A	Project	Bell	Config	
	Aircraft	Bell APT	Phase	
	Model	VSCL Subscale	Date	
	Title	**Logging Checklist**		

#	x	Test Item
1.		Confirm that the GCS Your Using is Connected to the VSCL WiFi
		In Terminal A on GCS
2.		ssh vscl@192.168.1.5
3.		cd bell_ws && source devel/setup.bash
4.		Type, but Don't Run Yet: roslaunch data_reformater start_logging.launch
		In Terminal B on GCS
5.		ssh vscl@192.168.1.5
6.		cd bell_ws && source devel/setup.bash
7.		Type, but Don't Run Yet: rostopic hz -w 1700 /dfti_data
		In Both Terminal A & B on GCS
8.		Click Enter.*
9.		If necessary, start stopwatch when logging starts.
10.		Confirm in terminal B that data is being logged at around 1700 hz.**
		Perform Mission
11.		Press control c in both terminals
12.		Enter Exit in both terminals
		Retrieve Data
13.		sftp vscl@192.168.1.5
14.		get <File Name>***
15.		exit
		Reformat Data
16.		Place the files in the folder as the Matlab file data_importer.m
17.		run the Matlab file data_importer.m****

Notes	
Approved	

* Logging will start when "Logging Started" is printed in terminal A <u>and</u> the first message is received in terminal B.

** Getting to 1700 hz may take a minute as the autopilot boots up. Wait for a steady 1700 hz before taking off. Small oscillations (standard deviation of 100 hz) are okay as long as the mean is 1700.

*** File will be in the home folder and named "DFTI_<Time Stamp>.csv"

**** This should automatically detect and export the data as .mat files with the same names. The format of the data is explained on the next page.

1 Explanation of Data Format

The files are .mat files. So you can load them into matlab by using the load command. Once these files are loaded you'll find 2 variables:

- hacks - This variable stores a list of times at which external excitation occurred.
- types - This is a list of strings containing the names of the variables I logged. One of the flights didn't log the motors commands (the one we didn't actually fly). I think because commands hadn't started being sent to the motors yet. The variables I logged are:
  - x (m) - horizontal position NED coordinates
  - y (m) - horizontal position NED coordinates
  - z (m) - vertical position in NED coordinates
  - phi (rad)- roll angle
  - theta (rad) - pitch angle
  - psi (rad) - yaw angle
  - q0 (unitless) - part of a quaternion representing orientation in NED coordinates.
  - q1 (unitless) - part of a quaternion representing orientation in NED coordinates.
  - q2 (unitless) - part of a quaternion representing orientation in NED coordinates.
  - q3 (unitless) - part of a quaternion representing orientation in NED coordinates.
  - u (m/s) - x axis velocity in the body frame.
  - v (m/s) - y axis velocity in the body frame.
  - w (m/s) - z axis velocity in the body frame.
  - p (rad/s) - x axis angular rate in the body frame.
  - q (rad/s) - y axis angular rate in the body frame.
  - r (rad/s) - z axis angular rate in the body frame
  - M1 (PWM) - Command to motor 1
  - M2 (PWM) - Command to motor 2
  - M3 (PWM) - Command to motor 3
  - M4 (PWM) - Command to motor 4
- results - The data is stored here in a nested structure format. It has the following parameters that can be accessed with dot indexing:
  - <Variable Type Name> - There is one parameter for each of the variables listed above. This is also a structure with the following parameters representing different types of data:

* raw - This is the data directly as recorded. The only post processing I did was to add phi, theta, and psi. This is also a structure with the following parameters (note each of the following data types also have the same parameters):
    - time - An array with time stamps representing when data points were taken.
    - data - An array with the actual data.
    - hz - An array showing the time (as a frequency in hz) between that measurement and the next measurement.
* zoh - The data in raw was logged at roughly 100 hz, but that rate fluctuated slightly. Additionally, the data wasn't logged synchronously. There was no way to check each of the variables at exactly the same time. As such, I created an array of times that spanned the same time span as the log, but had times at exactly 100 hz (0.01 seconds apart). I then used a zeroth order hold to fill in these values (If you don't know what a zeroth order hold is, essentially I used the value of that measurement at the most recent previous measurement.)
    - time
    - data
    - hz - Always 100.
* interpolate - I tried to solve the same problem as the zoh variable above, but this time by interpolating to get the data point at that time.
    - time
    - data
    - hz - Always 100.

Long story short, the data in raw is not synchronous nor at a constant rate, but the data in zoh and interpolate are synchronous and at exactly 100 hz. I suggest you use the interpolated data. To access the data you would use something like:

$$\begin{aligned} time\_stamp &= results.theta.interpolate.time; \\ theta &= results.theta.interpolate.data; \end{aligned} \tag{1}$$

The PWM signals to the motors can be converted to throttle percentage by subtracting 1000 and dividing by 1000. In other words:

$$delta\_t\_motor\_1 = (M1 - 1000)/1000 \tag{2}$$

Phi, theta, and psi contain the same information as the quaternion. In fact they were derived from this quaternion.

Contact Kameron Eves (KameronEves@gmail.com) with any questions.