Motor Testing System Setup

Purpose:

The Motor Testing System is designed to allow for the experimental determination of several performance parameters important to the dynamic behavior of the quadcopter (See *Mathematical Model* documentation). To obtain these parameters we measure RPM, thrust, torque, and a first order model time constant for the motor, ESC, and prop system. The overall Motor Test System includes designs for the required hardware and circuits, an Arduino Uno program that facilitates easy data collection, and a data analysis program that makes data analysis simple and fast. This document will focus only on the actual setup of the motor test system, and assumes the hardware and circuits are built and ready to be used. See the quick start documents for the *Comparator Circuit, Motor Modeling Hardware,* and *Data Analysis Program* for additional information about these aspects of the system.

Skills/Knowledge Required:

- Basic soldering (you can learn as you go if needed, just check out some YouTube guides)
- Fundamentals-level knowledge of circuit behavior (Ohm's law, KVL, KCL) is very helpful
- A little experience with Arduino platform and microprocessors (at least having done some example sketches and circuits) is also highly recommended.

Parts and Components (See Figure 1):

Name/Description	*Part No.*	*Qty.*	*Notes*
LiPo Battery	Project Dependent	1	
Power Meter (Opt.)	HK-010 or equiv.	1	Not required, but recommended.
ESC	Project Dependent	1	
Brushless DC Motor/Rotor	Project Dependent	1	
Computer (PC or Mac with serial USB port)	Project Dependent	1	Must have Arduino IDE installed
Arduino Uno (Or equivalent) with serial USB to computer wire	-	1	Program and circuit could be reconfigured for alt. microprocessor but this was not tested.
Comparator Circuit and Photosensor	Built by user	1	See "*Quick Start Comparator Circuit*"
Light Source	As available. A "bright", directed light source is recommended.	1	See notes in discussion section below
Power Supply Wiring and Connectors, heat shrink, electrical tape,	Project Dependent. We used XT60 and " Bullet" style connectors together with good success.	Varies	Choose wiring based on anticipated motor and ESC current draw by consulting an AWG chart.
Signal Wiring and Connectors	User preference. Servo style connectors and protoboard "jumper" wires may prove useful.	Varies	We recommend good quality stranded wire, a servo connector crimping tool, and some servo connection ends to make reliable connections.

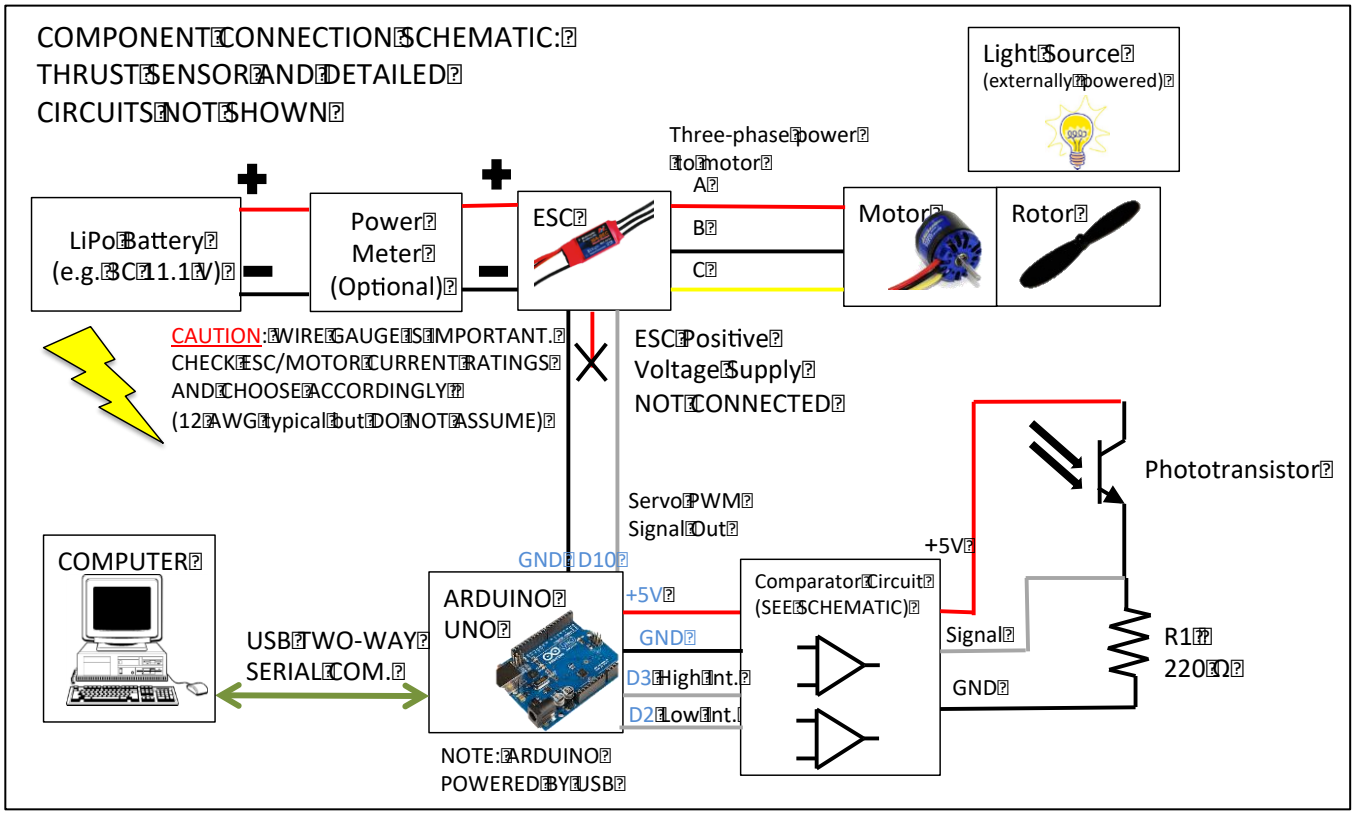

Figure 1. Motor Testing Connection Diagram

Discussion:

The actual work of constructing this circuit will largely consist of cutting wires to length, soldering connections onto wires for the power supply path, and crimping servo connecters onto the ESC wires. While perhaps a bit tedious, this work should be performed carefully, as the voltages and currents involved can easily damage the measurement equipment (best case), damage your computer (bad case), or burn down your house (worst case). I managed to have a jumper wire short our battery at one point, and the violent bright flash and burn I received taught me a lesson about keeping power voltages and signal voltages separated within the circuit. The pin assignments to the Arduino Uno are shown in blue. Not shown is how to actually construct the comparator circuit; see the Quick Start Comparator Circuit document for some details on this. Some thought as to how you want to wire the ground/comparator power to the Arduino can save you trouble later. For questions related to soldering connectors and crimping servo wires, search YouTube for videos; there are several quick tutorials that will explain it better than I can here.

The following will provide a brief walkthrough of the circuit and its connections. The rest is left to the reader!

LiPo Battery to Power Meter:

Our battery had an XT60 connector on the end. I connected this to the Power Meter's XT60 connector. Pretty easy so far huh?

Power Meter to ESC:

Our ESC came with no connectors attached, but our motor used bullet connectors (3.5 mm), so we installed these on both sides of the ESC. Interestingly, XT60 connectors and 3.5 mm bullet-style connectors can be mated quite nicely, so a patch cable was fashioned (about 1.5 ft) with bullet connector ends to wire the ESC to the Power Meter. If this is not the case for your components, just choose a connector that you want to use and make sure you make the patch cable long enough to allow comfortable testing (you will want to wait until you've built the test rig and have a quadcopter arm available before doing this).

ESC to Motor:

Our motor had bullet connectors (3.5 mm) pre-soldered on its three power wires. So we added bullet connectors to the ESC leads. Obviously make sure your ESC can handle the current your motor needs (check online resources about this), but you should already have done that by the time you're soldering components together.

ESC to Uno:

The ESC has a small servo wire coming off of it with three wires (black, red, white or similar). The black wire is ground (needed), the red wire is a ≈ 6 V supply (not used here), and the white wire carries the "servo signal" to the ESC from the UNO (needed). Make a patch wire from small signal wire (≈ 24-26 AMG stranded) with one side having a "female" servo connection (look up what is called "male" and "female" for servo connectors…it's probably not what you would guess!!!) and the other having whatever connection you want to put into the headers on the Uno (I used servo "female" pins (the needle looking one, which most people would call male) connectors neatly wrapped in electrical tape). Do not connect the red power line coming from the ESC to the board. Read up on ESC's and servo signals if you aren't familiar as having at least a cursory understanding of these may be important to the success of your project. I'll say this here in case I don't say it elsewhere: *SERVO SIGNALS ARE NOT JUST ORDINARY PWM SIGNALS*. Yes the pulse width is modified, but if you approach servo signals with typical PWM preconceptions you might

miss important details! Anyway this connection is a communication connection, with the Uno sending servo signals to the ESC, which are interpreted as throttle commands by the ESC. The ESC reads these commands and sets the motor speed (sort-of) proportionally. See the Mathematical Modeling documentation for a brief discussion of how we modeled this.

Uno to Computer:

We chose to power the Uno from the Computer. The Computer communicates with the Uno over a Serial connection. This allows the user to type throttle commands into the Arduino IDE Serial Monitor window and receive data collection points from the Uno in real time during testing. Just plug the USB cable into the Uno and the Computer and select the correct USB port and baud rate (115200) from the Arduino IDE when the time comes.

Comparator Circuit and Photosensor:

See the Comparator Circuit documentation for details on construction of the circuit downstream of the Uno. Looking from the Uno side of things, the comparator circuit needs a +5 V supply, a ground, and has two outputs to the Uno. These outputs connect to D2 and D3 on the Uno as these are the only external interrupt capable pins. Use whatever signal wire is convenient; we used protoboard jumper wires.

Pictures and Tips:

Figure 2 shows the battery, power meter, and ESC (mounted on the quadcopter's arm) for our setup.

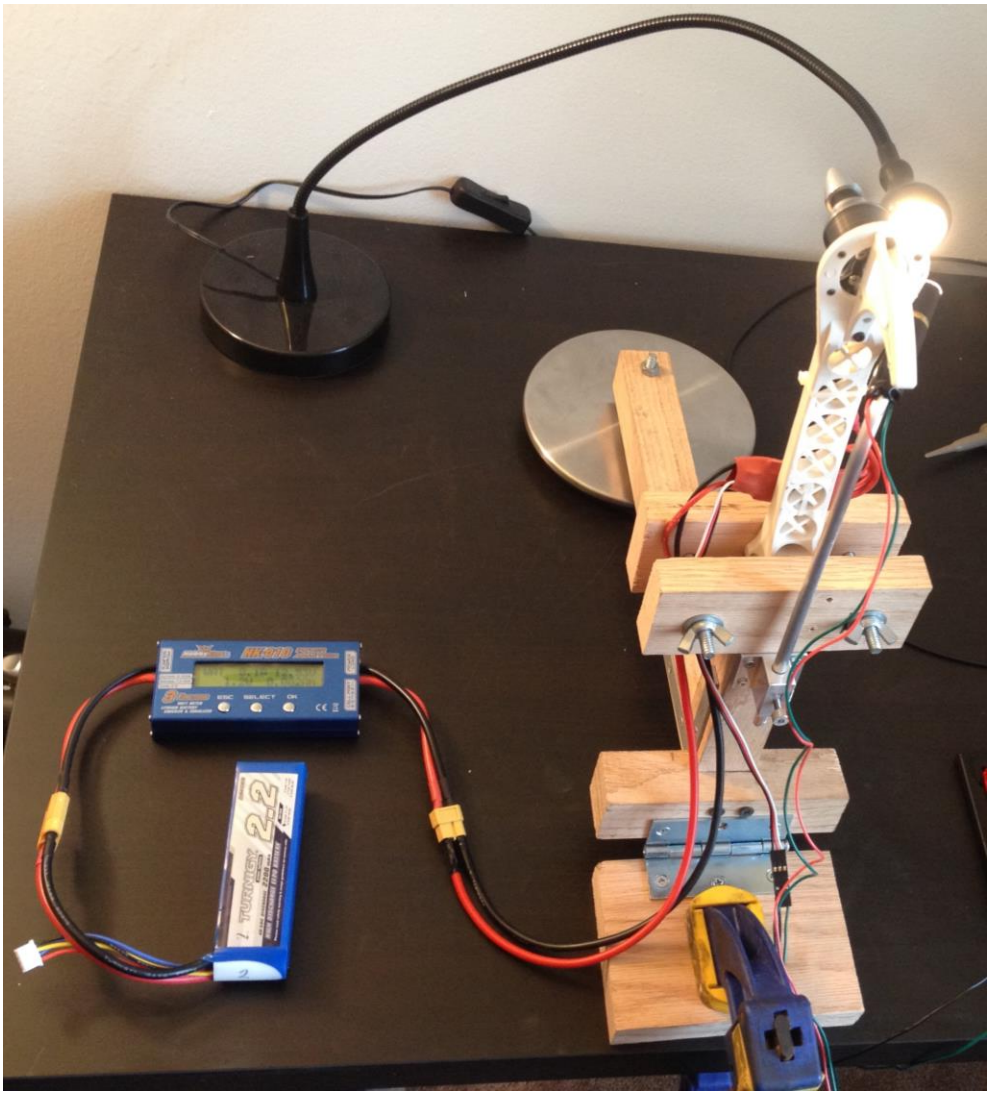

Figure 2. Battery, Power Meter, and ESC connections

Though not apparent in Figure 2, the XT60 connector from the power meter has a patch cable with bullet connectors plugged into it. These lead to the bullet connectors coming from the ESC. The ESC servo wire can be seen in the picture, as well as the phototransistor power and signal wires (red and green).

Figure 3 is provided for additional reference, but refer to the comparator circuit documentation for more info.

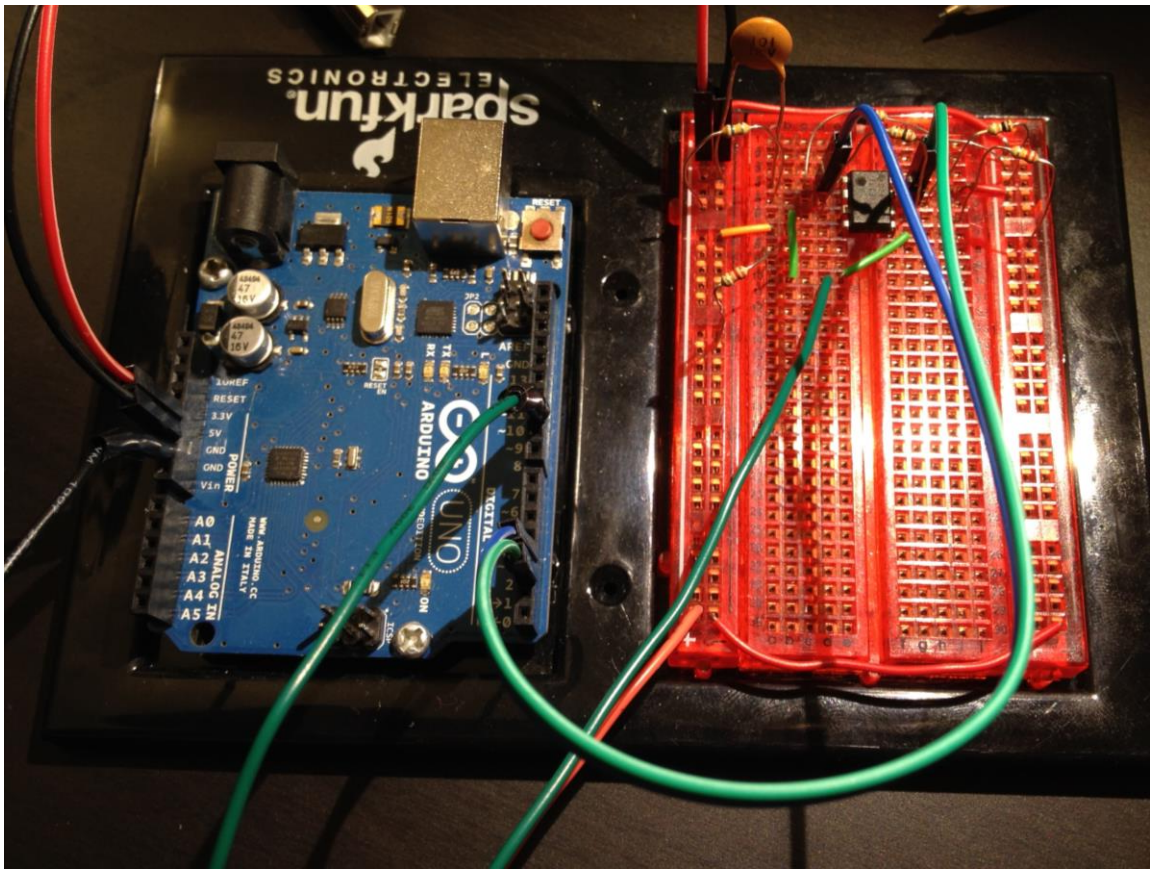

Figure 3. Arduino and comparator circuit view

In Figure 3, the comparator circuit and the Arduino board are shown for general reference. **Do not use this picture to try to build the circuit**, it is provided simply for a conceptual layout reference. Notice that the USB cable is not plugged into the Arduino in this picture.