE
            if(previous_state.state == 6){ change_state = true; }
        }else if(command_state.state == 5){ // LAUNCH
            if(previous_state.state == 4){ change_state = true; }
        }else if(command_state.state == 6){ // LAND
            if(previous_state.state == 7){ change_state = true; }
        }else if(command_state.state == 7){ // HOVER
```

```
            if(previous_state.state == 8){ change_state = true; }
        }else if(command_state.state == 8){ // MOVE
            if(previous_state.state == 7){ change_state = true; }
        }
    }
    // publish if target state and target position obtain
    if(change_state && change_pose){ publish_change = true; }

    //mel.unlock();
}

// callback for position change
void FlightTest::callbackPose(const boost::shared_ptr<collab_msgs::SubjectPose const> &msg){
    //mel.lock();
    current_pose = (*msg);

    // save current position in position compare topic
    /*compare_pose.position.x = current_pose.translation.x;
    compare_pose.position.y = current_pose.translation.y;
    compare_pose.position.z = current_pose.translation.z;
    compare_pose.position.w = current_pose.rotation.z;*/
    //ROS_INFO("pub: %d, pose: %d, state: %d",publish_change,change_pose,change_state);
    // set target position if not finished
    if(!publish_change && !change_pose && !last){
        //ROS_INFO("manuever: %i, state: %d",count,previous_state.state);// set rotation
        if(previous_state.state < 7){
            target_pose.rotation.z = PI;
        }else{
            if(count == 1 || count == 4 || count == 15){ target_pose.rotation.z =  PI/2; }
            else if(count == 2 || count == 7){ target_pose.rotation.z = 3*PI/2; }
            else if(count == 3 || count == 10){ target_pose.rotation.z = PI; }
            else{ target_pose.rotation.z = 0.0; }
        }

        // set initial position
        if(first){
            initialX = current_pose.translation.x;
            initialY = current_pose.translation.y;
            target_pose.translation.x = current_pose.translation.x;
            target_pose.translation.y = current_pose.translation.y;
            target_pose.translation.z = 0.0;
            first = false;
        }else{
            /*if(previous_state.state < 7){ target_pose.translation.z = 0.0; }
            else{*/ target_pose.translation.z = launchZ; /*}*/
            target_pose.translation.x = initialX;
            target_pose.translation.y = initialY;
        }

        // change position
        if(count == 4){ target_pose.translation.x += poseChange; }
        else if(count == 5){ target_pose.translation.x -= poseChange; }
        else if(count == 7){ target_pose.translation.y -= poseChange; }
        else if(count == 8){ target_pose.translation.y += poseChange; }
        else if(count == 10){ target_pose.translation.z += poseChange; }
        else if(count == 12){ // change in x and y
            target_pose.translation.x += poseChange;
            target_pose.translation.y += poseChange;
        }else if(count == 13){ // change in x and z
            target_pose.translation.y += poseChange;
            target_pose.translation.z += poseChange;
        }else if(count == 15){ // count = 14 change in y and z; change in x, y, and w
            target_pose.translation.x -= poseChange;
            target_pose.translation.y -= poseChange;
        }else if(count == 16){ // change in x, y, z, and w
            target_pose.translation.z += poseChange;
        }else if(count > 16){ land_request = true; } // count = 17 change in z

        // publishing condition
        change_pose = true;
    }

    /*// confirm target with land target
    if(land_request == true || (launch_request == true && !first)){
        target_pose.translation.x = initialX;
        target_pose.translation.y = initialY;
        if(land_request == true && previous_state.state < 7){
            target_pose.translation.z = 0.0;
        }else if(launch_request == true){
            if(previous_state.state >= 2){ target_pose.translation.z = launchZ; }
            else{ target_pose.translation.z = 0.0; }
        }
        if(previous_state.state == 6 || previous_state.state == 2){
            target_pose.rotation.z = PI;
        }else if(land_request == true){ target_pose.rotation.z = 0.0; }
    }  */

    // save current target in position compare topic
    /*compare_pose.target.x = target_pose.translation.x;
    compare_pose.target.y = target_pose.translation.y;
    compare_pose.target.z = target_pose.translation.z;
```

```
    compare_pose.target.w = target_pose.rotation.z;*/

    //mel.unlock();
}

// callback for status
void FlightTest::callbackStatus(const boost::shared_ptr<collab_msgs::SubjectStatus const> &msg){
    //mel.lock();
    status = (*msg);
    //mel.unlock();
}


/*
 * FILE: flight_testing_init.cpp
 * AUTHOR: William Willie Wells
 * DATE: May 2015
 */

// c++ include
#include <cmath>

// class include
#include "../../include/flight_testing/flight_testing.h"

// initialize parameters
void FlightTest::initParams(){
    // initialize private booleans
    change_state = true;
    change_pose = false;
    publish_change = false;
    launch_request = true;
    land_request = false;
    first = true;
    last = false;
    partial = false;
    test_enabled = false;

    // set launch height
    pnh.param("launch_height",launchZ,0.7);
    if(launchZ < 0.1 || launchZ > 1.4){ launchZ = 0.7; }

    // set test position change variance used in x, y, and z
    pnh.param("position_change",poseChange,0.2);
    if(fabs(poseChange) > 1.0){ poseChange = 0.5; }
    ROS_INFO("Z:%f, change:%f",launchZ,poseChange);

    // set relative length of each pre launch test
    pnh.param("test_length", test_length, 16);
    if(test_length < 8){ test_length = 16; }

    // set control rate
    pnh.param("ctrl_rate",ctrl_rate,25);
    if(ctrl_rate < 10 || ctrl_rate > 200){ ctrl_rate = 25; }

    // initialize private real numbers
    count = 0;
//    hover_count = 0;
    initialX = 1.9;
    initialY = -1.9;
}

// initialize subscribers
void FlightTest::initSubscribers(){
    // subscribe to position
    pose_sub = nh.subscribe<collab_msgs::SubjectPose>("subject_pose",1,&FlightTest::callbackPose,this);
    // subscribe to state
    state_sub = nh.subscribe<collab_msgs::SubjectCtrlState>("subject_ctrl_state",10,&FlightTest::callbackState,this);
    // subscribe to status
    status_sub = nh.subscribe<collab_msgs::SubjectStatus>("subject_status",1,&FlightTest::callbackStatus,this);
}

// initialize publishers
void FlightTest::initPublishers(){
    // publish position
    pose_pub = nh.advertise<collab_msgs::SubjectPose>("task_waypose",1,true);

    // publish state
    state_pub = nh.advertise<collab_msgs::SubjectCtrlState>("cmd_subject_ctrl_state",10,true);

    // publish prelaunch test
    test_pub = nh.advertise<collab_msgs::QuadCtrlInput>("motor_test_input",1,true);

    // publish ignition command
    ignition_pub = nh.advertise<collab_msgs::QuadCtrlInput>("ignition_input",1,true);

    // publish off command
    off_pub = nh.advertise<collab_msgs::QuadCtrlInput>("off_input",1,true);

    // publish launch or land command
    height_pub = nh.advertise<collab_msgs::QuadCtrlInput>("height_input",1,true);
```

```
}


/*
 * FILE: flight_testing_thread.cpp
 * AUTHOR: William Willie Wells
 * DATE: May 2015
 */

// class include
#include "../../include/flight_testing/flight_testing.h"

// unlock mutual exclusion to publish, sleep, and relock
void FlightTest::sleep(){
    mel.unlock();
    ros::Duration(3.0).sleep();
    mel.lock();
}

// startup UAV
void FlightTest::startup(){
    //
    if(previous_state.state == 0){
        // starting: do not need to land
        land_request = false;

        // transition to OFF
        initialQCI();
        publishQCI(1);
        sleep();

        // transition to STARTUP
        while(status.motors_status == 0){
            ignition(2);
            sleep();
        }
    }
}

//
void FlightTest::forceIdle(){
    // force idle state
    initialQCI();
    publishQCI(4);
    sleep();
}

// initial quad control input
void FlightTest::initialQCI(){
    target_qci.pitch =
    target_qci.roll =
    target_qci.yaw = 0.0;
    target_qci.thrust = -1;
    target_qci.pitch_ctrl =
    target_qci.roll_ctrl =
    target_qci.yaw_ctrl =
    target_qci.thrust_ctrl = 1;
    target_qci.altitude_ctrl =
    target_qci.gps_ctrl = 0;
}

// publish quad control input for non-flying states, states < 7 [Thread?]
void FlightTest::publishQCI(int state){
    ros::Rate pub_rate(ctrl_rate);
    //mel.lock();

    // synchronize time
    target_qci.header.stamp =
    target_pose.header.stamp =
    command_state.header.stamp = ros::Time::now();

    // publish pose
    target_pose.translation.x = initialX;
    target_pose.translation.y = initialY;
    if(state == 5){ target_pose.translation.z = launchZ; }
    else{ target_pose.translation.z = 0.0; }
    target_pose.rotation.z = 0.0;
    if(state > 1){ pose_pub.publish(target_pose); }

    //ROS_INFO("yaw: %f",target_qci.yaw);
    // publish quad control input
    if(state == 4){ test_pub.publish(target_qci); }
    else if(state == 2 || state == 3){ ignition_pub.publish(target_qci); }
    else if(state == 1){ off_pub.publish(target_qci); }
    else if(state == 6 || state == 5){ height_pub.publish(target_qci); }

    // publish state
    command_state.state = state;
    state_pub.publish(command_state);

    //mel.unlock();
```

```
    pub_rate.sleep();
}

// test pitch and roll
void FlightTest::test(){
    double move = 1.0;
    test_enabled = true;

    // perform motor tests
    for(int i=0;i<4*test_length+4;i++){
        // iniital quad control inputs
        initialQCI();

        // y positive, y negative, x positive, x negative
        if(i < test_length){ target_qci.pitch -= move; }
        else if(i < 3*test_length && i > 2*test_length-1){ target_qci.roll -= move; }
        else if(i < 2*test_length && i > 1*test_length-1){ target_qci.pitch += move; }
        else if(i < 4*test_length && i > 3*test_length-1){ target_qci.roll += move; }

        // publish quad control input
        publishQCI(4);
    }
    test_enabled = false;
}

// start or stop motors
void FlightTest::ignition(int state){
    initialQCI();
    if(state == 2){ target_qci.yaw = 1.0; }
    else{ target_qci.yaw = -1.0; }
    publishQCI(state);
}

// launch UAV
void FlightTest::launch(){
    // transition to LAUNCH
    initialQCI();
    publishQCI(5);
    sleep();

    // success, launched
    launch_request = land_request =
    publish_change = change_pose = change_state = false;
    ROS_INFO("Launched");

}

// land UAV
void FlightTest::land(){
    // land from moving or hovering state
    if(previous_state.state == 8 || previous_state.state == 7){
        // transition to HOVER
        command_state.state = 7;
        sleep();

        // transition to LAND
        initialQCI();
        while(current_pose.translation.z > 0.05){
            target_qci.altitude_ctrl = 1;
            publishQCI(6);
            //sleep();
        }

        // transistion to Idle
        initialQCI();
        publishQCI(4);

        // shutdown and auto transition to OFF
        while(status.motors_status == 1){
            ignition(3);
            sleep();
        }

        // transition to ESTOP
        command_state.state = 0;
        sleep();
    }

    // landing a success
    land_request = false;
    //mel.unlock();
}

// stop a thread
void FlightTest::stopThread(int t){
     switch(t){
         case(0): flight_thread->join(); break;
         case(1): poseState_thread->join(); break;
         //case(2): state_thread->join(); break;
         default:  break;
     }
```

```
}

// start a thread
void FlightTest::startThread(int t){
    switch(t){
        case(0):
            flight_thread = boost::shared_ptr<boost::thread>(new boost::thread(boost::bind(&FlightTest::flightTestThre
        case(1):
            poseState_thread = boost::shared_ptr<boost::thread>(new boost::thread(boost::bind(&FlightTest::poseStateTh
        //case(2):
        //    state_thread = boost::shared_ptr<boost::thread>(new boost::thread(boost::bind(&FlightTest::stateThread,t
        default: break;
    }
}

// position thread
void FlightTest::poseStateThread(){
    // set publishing rate
    ros::Rate pub_rate(ctrl_rate);

    // publish position data
    while(ros::ok()){
        //ROS_INFO("pub:%i, launch:%i, land:%i, state:%i, pose:%i, test:%i, count:%i",
        //    publish_change,launch_request,land_request,change_state,change_pose,test_enabled,count);
        if(publish_change && !test_enabled && command_state.state > 6){
            //mel.lock();
            //ROS_INFO("pub:%i, launch:%i, land:%i, state:%i, pose:%i, test:%i, count:%i",
            //publish_change,launch_request,land_request,change_state,change_pose,test_enabled,count);
            command_state.header.stamp =
            target_pose.header.stamp =  ros::Time::now();
            //ROS_INFO("X:%f, Y:%f, Z:%f, W:%f,C:%i",target_pose.translation.x,target_pose.translation.y,target_pose.t
            // publish new position
            pose_pub.publish(target_pose);
            state_pub.publish(command_state);

            change_state = false;
            change_pose = false;

            if(publish_change && !launch_request && command_state.state == 8){
                count++; land_request = false;
            }

            // reset publishing condition
            publish_change = false;

            //mel.unlock();
            pub_rate.sleep();
        }
    }
}

// main thread
void FlightTest::flightTestThread(){
    // start pose, and state threads
    //startThread(2);
    startThread(1);

    // main thread launchs, relies on callbacks, lands when count reachs 17
    while(ros::ok() && !last){
        // launch UAV
        if(launch_request){
            startup();
            while(previous_state.state != 4){ forceIdle(); }
            test();
            launch();
        }

        // condition to land
        if(count > 16){ land_request = true; }

        // land UAV
        if(land_request){ land(); last = true; }
    }

    // stop state, and pose threads
    stopThread(1);
    //stopThread(2);
}


/*
 * FILE: parse_serial.h
 * AUTHOR: William Willie Wells, modified from code
 *         written by: Carrick Detweiler and Najeeb Najeeb
 * DATE: May 2015
 */

#ifndef _PARSE_SERIAL_H_
#define _PARSE_SERIAL_H_

// ros includes
```

```
#include "ros/ros.h"
#include "std_msgs/UInt8MultiArray.h"
#include "std_msgs/Float64MultiArray.h"
#include "std_msgs/Float64.h"

// mit_asctec includes
#include <collab_msgs/SubjectCtrlState.h>
#include <collab_msgs/SubjectPose.h>

// c++ include
#include <boost/thread/thread.hpp>

// package message include
#include <flight_testing/Sensor.h>

// namespace
using namespace std;

// class
class ParseSerial{
    public:
        // con(de)structors
        ParseSerial();
        ~ParseSerial();

    private:
        // callbacks
        void callbackSerialRx(const boost::shared_ptr<std_msgs::UInt8MultiArray const>&);
        void callbackState(const boost::shared_ptr<collab_msgs::SubjectCtrlState const>&);
        void callbackTask(const boost::shared_ptr<collab_msgs::SubjectPose const>&);

        // initializers
        void initParams();
        void initPublishers();
        void initSubscribers();

        // main process
        void processByte();

        // ros objects
        ros::NodeHandle nh;

        // ros publisher
        ros::Publisher sensor, serial_tx;

        // ros subscribers
        ros::Subscriber serial_rx, flight, task;

        // thread object
        boost::mutex mel;

        // serial messages
        std_msgs::UInt8MultiArray message;
        std_msgs::Float64MultiArray float_message;
        collab_msgs::SubjectCtrlState current_state;
        collab_msgs::SubjectPose current_task;
        flight_testing::Sensor sensor_value;

        // state enumeration
        enum states{NOT_SYN, // waiting for '\r'
            READY, // '\r' received waiting for '\n'
            SENSOR1, // '\n' received, waiting for first sensor data
            SENSOR2, // previous sensor's data received, waiting for next sensor's data
            SENSOR3, // previous sensor's data received, waiting for next sensor's data
            SENSOR4, // previous sensor's data received, waiting for next sensor's data
        };
        volatile enum states state;

        // private variables
        bool decimal;
        //bool state_received;
        //bool task_received;
        volatile bool flying;
        int size;
        int place;
        unsigned int c;
        double value;
};
#endif // parse serial


/*
 * FILE: parse_serial.cpp
 * AUTHOR: William Willie Wells, modified from code
 *         written by: Carrick Detweiler and Najeeb Najeeb
 * DATE: May 2015
 */

// class include
#include "../../include/parse_serial/parse_serial.h"
```

```
// defines
#define ROS_PKG "flightTest"
#define ROS_NODE "parseSerial"

// constructor
ParseSerial::ParseSerial():nh(){
    // initialize parameters
    initParams();

    // initialize publishers
    initPublishers();

    // initialize subscribers
    initSubscribers();
}

// destructor
ParseSerial::~ParseSerial(){}

// c++ Main
int main(int argc, char **argv){
    // initialize ros node handle
    ros::init(argc,argv,ROS_NODE);

    // log start
    ROS_INFO("Started node %s",ROS_NODE);

    // create and start parse serial instance
    ParseSerial passCereal;

    // keep spinning
    ros::spin();

    // log stop
    ROS_INFO("Stopped node %s", ROS_NODE);

    return 0;
}


/*
 * FILE: parse_serial_callback.cpp
 * AUTHOR: Wiliiam Willie Wells, modified from code
 *         written by Carrick Detweiler and Najeeb Najeeb
 * DATE: May 2015
 */

// class include
#include "../../include/parse_serial/parse_serial.h"

// callback on serial data received
void ParseSerial::callbackSerialRx(const boost::shared_ptr<std_msgs::UInt8MultiArray const> &msg){
    mel.lock();

    // obtain message and the size of the packet
    message = (*msg);
    size = message.data.size();

    // process each byte in the message packet separately
    for(unsigned int i=0;i<size;i++){
        c = message.data.at(i);
        processByte();
    }

    mel.unlock();
}

// process individual bytes and publish sensor values
void ParseSerial::processByte(){
    // read and construct floating point sensor values a byte at a time
    if(state != NOT_SYN && state != READY){
        if(c == 44 || (state == SENSOR4 && c == 13)){ // sensor value end condition
            // scale value based on control board used
            if(state == SENSOR1 || state == SENSOR3){ value /= 49.44; }
            else{ value /= 14.9; }
        }else if(c == 32 || (state == SENSOR4 && c == 13)){ // reset flags
            decimal = false;
            place = 0;
        }else if(c == 46){ decimal = true; } // decimal point, '.'
        else if(c < 58 && c > 47){ // construct complete sensor value
            if(!decimal){
                if(value != 0){ value *= 10; }
                value += c - 48;
            }else{ place++; value += (c -48)/(10*place); }
        }
    }

    // state specific actions: change state, publish sensor value
    switch(state){
        case NOT_SYN:
            if(c == 13){ state = READY; } // '\r'
```

```
            else{ state = NOT_SYN; }
            break;
        case READY:
            if(c == 10){ state = SENSOR1; } // '\n'
            else{ state = NOT_SYN; value = 0.0; }
            break;
        case SENSOR1:
            if(c == 44){
                sensor_value.voltage_one = value;
                value = 0.0;
            }else if(c == 32){ state = SENSOR2; } // ' '
            break;
        case SENSOR2:
            if(c == 44){
                sensor_value.current_one = value;
                value = 0.0;
            }else if(c == 32){ state = SENSOR3; }
            break;
        case SENSOR3:
            if(c == 44){
                sensor_value.voltage_two = value;
                value = 0.0;
            }else if(c == 32){ state = SENSOR4; }
            break;
        case SENSOR4:
            if(c == 13){
                sensor_value.current_two = value;
                value = 0.0;
                //ROS_INFO("sensor values %f",sensor_value.current_one);
                sensor_value.header.stamp = ros::Time::now();
                sensor.publish(sensor_value);
                state = READY;
            }
            break;
        default: state = NOT_SYN; break;
    }
}

// callback for operating range
void ParseSerial::callbackState(const boost::shared_ptr<collab_msgs::SubjectCtrlState const> &msg){
    mel.lock();
    // indicate whether the vehicle is in state 7 or 8
    current_state = (*msg);
    if(current_state.state > 6 && flying == false){
        flying = true;
    }else if(current_state.state < 7 && flying == true){
        flying = false;
    }
    sensor_value.flying = flying;
    // state_received = true;
    mel.unlock();
}

// callback for task segmentation
void ParseSerial::callbackTask(const boost::shared_ptr<collab_msgs::SubjectPose const> &msg){
    mel.lock();
    // indicate which task sequence corresponds to a particular sensor value
    current_task = (*msg);
    sensor_value.task = current_task.header.seq;

    //
    mel.unlock();
}


/*
 * FILE: parse_serial_init.cpp
 * AUTHOR: Wiliiam Willie Wells modified from code written
 *         by Carrick Detweiler and Najeeb Najeeb
 * DATE: May 2015
 */

// class include
#include "../../include/parse_serial/parse_serial.h"

// initialize parameters
void ParseSerial::initParams(){
    // initialize private booleans
    decimal = false;
    //state_received = false;
    //task_received = false;
    flying = false;

    // initialize state
    state = NOT_SYN;

    // initialize private real numbers
    size = 1;
    place = 0;
    c = 0;
    value = 0.0;
```

```
}

// initialize subscribers
void ParseSerial::initSubscribers(){
    // subscribe to communication serial port receiver
    serial_rx = nh.subscribe("device_rx_data",10,&ParseSerial::callbackSerialRx,this);

    // subscribe to  state
    flight = nh.subscribe<collab_msgs::SubjectCtrlState>("subject_ctrl_state",10,&ParseSerial::callbackState,this);

    // subscribe to task
    task = nh.subscribe<collab_msgs::SubjectPose>("task_waypose",10,&ParseSerial::callbackTask,this);
}

// initialize publishers
void ParseSerial::initPublishers(){
    // publish each sensor value on separate topic
    sensor = nh.advertise<flight_testing::Sensor>("sensor_data",10,true);
}


<launch>

  <node name="ard_driver" type="ardrone_driver" pkg="ardrone_autonomy" args="-ip 192.168.2.5" output="log"/>

  <group ns="a">
    <include file="$(find flight_testing)/launch/ardrone_common.launch" />

    <!-- Even indoors we need GPS, so that subject_status messages are published -->
    <include file="$(find collab_launch)/common/poll_gps.launch" />
    <include file="$(find collab_launch)/common/poll_imu.launch" />

    <node name="ad_prot" type="ad_prot" pkg="ad_protocol" output="screen" />

    <node name="robot_monitor" type="asctec_monitor" pkg="supervision"/>
    <node name="robot_trans" type="asctec_trans" pkg="translation"/>

    <!--param name="sim_latitude" value="40.821187"/>
    <param name="sim_longitude" value="-96.704359"/-->

    <!-- PID Controller -->
    <node name="pid_ctrl" type="quad_pid" pkg="pose_control">
      <param name="ctrl_rate" value="100.0"/>

      <!-- Filtering and control output options -->
      <!-- Fly under Vicon control, so the thrust should emulate the
        manual throttle control, not the barometric pressure control. -->
      <param name="enable_baro_ctrl_mode" value="false" />
      <!-- Not flying in barometric pressure control, so do not collapse
        the deadband. -->
      <param name="collapse_deadband" value="false" />
      <!-- Disable the IIR filter on thrust output, because Vicon control
        gives a good height estimate, resulting in smooth flight. -->
      <param name="enable_thrust_iir_filter" value="false" />
      <!-- Vicon control does not require use to smooth derivative input,
        so disable filtering on this term -->
      <param name="enable_derv_fir_filter" value="true" />

      <!-- PITCH -->
      <param name="k_prop_pitch" value="0.5"/>
      <param name="k_intg_pitch" value="0.0"/>
      <param name="k_derv_pitch" value="0.25"/>
      <param name="pitch_offset" value="0.0"/>
      <param name="pitch_intg_min_sat" value="0.0"/>
      <param name="pitch_intg_max_sat" value="0.0"/>

      <!-- ROLL  -->
      <param name="k_prop_roll" value="0.5"/>
      <param name="k_intg_roll" value="0.0"/>
      <param name="k_derv_roll" value="0.25"/>
      <param name="roll_offset" value="0.0"/>
      <param name="roll_intg_min_sat" value="0.0"/>
      <param name="roll_intg_max_sat" value="0.0"/>

      <!-- YAW  -->
      <param name="k_prop_yaw" value="1.0"/>
      <param name="k_intg_yaw" value="0.0"/>
      <param name="k_derv_yaw" value="0.4"/>
      <param name="yaw_offset" value="0.0"/>
      <param name="yaw_intg_min_sat" value="0.0"/>
      <param name="yaw_intg_max_sat" value="0.0"/>

      <!-- THRUST   -->
      <param name="k_prop_thrust" value="0.4"/>
      <param name="k_intg_thrust" value="0.0"/>
      <param name="k_derv_thrust" value="-0.2"/>
      <param name="thrust_offset" value="0.9"/>
      <param name="thrust_intg_min_sat" value="-0.75"/>
      <param name="thrust_intg_max_sat" value="1.0"/>

    </node>
```

```
    <!-- Graphical flight control -->
    <!--node name="gui_fly" type="gui_fly" pkg="gui_fly">
      <param name="min_y" value="-2.0"/>
      <param name="max_y" value="2.0"/>
      <param name="min_x" value="-2.0"/>
      <param name="max_x" value="2.0"/>
      <param name="min_z" value="0.0"/>
      <param name="max_z" value="2.5"/>
    </node-->

    <!-- Graphical monitoring -->
        <node name="gui_monitor" type="gui_monitor.py" pkg="gui_fly" />

  </group>

</launch>


<launch>

  <node name="ard_driver" type="ardrone_driver" pkg="ardrone_autonomy" args="-ip 192.168.2.5" output="log"/>

</launch>


<launch>
  <!-- ============================Remote Control Communication================================ -->
  <node name="robot_comm" type="serial_comm" pkg="communication">
    <remap from="serial_rx_data" to="robot_rx_data"/>
    <remap from="serial_tx_data" to="robot_tx_data"/>
    <param name="serial_device" value="/dev/ttyUSB0"/>
    <param name="serial_baudrate" value="57600"/>
  </node>

  <!-- ========================================GUI============================================ -->
      <!-- Get information such as motor and battery status from UAV -->
  <node name="poll_status" type="rostopic" pkg="rostopic"
    args="pub -r 1.0 'robot_poll' collab_msgs/AsctecPoll
    '{header: auto, packets: 1}'"/>
      <!-- Poll IMU information -->
  <node name="poll_imu" type="rostopic" pkg="rostopic"
    args="pub -r 5.0 'robot_poll' collab_msgs/AsctecPoll
    '{header: auto, packets: 4}'"/>
      <!-- Get GPS information from UAV -->
  <node name="poll_gps" type="rostopic" pkg="rostopic"
    args="pub -r 4.0 'robot_poll' collab_msgs/AsctecPoll
    '{header: auto, packets: 128}'"/>

      <!-- graphical user interface monitoring -->
  <node name="gui_monitor" type="gui_monitor.py" pkg="gui_fly"/>

  <!-- ==================Ascending Technologies to ROS Communication=================== -->
      <!-- flight state, protocol, subject_status, translation of commands to AscTec -->
  <node name="ctrl_state_machine" type="manage_subject_ctrl" pkg="state_machine"
    output="screen"/>
  <node name="robot_prot" type="asctec_prot" pkg="protocol"/>
  <node name="robot_monitor" type="asctec_monitor" pkg="supervision"/>
  <node name="robot_trans" type="asctec_trans" pkg="translation"/>

</launch>


<launch>
  <!-- flight_testing.launch, created by: Wiliam Willie Wells, in May 2015 -->

  <!-- ========================================VICON Object============================================ -->
  <group ns="a">
    <node pkg="vicon_pose" type="vicon_pose" name="conv_q_p" output="screen">
      <param name="object_name" value="HULK_SMASH" type="str" />
      <!-- "unl_comp_sci", "UNL_COMP_SCI" -->
    </node>
  </group>

  <!-- ======================MIT AscTec Protocol and Communication====================== -->
    <!-- movement at constant velocity -->
  <include file="$(find flight_testing)/launch/hummingbird_indoor.launch" />
  <!--include file="$(find flight_testing)/launch/arducopter.launch" /-->

  <!-- ===================================Bag Recording============================================ -->
    <!-- custom bag placement, -a -o records a subset, -->
    <!-- scp -r /path/bagDirectory wwells@cse.unl.edu:/home/research3/elbaum/nimbus/rosBags/failure_testing -->
    <!-- sed -i.bak s/unl_comp_sci#/crash#+1/ path/flight_testing.launch -->

  <node pkg="rosbag" name="record" type="record" args="-O $(find flight_testing)/bags/hummingbird/hulk_smash/hummingb
  <!-- sed -i.bak s/unl_comp_sci#/crash#+1/ path/bat -->
  <!-- ./path/bat -->

  <group ns="a">
  <!-- ================================Arduino Communication============================== -->
    <node name="arduino_comm" type="serial_comm" pkg="communication">
```

```
    <remap from="serial_rx_data" to="device_rx_data"/>
    <remap from="serial_tx_data" to="device_tx_data"/>
    <param name="serial_baudrate" value="9600"/>
    <param name="serial_device" value="/dev/ttyACM0"/>
  </node>

  <node name="parse_serial" type="parse_serial" pkg="flight_testing" output="screen"></node>

<!--=====================================Main Node=====================================-->
  <node required="true" name="flight_testing" type="flight_testing" pkg="flight_testing" output="screen">
    <param name="position_change" value="0.5"/>
    <param name="launch_height" value="0.7"/>
    <param name="ctrl_rate" value="30"/>
    <param name="test_length" value="24"/><!-- relative length of 4 directional ground test -->
  </node>

 </group>
</launch>


<launch>
      <!-- Namespace for one UAV. Multiple UAVs require multiple namespaces -->
 <group ns="a">
  <include file="$(find flight_testing)/launch/communication.launch"/>

  <!--=====================================PID Controller=====================================-->
  <node name="pid_ctrl" type="quad_pid" pkg="pose_control">
    <param name="ctrl_rate" value="30"/>

        <!-- Filtering and control output options -->
        <!-- Fly under Vicon control, so the thrust should emulate the
             manual throttle control, not the barometric pressure control. -->
    <param name="enable_baro_ctrl_mode" value="false" />
        <!-- Not flying in barometric pressure control, so do not collapse the deadband. -->
    <param name="collapse_deadband" value="false" />
        <!-- Disable the IIR filter on thrust output, because Vicon control
           gives a good height estimate, resulting in smooth flight. -->
    <param name="enable_thrust_iir_filter" value="false" />
        <!-- Vicon control does not require us to smooth derivative input,
           so disable filtering on this term -->
    <param name="enable_derv_fir_filter" value="false" />

        <!-- PITCH -->
    <param name="k_prop_pitch" value="-0.2"/>
    <param name="k_intg_pitch" value="-0.06"/>
    <param name="k_derv_pitch" value="-0.3"/>
    <param name="pitch_offset" value="0.0"/>
    <param name="pitch_intg_min_sat" value="-1.0"/>
    <param name="pitch_intg_max_sat" value="1.0"/>

        <!-- ROLL  -->
    <param name="k_prop_roll" value="-0.2"/>
    <param name="k_intg_roll" value="-0.06"/>
    <param name="k_derv_roll" value="-0.3"/>
    <param name="roll_offset" value="0.0"/>
    <param name="roll_intg_min_sat" value="-1.0"/>
    <param name="roll_intg_max_sat" value="1.0"/>

        <!-- YAW  -->
    <param name="k_prop_yaw" value="-0.5"/>
    <param name="k_intg_yaw" value="0.0"/>
    <param name="k_derv_yaw" value="0.0"/>
    <param name="yaw_offset" value="0.0"/>
    <param name="yaw_intg_min_sat" value="0.0"/>
    <param name="yaw_intg_max_sat" value="0.0"/>

        <!-- THRUST   -->
    <param name="k_prop_thrust" value="0.4"/>
    <param name="k_intg_thrust" value="0.1"/>
    <param name="k_derv_thrust" value="0.2"/>
    <param name="thrust_offset" value="0.9"/>
    <param name="thrust_intg_min_sat" value="-0.75"/>
    <param name="thrust_intg_max_sat" value="1.0"/>
  </node>

  <!--=========================State Specific Position Controllers=========================-->

  <node name="pid_waypose_arb" type="arb_subject_ctrl" pkg="arbitration" output="screen">
    <remap from="shape_shifter" to="pid_waypose"/>
    <rosparam param="state_mappings">
      [
      [ "launch_waypose", [ 5 ] ],
      [ "land_waypose", [ 6 ] ],
      [ "task_waypose", [ 7,8 ] ],
      ]
    </rosparam>
  </node>

  <node name="launch_ctrl" type="delta_ctrl" pkg="pose_control" output="screen">
    <remap from="delta_waypose" to="launch_waypose"/>
    <param name="ctrl_rate" value="30.0"/>
```

```
    <param name="delta_rate" value="0.5"/>
    <param name="delta_term" value="translation_z"/>
    <param name="delta_state" value="5"/>
  </node>

  <node name="land_ctrl" type="delta_ctrl" pkg="pose_control" output="screen">
    <remap from="delta_waypose" to="land_waypose"/>
    <param name="ctrl_rate" value="30.0"/>
    <param name="delta_rate" value="-0.2"/>
    <param name="delta_term" value="translation_z"/>
    <param name="delta_state" value="6"/>
  </node>

  <!--======================State Specific PID Controller Input=======================-->

  <node name="ctrl_input_arb" type="arb_subject_ctrl" pkg="arbitration" output="screen">
    <remap from="shape_shifter" to="quad_ctrl_input"/>
    <rosparam param="state_mappings">
      [
      [ "off_input", [ 1 ] ],
      [ "ignition_input", [ 2, 3 ] ],
      [ "motor_test_input", [ 4 ] ],
      [ "height_input", [ 5, 6 ] ],
      [ "pid_input", [ 7, 8 ] ],
      ]
    </rosparam>
  </node>

 </group>
</launch>
```

C.2 MATLAB code

```
function [delay,packet_loss,flag,max_delay] = delayed_packet(topic_time,...
    begin_time,end_time,m,gap,gaptime_flag)
%delayed_packet: Return delayed packet timing information
%   Returns a set of time pairs which are the start and end of a delay in
%   delay; packet_loss contains the last abnormal delay time and length of
%   the corresponding delay; flag is initialized to 0 and set to 1 if there
%   is a delay in the interval; max_delay records the longest delay and the
%   time it occurred
    count = 1;
    delay = zeros(1,2);
    packet_loss = zeros(2,1);
    flag = 0;
    max_delay =  zeros(2,1);
    for i = 2:1:m
        if topic_time(i) > begin_time && topic_time(i) < end_time && topic_time(i) > 14.0
            gap_time = topic_time(i) - topic_time(i-1);
            if (gaptime_flag == 0 && gap_time > gap) || (gaptime_flag == 1 && abs(1.0 - gap_time) > gap)
                if gap_time > max_delay(1,1)
                    max_delay(1,1) = gap_time;
                    max_delay(2,1) = topic_time(i-1);
                end
                packet_loss(1,1) = gap_time;
                packet_loss(2,1) = topic_time(i-1);
                flag = 1; % delay of topic

                delay(count) = topic_time(i-1);
                delay(count+1) = topic_time(i);
                count = count + 2;
            end
        end
    end
end


function [signal1, signal2, m] = equalizePower(signal1,signal2,version)
%equalizePower: removes noisy values and returns equal sized arrays
%   Forces larger length signal to smaller length and returns both signals
%   and the length; version is based on changes in power toipic output
    mc = length(signal2');
    mv = length(signal1');

    % smooth out noise
    signal1 = smoothNoise(signal1,15.0,9.0,mv)';
    signal2 = smoothNoise(signal2,30.0,0.0,mc)';

    % equalize amount of data points
    if version < 3
        if mc < mv
            signal1 = signal1(1:mc);
            m = mc;
        elseif mc > mv
            signal2 = signal2(1:mv);
            m = mv;
        else
            m = mv;
        end
```

```
    else
        m = mv;
    end
end


function [secs,nsecs,foi_1,foi_2,foi_3,foi_4,foi_5,foi_6,foi_7,foi_8] = importfile(filename, startRow, endRow, topic)
% relative time (sample time), seconds, foi = fields of interest
%IMPORTFILE Import numeric data from a text file as column vectors.
%   [secs,,nsecs,foi_1,foi_2,foi_3,foi_4,foi_5,foi_6,foi_7,foi_8]
%   = IMPORTFILE(filename, startRow, endRow, topic) Reads data from rows
%   STARTROW through ENDROW of text file FILENAME. Topic specifies the
%   particular format to use when importing the data;
%
%    See also TEXTSCAN.
% Auto-generated by MATLAB on 2015/05/28 13:32:31

%% Initialize variables.
delimiter = ',';
if nargin<=2
    startRow = 2;
    endRow = inf;
end

%% Format string for each line of text:
if topic == 0 % sensor v0
    formatSpec = '%f%[^\n\r]';
elseif topic == 1 % position
    formatSpec = '%*s%f%f%f%f%f%f%f%f%f%s%[^\n\r]';
elseif topic == 2 % state
    formatSpec = '%*s%f%f%f%f%s%f%[^\n\r]';
elseif topic == 3 % status
    formatSpec = '%f%f%f%f%s%f%f%f%f%f%f%s%f%f%[^\n\r]';
elseif topic == 4 % vicon
    formatSpec = '%s%s%f%f%f%f%f%f%f%f%f%f%[^\n\r]';
elseif topic == 5 % control input
    formatSpec = '%f%d%s%f%f%f%f%f%f%f%f%f%f%[^\n\r]';
elseif topic == 6 % sensor v1
    formatSpec = '%f%s%f%f%f%f%f%[^\n\r]';
elseif topic == 7 % sensor v2
    formatSpec = '%f%f%f%s%f%f%f%f%f%f%[^\n\r]';
else
    return;
end

% For more information, see the TEXTSCAN documentation.
%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to format string.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
if topic < 6
    dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', delimiter, 'EmptyValue' ,NaN,'Head
    for block=2:length(startRow)
        frewind(fileID);
        dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, 'Delimiter', delimiter, 'Empty
        for col=1:length(dataArray)
            dataArray{col} = [dataArray{col};dataArrayBlock{col}];
        end
    end
else
    dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', delimiter, 'HeaderLines', startRow(
    for block=2:length(startRow)
        frewind(fileID);
        dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1, 'Delimiter', delimiter, 'Heade
        for col=1:length(dataArray)
            dataArray{col} = [dataArray{col};dataArrayBlock{col}];
        end
    end
end
%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and regenerate the
% script.

%% Allocate imported array to column variable names
if topic == 0 % sensor
    secs = dataArray{:, 1};
elseif topic == 1 % position
    foi_7 = dataArray{:, 1};
    secs = dataArray{:, 3};
    nsecs = dataArray{:, 2};
    foi_5 = dataArray{:, 5};
    foi_6 = dataArray{:, 6};
    foi_4 = dataArray{:, 7};
```

```
    foi_1 = dataArray{:, 8};
    foi_2 = dataArray{:, 9};
    foi_3 = dataArray{:, 10};
elseif topic == 2 % state
    foi_7 = dataArray{:, 1};
    secs = dataArray{:, 3};
    nsecs = dataArray{:, 2};
    foi_2 = dataArray{:, 2};
    foi_1 = dataArray{:, 4};
elseif topic == 3 % status
    foi_1 = dataArray{:, 2};
    foi_7 = dataArray{:, 6};
    nsecs = dataArray{:,7};
    secs = dataArray{:,8};
    foi_2 = dataArray{:, 9};
    foi_3 = dataArray{:, 11};
    foi_4 = dataArray{:, 14};
elseif topic == 4 % vicon
    foi_8 = dataArray{:, 1};
    foi_7 = dataArray{:, 3};
    secs = dataArray{:, 5};
    nsecs = dataArray{:, 4};
    foi_1 = dataArray{:, 7};
    foi_2 = dataArray{:, 8};
    foi_3 = dataArray{:, 9};
    foi_4 = dataArray{:, 10};
    foi_5 = dataArray{:, 11};
    foi_6 = dataArray{:, 12};
elseif topic == 5 % control input
    foi_5 = dataArray{:, 2};
    foi_7 = dataArray{:, 4}; % controller sequence
    nsecs = dataArray{:, 5};
    secs = dataArray{:, 6};
    foi_1 = dataArray{:, 7};
    foi_2 = dataArray{:, 9};
    foi_3 = dataArray{:, 11};
    foi_4 = dataArray{:, 13};
    foi_6 = dataArray{:, 4};
elseif topic == 6 % modified sensor message
    foi_3 = dataArray{:, 1}; % flying
    nsecs = dataArray{:, 4};
    secs = dataArray{:, 5};
    foi_2 = dataArray{:, 6}; % task
    foi_1 = dataArray{:, 7}; % value
elseif topic == 7 % version three sensor message
    nsecs = dataArray{:, 6};
    secs = dataArray{:, 7};
    foi_1 = dataArray{:, 9};
    foi_2 = dataArray{:, 10};
    foi_3 = dataArray{:, 1};
    foi_4 = dataArray{:, 2};
    foi_5 = dataArray{:, 8};
    foi_6 = dataArray{:, 3};
end


function [t_s,t_ns] = minTime(t1_s, t1_ns, t2_s,t2_ns)
%minTime returns the minimum unix time pair
%   Subtracts time 2 from time 1; returns time 2 if the result is negative
%   and time 1 if the result is positive.
    t = t1_s - t2_s...
        + (t1_ns - t2_ns)/1000000000;
    if t < 0
        t_s = t1_s;
        t_ns = t1_ns;
    else
        t_s = t2_s;
        t_ns = t2_ns;
    end

end


function plot_line(points,data)
%plot_line: plots vertical lines at each point in points
%   plots a vertical line at each point in points from the minimum to
%   the maximum value in data alternating color for each point in points.
    if points(1) ~= 0
        for i=1:2:length(points)
            y = min(data):(max(data)-min(data))/100:max(data);
            x = zeros(1,length(y));
            x2 = zeros(1,length(y));
            x(:) = points(i);
            plot(x,y,'m.')
            hold on
            x2(:) = points(i+1);
            plot(x2,y,'y.')
        end
    end
end
```

```
function [flight_stats]=plot_topic(test_set,vehicle,start_test,end_test,save_plot,view_plot)
%% Plot Flight Testing UAV Statistics
if save_plot == 1
    mkdir(sprintf('../plot/%s/',test_set))
end

% max topic data
nsp = 20000;
nscs = nsp/100;
nss = nscs;
nv = nsp;
nqci = nsp/10;
nse = nsp/10;

% results matrix
flight_stats = zeros(14,4);

% intermediate matrices
failed = 0;
fail_type = zeros(5,4);
n_test = end_test - start_test + 1;
oap = zeros(n_test,3);
task_power = zeros(n_test,2);
pmax = oap;
ot = zeros(n_test,2);
precision = zeros(n_test,5);
packet_delay = zeros(2,6);
fprintf('%s\n',vehicle);
for test = start_test:1:end_test
    test_index = test - start_test + 1;
    fail = 0;

    close all
    if save_plot == 1
        mkdir(sprintf('../plot/%s/%s%i/',test_set,vehicle,test))
    end
    % data version difference
    if (strcmp(vehicle,'herbie') == 1 && test > 20) || strcmp(vehicle,'morpheus') == 1
        v2 = 1;
        begin_task2 = zeros(1,2);
    elseif (strcmp(vehicle,'hulk_smash') == 1 && test > 3) || (strcmp(vehicle,'jenny') == 1 && test > 110)
        v2 = 2;
    else
        v2 = 0;
    end
    % import data from csv files
    [status_s,status_ns,voltage,motors_status,serial_mode,...% subject status
        waypose_status]= importfile_csv_multiple(sprintf('../csv/%s/%s%i/status_%s%i.csv',...
        test_set,vehicle,test,vehicle,test),2,nss,3);
%     [r_status_s,r_status_ns,r_voltage,r_motors_status,r_serial_mode,...% subject status
%         r_waypose_status]= importfile_csv_multiple(sprintf('../csv/%s/%s%i/robot_status_%s%i.csv',...
%         test_set,vehicle,test,vehicle,test),2,nss,3);
    [sp_s,sp_ns,x,y,z,rotation_sp]= importfile_csv_multiple(sprintf('../csv/%s/%s%i/position_%s%i.csv',...
        test_set,vehicle,test,vehicle,test),2,nsp,1); % subject pose
    [task_s,task_ns,x_t,y_t,z_t,r_t]= importfile_csv_multiple(sprintf('../csv/%s/%s%i/task_%s%i.csv',...
        test_set,vehicle,test,vehicle,test),2,nscs,1); % task waypose
    [vicon_s,vicon_ns,r_x,r_y,r_z,x_v,y_v,... % vicon
        z_v]= importfile_csv_multiple(sprintf('../csv/%s/%s%i/vicon_%s%i.csv',...
        test_set,vehicle,test,vehicle,test),2,nv,4);
    [qci_s,qci_ns,pitch,roll,thrust,rotation_qci,flying,... % Quad Control Input
        control_qci] = importfile_csv_multiple(sprintf('../csv/%s/%s%i/control_input_%s%i.csv',...
        test_set,vehicle,test,vehicle,test),2,nqci,5);
    [state_s,state_ns,state] = importfile_csv_multiple(sprintf('../csv/%s/%s%i/state_%s%i.csv',...
        test_set,vehicle,test,vehicle,test),2,nscs,2); % state
    [command_state_s,command_state_ns,command_state] = importfile_csv_multiple(sprintf('../csv/%s/%s%i/command_state_%
        test_set,vehicle,test,vehicle,test),2,nscs,2); % state
    if v2 == 1 % voltage and current
        [volt1_s,volt1_ns,voltage1,power_task,volt1_flying] =...
            importfile_csv_multiple(sprintf('../csv/%s/%s%i/sensor1_%s%i.csv',...
            test_set,vehicle,test,vehicle,test),2,nse,6);
        [volt2_s,volt2_ns,voltage2,power_task,volt2_flying] =...
            importfile_csv_multiple(sprintf('../csv/%s/%s%i/sensor3_%s%i.csv',...
            test_set,vehicle,test,vehicle,test),2,nse,6);
        [current_s,current_ns,current1,power_task,current1_flying] =...
            importfile_csv_multiple(sprintf('../csv/%s/%s%i/sensor2_%s%i.csv',...
            test_set,vehicle,test,vehicle,test),2,nse,6);
        [current_s,current_ns,current2,power_task,...
            current2_flying] = importfile_csv_multiple(sprintf('../csv/%s/%s%i/sensor4_%s%i.csv',...
            test_set,vehicle,test,vehicle,test),2,nse,6);
    elseif v2 == 2
        [power_s,power_ns,voltage1,voltage2,current1,current2,...
            power_task,power_flying] = importfile_csv_multiple(sprintf('../csv/%s/%s%i/sensor_%s%i.csv',...
            test_set,vehicle,test,vehicle,test),2,nse,7);
    else
        voltage1 = importfile_csv_multiple(sprintf('../csv/%s/%s%i/sensor1_%s%i.csv',...
            test_set,vehicle,test,vehicle,test),2,nse,0);
        voltage2 = importfile_csv_multiple(sprintf('../csv/%s/%s%i/sensor3_%s%i.csv',...
            test_set,vehicle,test,vehicle,test),2,nse,0);
        current1 = importfile_csv_multiple(sprintf('../csv/%s/%s%i/sensor2_%s%i.csv',...
            test_set,vehicle,test,vehicle,test),2,nse,0);
        current2 = importfile_csv_multiple(sprintf('../csv/%s/%s%i/sensor4_%s%i.csv',...
```

```
        test_set,vehicle,test,vehicle,test),2,nse,0);
end

%% topic sizes and time conversion
no_data = 0;

% blocked cmmunication, outgoing communication was blocked from
% controlling laptop by other device
if size(status_s) == [0,1]
    no_data = 1;
    fail = 1;
else % time conversion and sizes
    % find first time as reference 0
    [start_s,start_ns] = minTime(vicon_s(1,1),vicon_ns(1,1),status_s(1,1),status_ns(1,1));
    [start_s,start_ns] = minTime(start_s,start_ns,sp_s(1,1),sp_ns(1,1));%[start_s,start_ns] = minTime(start_s,star
    [start_s,start_ns] = minTime(start_s,start_ns,state_s(1,1),state_ns(1,1));
    [start_s,start_ns] = minTime(start_s,start_ns,task_s(1,1),task_ns(1,1));

    % convert time vectors from unix time to time from start
    [ms,state_time] = unixTime2TestTime(start_s,start_ns,state_s',state_ns');
    % crop bag
    cut_time = state_time(ms);
    % state
    if ms > 70
        k = 50;
    else
        k = floor(ms*0.75);
    end
    for i=k:1:ms
        if state(i) == 0 && state(i-1) == 0 && state(i-2) == 0
            cut_time = state_time(i);
            ms = i;
            state_time = state_time(1:ms);
            state = state(1:ms);
            break;
        end
    end
    [mcs,command_state_time] = unixTime2TestTime(start_s,start_ns,command_state_s',command_state_ns');
    for j = mcs:-1:1
        if abs(command_state_time(j) - cut_time) < 1.0
            command_state = command_state(1:j);
            command_state_time = command_state_time(1:j);
            mcs = j;
            break;
        end
    end

    % status
    [mss,status_time] = unixTime2TestTime(start_s,start_ns,status_s',status_ns');
    for j = mss-1:-1:1
        if status_time(j) < cut_time
            mss = j + 1;
            break;
        end
    end
    voltage = voltage(1:mss);
    motors_status = motors_status(1:mss);
    serial_mode = serial_mode(1:mss);
    waypose_status = waypose_status(1:mss);
    status_time = status_time(1:mss);

    % subject pose
    [mx,sp_time] = unixTime2TestTime(start_s,start_ns,sp_s',sp_ns');
    for j = mx-1:-1:1
        if sp_time(j) < cut_time
            mx = j + 1;
            break;
        end
    end
    sp_time = sp_time(1:mx);
    x = x(1:mx);
    y = y(1:mx);
    z = z(1:mx);
    rotation_sp = rotation_sp(1:mx);

    % task
    [mxt,task_time] = unixTime2TestTime(start_s,start_ns,task_s',task_ns');

    % vicon
    [ma,vicon_time] = unixTime2TestTime(start_s,start_ns,vicon_s',vicon_ns');
    for j = ma-1:-1:1
        if vicon_time(j) < cut_time
            ma = j + 1;
            break;
        end
    end
    vicon_time = vicon_time(1:ma);
    r_x = r_x(1:ma);
    r_y = r_y(1:ma);
    r_z = r_z(1:ma);
    x_v = x_v(1:ma);
```

```
y_v = y_v(1:ma);
z_v = z_v(1:ma);

% power
if v2 == 1
    [mv1,sensor1_time] = unixTime2TestTime(start_s,start_ns,volt1_s',volt1_ns');
    [mv2,sensor2_time] = unixTime2TestTime(start_s,start_ns,volt2_s',volt2_ns');
elseif v2 == 2
    [mv1,sensor1_time] = unixTime2TestTime(start_s,start_ns,power_s',power_ns');
    mv2 = mv1;
    sensor2_time = sensor1_time;
else
    mv1 = length(voltage1');
    mv2 = length(voltage2');
    if mv1 < 600
        v_freq = 0.2;
    elseif mv1 < 1200 && mv1 > 600
        v_freq = 0.1;
    else
        v_freq = 0.05;
    end
    sensor1_time = 0.0:v_freq:mv1*v_freq;
    sensor2_time = 0.0:v_freq:mv2*v_freq;
end
for j = mv1-1:-1:1
    if sensor1_time(j) < cut_time
        mv1 = j + 1;
        break;
    end
end
mv1 = min([mv1,mv2]) ;
mv2 = mv1;
voltage1 = voltage1(1:mv1);
voltage2 = voltage2(1:mv2);
if v2 == 0
    sensor1_time = 0.0:v_freq:mv1*v_freq;
    sensor2_time = 0.0:v_freq:mv2*v_freq;
else
    sensor1_time = sensor1_time(1:mv1);
    sensor2_time = sensor2_time(1:mv2);
end
%       if max(voltage1) > 13 || max(voltage2) > 13
%           fprintf('high voltage on test: %i\n',test)
%       end
%       if min(voltage1) < 9 || min(voltage2) < 9
%           fprintf('low voltage on test: %i\n',test)
%       end
%       if max(current1) > 30 || max(current2) > 30
%           fprintf('high current on test: %i\n',test)
%       end
%       if min(current1) < 9 || min(current2) < 9
%           fprintf('low voltage on test: %i\n',test)
%       end

% quad control input
[mq,qci_time] = unixTime2TestTime(start_s,start_ns,qci_s',qci_ns');
% remove time travel
uniform_time_qci = 0:0.05:100;
uniform_time_qci = uniform_time_qci(1:mq);
%qci_time(1) = 0.0;
flight_begin_qci = 1;
flight_end_qci = mq;
for i = 1:1:mq
    if qci_time(i) > 0.0
        flight_qci = 0;

        for j=1:1:3
            if qci_time(i+j) <= 0.0
                flight_qci = 1;
            end
        end
        if flight_qci == 0
            flight_begin_qci = i;
            break;
        end
    end
end
for i = mq:-1:1
    if qci_time(i) > 0.0 && qci_time(i-1) > 0.0
        flight_end_qci = i;
        break;
    end
end
for i = 1:1:ms
    if state(i,1) == 4
        begin_dual = state_time(i);
        break;
    end
end
for i = 1:1:ms
    if state(i,1) == 7
```

```
        end_dual = state_time(i-1);
        break;
    end
end

% Quad Control Input Time
for i = 1:1:mq
    if qci_time(i) > 0.0
        c_time = qci_time(i);
        k = i;
        break;
    end
end
% approximate control rate used
ctrl_rate = qci_time(floor(mq/2)) - qci_time(floor(mq/2)-1);
i = 1;
while ctrl_rate < 0.01 && i < mq
    ctrl_rate = qci_time(floor(mq/2)+i) - qci_time(floor(mq/2)-(i+1));
    i = i + 1;
end
% change qci time prior to dual publishers and during dual
% publishing
for i = k:-1:1
    uniform_time_qci(i) = c_time - (k-i)*ctrl_rate;
end
for i = 1:1:mq
    if i < k
        qci_time(i) = uniform_time_qci(i);
    end
    if i > k && qci_time(i) <= 0.0
        for j = 1:1:3
            if qci_time(i-j) > 0.0
                if qci_time(i-j) <= end_dual
                    qci_time(i) = qci_time(i-j);
                else
                    qci_time(i) = qci_time(i-j)+ctrl_rate;
                end
                break;
            end
        end
    end
end
% change qci times to fluid times between test and launch
first_qci = 0;
fly_flag = 0;
for i = 1:1:mq
    if qci_time(i) >= begin_dual && qci_time(i) < end_dual
        test_qci = 0;

        for j=1:1:3
            if qci_time(i) ~= qci_time(i+j) && qci_time(i+3) <= end_dual
                test_qci = 1;
            end
        end
        if test_qci == 0
            if first_qci == 0
                first_qci = 1;
            else
                qci_time(i) = qci_time(i-1) + ctrl_rate;
            end
        end
    end
    if fly_flag == 0 && flying(i) == 1
        fly_flag = 1;
        qtime = qci_time(i);
        flight_begin_pose = i;
        for k = i:-1:1
            if qtime - qci_time(k) >= 1.0
                qtime = qci_time(k);
                flight_begin_qci = k;
                break;
            end
        end
        %flight_begin_qci = i;%qci_time(i);
    end
end
for j = mq:-1:1
    if fly_flag == 1 && flying(j) == 1
        fly_flag = 0;
        qtime = qci_time(j);
        flight_end_pose = j;
        for k = j:-1:1
            if qtime - qci_time(k) >= 5.0
                qtime = qci_time(k);
                flight_end_qci = k;
                break;
            end
        end
        %qci_time(j);
    end
end
```

```
        for j = mq-1:-1:1
            if qci_time(j) < cut_time
                mq = j + 1;
                if mq < flight_end_pose
                    flight_end_pose = mq;
                end
                break;
            end
        end

        qci_time = qci_time(1:mq);
        pitch = pitch(1:mq);
        roll = roll(1:mq);
        thrust = thrust(1:mq);
        rotation_qci = rotation_qci(1:mq);
        flying = flying(1:mq);
        control_qci = control_qci(1:mq);
%         [qci_time(flight_begin_qci) qci_time(flight_begin_pose) qci_time(flight_end_qci) qci_time(flight_end_pose)]
%         [vicon_time(1) vicon_time(ma) sp_time(1) sp_time(mx) state_time(1) state_time(ms) command_state_time(1) com
%         [status_time(1) status_time(mss) sensor1_time(1) sensor1_time(mv1) task_time(1) task_time(mxt)]
    end

    %% Failure cases
    fail_flag = zeros(5,4);

    % if communication blocked
    if no_data == 1
        if view_plot == 1
            fprintf('communication from controlling station was blocked')
            fprintf(' by external device\n')
            fprintf('test: %i failed\n',test)
        end
    else % if communication established proceed
        % object has flipped
        gap = 0.6;
        d = 200;
        for i = 1:1:mq
            if (abs(pitch(i,1)) > 0.3 && abs(pitch(i,1)) < 1.0) || (abs(roll(i,1)) > 0.3 && abs(roll(i,1)) < 1.0)
                if (max(r_x) > gap && min(r_x) < -gap) || (max(r_y) > gap && min(r_y) < -gap)
                    for i = 1:1:ma - d
                        if r_x(i) > gap
                            for j = 1:1:d
                                if r_x(i+j) < gap
                                    break;
                                elseif j == d
                                    fail_flag(4,1) = 1;
                                    fail = 1;
                                end
                            end
                        elseif r_y(i) > gap
                            for j = 1:1:d
                                if r_y(i+j) < gap
                                    break;
                                elseif j == d
                                    fail_flag(4,1) = 1;
                                    fail = 1;
                                end
                            end
                        end
                        if fail_flag(4,1) == 1
                            break;
                        end
                    end
                end
            end
        end

        % momentary and sustained extreme rotation around x or y axis
        % rotation failure indicator for misinterpretation of object
        gap = 0.5;
        if (max(r_x) > gap || max(r_y) > gap || min(r_x) < -gap || min(r_y) < -gap)
            pass_count = zeros(4,1);
            fail_flag(1,4) = 1;
            for i = 1500:1:ma - 1500
                if r_x(i,1) > gap
                    pass_count(1,1) = pass_count(1,1) + 1;
                elseif r_x(i,1) < -gap
                    pass_count(2,1) = pass_count(2,1) + 1;
                elseif r_y(i,1) > gap
                    pass_count(3,1) = pass_count(3,1) + 1;
                elseif r_y(i,1) < -gap
                    pass_count(4,1) = pass_count(4,1) + 1;
                else
                    pass_count = zeros(4,1);
                end
                if max(pass_count) == 50;
                    fail = 1;
                    break;
                end
            end
        end
```

```
%--

%--TASK Propagation
% fail indicator for delayed startup and incorrect task propagation
if strcmp(vehicle,'morpheus') == 0
flight_state=zeros(1,2);
for j = 2:1:ms-5
    if state(j,1) == 8 % Extra Task Received
        if state(j+1,1) == 8
            fail = 1;
            fail_flag(2,4) = 1;
        end
        flight_state(1) = flight_state(1) + 1;
    elseif state(j,1) == 7 % Did Not Receive Task
        if state(j+1,1) == 7 && state(j+2,1) == 8 && state(j-1,1) ~= 4
            fail = 1;
            fail_flag(2,3) = 1;
        end
        flight_state(2) = flight_state(2) + 1;
%       elseif state(j,1) == 2 && state(j+5,1) == 2
%          fail_flag(1,4) = 1;
    end
end
if flight_state(1) > 17 && strcmp(vehicle,'morpheus') == 0
    fail = 1;
    fail_flag(2,4) = 1;
end
if flight_state(2) > 20 && strcmp(vehicle,'morpheus') == 0
    fail = 1;
    fail_flag(2,3) = 1;
end
end
% subject status task propagation failure
for i=15:1:mss-15
    if waypose_status(i,1) == 0 && waypose_status(i+1,1) == 0
        if waypose_status(i+2,1) == 1 && waypose_status(i+3,1) == 1 && waypose_status(i+4,1) == 0
            fail_flag(3,3) = 1;
        end
    elseif waypose_status(i,1) == 1 && waypose_status(i+1,1) == 1
        if waypose_status(i+2,1) == 0 && waypose_status(i+3,1) == 0 && waypose_status(i+4,1) == 1
            fail_flag(3,3) = 1;
        end
    end
end

begin_task = 1;
end_task = mxt;
% Missing and extra tasks
for j = 1:1:mxt
    if r_t(j) > 4.0
        if strcmp(vehicle,'morpheus') == 1
            begin_task = j-4;
            break;
        end
        for i = j-1:-1:1
            if r_t(i) == 0
                if i > 2 && r_t(i-1) > 3 && r_t(i-2) == 0 && r_t(i+1) < 3
                    begin_task = i-2;% single zero task inserted
                else
                    begin_task = i;
                end
                break;
            end
        end
        break;
    end
end
for k = mxt:-1:5
    if r_t(k-4) ~= 0.0 && r_t(k) == 0.0 && r_t(k-5) ~= 0.0
        end_task = k;
        break;
    end
end
tasks = end_task - begin_task + 1;
if strcmp(vehicle,'morpheus') == 0
% failure indicator for extra and missing tasks
if tasks > 36
    fail_flag(2,1) = 1;
    fail = 1;
elseif tasks < 36
    fail_flag(2,2) = 1;
    fail = 1;
end

% failure indicator for an extra and a missing task on one run
xyzr_change = zeros(4,2);
xyzr_task = [4 4; 6 4; 6 5; 6 4];
for j = begin_task:1:end_task
    % check for missing x change
    if x_t(j) > x_t(begin_task)
        xyzr_change(1,1) = xyzr_change(1,1) + 1;
```

```
    elseif x_t(j) < x_t(begin_task)
        xyzr_change(1,2) = xyzr_change(1,2) + 1;
    end
    % check for missing y change
    if y_t(j) > y_t(begin_task)
        xyzr_change(2,1) = xyzr_change(2,1) + 1;
    elseif y_t(j) < y_t(begin_task)
        xyzr_change(2,2) = xyzr_change(2,2) + 1;
    end
    % check for missing z change
    if z_t(j) > z_t(begin_task)
        xyzr_change(3,1) = xyzr_change(3,1) + 1;
    end
    % check for missing r change
    if r_t(j) < 2 && r_t(j) > 1
        xyzr_change(4,1) = xyzr_change(4,1) + 1;
    elseif r_t(j) < 5 && r_t(j) > 4
        xyzr_change(4,2) = xyzr_change(4,2) + 1;
    elseif r_t(j) < 4 && r_t(j) > 3
        xyzr_change(3,2) = xyzr_change(3,2) + 1;
    end
end
%xyzr_change - xyzr_task
if sum(sum(abs(xyzr_change - xyzr_task))) ~= 0  && tasks == 36
    fail = 1;
    for i = 1:1:4
        for j = 1:1:2
            if xyzr_change(i,j) - xyzr_task(i,j) > 0
                fail_flag(2,1) = 1;
            elseif xyzr_change(i,j) - xyzr_task(i,j) < 0
                fail_flag(2,2) = 1;
            end
        end
    end
end
%--
end

%-- SIGNAL DELAY
loss = zeros(1,2);max_delay=zeros(2,7);
%[qci_time(flight_begin_qci) qci_time(flight_end_qci) qci_time(flight_begin_pose) qci_time(flight_end_pose) ta

% Vicon packet delay
if strcmp(vehicle,'morpheus') == 1
    [loss,packet_delay(:,6),flags,max_delay(:,6)] = delayed_packet(vicon_time,...
        task_time(begin_task),task_time(end_task),ma,0.05,0);
else
    [loss,packet_delay(:,6),flags,max_delay(:,6)] = delayed_packet(vicon_time,...
        qci_time(flight_begin_pose),qci_time(flight_end_pose),ma,0.05,0);
end
fail_flag(1,3) = flags;

% state packet delay
[loss,packet_delay(:,4),flags,max_delay(:,4)] = delayed_packet(state_time,...
    task_time(begin_task),task_time(end_task),ms,0.2,1);
%fail_flag(3,4) = flags;

% command state packet delay
if strcmp(vehicle,'morpheus') == 1
    [loss,packet_delay(:,3),flags,max_delay(:,3)] = delayed_packet(command_state_time,...
        task_time(begin_task),task_time(end_task),mcs,2.3,0);
else
    [loss,packet_delay(:,3),flags,max_delay(:,3)] = delayed_packet(command_state_time,...
        qci_time(flight_begin_pose),qci_time(flight_end_qci),mcs,1.3,0);
end
fail_flag(3,4) = flags;

% task packet delay
if strcmp(vehicle,'morpheus') == 1
    [loss_task,packet_delay(:,2),flags,max_delay(:,2)] = delayed_packet(task_time,...
        task_time(begin_task),task_time(end_task),mxt,2.3,0);
else
    [loss_task,packet_delay(:,2),flags,max_delay(:,2)] = delayed_packet(task_time,...
        qci_time(flight_begin_pose),qci_time(flight_end_qci),mxt,1.3,0);
end
fail_flag(4,4) = flags;

% subject status packet delay
if strcmp(vehicle,'morpheus') == 1
    [loss,packet_delay(:,5),flags,max_delay(:,5)] = delayed_packet(status_time,...
        task_time(begin_task),task_time(end_task),mss,2.3,0);%0.15
else
    [loss,packet_delay(:,5),flags,max_delay(:,5)] = delayed_packet(status_time,...
        qci_time(flight_begin_pose),qci_time(flight_end_pose),mss,1.3,0);%0.15
end
fail_flag(3,3) = flags;

if strcmp(vehicle,'morpheus') == 0
    % subject pose packet delay
    [loss,packet_delay(:,7),flags,max_delay(:,7)] = delayed_packet(sp_time,...
        qci_time(flight_begin_pose),qci_time(flight_end_pose),mx,...
```

```
        ctrl_rate*10,0);
else
    [loss,packet_delay(:,7),flags,max_delay(:,7)] = delayed_packet(sp_time,...
        task_time(begin_task),task_time(end_task),mx,...
        ctrl_rate*10,0);
end
fail_flag(4,3) = flags;

if strcmp(vehicle,'morpheus') == 0
    % quad control input packet delay
    [loss,packet_delay(:,1),flags,max_delay(:,1)] = delayed_packet(qci_time,...
        qci_time(flight_begin_pose),qci_time(flight_end_qci),mq,...
        ctrl_rate*10,0);
    fail_flag(4,2) = flags;
end
% packet delay failure based on task and quad control input
if fail_flag(4,2) == 1 || fail_flag(4,4) == 1 || fail_flag(3,4) == 1
    fail = 1;
end
if view_plot == 1 && sum(sum(packet_delay)) > 0.0
    fprintf('test: %i qci \t task \t ctrl state \t state \t status \t vicon \t pose\n',test);
    max_delay
    packet_delay
end
%--

%--SUBJECT STATUS
% fail indicator for lost of serial communication, manual
% control taken and motors fail to turn off
if strcmp(vehicle,'morpheus') == 0
    serial_flag = 0;
    motors_flag = 0;
    begin_flight = 0;
    end_flight = state_time(ms);
    for j = 1:1:ms
        if state(j) > 6
            begin_flight = state_time(j);
            break;
        end
    end
    for j = ms:-1:begin_flight
        if state(j) > 6
            end_flight = state_time(j);
            break;
        end
    end
    %[test begin_flight end_flight end_flight - begin_flight]

    if sum(serial_mode) < 37
        mc_time = 0.0;
        fail = 1;
        fail_flag(1,1) = 1; % manual control taken
    end

    for i = 1:1:mss
        % Loss of serial communication
        if serial_mode(i,1) == 1 && serial_flag == 0
            serial_flag = 1; % serial comms established
        end
        if serial_flag == 1 && serial_mode(i,1) == 0
            serial_flag = 2; % serial comms lost
            mc_time = status_time(i);
            for j = i:1:mss
                if serial_mode(j,1) == 1
                    if status_time(i) >= begin_flight && status_time(i) <= end_flight
                        fail = 1;
                    end
                    fail_flag(1,2) = 1; % Loss of serial communication
                    break;
                end
            end
            % manual control taken
            if fail_flag(1,2) == 0 && motors_status(i,1) == 1
                fail = 1;
                fail_flag(1,1) = 1; % manual control taken
            end
        end

        % motors turned off momentarily
        if motors_status(i,1) == 1 && motors_flag == 0
            motors_flag = 1; % motors on
        end
        if motors_flag == 1 && motors_status(i,1) == 0
            motors_flag = 2;
            for j = i:1:mss
                if motors_status(j,1) == 1
                    fail = 1;
                    fail_flag(3,1) = 1; % motors turned off momentarily
                    break;
                end
            end
```

```
                end
            end
        end
        % motors did not turn off
        if motors_status(mss,1) ~= 0 && state(ms,1) == 0
%             fail = 1;
            fail_flag(3,1) = 1;
        end

        % fail indicator for loss of GPS control
        for i = 2:1:mq-5
            if flying(i-1,1) == 1 && flying(i+1,1) == 1 && flying(i,1) == 0
                fail = 1;
                fail_flag(3,2) = 1;
                break;
            end
        end
        %--

        % update fail type count
        fail_type = fail_type + fail_flag;
        % update total failed
        failed = failed + fail;
        % print fail types on test
        if sum(sum(fail_flag)) > 0
            if view_plot == 1
                if fail_flag(1,1) == 1
                    fprintf('manual control taken on test: %i at time: %f\n',test, mc_time)
                end
                if fail_flag(1,2) == 1
                    fprintf(' loss of serial communication on test: %i\n',test)
                end
                if fail_flag(1,3) == 1
                    fprintf('  loss of vicon on test: %i\n',test)
                end
                if fail_flag(1,4) == 1
                    fprintf('    Vicon lost object momentarily on test: %i\n',test)
                end
                if fail_flag(2,1) == 1
                    fprintf('   Extra task(s) added on test: %i\n',test)
                end
                if fail_flag(2,2) == 1
                    fprintf('   missing task on test: %i\n',test)
                end
                if fail_flag(2,3) == 1
                    fprintf('Two hover states back to back on test: %i\n',test)
                end
                if fail_flag(2,4) == 1
                    fprintf('Two Move states back to back on test: %i\n',test)
                end
                if fail_flag(3,1) == 1
                    fprintf('  motor response incorrect while on ground on test: %i\n',test)
                end
                if fail_flag(3,2) == 1 % 3,2
                    fprintf('Competing PID controllers on test: %i\n',test)
                end
                if fail_flag(3,3) == 1 % 3,3
                    fprintf('  Subject status task propagation inaccurate on test: %i\n',test)
                end
                if fail_flag(3,4) == 1 % 3,4
                    fprintf('     command state messages abnormal delay on test: %i\n',test)
                end
                if fail_flag(4,1) == 1
                    fprintf('vehicle may have flipped on test: %i\n',test)
                end
                if fail_flag(4,2) == 1 % 3,2
                    fprintf('  quad control input messages abnormal delay on test: %i\n',test)
                end
                if fail_flag(4,3) == 1 % 3,3
                    fprintf('     Subject pose messages dropped on test: %i\n',test)
                end
                if fail_flag(4,4) == 1 % 3,3
                    fprintf('     task waypose messages abnormal delay on test: %i\n',test)
                end
                if fail == 1
                    fprintf('test: %i failed\n',test)
                end
            end
        end
    end
    %% time calculation
    ot(test_index,2) = fail;
    if no_data == 0%
        ot(test_index,1) = max([vicon_time(ma) sp_time(mx) state_time(ms) status_time(mss) task_time(mxt)]);
%         if ot(test_index,1) >= 70 && view_plot == 1
%             [test ot(test_index,1)]
%         end
    end

%     % calculate time spent in each state
%     if test_index == 1
```

```
%          ti = 0;
%      end
%
%      if fail == 0
%          current_state_index = 1;
%          transition = 0;
%          ti = ti +1;
%          for i = 2:1:ms
%              if state(i,1) ~= state(i-1,1)
%                  transition = transition + 1;
%                  trans_time(ti,transition,1) = state(i-1);
%                  trans_time(ti,transition,2) = state_time(i) - state_time(current_state_index);
%                  trans_time(ti,transition,3) = state(i);
%                  current_state_index = i;
%              end
%          end
%      end

      %%% Power Calculation Totals
      oap(test_index,3) = fail;
      pmax(test_index,3) = fail;

      %[max(voltage1) max(voltage2);min(voltage1) min(voltage2)]
      %[max(current1) max(current2);min(current1) min(current2)]

      % power sensor 1
      [voltage1, current1, mv1] = equalizePower(voltage1,current1,v2);
      mc1 = mv1;

      % power sensor 2
      [voltage2, current2, mv2] = equalizePower(voltage2,current2,v2);
      mc2 = mv2;
      m = min(mv1,mv2);

      % separate hover and move task operating average power
      if fail == 0
          begin_sensor_time=sync_time(qci_time,sensor1_time,flight_begin_qci);
          end_sensor_time=sync_time(qci_time,sensor1_time,flight_end_qci);

          toggle_time = sensor1_time(begin_sensor_time);
          toggle_flag = 1;
          power_task = zeros(2,2);
          task_count = zeros(1,2);
          if strcmp(vehicle,'morpheus') == 0
              move_time = 0.9;
          else
              move_time = 1.9;
          end
          for i = begin_sensor_time:1:end_sensor_time
              if sensor1_time(i) - sensor1_time(begin_sensor_time) < 36.0 && sensor1_time(i) - toggle_time > move_time
                  if toggle_flag == 1
                      toggle_flag = 2;
                      move_time = 0.9;
                  else
                      toggle_flag = 1;
                      if strcmp(vehicle,'morpheus') == 0
                          move_time = 0.9;
                      else
                          move_time = 1.9;
                      end
                  end
                  toggle_time = sensor1_time(i);
              elseif sensor1_time(i) - toggle_time > 0.9
                  toggle_flag = 2;
              end
              power_task(toggle_flag,1) = power_task(toggle_flag,1) + current1(i)*voltage1(i);
              power_task(toggle_flag,2) = power_task(toggle_flag,2) + current2(i)*voltage2(i);
              task_count(toggle_flag) = task_count(toggle_flag) + 2;
          end
          task_power(test_index,1) = (power_task(1,1) + power_task(1,2)) / task_count(1);
          task_power(test_index,2) = (power_task(2,1) + power_task(2,2)) / task_count(2);

          L=begin_sensor_time;
          R=end_sensor_time;
          %[test L R]
          % estimate operating average power
          oav = sum(voltage1(L:R))/(R-L+1);
          oac = sum(current1(L:R))/(R-L+1);
          oap(test_index,1) = oav.*oac;
          oav = sum(voltage2(L:R))/(R-L+1);
          oac = sum(current2(L:R))/(R-L+1);
          oap(test_index,2) = oav.*oac;

          % find maximum power
          pmax(test_index,1) = max(voltage1.*current1);
          pmax(test_index,2) = max(voltage2.*current2);
          %--
      end

      %%% Position Calculation Totals
      if test_index == 1
```

```
        max_x = 0;max_y = 0;max_r = 0;
    end
    if fail == 0
    se_dif_r = rotation_sp(1) - rotation_sp(mx);
    if se_dif_r > pi()
        se_dif_r = se_dif_r - 2*pi();
    elseif se_dif_r < -pi()
        se_dif_r = se_dif_r + 2*pi();
    end
%     if abs(x_v(1) - x_v(ma)) > max_x
%         max_x = abs(x_v(1) - x_v(ma));
%         %[test max_x max_y max_r]
%     end
%     if abs(y_v(1) - y_v(ma)) > max_y
%         max_y = abs(y_v(1) - y_v(ma));
%         %[test max_x max_y max_r]
%     end
    if abs(se_dif_r) > max_r
        max_r = abs(se_dif_r);
        %[test max_x max_y max_r]
    end
    %[test x_v(1) - x_v(ma) y_v(1) - y_v(ma) se_dif_r;]
    end

    precision(test_index,5) = fail;
    if fail == 0
        % reset off zero
        for k = 1:1:mxt
            if x_t(k,1) ~= 0.0
                x_t(1:k-1,1) = x_t(k,1);
                y_t(1:k-1,1) = y_t(k,1);
                z_t(1:k-1,1) = 0.0; %
                z_t(end_task + 2:mxt)=0.0;%
                break;
            end
        end

        % time synchronization
        k = 1;
        pos_sync = zeros(mxt,2);
        for i = begin_task-1:1:end_task+1
            for j = k:1:mx
                if i > 1 && i < mxt && task_time(i) < sp_time(j)
                    pos_sync(i,1) = i;
                    pos_sync(i,2) = j;
                    k = j;
                    break;
                end
            end
        end

        % generate task array same length as position array for each
        % dimension
        start = pos_sync(begin_task,2);
        stop = pos_sync(end_task,2);
        move_count = start;
        prev_count = 0;

        %ma=mx;vicon_time=sp_time;x_v=x;y_v=y;z_v=z; r_v = rotation_sp;
        % initialize task arrays
        xyzr_f = zeros(mx,4);
        xyzr_f(1:ma,1) = x_t(1,1);
        xyzr_f(1:ma,2) = y_t(1,1);
        xyzr_f(1:ma,3) = z_t(1,1);
        if rotation_sp(1,1) > 6
            xyzr_f(1:ma,4) = 2*pi();
        else
            xyzr_f(1:ma,4) = 0.0;
        end
        % construct linear task arrays
        for j = begin_task:1:end_task - 1
            d_x = x_t(j+1,1) - x_t(j,1);
            d_y = y_t(j+1,1) - y_t(j,1);
            d_z = z_t(j+1,1) - z_t(j,1);
            if j == begin_task
                d_r = r_t(j+1,1) - r_t(j-1,1);
            else
                d_r = r_t(j+1,1) - r_t(j,1);
            end
            sync_diff = pos_sync(j+1,2) - pos_sync(j,2)-1;
            if sync_diff < 1
                [test begin_task end_task j sync_diff]
            end
            while move_count < prev_count + sync_diff + 1
                move_count = move_count + 1;
                xyzr_f(move_count,1) = x_t(j,1)...
                    + d_x*((move_count-prev_count)/sync_diff);
                xyzr_f(move_count,2) = y_t(j,1)...
                    + d_y*((move_count-prev_count)/sync_diff);
                xyzr_f(move_count,3) = z_t(j,1)...
                    + d_z*((move_count-prev_count)/sync_diff);
```

```
            if j ~= begin_task && j ~= begin_task + 1
                xyzr_f(move_count,4) = r_t(j,1)...
                    + d_r*((move_count-prev_count)/sync_diff);
            end
        end
        prev_count = move_count;
    end

    % time syncronization offset
    offset = 901;

    % Subtract actual from ideal position
    fs = zeros(ma-offset+1,4);
    fs(:,1) = xyzr_f(1:ma-offset+1,1) - x_v(offset:ma,1);
    fs(:,2) = xyzr_f(1:ma-offset+1,2) - y_v(offset:ma,1);
    fs(:,3) = xyzr_f(1:ma-offset+1,3) - z_v(offset:ma,1);

    % subtract actual
    r_task = xyzr_f(1:ma,4);
    r_actual = rotation_sp(offset:mx);

    % Subtract actual from ideal rotation
    i_r = min(length(r_task),length(r_actual));
    mf = length(fs');
    for i = 1:1:i_r
        if r_task(i,1) == 0 && r_actual(i,1) > 6
            fs(i,4) = 2*pi() - r_actual(i,1);
        elseif r_task(i,1) > 6 && r_actual(i,1) < 1
            fs(i,4) = r_actual(i,1) - 0.0;
        else
            fs(i,4) = r_task(i,1) - r_actual(i,1);
        end
        if fs(i,4) > pi()
            if r_task(i,1) > 6
                fs(i,4) = r_actual(i,1) - 0.0;
            end
        end
    end

    % calculate error measure, MAE
    precision(test_index,1) = sum(abs(fs(:,1)))/mf;%sqrt(sum(fs(:,1).*fs(:,1)))/mf;
    precision(test_index,2) = sum(abs(fs(:,2)))/mf;%sqrt(sum(fs(:,2).*fs(:,2)))/mf;
    precision(test_index,3) = sum(abs(fs(:,3)))/mf;%sqrt(sum(fs(:,3).*fs(:,3)))/mf;
    precision(test_index,4) = sum(abs(fs(:,4)))/mf;%sqrt(sum(fs(:,4).*fs(:,4)))/mf;

    % plot time synchronization position graphs
    if view_plot == 1
        figure
        plot(fs(:,1),'k.')
        hold on
        plot(fs(:,2),'cx')
        plot(fs(:,3),'b+')
%         if loss_vicon(1) ~= 0
%             for i=1:1:length(loss_vicon)
%                 %
%                 plot(loss_vicon(i),min(min(fs)):max(max(fs)))
%             end
%         end

        figure
        plot(fs(:,4),'k*')

        % rotation actual versus tasked
        figure,%[length(r_task) length(r_actual) length(sp_time(offset:mx)) length(vicon_time)]
        plot_line(loss_task,r_task)
        hold on
        plot(vicon_time,r_task,'bx')
        grid on
        grid minor
        plot(sp_time(1:mx-offset+1),r_actual,'c')
        hold off

        % print figure
        if save_plot == 1
            print(sprintf('../plot/%s/%s%i/rotation_error_%s%i',...
                test_set,vehicle,test,vehicle,test),'-djpeg')
        end

        % spatial actual versus tasked
        figure
        subplot(3,1,1) % x
        plot_line(loss_task,x_v)
        hold on
        plot(vicon_time,x_v,'c')
        grid on
        grid minor
        plot(vicon_time(offset:ma),xyzr_f(1:ma-offset+1,1),'b')
        hold off

        subplot(3,1,2) % y
        plot_line(loss_task,xyzr_f(:,2))
```

```
            hold on
            plot(vicon_time,y_v,'c')
            grid on
            grid minor
            plot(vicon_time(offset:ma),xyzr_f(1:ma-offset+1,2),'b')
            hold off

            subplot(3,1,3) % z
            plot_line(loss_task,xyzr_f(:,3))
            hold on
            plot(vicon_time,z_v,'c')
            grid on
            grid minor
            plot(vicon_time(offset:ma),xyzr_f(1:ma-offset+1,3),'b')
            hold off

            % print figure
            if save_plot == 1
                print(sprintf('../plot/%s/%s%i/position_error_%s%i',...
                    test_set,vehicle,test,vehicle,test),'-djpeg')
            end
        end
    end

    %% control
    if view_plot == 1 && no_data == 0
        figure
        plot_line(loss_task,control_qci)
        hold on
        plot(qci_time,control_qci,'b.')
        grid on
        grid minor
        title('Quad Control Input Control Sequence')
        xlabel('Time (seconds)')
        ylabel('Sequence Number')
        % print figure
        if save_plot == 1
            print(sprintf('../plot/%s/%s%i/control_%s%i',...
                test_set,vehicle,test,vehicle,test),'-djpeg')
        end
    end

    %% voltage, current, power

    % voltage
    if view_plot == 1 && no_data == 0
        figure
        subplot(2,2,2),
%         if v2 == 1
            plot_line(loss_task,voltage1)
            plot(sensor1_time(1:mv1),voltage1,'k+')
%         else
%             plot(voltage1,'k+')
%         end
        hold on
%         if v2 == 1
            plot(sensor2_time(1:mv2),voltage2,'cx')
            xlabel('Time (seconds)')
%         else
%             plot(voltage2,'cx')
%             xlabel('Discrete Time Sample (n)')
%         end
        grid on
        grid minor
        title('Sensor Voltage')
        ylabel('Voltage (volts)')
        %legend('Sensor 1','Sensor 2')
        hold off

        % current
        subplot(2,2,3),
%         if v2 == 1
            plot_line(loss_task,current1)
            plot(sensor1_time(1:mv1),current1,'k+')
%         else
%             plot(current1,'k+')
%         end
        hold on
%         if v2 == 1
            plot(sensor2_time(1:mv2),current2,'cx')
            xlabel('Time (seconds)')
%         else
%             plot(current2,'cx')
%             xlabel('Discrete Time Sample (n)')
%         end

        grid on
        grid minor
        title('Sensor Current')
        ylabel('Current (Amperes)')
        % legend('Sensor 1','Sensor 2')
```

```
        hold off

        %[L, LP1,R]
        % plot power figure
        subplot(2,2,1),
%         if v2 == 1
            plot(sensor1_time(1:mv1),current1.*voltage1,'k+')
%         else
%             plot(current1.*voltage1,'k+')
%         end
        hold on
%         if v2 == 1
            plot(sensor2_time(1:mv2),current2.*voltage2,'cx')
            xlabel('Time (seconds)')
%         else
%             plot(current2.*voltage2,'cx')
%             xlabel('Discrete Time Sample (n)')
%         end
        if fail == 0
            plot(sensor1_time(L),100,'r^')
            %plot(LP1,100,'b^')
            plot(sensor1_time(R),100,'y^')
        end
        grid on
        grid minor
        title('Apparent Power')
        ylabel('Power (Watts)')
        %legend('Sensor 1','Sensor 2')
        hold off

        % "battery" voltage
        subplot(2,2,4),
        plot_line(loss_task,voltage)
        hold on
        plot(status_time,voltage,'b-')
        grid on
        grid minor
        title('Voltage (via Status Node)')
        xlabel('Time (seconds)')
        ylabel('Voltage (volts)')
        %legend('Status Voltage')

        % print figure
        if save_plot == 1
            print(sprintf('../plot/%s/%s%i/power_%s%i',...
                test_set,vehicle,test,vehicle,test),'-djpeg')
        end
    end

    %% position Plot

    % 3-D position
    if view_plot == 1 && no_data == 0
        figure
        plot3(x,y,z,'c-')
        hold on
        plot3(x_t,y_t,z_t,'b-')
        grid on
        grid minor
        title(sprintf('Path of %s%i',vehicle,test))
        xlabel('Position in X Direction (meters)')
        ylabel('Position in Y Direction (meters)')
        zlabel('Position in Z Direction (meters)')
        %legend('Actual Path', 'Ideal Path')

        % print
        if save_plot == 1
            print(sprintf('../plot/%s/%s%i/task_path_%s%i',...
                test_set,vehicle,test,vehicle,test),'-djpeg')
        end

        % 1-D position with respect to time
        figure
        m = size(x);
        subplot(2,2,2),
        plot_line(loss_task,z)
        hold on
        plot(sp_time,z,'b+')
        plot(sp_time,y,'cx')
        plot(sp_time,x,'k.')
        grid on
        grid minor
        title('Subject Pose')
        xlabel('Time (seconds)')
        ylabel('Position in X, Y, and Z Direction (meters)')
        axis([0 sp_time(mx) min(min(min(z,y),x)) - .1 max(max(max(z,y),x)) + .1])
        %legend('X Position','Y Position','Z Position')
        hold off

        % pitch roll and thrust
        subplot(2,2,4),
```

```
    plot_line(loss_task,thrust)
    hold on
    plot(qci_time,thrust,'b+')
    plot(qci_time,pitch,'cx')
    plot(qci_time,roll,'k.')
    axis([0 qci_time(mq) -1.1 1.1])
    grid on
    grid minor
    title('Quad Control Input Pitch, Roll, and Thrust')
    xlabel('Time (seconds)')
    ylabel('Variance(unit circle)')
    % legend('Thrust','Pitch','Roll')
    hold off

    % vicon position
    subplot(2,2,3), plot_line(loss_task,z_v)
    hold on
    plot(vicon_time,y_v,'cx')
    plot(vicon_time,x_v,'k.')
    plot(vicon_time,z_v,'b+')
    axis([0 vicon_time(ma) min(min(min(z_v,y_v),x_v)) - .1 max(max(max(z_v,y_v),x_v)) + .1])
    grid on
    grid minor
    title('Vicon Position')
    xlabel('Time (seconds)')
    ylabel('Position in X, Y, and Z Direction (meters)')
    %legend('Position in X','Position in Y','Position in Z')
    hold off

    % Task 1-D position with respect to time
    subplot(2,2,1), plot_line(loss_task,z_t)
    hold on
    plot(task_time,y_t,'cx')
    plot(task_time,x_t,'k.')
    plot(task_time,z_t,'b+')
    axis([0 task_time(mxt) min(min(min(z_t,y_t),x_t)) - .1 max(max(max(z_t,y_t),x_t)) + .1])
    grid on
    grid minor
    title('Task Waypose')
    xlabel('Time (seconds)')
    ylabel('Position in X, Y, and Z Direction (meters)')
    %legend('X Position','Y Position','Z Position')
    hold off

    % print figure
    if save_plot == 1
        print(sprintf('../plot/%s/%s%i/position_%s%i',...
            test_set,vehicle,test,vehicle,test),'-djpeg')
    end
end

%% rotation

% rotation (subject_pose)
if view_plot == 1 && no_data == 0
    figure
    subplot(2,2,2), plot_line(loss_task,rotation_sp)
    hold on
    plot(sp_time,rotation_sp,'k.')
    axis([0 sp_time(mx) min(rotation_sp) - .1 max(rotation_sp) + .1])
    grid on
    grid minor
    title('Subject Pose Rotation')
    xlabel('Time (seconds)')
    ylabel('Rotation Around Z Axis (radians)')
    %legend('Rotation')

    % rotation control input
    subplot(2,2,4), plot_line(loss_task,rotation_qci)
    hold on
    plot(qci_time,rotation_qci,'cx')
    grid on
    grid minor
    axis([0 qci_time(mq) min(rotation_qci) - .1 max(rotation_qci) + .1])
    title('Quad Control Input Rotation')
    xlabel('Time (seconds)')
    ylabel('Rotation Around Z Axis (unit circle)')
    %legend('Rotation')

    % rotation task
    subplot(2,2,1), plot_line(loss_task,r_t)
    hold on
    plot(task_time,r_t,'b+')
    axis([0 task_time(mxt) min(r_t) - .1 max(r_t) + .1])
    grid on
    grid minor
    title('Task Rotation')
    xlabel('Time (seconds)')
    ylabel('Rotation Around Z Axis (radians)')
    %legend('Rotation')
```

```
        % vicon rotation
        subplot(2,2,3), plot_line(loss_task,r_z)
        hold on
        plot(vicon_time,r_z,'b+')
        axis([0 vicon_time(ma) min(min(min(r_z,r_y),r_x)) - .1 max(max(max(r_z,r_y),r_x)) + .1])
        plot(vicon_time,r_y,'cx')
        plot(vicon_time,r_x,'k.')
        grid on
        grid minor
        title('Vicon Rotation')
        xlabel('Time (seconds)')
        ylabel('Rotation Around an Axis (unit circle)')
        %legend('Rotation Around X','Rotation Around Y','Rotation Around Z')
        hold off
        % print figure
        if save_plot == 1
            print(sprintf('../plot/%s/%s%i/rotation_%s%i',...
                test_set,vehicle,test,vehicle,test),'-djpeg')
        end
    end

    %% state
    if view_plot == 1 && no_data == 0
        figure
        subplot(3,1,1), plot_line(loss_task,state)
        hold on
        plot(command_state_time,command_state,'b+')
        plot(state_time,state,'k.')
        grid on
        grid minor
        axis([0 state_time(ms) 0 - .5 8 + .5])
        title(sprintf('State Sequence of %s%i',vehicle,test))
        xlabel('Time (seconds)')
        ylabel('Current State')
        %legend('State')

        % binary status
        subplot(3,1,2), plot_line(loss_task,motors_status)
        hold on
        plot(status_time,motors_status,'k-')
        grid on
        grid minor
        plot(status_time,serial_mode,'c-')
        plot(status_time,waypose_status,'bx')
        axis([0 status_time(mss) -0.3 1.3])
        title('Binary Status Flags')
        xlabel('Time (seconds)')
        ylabel('Boolean Value')
        %legend('Motors','Serial Mode','Waypose')
        hold off

        % gps control Status
        subplot(3,1,3), plot_line(loss_task,flying)
        hold on
        plot(qci_time,flying,'b-')
        grid on
        grid minor
        axis([0 qci_time(mq) -0.3 1.3])
        title('Quad Control Input GPS Control Status')
        ylabel('Currently Flying')
        xlabel('Time (seconds)')
        %legend('GPS Control Status')
    end

    % print figure
    if save_plot == 1 && no_data == 0
        print(sprintf('../plot/%s/%s%i/state_status_%s%i',...
            test_set,vehicle,test,vehicle,test),'-djpeg')
    end
end

%% failure
fail_type(5,2) = failed;
fail_type(5,3) = n_test;
fail_type(5,4) = (failed/n_test)*100;
[fail_type(5,2) fail_type(5,4)]
flight_stats(1:5,:) = fail_type;

% if view_plot == 1
%     figure
%     bar(fail_type(1:2,:))
%     grid on
%     grid minor
%     title('Failure Types')
%     ylabel('Number of Failures')
%     xlabel('ManContr LostSeri LstVicon SlwStart TaskSent TaskRece NRecTask NSendTsk')
%
%     % print figure
%     if save_plot == 1
%         print(sprintf('plot/%s/failure_%s',test_set,vehicle),'-djpeg')
%     end
```

```
% end
%% Power
% average operating power
figure
subplot(2,1,1)

% segment pass and fail test runs and plot data
pass_count = 0;
fail_count = 0;
pass = zeros(mv1,4);
fail= zeros(mv1,4);
for i = 1:1:n_test
    if oap(i,3) == 0
        plot(i,oap(i,1),'k+')
        if i == 1
            hold on
        end
        plot(i,oap(i,2),'cx')
        pass_count = pass_count + 1;
        pass(pass_count,1:2) = oap(i,1:2);
        pass(pass_count,3:4) = task_power(i,:);
        if oap(i,1) == 0 || oap(i,2) == 0
            [vehicle]
            [i]
        end
    else
        plot(i,oap(i,1),'r+')
        if i == 1
            hold on
        end
        plot(i,oap(i,2),'rx')
        fail_count = fail_count + 1;
        fail(fail_count,1:2) = oap(i,1:2);
        fail(fail_count,3:4) = task_power(i,:);
    end
end
hold off
pass = pass(1:pass_count,:);
fail = fail(1:fail_count,:);

% record power related statistics
power_stats = zeros(4,4);
power_stats(1,:) = mean(pass);
power_stats(2,:) = var(pass);
power_stats(4,1:2) = [min(min(pass(:,1:2))) max(max(pass(:,1:2)))];

% add details to plot
grid on
grid minor
title('Average Operating Apparent Power')
xlabel('Test run')
ylabel('Power (Watts)')

subplot(2,1,2)
% maximum power
% segment pass and fail test runs and plot data
pass_count = 0;
fail_count = 0;
pass = zeros(n_test,2);
fail= zeros(n_test,2);
for i = 1:1:n_test
    if pmax(i,3) == 0
        plot(i,pmax(i,1),'k+')
        if i == 1
            hold on
        end
        plot(i,pmax(i,2),'cx')
        pass_count = pass_count + 1;
        pass(pass_count,:) = pmax(i,1:2);
    else
        plot(i,pmax(i,1),'r+')
        if i == 1
            hold on
        end
        plot(i,pmax(i,2),'rx')
        fail_count = fail_count + 1;
        fail(fail_count,:) = pmax(i,1:2);
    end
end
hold off
pass = pass(1:pass_count,:);
fail = fail(1:fail_count,:);

% record power related statistics
power_stats(3,:) = [mean(pass) min(min(pass)) max(max(pass))];
power_stats(4,3:4) = var(pass);
flight_stats(6:9,:) = power_stats;

% text(1, min(pmax(:,1)) + 0.7,...
%     sprintf('Maximum Power: Average = [%f %f],Range = [%f,%f]',...
%     power_stats(3,1),power_stats(3,2),power_stats(3,3),power_stats(3,4)))
```

```
grid on
grid minor
title('Maximum Apparent Power')
xlabel('Test run')
ylabel('Power (Watts)')

% print figure
if save_plot == 1
    print(sprintf('../plot/%s/maximum_power_%s',test_set,vehicle),'-djpeg')
end

%% position
% [max_x max_y max_r]
% plot and segment pass and fail tests prior to generating statistics
figure
pass_count = 0;
fail_count = 0;
pass = zeros(n_test,4);
fail= zeros(n_test,4);
for i = 1:1:n_test
    if precision(i,5) == 0
        plot(i,precision(i,1),'k.')
        if i == 1
            hold on
        end
        plot(i,precision(i,2),'cx')
        plot(i,precision(i,3),'b+')
        pass_count = pass_count + 1;
        pass(pass_count,:) = precision(i,1:4);
        %[i precision(i,1:3)/ot(i,1)*1000 precision(i,4)/ot(i,1)]
    else
        plot(i,precision(i,1),'r.')
        if i == 1
            hold on
        end
        plot(i,precision(i,2),'rx')
        plot(i,precision(i,3),'r+')
        fail_count = fail_count + 1;
        fail(fail_count,:) = precision(i,1:4);
    end
end
pass = pass(1:pass_count,:);
fail = fail(1:fail_count,:);

% calculate and store position statistics
error_stats = zeros(4,4);
error_stats(1,:) = mean(pass);%pass
error_stats(1,1:3) = error_stats(1,1:3)*100;
error_stats(2,:) = var(pass);
error_stats(2,1:3) = error_stats(2,1:3)*100;
total_error = pass(:,1).*pass(:,1) + pass(:,2).*pass(:,2) + pass(:,3).*pass(:,3);
error_stats(3,:) = [min(min(pass(:,1:3)))*100 max(max(pass(:,1:3)))*100 0 mean(sqrt(total_error))*100];
flight_stats(10:12,:) = error_stats(1:3,:);

grid on
grid minor
title('Position Precision Estimate')
xlabel('Test run')
ylabel('Position (Meters)')

%% time
% plot and segment pass and fail tests prior to generating statistics
figure
pass_count = 0;
fail_count = 0;
pass = zeros(n_test,1);
fail= zeros(n_test,1);
for i = 1:1:n_test
    if ot(i,2) == 0
        plot(i,ot(i,1),'kx')
        if i == 1
            hold on
        end
        pass_count = pass_count + 1;
        pass(pass_count,1) = ot(i,1);
    else
        plot(i,ot(i,1),'rx')
        if i == 1
            hold on
        end
        fail_count = fail_count + 1;
        fail(fail_count,1) = ot(i,1);
    end
end
pass = pass(1:pass_count,:);
fail = fail(1:fail_count,:);

% calculate and store time
time_stats = zeros(1,4);
time_stats(1,:) = [mean(pass) var(pass) min(min(pass)) max(max(pass))];
```

```
% position normalized statistics calculated here due to needing mean(time) value
error_stats(4,1:3) = error_stats(1,1:3)*10/mean(pass);
error_stats(4,4) = error_stats(1,4)/mean(pass);
flight_stats(12,3) = error_stats(3,4)*10/mean(pass);
flight_stats(14,:) = time_stats;
flight_stats(13,:) = error_stats(4,:);

% format plot
text(1, min(ot(:,1)) + 0.1,sprintf('Average = %f, Range = [%f %f]',...
    time_stats(1,1),time_stats(1,3),time_stats(1,4)))

% edit plot
grid on
grid minor
title('Time to Perform Task Set')
xlabel('Test run')
ylabel('Time (secs)')

%trans_time
%[max(trans_time(:,:, 2)); min(trans_time(:,:, 2));mean(trans_time(:,:, 2));mode(round(trans_time(:,:, 2)))]

% print figure
if save_plot == 1
    print(sprintf('../plot/%s/total_time_%s',test_set,vehicle),'-djpeg')
end


function [ signal ] = smoothNoise(signal,upper_bound,lower_bound,sizeSig)
%smoothNoise: smooth noisy data points
%   smooth out signal noise outside of [lower_bound, upper_bound]
    for i = 1:1:sizeSig
        if signal(i,1) > upper_bound || signal(i,1) < lower_bound
            if i > 1
                signal(i,1) = signal(i-1,1);
            else
                signal(i,1) = signal(i+1,1);
            end
        end
    end
end


function [index2] = sync_time(time1,time2,index1)
%sync time given an index of one time vector return corresponding index
%   in secopnd time vector
    for i=1:1:length(time2)
        if time2(i) > time1(index1)
            index2 = i;
            return;
        end
    end
    index2 = length(time2);
end


function uav_comparison(select_vehicle,save_plot,view_plot,fname)
%uav_comparison returns UAV stats
%   Output: Excel file with failure, power, position, and time statistics
%   of the selected unmanned aerial systems tested
n = length(select_vehicle');
comparison =  zeros(n,13,4);

% non volatile column values
vehicles = char({'CRASH'; 'HERBIE'; 'JENNY'; 'UNL_COMP_SCI'; 'HULK_SMASH'; 'MORPHEUS'});
propellers = char({'AscTec Safety Propellers'; 'ARDrone Propellers';...
    'AscTec Normal Propellers'; 'AscTec Safety Propellers';...
    'AscTec Safety Propellers'; 'ARDrone Propellers'});
firmwares = char({'AutoPilot LowLevel #20636 V3.12, AutoPilot HighLevel V3.11';...
    'AutoPilot LowLevel #20632 V3.12, AutoPilot HighLevel V3.0';...
    'AutoPilot LowLevel #20633 V2.14, AutoPilot HighLevel V3.0';...
    'AutoPilot LowLevel #20637 V3.12, AutoPilot HighLevel V3.0';...
    'AutoPilot LowLevel #20502 V3.12, AutoPilot HighLevel V3.11';...
    'ARDrone'});
costs = [inf; inf; inf; inf; inf; inf];

% variable row selection matrices
vehicle=char(zeros(n,length(vehicles(1,:))));
propeller=char(zeros(n,length(propellers(1,:))));
firmware=char(zeros(n,length(firmwares(1,:))));
cost=zeros(n,1);

% determine flight characteristics of selected vehicles
for j = 1:1:n
    i = select_vehicle(j);
    vehicle(j,:) = vehicles(i,:);
    propeller(j,:) = propellers(i,:);
    firmware(j,:) = firmwares(i,:);
    cost(j,:) = costs(i,1);

    % determine flight characteristics of selected vehicle for a specific range of tests
    if i == 1
```

```
            [comparison(j,:,:)]=plot_topic('crash','crash',1,510,...
                save_plot,view_plot);
            elseif i == 2
                [comparison(j,:,:)]=plot_topic('herbie','herbie',1,100,save_plot,view_plot);
            elseif i == 3
                [comparison(j,:,:)]=plot_topic('jenny','jenny',1,110,save_plot,view_plot);
            elseif i == 4
                [comparison(j,:,:)]=plot_topic('unl_comp_sci','unl_comp_sci',1,100,save_plot,view_plot);
            elseif i == 5
                [comparison(j,:,:)]=plot_topic('hulk_smash','hulk_smash',1,100,save_plot,view_plot);
            elseif i == 6
                [comparison(j,:,:)]=plot_topic('morpheus','morpheus',1,100,save_plot,view_plot);
        end
    end

    % Failure Statistics Arrays
    reliability = comparison(:,3,4);
    failedTests = comparison(:,3,2);
    totalTests = comparison(:,3,3);
    manualControl = comparison(:,1,1);
    serialCommunicationLost = comparison(:,1,2);
    viconDelay = comparison(:,1,3);
    viconLostObject = comparison(:,1,4);
    extraTaskSent = comparison(:,2,1);
    extraStateReceived = comparison(:,2,4);
    didNotReceiveState = comparison(:,2,3);
    didNotSendTask = comparison(:,2,2);
    motorsStayedOn = comparison(:,3,1);
    vehicleFlipped = comparison(:,4,1);
    quadControlInputDelay = comparison(:,4,2);
    subjectPoseDelay = comparison(:,4,3);
    taskedPoseDelay = comparison(:,4,4);

     title_fail = {'Vehicle Type and Failure Statistics'};
     units_fail = {'(type)' '(version)' '(U.S. Dollars)' '(%)' '(count)' ...
         '(count)' '(count)' '(count)' '(count)' '(count)' '(count)'...
         '(count)' '(count)' '(count)' '(count)' '(count)'};
     uas_fail = table(vehicle,propeller,firmware,cost,reliability,failedTests,...
         totalTests,manualControl,serialCommunicationLost,...
         extraTaskSent,extraStateReceived,didNotReceiveState,didNotSendTask,...
         viconLostObject,viconDelay,motorsStayedOn,vehicleFlipped,...
         quadControlInputDelay,subjectPoseDelay,taskedPoseDelay);

    % Power Statistics Arrays
    meanPower = (comparison(:,5,1)+comparison(:,5,2))/2;
    moveMean = comparison(:,5,3);
    hoverMean = comparison(:,5,4);
    meanPowerRange = comparison(:,8,1:2);
    meanPowerVariance = (comparison(:,6,1)+comparison(:,6,2))/2;
    movePowerVariance = comparison(:,6,3);
    hoverPowerVariance = comparison(:,6,4);
    maximumPower = (comparison(:,7,1)+comparison(:,7,2))/2;
    maximunPowerRange = comparison(:,7,3:4);
    maximumPowerVariance = (comparison(:,8,3)+comparison(:,8,4))/2;

     title_power = {'Power Statistics'};
     units_power = {'(Watts)' '(Watts)' '(Watts)' '(Watts)' '(Watts)' '  '...
         '  ' '  ' '(Watts)' '(Watts)' '(Watts)' '  '};
     uas_power = table(vehicle,meanPower,moveMean,hoverMean,meanPowerRange,...
         meanPowerVariance,movePowerVariance,hoverPowerVariance,maximumPower,...
         maximunPowerRange,maximumPowerVariance);

     % Position Statistics Arrays
     totalSpatialError = comparison(:,11,4);
    totalMeanError_x = comparison(:,9,1);
    totalMeanError_y = comparison(:,9,2);
    totalMeanError_z = comparison(:,9,3);
    totalMeanError_rotation = comparison(:,9,4);
    spatialErrorPerTask = comparison(:,11,3);
    meanErrorPerTask_x = comparison(:,12,1);
    meanErrorPerTask_y = comparison(:,12,2);
    meanErrorPerTask_z = comparison(:,12,3);
    meanErrorPerTask_rotation = comparison(:,12,4);
    varianceErrorPerTask_x = comparison(:,10,1);
    varianceErrorPerTask_y = comparison(:,10,2);
    varianceErrorPerTask_z = comparison(:,10,3);
    varianceErrorPerTask_rotation = comparison(:,10,4);

     title_position = {'Position Error Statistics'};
     units_position = {'(centimeters - cm)' '(cm)' '(cm)' '(cm)' '(radians)'...
         '(millimeters - mm)' '(mm)' '(mm)' '(mm)' '(radians)' '  '  '  '  '  '};
     uas_position = table(vehicle,totalSpatialError,...
         totalMeanError_x,totalMeanError_y,totalMeanError_z,...
         totalMeanError_rotation,spatialErrorPerTask,meanErrorPerTask_x,...
         meanErrorPerTask_y,meanErrorPerTask_z,meanErrorPerTask_rotation,...
         varianceErrorPerTask_x,varianceErrorPerTask_y,varianceErrorPerTask_z,...
         varianceErrorPerTask_rotation);

     % Time Statistics Arrays
    timeMean = comparison(:,13,1);
    minimumTime = comparison(:,13,3);
```

```
maximumTime = comparison(:,13,4);
timeVariance = comparison(:,13,2);

 title_time = {'Test Execution Time Statistics'};
 units_time = {'(seconds - s)' '(s)' '(s)' ' '};
 uas_time = table(vehicle,timeMean,...
      minimumTime,maximumTime,...
      timeVariance);

% write to excel file: loop possibly
base=3;
xlswrite(fname,title_fail,1,sprintf('G%i',0*n+base-2))
xlswrite(fname,units_fail,1,sprintf('B%i',0*n+base-1))
writetable(uas_fail,fname,'Sheet',1,'Range',...
     sprintf('A%i:T%i',base,n+base))

space = 4;
base = base + space;
xlswrite(fname,title_power,1,sprintf('G%i',1*n+base-2))
xlswrite(fname,units_power,1,sprintf('B%i',1*n+base-1))
writetable(uas_power,fname,'Sheet',1,'Range',...
     sprintf('A%i:M%i',n+base,2*n+base))

base = base + space;
xlswrite(fname,title_position,1,sprintf('G%i',2*n+base-2))
xlswrite(fname,units_position,1,sprintf('B%i',2*n+base-1))
writetable(uas_position,fname,'Sheet',1,'Range',...
     sprintf('A%i:O%i',2*n+base,3*n+base))

base = base + space;
xlswrite(fname,title_time,1,sprintf('C%i',3*n+base-2))
xlswrite(fname,units_time,1,sprintf('B%i',3*n+base-1))
writetable(uas_time,fname,'Sheet',1,'Range',...
     sprintf('A%i:E%i',3*n+base,4*n+base))

% output finish executing in French
fprintf('fini\n')
end


function [m,t] = unixTime2TestTime(sync_s,sync_ns,topic_s,topic_ns)
%unixTime2TestTime: convert unix time to a time from start in seconds
%    return size (m) of time array and times (t) from start of test
     m = length(topic_s);
     t = zeros(1,m);
     for i = 1:1:m
          t(i) = topic_s(i) - sync_s...
               + (topic_ns(i) - sync_ns)/1000000000;
     end
end
```

C.3 Arduino AttoPilot code

```
/*
AttoPilot Current and Voltage Sensing Demo
N.Poole, Sparkfun Electronics, 2011

"I don't care what you do with it, and neither does the script." (apathyware)

Physical Connections:
-------------------------
Arduino  | Peripherals
-------- | ---------------
Pin 3  --- SerLCD "RX"
Pin A0 --- AttoPilot "V"
Pin A1 --- AttoPilot "I"
GND ------ AttoPilot "GND"
GND ------ SerLCD "GND"
5V ------- SerLCD "VCC"

This demo will read the Voltage and Current from the "AttoPilot Voltage and Current Sense Board,"
convert the raw ADC data to Volts and Amps and display them as floating point numbers on the
Serial Enabled LCD. (If you would like to do without the Serial LCD, I have included commented code
for reading the results through the Serial Terminal.)

*/

int VRaw1; //This will store our raw ADC data
int IRaw1;
int VRaw2;
int IRaw2;

void setup() {

Serial.begin(9600);

}
```

```
void loop() {

  //Measurement
  VRaw1 = analogRead(A0);
  IRaw1 = analogRead(A1);
  VRaw2 = analogRead(A4);
  IRaw2 = analogRead(A5);

  Serial.print("\r\n");
  Serial.print(VRaw1);
  Serial.print(", ");
  Serial.print(IRaw1);
  Serial.print(", ");
  Serial.print(VRaw2);
  Serial.print(", ");
  Serial.print(IRaw2);

  delay(50);

}
```