LabVIEW Motor Thrust Test Automation

By: Pranav Maroli

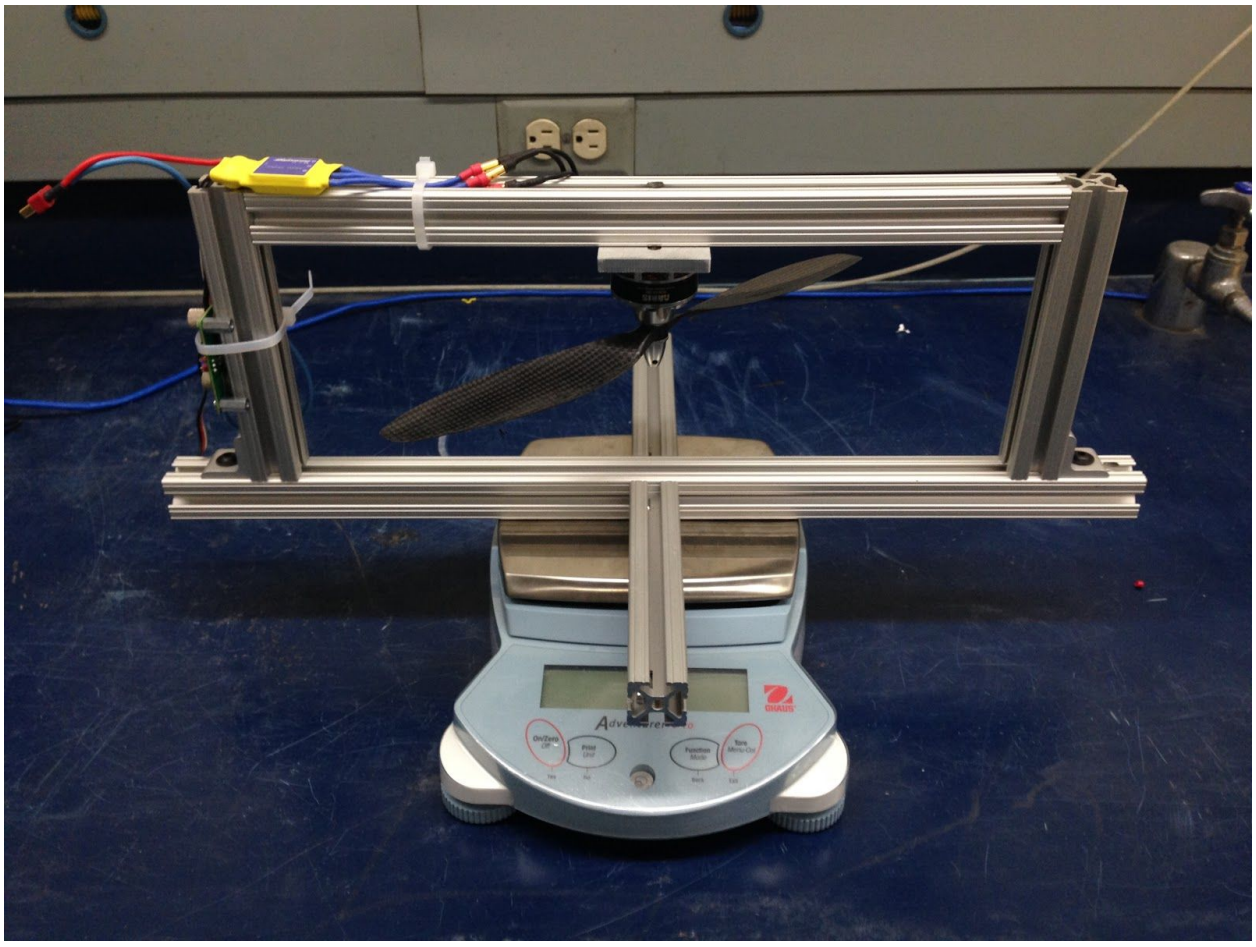

Table of Contents

Overview

In order to optimize the battery life of a quadcopter, the Airdrop Mini UAV Capstone team had been running tests on motors to determine the optimal thrust to amp draw ratio using different propellers and batteries. Previously, these tests consisted mainly of setting up and manually recording the amp draw and thrust in an Excel document; this way of conducting the tests was extremely slow and inefficient. The LabVIEW program that I have written streamlines this process and allows tests to be run at 2-3 times the original speed. Propellers can be switched out easily and minimal configuration is required. The use of object oriented programming allows the use of a variety of different hardware options and clean integration with the system.

Features

- Interchangeable hardware with use of object oriented programming
- User-defined test parameters
- Straightforward GUI
- Save data to Excel and plot to image file
- Allows for multiple test cases, numbering data files accordingly
- Emergency abort button

Setup (Hardware)

- Obtain the 8020 extrusion structure supplied with this package and secure the motor to it using the motor mount plate.
- Connect the Electronic Speed Controller wires accordingly, making sure the motor will spin in such a way that the thrust generated will push **down**.
- Wire the speed controller and battery grounds to the ammeter and power the ammeter with a steady 3.3V.
- Connect the scale to a COM port on the PC that is being used and configure any serial connection settings needed.
- Secure all electronics to the 8020 extrusion structure so that they do not get in the way of the spinning propeller.
- Secure the 8020 extrusion structure to the scale in a stable orientation using heavy duty tape; align the shaft of the motor approximately to the center of the scale.
- Wire analog inputs and outputs to available DAQ card as required.

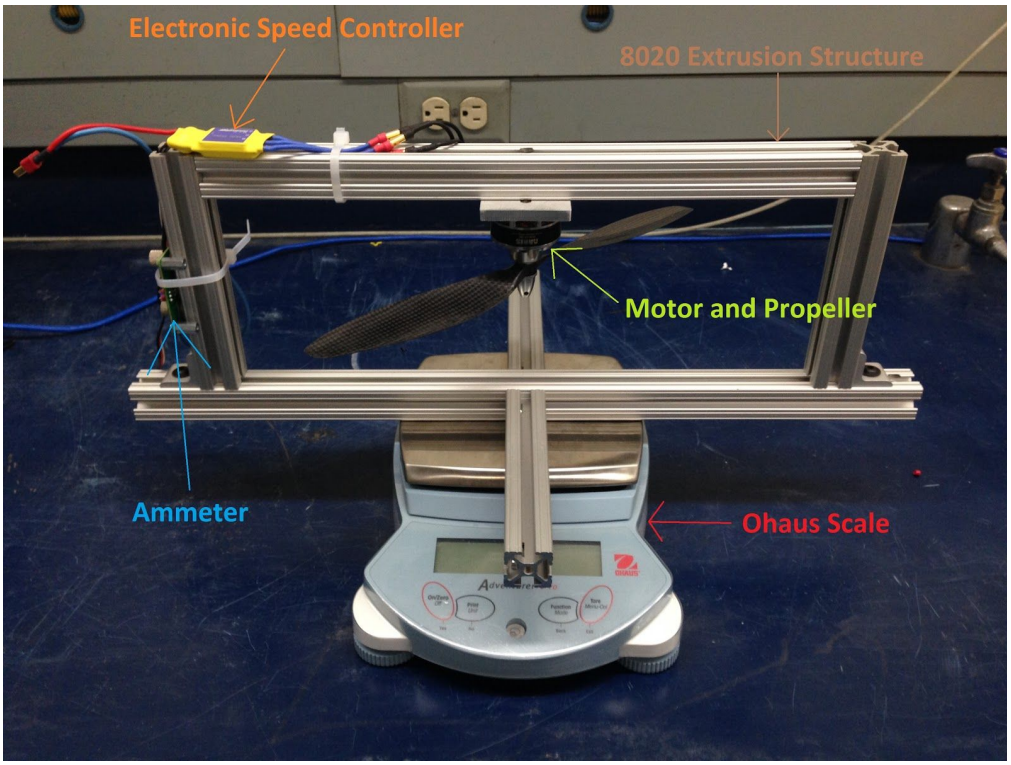

Figure 1: Hardware Setup

Installation and Use (Software)

To begin using the LabVIEW software package, simply download the .zip file, extract it, and open of the .VI file that is in the main folder. All drivers for the OHaus AV8101 scale are already installed in this package; use of a different scale would require that the drivers be downloaded and installed through the National Instruments website (this will be covered in a later section).

When the VI is run, it will prompt the user to calibrate the Electronic Speed Controller (ESC) and select a destination folder in which to save the data files. Instructions guiding the user on how to do those things are integrated into the VI. Once LabVIEW has calibrated the ESC, a dialog box will pop up indicating that it has been done; at this point, the test is ready to be run.

Before pressing the "Run Test" button on the front panel, **make sure the test fixture is secured and clear of any personnel**. After safety precautions have been taken, simply press the button and run the test; the default values on the front panel are recommended to obtain the best results, but they may be changed as needed.

Once the test is complete, the user will be prompted to save the test results. If he or she wishes to do so, they will be saved in the folder specified previously; subsequent tests will be saved in numerical order beginning with "Filename_01."

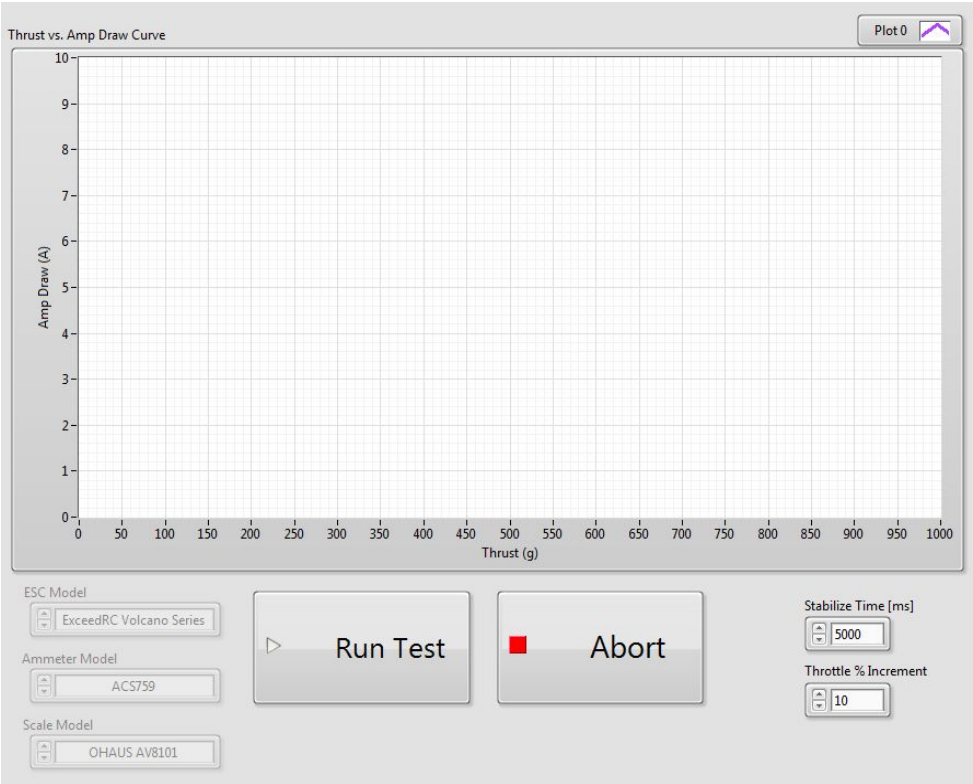

Figure 2: Front Panel

Note : To stop the motor at any time, press the "Abort" button

Code Overview

This project makes use of a state machine architecture. The states in a typical run include: “Wait for Go,” “First Signal,” “Wait to Stabilize,” “Record XY,” “Increment Signal,” and “Save.” The state machine architecture made the most sense for this project because the experiment inherently had these states when done manually. After a certain throttle value was sent to the motor, the operator had to wait until the readings on the scale and ammeter had stabilized, and then she had to record them in order to get a thrust vs. amp draw curve for the motor and propeller. A snapshot of the main VI is shown below:

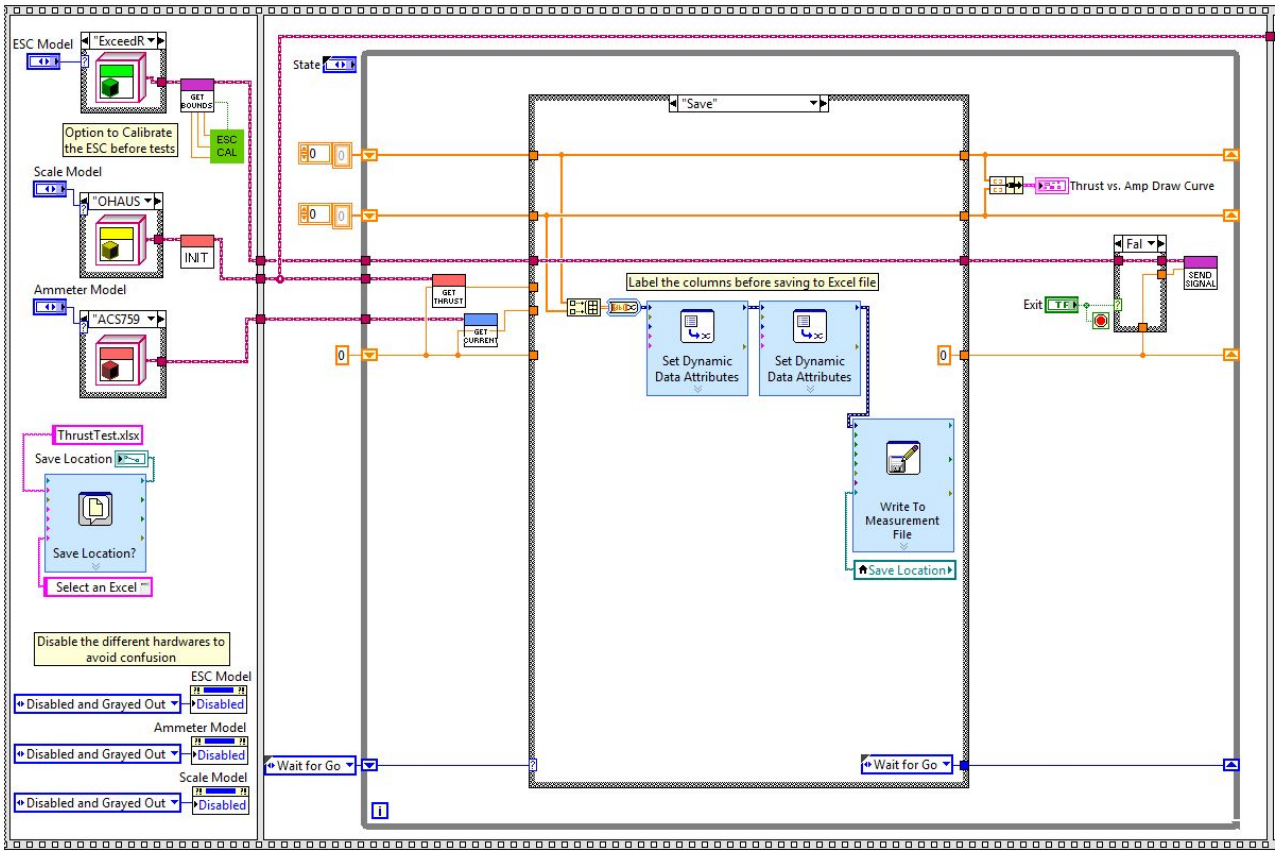

Figure 3: Main VI Block Diagram

It should be noted here that because of object oriented programming, and piece of hardware that can serve the same function as an ammeter or a scale can be used in this code without restructuring the code at all.

One essential subVI that is used in this project is called “Calibrate ESC.” This VI sends a PWM signal to the ESC when it is turned on so that the controller knows what the maximum and minimum ranges of PWM duty cycle are for the particular transmitter (the DAQ card and LabVIEW in this case).

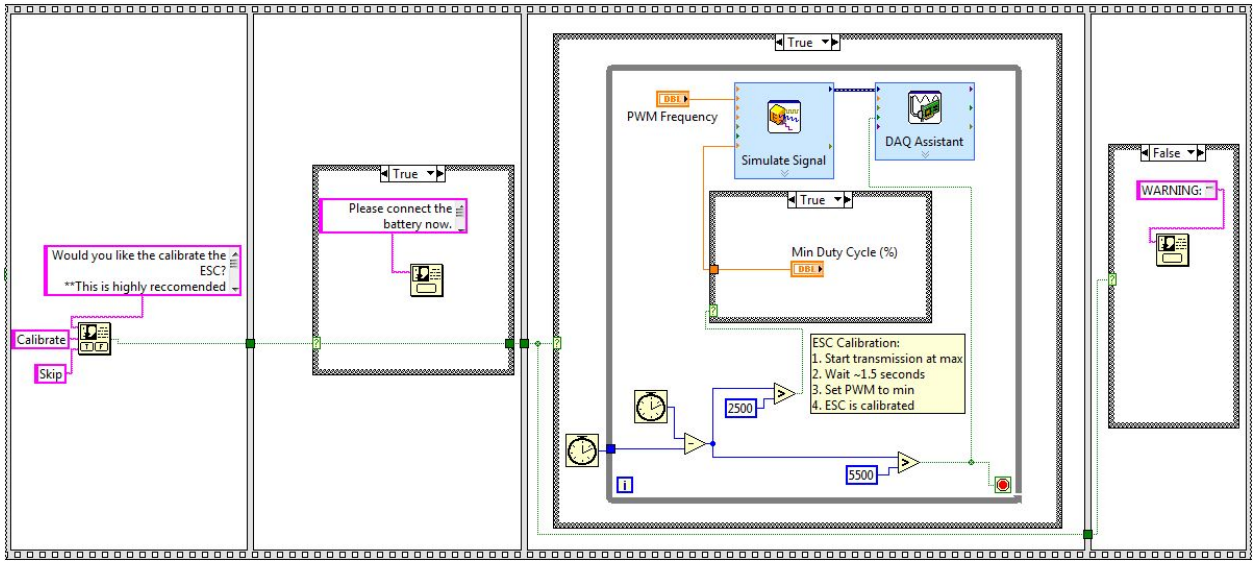

Figure 4: Calibrate ESC Block Diagram

The parameters for the maximum and minimum duty cycles as well as the frequency are all stored in class private data that is initialized when a new class for the hardware is set up.

Future improvements to this code would be mainly focused on the user interface. I initially imagined that the user would have an option to select which motor and propeller they are using so that the save file name would automatically be set to something that would reflect the test case. Minor changes like this and reworking the GUI to improve the quality of life of the end-user would not be too difficult to implement given more time. Ideally, I would like this code to be robust enough so that it could be used by somebody who has minimal knowledge of the project and no knowledge of LabVIEW.

Scalability and Development

Anybody familiar with LabVIEW should be able to easily look at the code and know what is going on. The main issues that I see a development engineer having with the code is on how to add a class for additional hardware, so I will outline it in this section.

The end user may be using a different ESC than what is configured in this setup, but adding a new ESC class is not difficult. Simple create a new class in the project and change it's inheritance to the parent "ESC" class. Then populate its class private data with values for Max Duty Cycle, Min Duty Cycle, and PWM Frequency according to the data from the manufacturer.

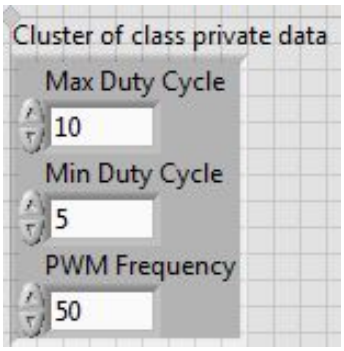

Figure 5: ESC Class Private Data

Additionally, a different scale may be used. Add a class in a similar fashion to the ESC, but instead populate the "Serial Configuration" cluster with the COM settings that the new scale uses. Finally, check to see if National Instruments has a driver for the scale, as it will make implementation much easier.

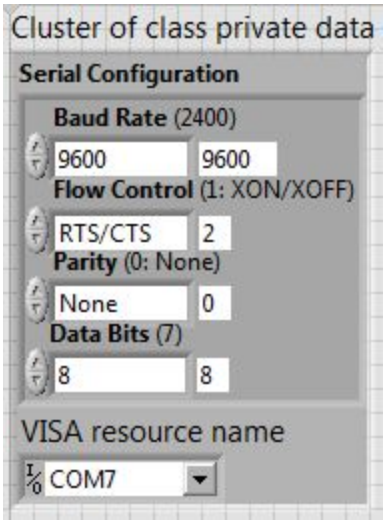

Figure 6: Scale Class Private Data

Challenges & Concerns

One of the biggest challenges I faced while doing this project was interfacing with the scale. Other hardware interfacing was as simple as an analog voltage connection, which is extremely simple with the DAQ card provided. Most laboratory scales use a serial connection to communicate with computers, and having no previous knowledge of this type of connection, it was a little daunting at first. After doing some reading and becoming more acquainted with the serial connection interface, things started to make more sense; once I found that National Instruments had drivers for the particular scale that I wanted to use, things became exponentially more simple. After a day or so of playing with the COM settings on the computer and the scale, I was able to get scale reading easily. Though this was the biggest hardship in the project, I think I learned more about hardware interfacing than anything else doing this project.

The main concern that I have for end users is safety. Currently, there is no hard shutoff implemented outside of LabVIEW; a mechanical shut off switch or breaker would be ideal for safety concerns. Additionally, if LabVIEW crashes or the program terminates unexpectedly, the voltage signals that were being sent last will still be sent continuously. This is less of a problem because the speed controllers require a PWM input, but a steady voltage is enough to keep the motors running.