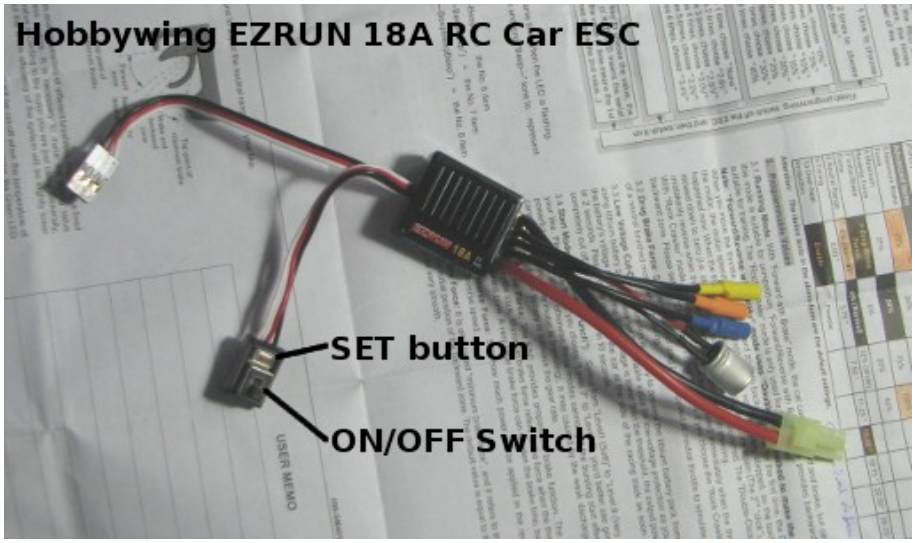
Programming and Calibration of EZRUN 18A ESCs

Having worked out that the occasional cutout of one of the thrusters in the Pool Test was due to the **HobbyKing® ™ Brushless Car ESC 10A w/ Reverse ESCs** overload protection rather than just an inappropriately high battery protection voltage threshold, I replaced the electronic speed controllers with the **HobbyWing EZRUN 18A ESCs** as used by the OpenROV project. Seeing as the original ROV design made allowance for these this was an easy substitution. The big difference is in the programming and calibration of these ESCs which don't conform to the typical calibration methods used and require a little more user input.

This page describes the method for programming the **HobbyWing EZRUN 18A ESCs**, their settings, and the subsequent calibration of the ESC using an Arduino. The Arduino was used to calibrate the ESCs to ensure the control signals were compatible with the main ROV control sketches. Because of the need for user input during the calibration sequence, the calibration process will not be able to be incorporated into the main ROV control sketches as was originally intended in the development of the ROV control system.



Programming and calibration of the **HobbyWing EZRUN 18A ESCs** was carried out without the ESCs being installed on the ROV electronics pod. Because the switch and “SET” button on the ESC are both accessible when the ESC is installed calibration and programming can be done at any time.

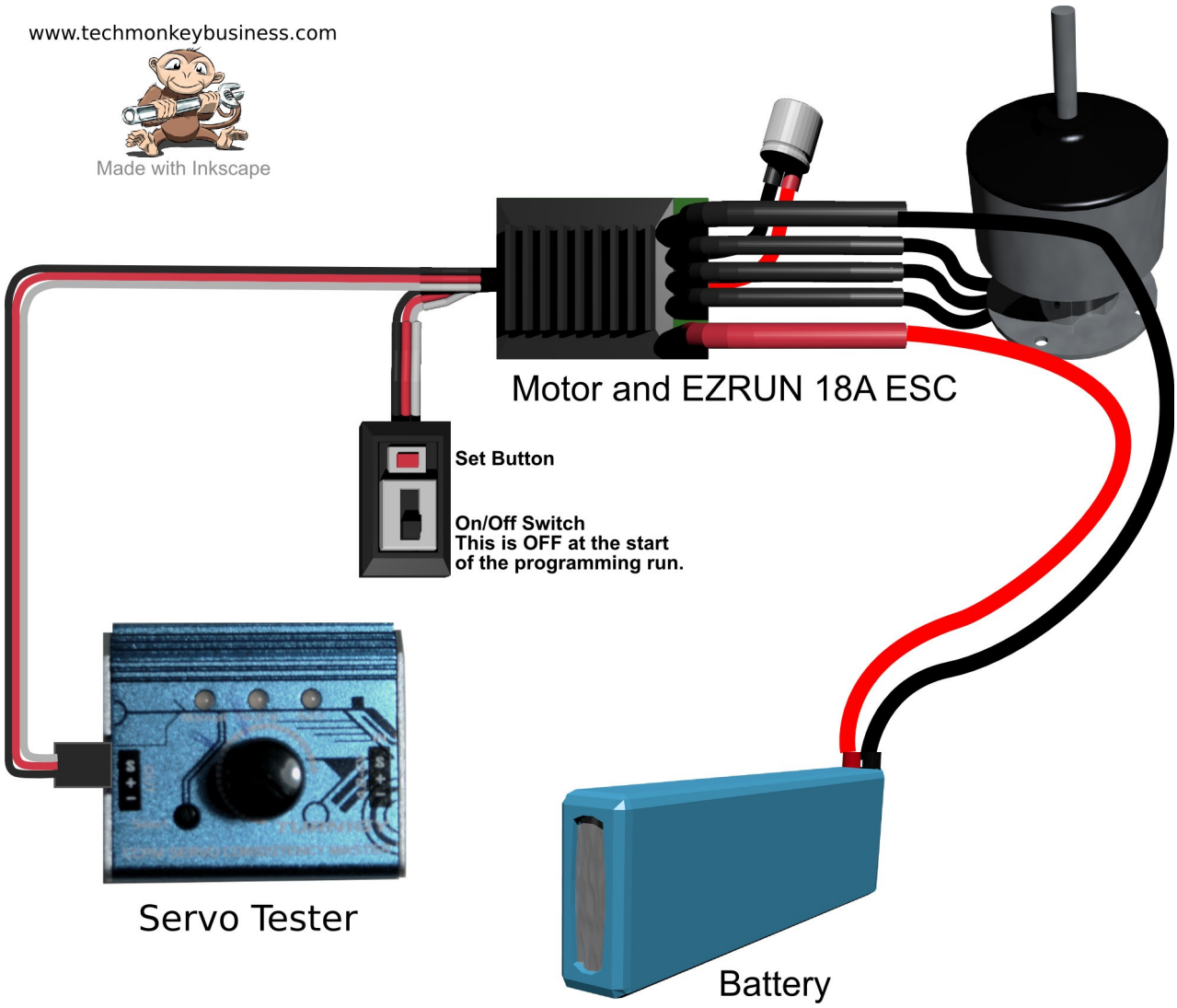
Basic Specifications

Continuous Current:	18A
Burst Current:	50A
Resistance:	0.01Ω
Suitable for Battery Cells:	2S LiPo and 3S LiPo
BEC Output	6V/1A

There is a suggestion that 3 cell LiPo needs a cooling fan. This is not used in the OpenROV and hopefully the thermal conductivity through the wall of the ROV electronics pod and the lower voltages associated with the $LiFePO_4$ batteries will be enough to keep the temperatures within a tolerable level.

Programming the ESC

To program each **HobbyWing EZRUN 18A ESC** I hooked them up to a battery and Servo Tester as shown below. The servo tester was set to the middle of its range to simulate a neutral throttle position. Initially the ESC switch should be OFF.



A word of warning about the default settings on the **HobbyWing EZRUN 18A ESCs**: The default settings from the factory **DO NOT** match the default settings listed in the user manual. The most significant of these is the *Low Voltage Cut-Off Threshold*. Its default is “Non-Protected” which means the ESC will not protect your battery from low voltage. If you do not set this appropriate to your battery type and are expecting the ESC to protect your nice new LiPo battery as the manual suggests it will, you will quickly end up with a damaged battery or worse.

While on the topic of battery protection, the *Low Voltage Cut-Off Threshold* for LiFePO4 batteries is 2.8V per cell. Lower voltages risk permanent damage to the battery or at least severely shortening its life.

ESC Program Settings

Here is a table of the settings used for the ROV and the actual default settings on my **HobbyWing EZRUN 18A ESCs** as they came from the supplier. The selected settings for the ROV are coloured orange and the actual factory default settings are in bold.

Program Table

			1	2	3	4	5	6	7	8	9
	Beep Codes ● = short ▬ = long		●	●●	●●●	●●●●	▬	▬●	▬●●	▬●●●	▬●●●●
1	●	Running Mode	Forward with Brake	**Forward/Reverse with Brake**	Rock Crawler						
2	●●	Drag Brake Force	0%	5%	**10%**	20%	40%	60%	80%	100%	
3	●●●	Low Voltage Cut-Off Threshold	**Non-Protected**	2.6V/Cell	2.8V/Cell	3.0V	3.2V	3.4V			
4	●●●●	Start Mode	Level 1 (Soft)	Level 2	L3	L4	**L5**	L6	L7	L8	L9
5	▬	Maximum Brake Force	25%	50%	75%	**100%**					
6	▬●	Maximum Reverse Force	25%	**50%**	75%	100%					
7	▬●●	Initial Brake Force	**=Drag Brake Force**	0%	20%	40%					
8	▬●●●	Neutral Range	6% (Narrow)	**9% (Normal)**	12%						
9	▬●●●●	Timing	0.00°	3.75°	7.50°	11.25°	**15.00°**	18.75°	22.50°	26.25°	
10	▬▬	Overheat Protection	**Enable**	Disable							

To access the programming mode turn off the ESC, then **press and hold the SET key** as you turn it on again. **Keep holding the SET key** as the ESC does a series of 13 short beeps while flashing the red LED, and then **keep holding the SET key** as the ESC signals which programmable item it is up to. It will signal which item it is at by beeping a sequence of short and long beeps and flashing the Green LED. Only release the SET key when you have got to the item you want to set.
After you release the set key, the ESC will signal what its current setting is by beeping and flashing its red LED in a sequence of short and long beeps. To change the setting, press the Set button. The ESC will signal the new setting. Keep pressing the SET key to advance through the settings until you get to the one you want. When you reach the setting you want, turn the ESC off. To set other items, hold the SET button and turn the ESC back on again, repeating the process describes above. Believe me when I say it takes a while to do.

To illustrate this with an example, if you want to set the Low Voltage Cut-Off Threshold to 2.8V/cell, press and hold the SET key as you turn on the ESC. Keep holding it while the ESC beeps 13 times while flashing the RED LED, then flashes the Green LED once. Keep holding the SET button until the ESC signals that it is at item No 3 with three short beeps and three flashes of the green LED. Release the SET key. The ESC will signal with one beep and flash of its red LED that its current setting is 1 – Non-protected. Press the Set key once and the ESC will signal with two beeps and two flashes of its red LED that it is now set for 2.6V/cell. Press the SET button again and it will beep three times and flash the red LED three times to signal that it is now set for 2.8V/cell. Switch the ESC off. To set other items, go through the process again.

Factory Reset

To return to the default settings if you truly muck it up, set the throttle to neutral and turn on the ESC. Press and hold the SET button for 3 seconds. The ESC's green and red LEDs will both flash and the ESC will be back to it's default settinsg read for you to start the process again.

Testing the ESC

Having the Servo Tester attached to the ESC means that you can quickly test some of the settings once you have completed programming it. With the servo tester knob at the midpoint of its range, switch the ESC on. Moving the knob either way should give a nice smooth acceleration and deceleration in either direction.

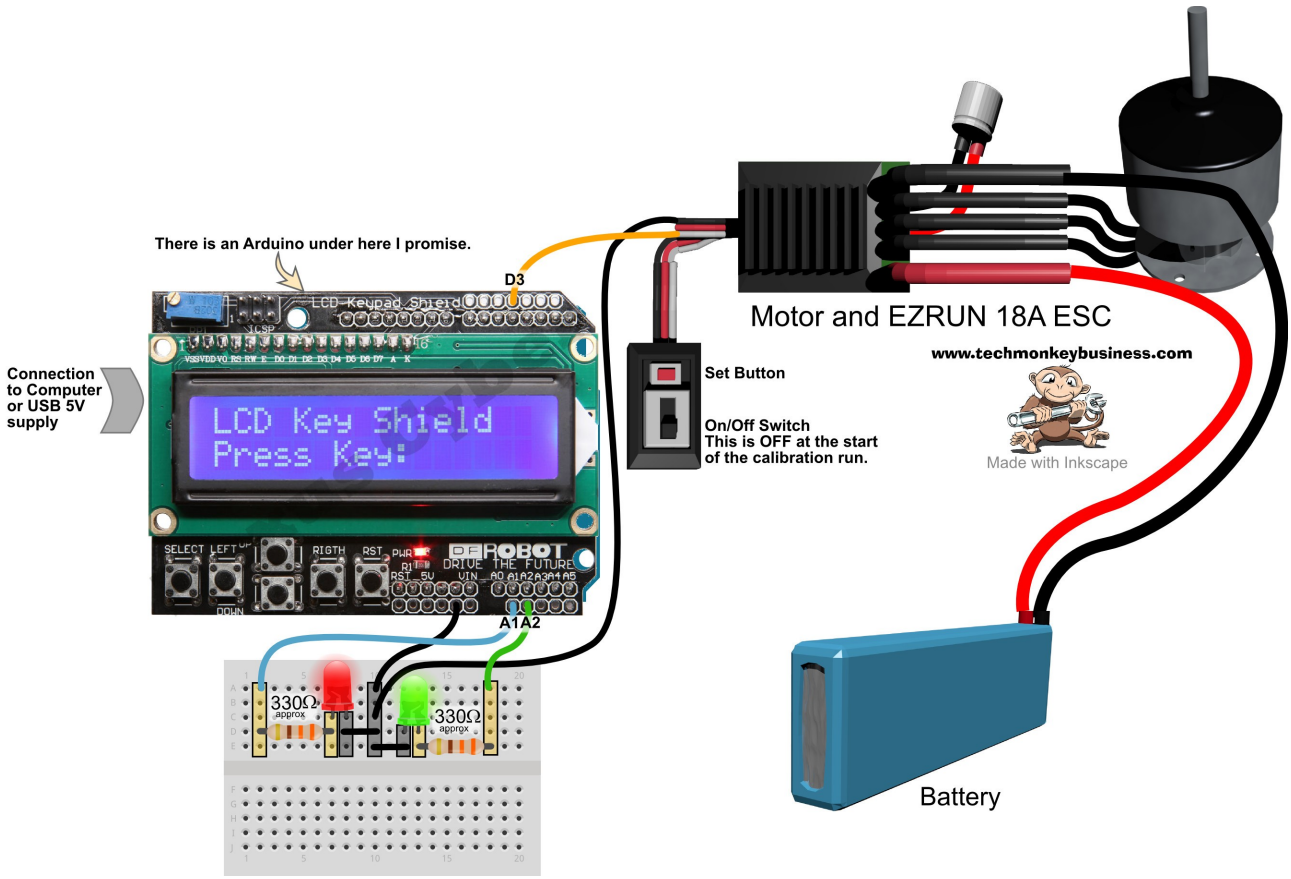
Calibrating the ESC

Once you have programmed each setting as outlined in the table above, the ESC can be calibrated.

Because this was going to be run by an Arduino and the ROV sketch defines the PWM signals for the servo positions, it is best to use an Arduino to calibrate the ESC by providing the same PWM signals. The circuit shown below uses the Arduino to provide the servo signals to the ESC in the correct sequence required for calibration. Two LEDs and a 16x2 LCD display shield provide instructions to the user about what they need to do.

The LCD shield is not strictly required, but it tells the user when to press the SET button.

The maximum and minimum PWM signal figures in the sketches `ESC.attach(3,600,2250);` command must match the figures used in the ROV sketch. These figures may vary depending on the ESCs you are using.



The connections to the Arduino are;

- the servo control wire of the ESC is connected to digital pin D3 (which is suitable for PWM)
- the red LED connects to analogue pin A1
- the green LED connects to analogue pin A2
- the LCD shield occupies digital pins 8, 9, 4, 5, 6, 7, and 10
- power is supplied to the Arduino from a computer or USB supply.

The Process

Once the sketch is loaded into the Arduino and an ESC is connected as shown above the LCD screen will signal the user to connect the battery (if it isn't already connected) switch on the ESC while holding down the SET button on the ESC. The ESC will start flashing its on-board red LED and beeping. Release the SET key as soon as the ESC starts beeping (holding it too long will cause the ESC to go into programming mode)

After a few seconds, the Arduino will signal that it is going to set the throttle to the neutral position. It will flash its own red LED for 10 seconds and then send the servo signal for a neutral throttle position to the ESC. The LCD screen will tell the user to “Press SET Button”. The sketch allows 10 seconds for the user to do this. When they press the button the ESC should signal the fact by flashing its on-board green LED once and beeping once.

After a few seconds, the Arduino will signal that it is going to set the throttle to the maximum forward speed position. It will flash its own red LED for 10 seconds and then send the servo signal for a maximum forward throttle position to the ESC. The LCD screen will tell the user to “Press SET Button”. The sketch allows 10 seconds for the user to do this. When they press the button the ESC should signal the fact by flashing its on-board green LED twice and beeping twice.

After a few seconds, the Arduino will signal that it is going to set the throttle to the maximum reverse speed position. It will flash its own red LED for 10 seconds and then send the servo signal for a maximum reverse throttle position to the ESC. The LCD screen will tell the user to “Press SET Button”. The sketch allows 10 seconds for the user to do this. When they press the button the ESC should signal the fact by flashing its on-board green LED three times and beeping three times.

The Arduino will allow this to sink into the ESC and signal completion of the process by flashing its own green LED. The ESC can be turned off and removed from the system.

To do another ESC, connect another ESC as shown and hit the Arduino's reset button to begin the process again.

EZRUN Calibration Sketch

```
/* ROV_EZRUN_Calibratorv0.ino
by Hamish Trolove 1 Dec 2015
www.techmonkeybusiness.com
The Hobbywing EZRUN 18A ESCs used now in the ROV need to be
calibrated to the Arduino's output signals.  The EZRUN works a
little differently from most ESCs in that it requires the
operator to push the "Set" button on the ESC at each point
in the throttle movement.
This sketch will provide the throttle motions and instruct the
user what to do through the attached 16x2 LCD display and when to
press the "Set" button on the ESC by flashing a red LED.
A flashing green LED will signal the end of the process.

An LCD Button shield is used in this sketch but this.
Connection pins for this are: 8, 9, 4, 5, 6, 7, 10

The connections are:
9-12V onto the ESC when needed.
Servo control pin for ESC on Arduino pin D3
Red LED on Arduino Analogue Pin A1
Green LED on Arduino Analogue Pin A2

*/

#include <Servo.h>
#include <LiquidCrystal.h>
#define LCD_BACKLIGHT_PIN 10  // D10 controls LCD backlight

LiquidCrystal lcd( 8, 9, 4, 5, 6, 7 );   //Pins for the freetronics
    //16x2 LCD shield.
    // While this particular display has some buttons as well we will
    // not bother to use them.

Servo ESC;  //Servo object called "ESC"
int throttle = 0;  //Variable for the throttle setting.
const int grnLEDpin = A2;
const int redLEDpin = A1;


void setup()
{
  digitalWrite( LCD_BACKLIGHT_PIN, HIGH );
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);

  pinMode(grnLEDpin,OUTPUT);
  pinMode(redLEDpin,OUTPUT);
  ESC.attach(3,600,2250); //attach the ESC to pin 3
  //Due to problems with the ESC recognising the maximum
  //position at the default settings, the figures after
  //the pin number are the microsecond signals for the
```

```
  //minimum and maximum that the ESC will recognise.
  // 600 and 2250 work.
  throttle = 0;  //Set the throttle to minimum
  ESC.write(throttle); //Set the ESC signal to minimum ie 100% reverse.
  //At this point the ESC's power should be connected.
  lcd.clear();  //make sure screen is clear again.
  lcd.setCursor(0,0);  //Move cursor to top left corner
  lcd.print("Connect Battery");
  lcd.setCursor(0,1);
  lcd.print("Hold SET 2 sec");
  delay(10000);  //Allow the user time to connect the battery to
  //the ESC and hold the SET button.  The button MUST be released
  //before the ESC's red light stops flashing otherwise the ESC
  //will enter programming mode.

  lcd.clear();  //make sure screen is clear again.
  lcd.setCursor(0,0);  //Move cursor to top left corner
  lcd.print("Moving 2 Neutral");

  redBlink();  //Jump to the Blinking red LED subroutine
  throttle = 90;  //Set the throttle to 90 degrees (neutral)
  ESC.write(throttle);  //Set the ESC signal to the neutral position.
  lcd.clear();  //make sure screen is clear again.
  lcd.setCursor(0,0);  //Move cursor to top left corner
  lcd.print("Press SET Button");

  delay(10000);  // allow a 10 second delay for the ESC to signal
  //that it is done.
  lcd.clear();  //make sure screen is clear again.
  lcd.setCursor(0,0);  //Move cursor to top left corner
  lcd.print("Moving 2 Max");

  redBlink();  //Jump to the Blinking red LED subroutine
  throttle = 180;  //Set throttle to the maximum forward position.
  ESC.write(throttle);
  lcd.clear();  //make sure screen is clear again.
  lcd.setCursor(0,0);  //Move cursor to top left corner
  lcd.print("Press SET Button");

  delay(10000);  // allow a 10 second delay for the ESC to signal
  //that it is done.
  lcd.clear();  //make sure screen is clear again.
  lcd.setCursor(0,0);  //Move cursor to top left corner
  lcd.print("Moving 2 Backwd");

  redBlink();  //Jump to the Blinking red LED subroutine
  throttle = 0;  //Set throttle to the maximum reverse position.
  ESC.write(throttle);
  lcd.clear();  //make sure screen is clear again.
  lcd.setCursor(0,0);  //Move cursor to top left corner
  lcd.print("Press SET Button");

  delay(10000);   //The ESC should now be calibrated.
  lcd.clear();  //make sure screen is clear again.
  lcd.setCursor(0,0);  //Move cursor to top left corner
  lcd.print("Done.");
}

void loop()
{
```

```
  //Blink the green LED to signal end of process
  digitalWrite(grnLEDpin, HIGH);
  delay(1000);
  digitalWrite(grnLEDpin, LOW);
  delay(1000);
}

void redBlink()
{
  //Blink the redLED to signal the user to press the "Set" button

  for(int i = 1; i < 6; i++)  //do the sequence for 10 seconds.
  {
    digitalWrite(redLEDpin, HIGH);
    delay(1000);
    digitalWrite(redLEDpin, LOW);
    delay(1000);
  }
}
```

I trust this helps make things a bit easier for you as you set up a **Hobbywing EZRUN 18A ESC** for use in whatever project you are applying them.

Hamish Trolove

www.techmonkeybusiness.com