Like many of you, I’ve jumped on the quadcopter bandwagon and working on making one of my own. I got myself an Arduino Uno, Neewer A2212, 1000KV motors and HW30 ESC. I spent a couple of months learning how to program on the Arduino and made a self-balancing robot as my first project (blog on that coming soon).

I started working on building the quadcopter last week. The trickiest part so far has been learning how to use the ESC. I spent the whole day on this yesterday and finally figured it out. It’s amazing how much misleading information there is out there about using ESCs. There is two page user manual available, and while it contains accurate information, it can be a bit hard to understand. So hopefully this article would be useful for you.

First off, a quick intro about what ESCs are and why they are needed. ESC stands for Electronic Speed Control. I’m using them to drive my brushless motors. Motors work due to the interaction of the magnetic field created by the current passing through the armature and the magnetic field of the permanent magnets mounted on the motor chassis. Regular motors have brushes that touch the contact points of a motors armature and feed current to the armature. These contacts points are laid out in such a way that as soon as the armature rotates to line up with the magnetic field of the permanent magnets, the contacts switch and the polarity of the current being fed to the armature coil reverses. This recreates the imbalance in the alignment of the magnetic fields leading to a continuous rotatory movement.

Since there is physical contact involved between the brushes and the armature, regular motors are subject to electrical noise and wear over time. In a brushless motor, the moving parts are switched. The chassis is the rotor and armature the stator. Current is fed directly to the armature instead of through brushes. This architecture eliminates physical parts rubbing against each other leading to better durability, efficiency and precision. However these benefits don’t come for free. In order to generate the rotating magnetic field necessary to create the rotation, brushless motors need to be fed a rotating electrical current. You can’t just connect them to a battery and expect the rotor to start spinning. This is where ESCs come in.

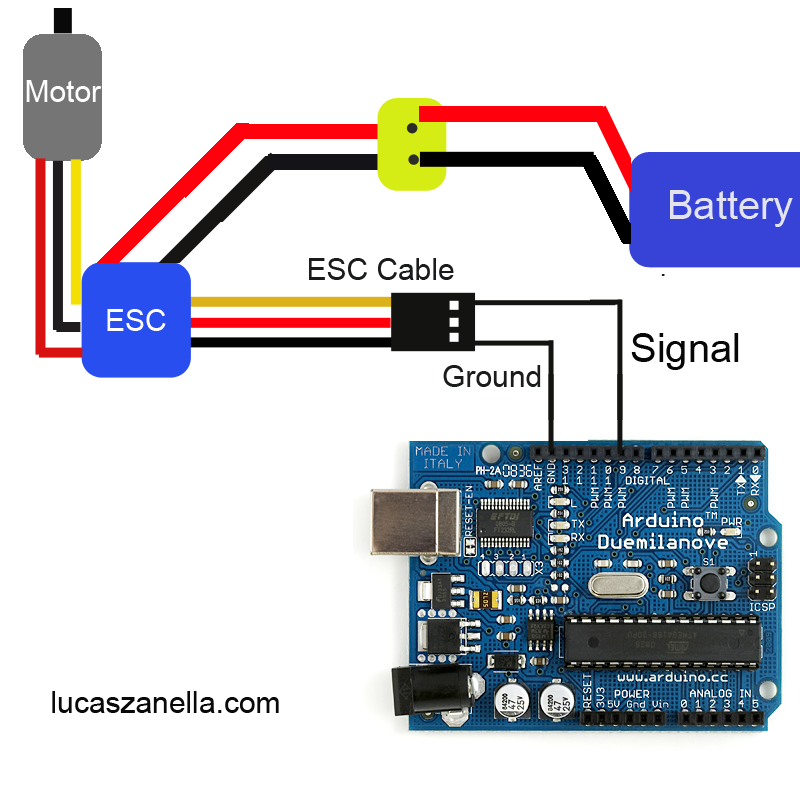
The role of ESC is to act as electronic brushes. The ESC takes a DC input voltage and converts that into a rotating (cyclical) electric current. The cycle frequency depends upon the PWM input sent to the ESC from the controlling microprocessor. The min and max cycle frequency can be set by the user, and this process is called “arming” the ESC. For a beginner, all this terminology can be a bit daunting, so good to understand the basic concepts.

Now the question is how to arm the ESC and this is where all the confusion begins. At this point, everything I say will refer to the ESC I’m using – HW30, so while the general concepts might apply to other ESCs as well, the details might not. So be careful.

Before we learn how to arm an ESC, there is another concept called “throttle” to be familiar with. I suppose this originated from the use of ESCs in remote controlled small planes which used a radio controller “throttle” to adjust the speed of the propeller. For an Arduino user, this just means the pulse width of the PWM signal sent to the ESC. Setting the low and high throttle translates to setting the min/max RPM of your brushless motor.

**Connecting the ESC to the motor and Arduino**

Connecting the ESC to the motor and Arduino is already well covered in numerous article, so just refer to any of those. Here’s an illustration:



**Using ESCs**

There are three main things you can do with the HW30 ESC

1. Program it – this means settings such as low voltage protection, startup mode etc. For a complete list, see the attached user manual. I found the defaults ok for my purpose.
2. Set the throttle range. You would likely have to do this once. Once the throttles are set, the values are persistent.
3. Use the ESC

**Setting the Throttle Range**

The key steps are:

1. Set the initial throttle position to high.
2. Turn on the ESC. It is very important that ESC is getting a high signal on the PWM pin when it is turned on. In your code, you should send a high signal and the wait for user input so you have time to turn the ESC on. On turn on, the ESC will emit a 1-2-3 (3 tones of different frequencies in rapid succession).
3. The ESC will emit a beep-beep tone, indicating that the high throttle range has been set
4. As soon as your hear the beep-beep, send the low throttle signal. The ESC will respond by emitting several beeps (the number depends on the input voltage of the power supply connected to the ESC). These beeps will be followed by a long beep, indicating that the low point has been correctly set.

The key is to ensure that the Arduino is powered on and sending a high signal to the PWM pin connected to the ESC *before* the ESC is powered on. Otherwise the ESC will get a random signal as input and there is no telling what will happen. If you don’t hear the beeps in the sequence mentioned here, figure out what’s wrong before proceeding. Some common sources of error are:

* Input voltage is not in the acceptable range (ESC will emit beep-beep with a time interval of about 1 sec)
* PWM signal is abnormal (ESC will emit beeps with a time interval of about 2 sec)

For a full list of errors, see the manual.

**Programming the ESC**

If you need to change the default ESC settings, keep sending the high signal for about 5 seconds after you hear the two beeps corresponding to the ESC registering the high throttle signal. You’ll hear a special tone indicating the ESC has entered the program mode. Upon entering the program mode, the ESC will emit different beeps corresponding to different settings. When you hear the beep corresponding to the setting you want to change, send a low PWM signal right away. The ESC will emit beeps corresponding to the options for that particular setting. As soon as you hear the option you want to set, send a high PWM signal. The ESC will emit a particular beep tone indicating that the new value for the setting has been registered. See the manual for details about different settings.

**Using the ESC once the throttle range has been set**

Just start the Arduino with a low PWM signal. Now turn the ESC on, so it sees a low PWM signal as the input. As you change the PWM signal to high, the motor should rotate faster.

**Important Note:** keep in mind that the load you put on the motor will have an impact on the max RPM that can be achieved. So for example when you connect the propeller to the brushless motor, the max RPM the motor can spin the propeller (for a given power being fed from your power supply) will be less than what the motor can achieve when not under load. So, the motor might top out before you move the throttle to the top position. Once the motor has topped out, moving the throttle higher will cause the motor to behave unpredictably. This is fundamentally because the ESC is sending a rotating electric current with a frequency dependent on the PWM input. However the capacity of the motor to respond to this rotating current is dependent on the input power and the load applied. So, a good practice is to set the high throttle to a position that the motor can achieve when subjected to the expected operating load. Otherwise, you’ll be wasting part of your throttle range that the motor can’t achieve.