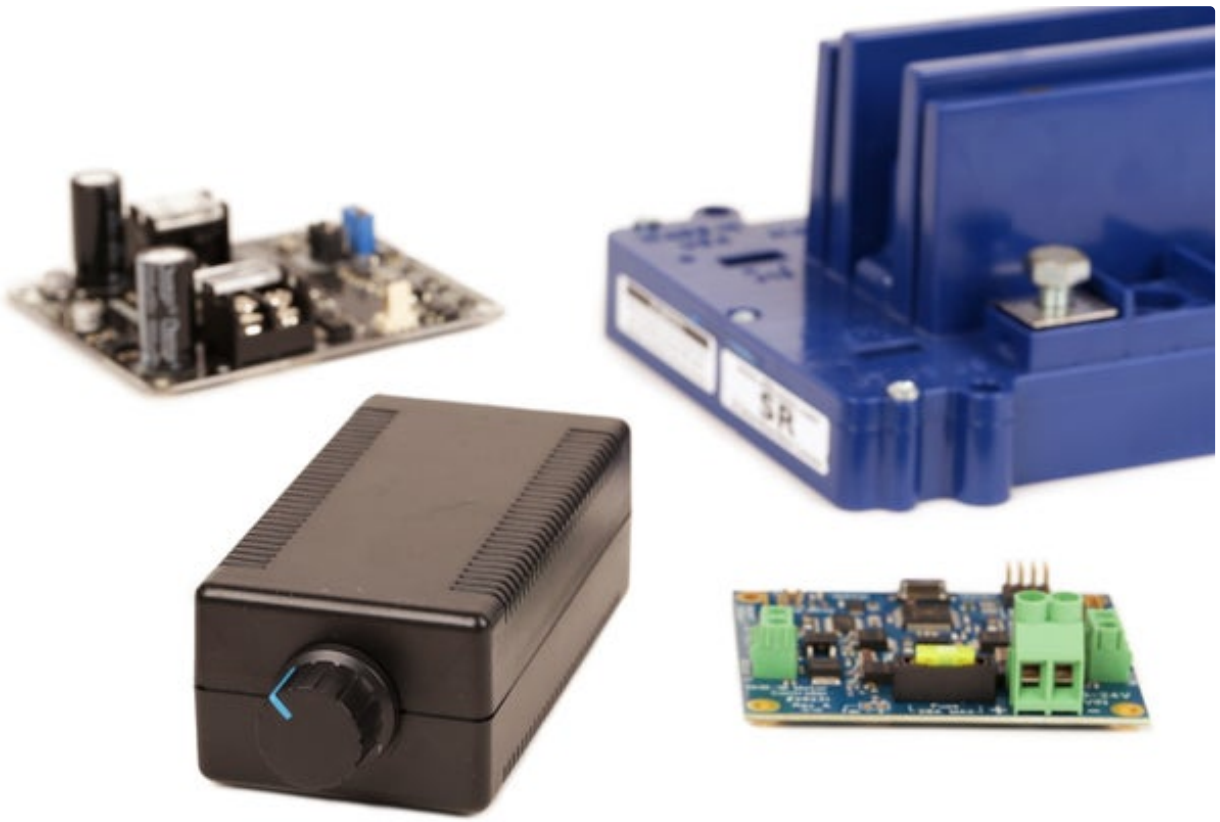
LESSON 3 :

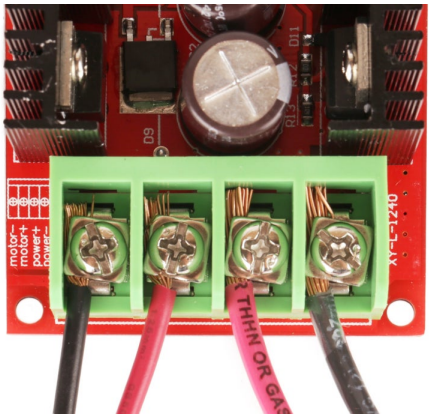
with in

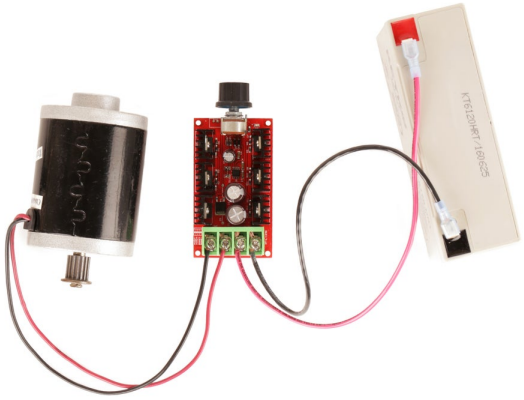


To control a large motor's speed, you need a motor controller. There are a range of motor controllers available on the market to deal with motors up to about 50A, but there are only a small handful capable of dealing with very high current electric motors (over 50A). Over the course of this lesson we are going to explore four different motor controllers, and review how to use them. By the end of this lesson you should be more than confident to power up a brushed DC motor of nearly any (reasonably sane) size and control its speed, with or without an Arduino board.

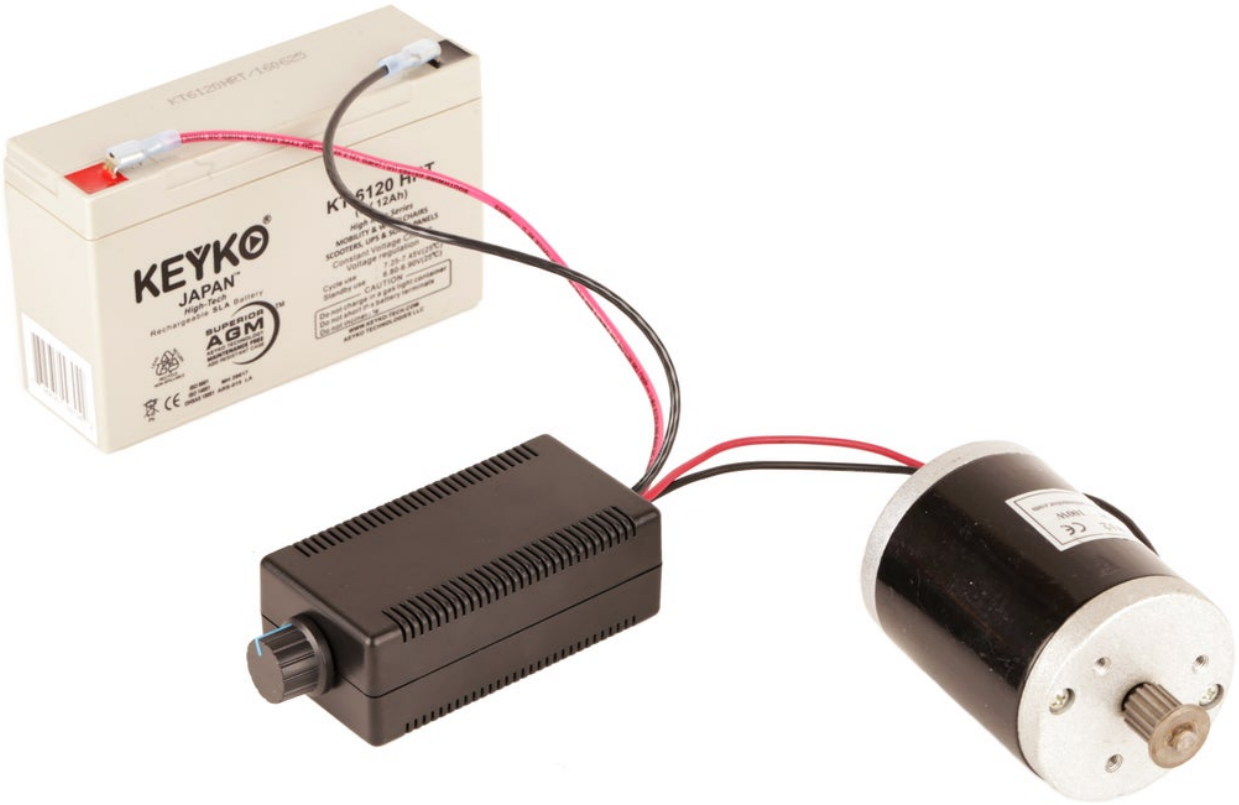
The easiest way to control a relatively low current 12-24V motor is by using a generic analog DC motor speed controller. This type of controller has a potentiometer to vary the speed of the motor. These controllers can be found with a wide range of power ratings. However, this type of controller is typically best for motors in the 5A to 20A range. For this example I selected a speed controller rated at 30A. The reason for this is the largest motor I am looking to control has a stall current no more than 15A, and it is advisable to get a controller rated for twice as much as your motor's typical operating current.

This type of generic speed controller is best when you want an easy solution that is pre-made and can manually control (with a knob) the speed of the motor in one direction. The shortcoming of this type of controller is that they are typically unable to reverse the motor direction, and not suited for microcontroller control.



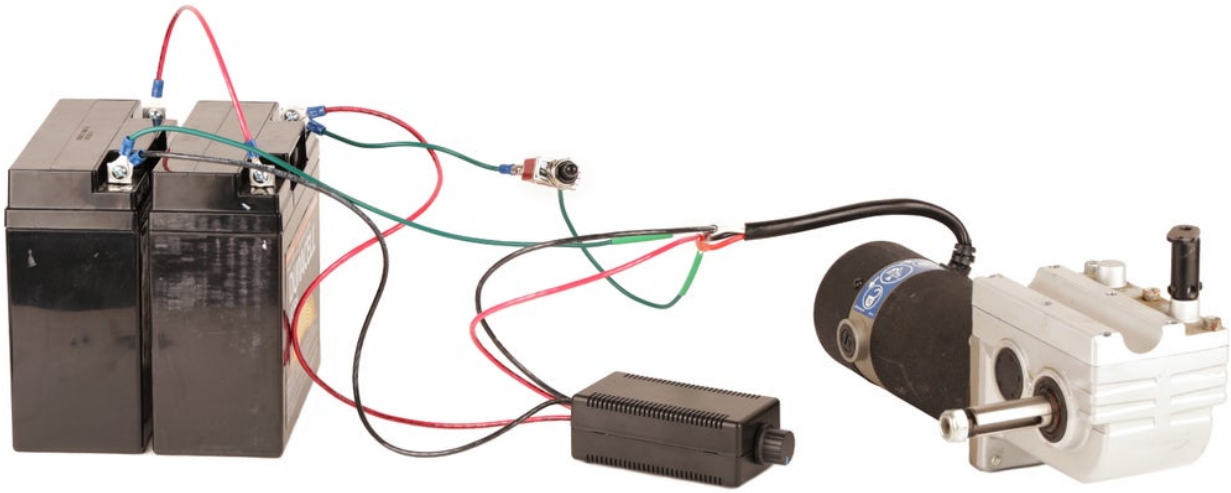


To wire up a DC speed controller, you connect the motor power cables to the motor screw terminals on the controller, and the battery wires to appropriate battery screw terminals on the controller. Be mindful the wires are being gripped firmly and none of the wire strands have gotten loose and are sticking out.



Once the wires are attached, close the case back up. There is potential high current and high heat on the circuit board. The enclosure will prevent shocks, burns and electrical shorts.

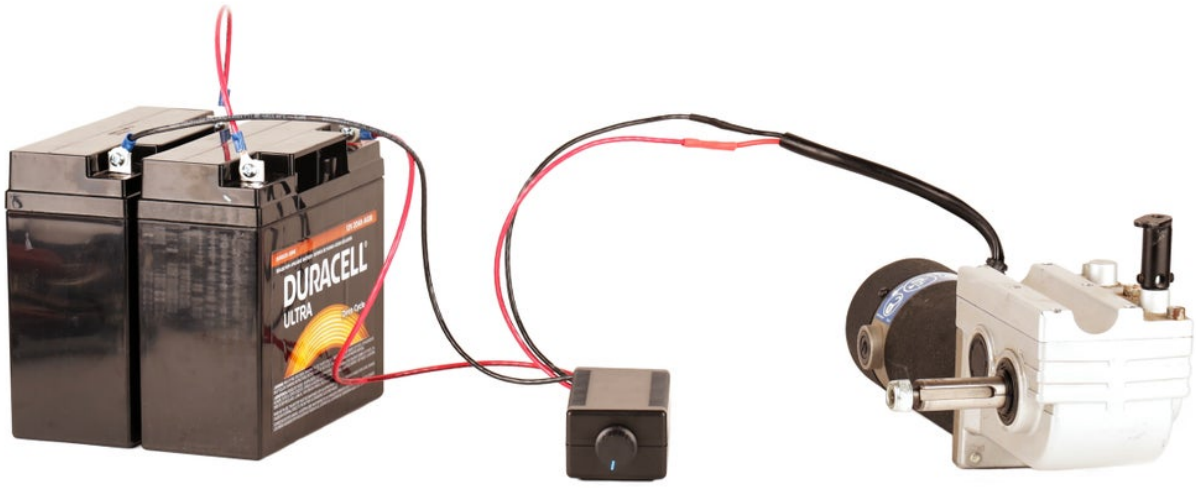
To control the motor simply turn the knob. If everything is wired correctly, your motor's speed should begin to ramp up as the knob is turned clockwise. As you will notice, with this type of controller, your motor will only ever rotate in one direction.



Wiring up a wheelchair motor to a DC speed controller is a little bit different because of the electronic brake attached to the back of the motor. In normal wheelchair operation, the brake is a fail safe to prevent the wheelchair from moving when it is not powered. So, if something goes wrong and the brake is not energized with 24V by the wheelchair control circuit, the wheelchair motor won't spin.

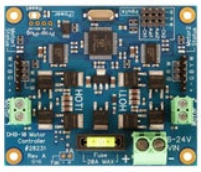
This can result in catastrophe when using standard (non-wheelchair) controllers. If you power the motor without releasing the brake, you will stall the motor (keep it from spinning) and force the wheelchair motor to draw its maximum current. The reason for this is that the motor is fighting against the brake as hard as it can to try to spin, and will draw as much current as it is able while doing so. If this happens, there is a chance you will overwhelm and eventually fry the motor controller and release its magic smoke!

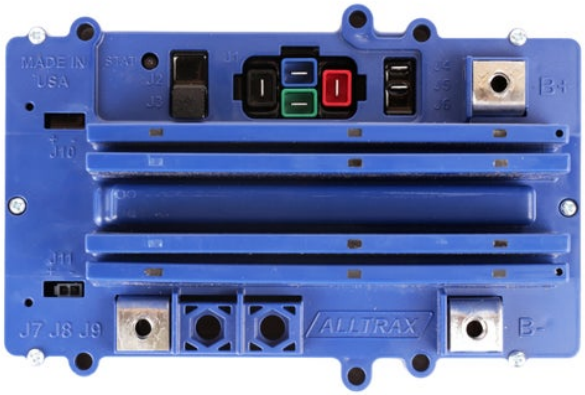
If you keep the electronic brake installed, it is important to connect the brake wires directly to the battery terminals. In the example above, two green wires were used to extend the two white brake wires (so they are easier to see) and attached to the battery bank. One wire has a switch connected in series with it so that the brake can be toggled on and off. If it is attached without a switch, the electromagnet inside the brake will eventually drain the batteries. The switch must be turned on before the motor's speed is adjusted using the controller.



A better solution for connecting a wheelchair motor to this (or any) motor controller is to remove the

brake from the motor altogether. Fortunately for you, I have posted detailed instructions for removing an electric brake from a wheelchair motor. Once the brake is removed, you can simply connect it to the controller as you would any other DC motor.

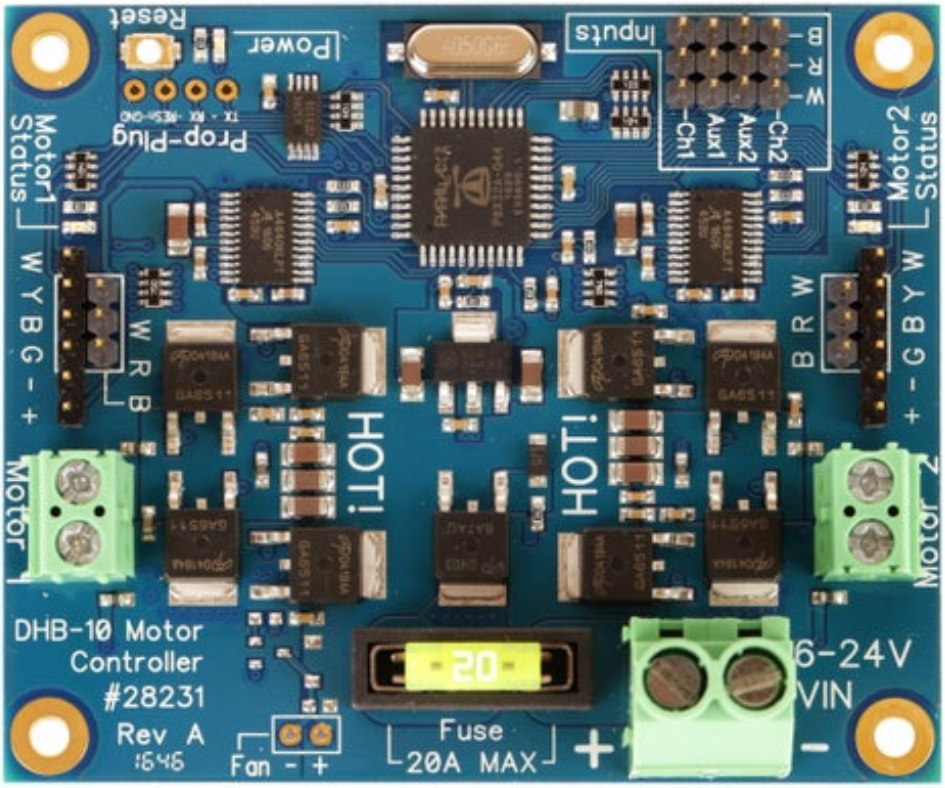




To control the speed of larger motor using an Arduino, you would need a motor controller board.

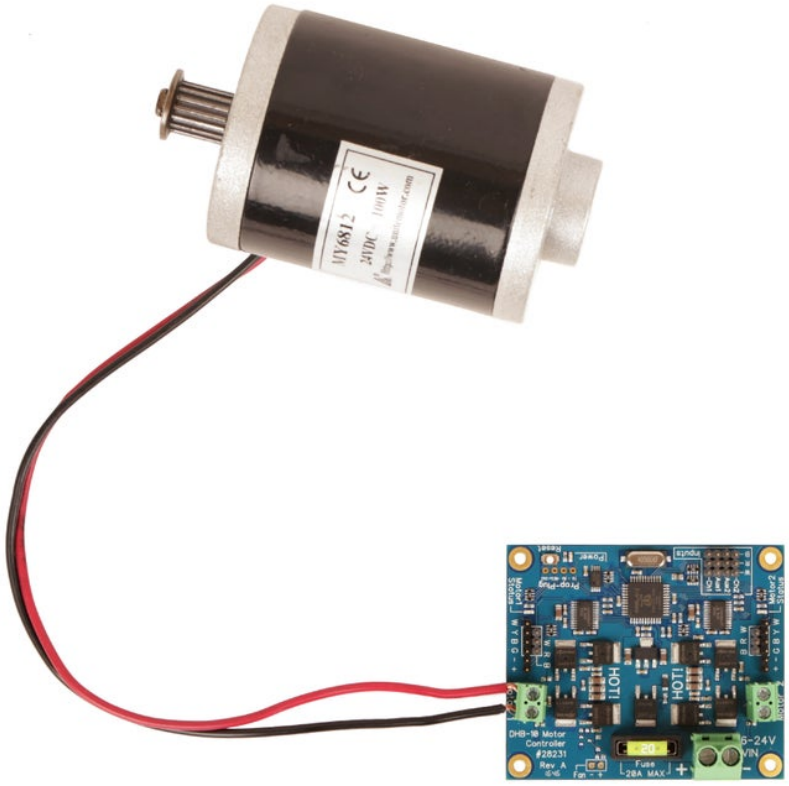
Many motor controller boards that interface with microcontrollers are H-bridge based, such as the Parallax DHB-10 and Cytron MD30C controllers. This is a special type of circuit that allows you to reverse the voltage polarity of a motor's power supply. This, in turn, changes the motor direction. You can learn more about H-bridges in the Motors and Motion Lesson of my Robotics Class, and see it in action in the Reversing a Motor lesson of this class.

However, when you begin controlling motors of 100A or greater, these boards are typically only designed to control speed, and not reverse direction, such as the Alltrax 48300 controller. These types of controllers are typically used in electric vehicles which use external circuitry to reverse motor direction.

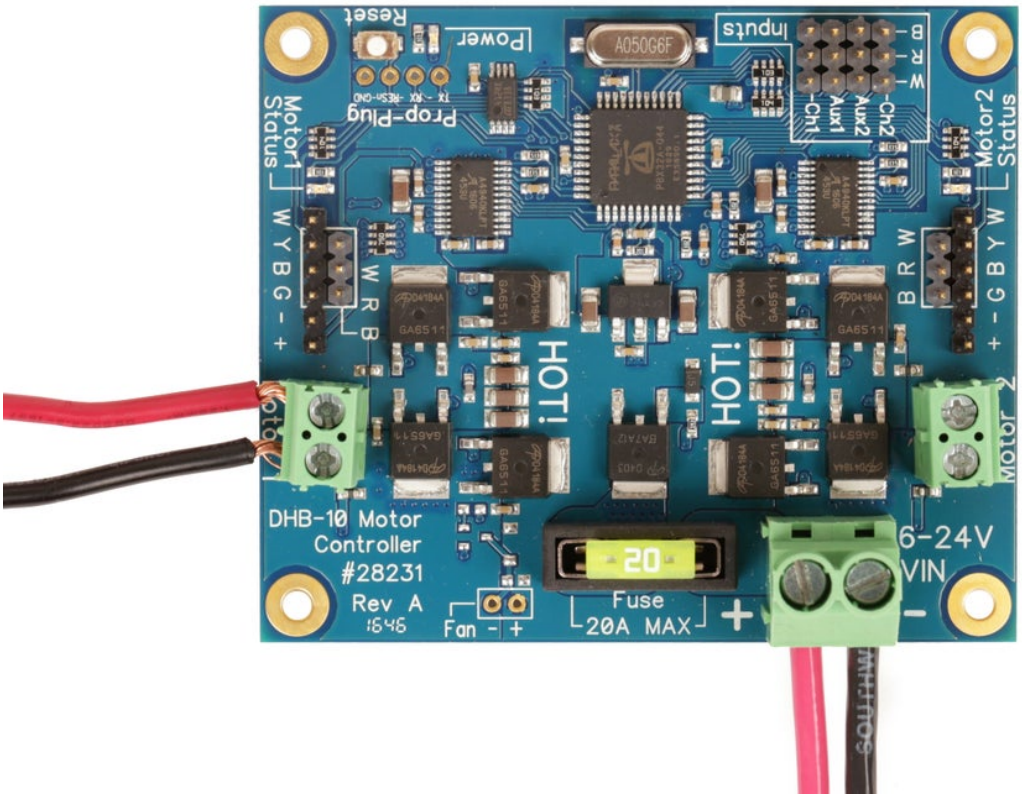


The Parallax DHB-10 Dual H-Bridge 10 Amp Motor Controller is good for controlling motors with a stall current of around 5A, such as the MY68 DC motor. This circuit board can control motors with a power supply of up to 24V and handle currents up to 10A continuous and brief surge current up to 12A. However, you shouldn't plan on running a motor with a 10A stall current off of this board. As already mentioned, it is wise to get a motor controller that is rated for double the motor's stall current.

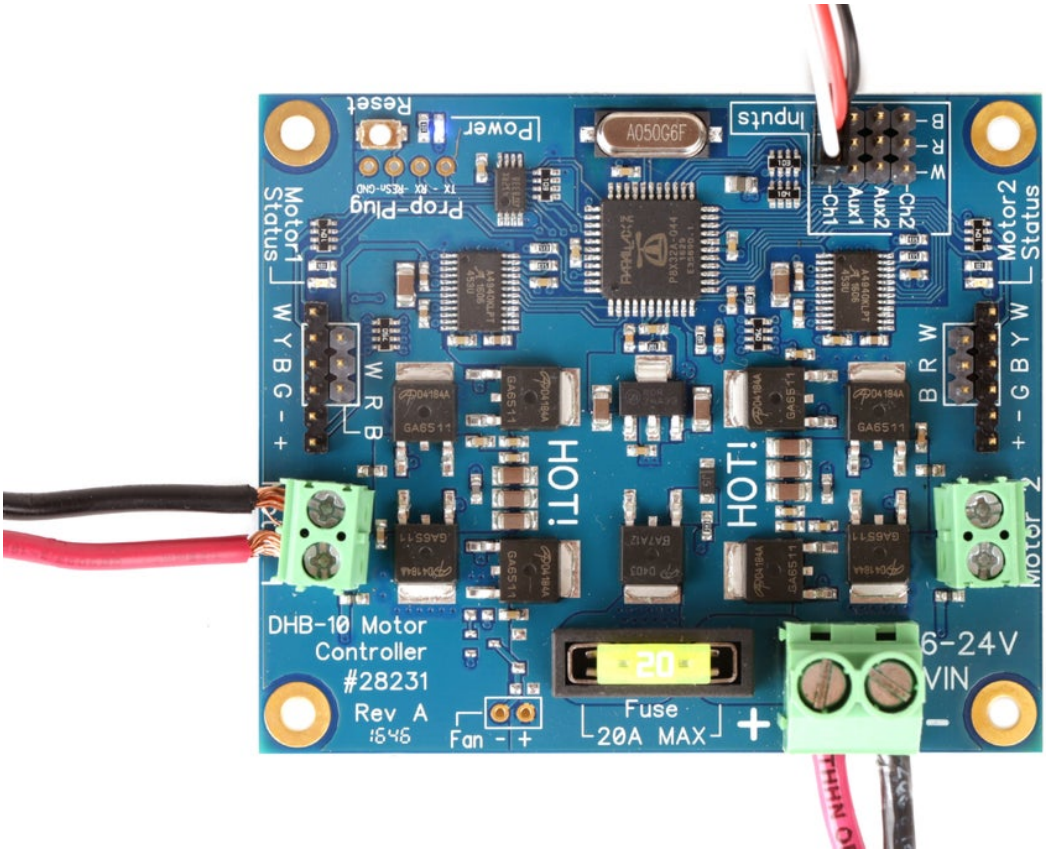
Also note that there is a 20A fuse on this board. Even though a single channel can theoretically handle a 12A surge, if you were to connect two motors, and have power surges of 12A on both channels at once, it could potentially blow the safety fuse.



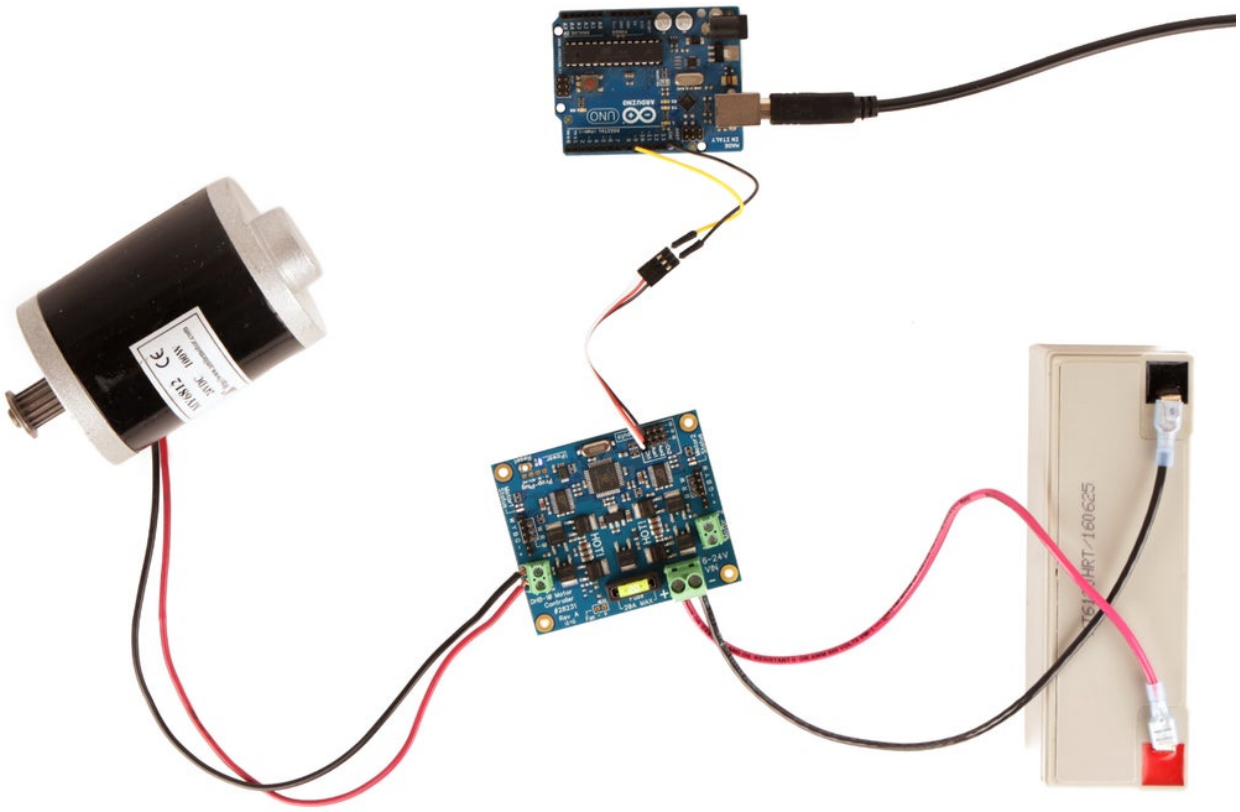
To control the motors, we need to connect the motor to the 10A motor 1 terminal. This is the green terminal with the set screw helpfully labeled "Motor 1." If we had an additional motor, we could also connect it to "Motor 2."



Once the motor is connected, connect the power supply to the terminal labeled "6-24V VIN." In case it was not clear, VIN stands for voltage-in. For this demonstration I am using a 12V 6Ah battery.



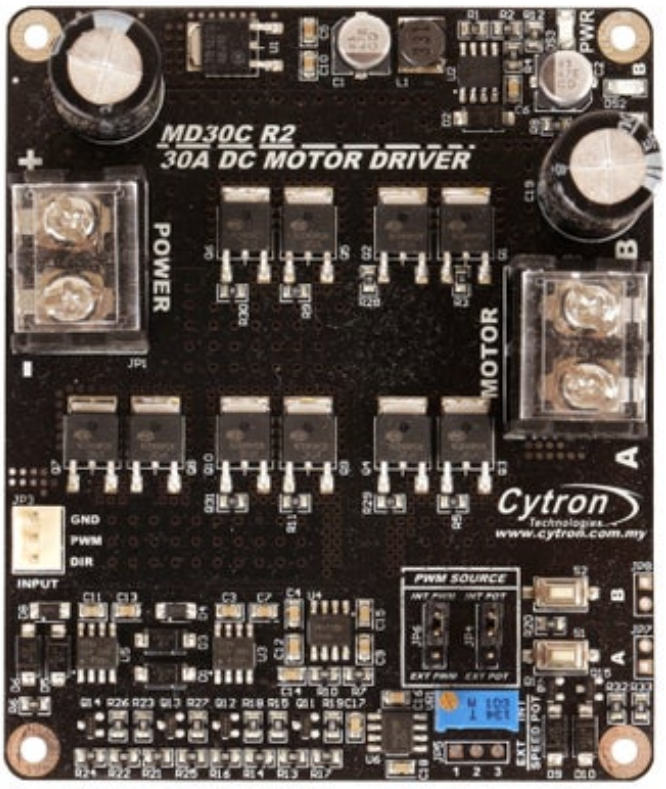
The next order of business is to connect a microcontroller to the motor controller board. This controller board can be controlled like a servo motor. Therefore, it should not be surprising that we need to plug a servo extension cable into the Channel 1 header pin inputs (labeled "Ch1"), with the black, red and white wires lining up appropriately ("WRB" - as labeled).



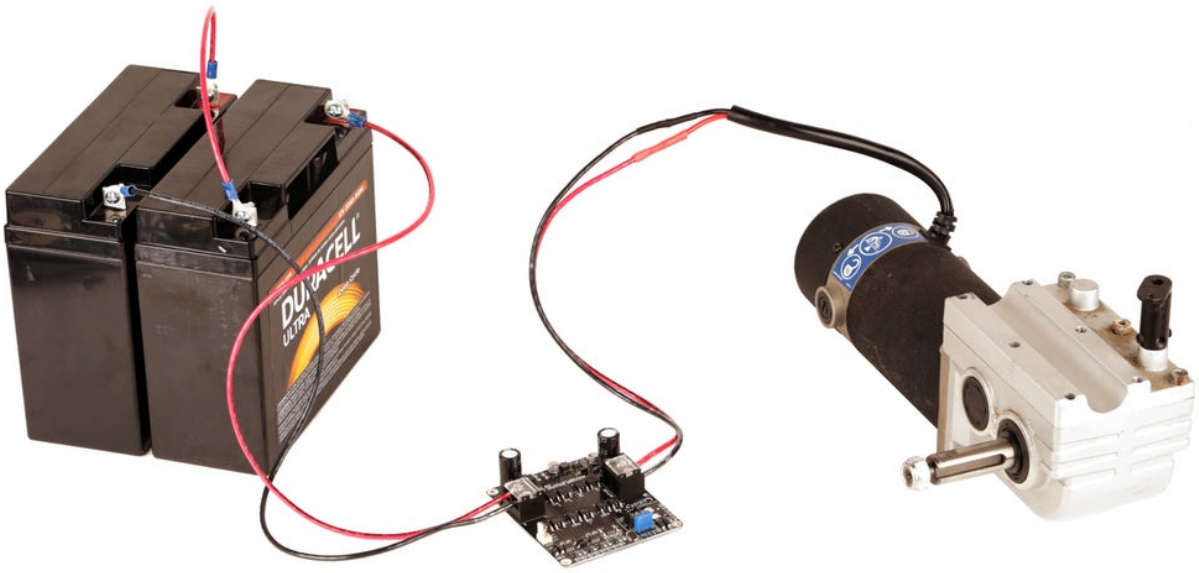
Using hookup wire, the white wire from the servo extension cable should should then be connected to Digital Pin 9, and the black wire with the Arduino's ground pin. The red wire can be ignored.

After all of the connections are made, the following code can be uploaded to the Arduino to control the speed of the motor:

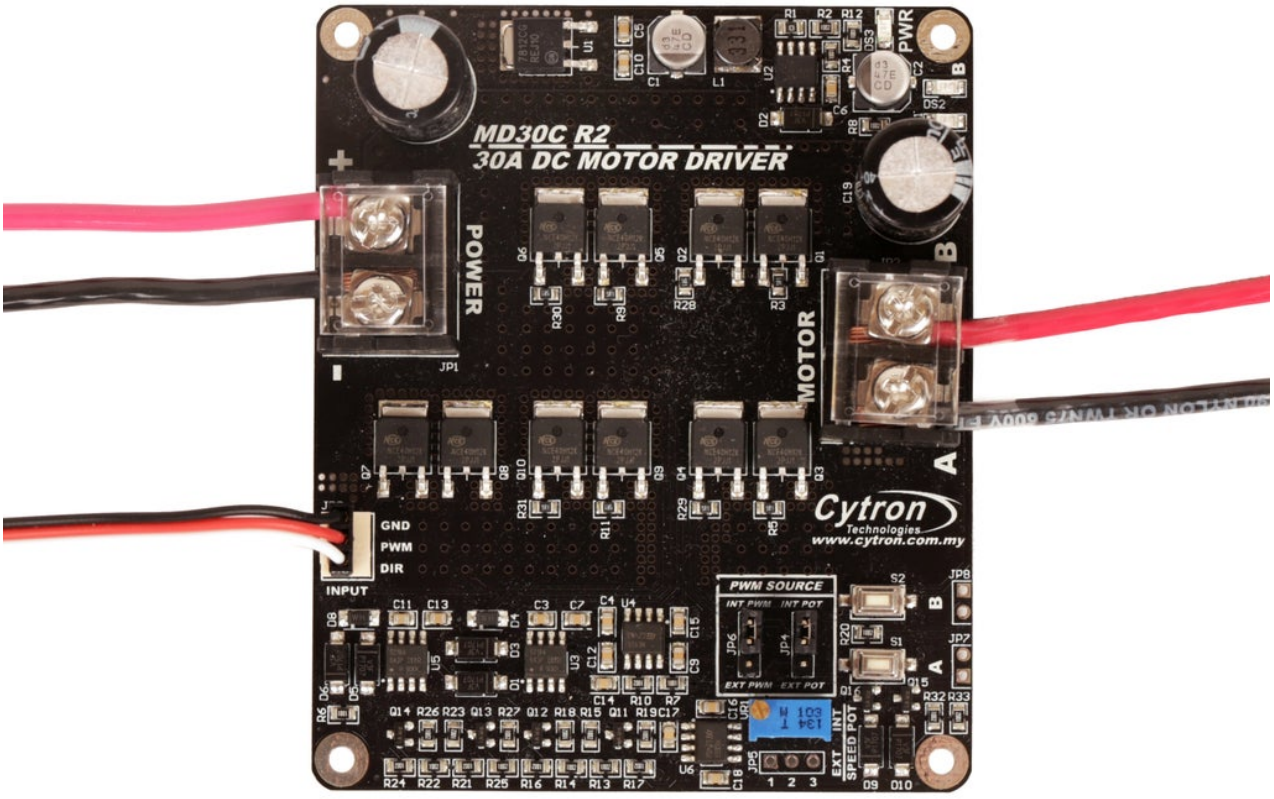
```
// Include the servo library
#include <Servo.h>

// Create a servo instance
Servo servo1;

void setup() {

   servo1.attach(9);

   // Set the servo to its zero position and wait one second.
   // This may number may need to be tweaked slightly until spinning stops.
   servo1.writeMicroseconds(1497);
   delay(1000);

}

void loop() {

   // Spin slowly for two seconds
   servo1.writeMicroseconds(1400);
   delay(2000);

   // Spin a little faster for two seconds
   servo1.writeMicroseconds(1200);
   delay(2000);

   // Spin even faster for two seconds
   servo1.writeMicroseconds(800);
   delay(2000);

}
```



The Cytron 30A 5-30V Single Brushed DC Motor Driver is similar to the Parallax motor controller, but can only control one motor. However, it can handle three times as much current (30A) and a considerably high current spike (80A). This makes this controller suitable for controlling motors with stall currents from 15-20A such as the wheelchair motor. Unlike the Parallax board which could feasibly control 2 motors, this one is single channel and can only control a single motor.

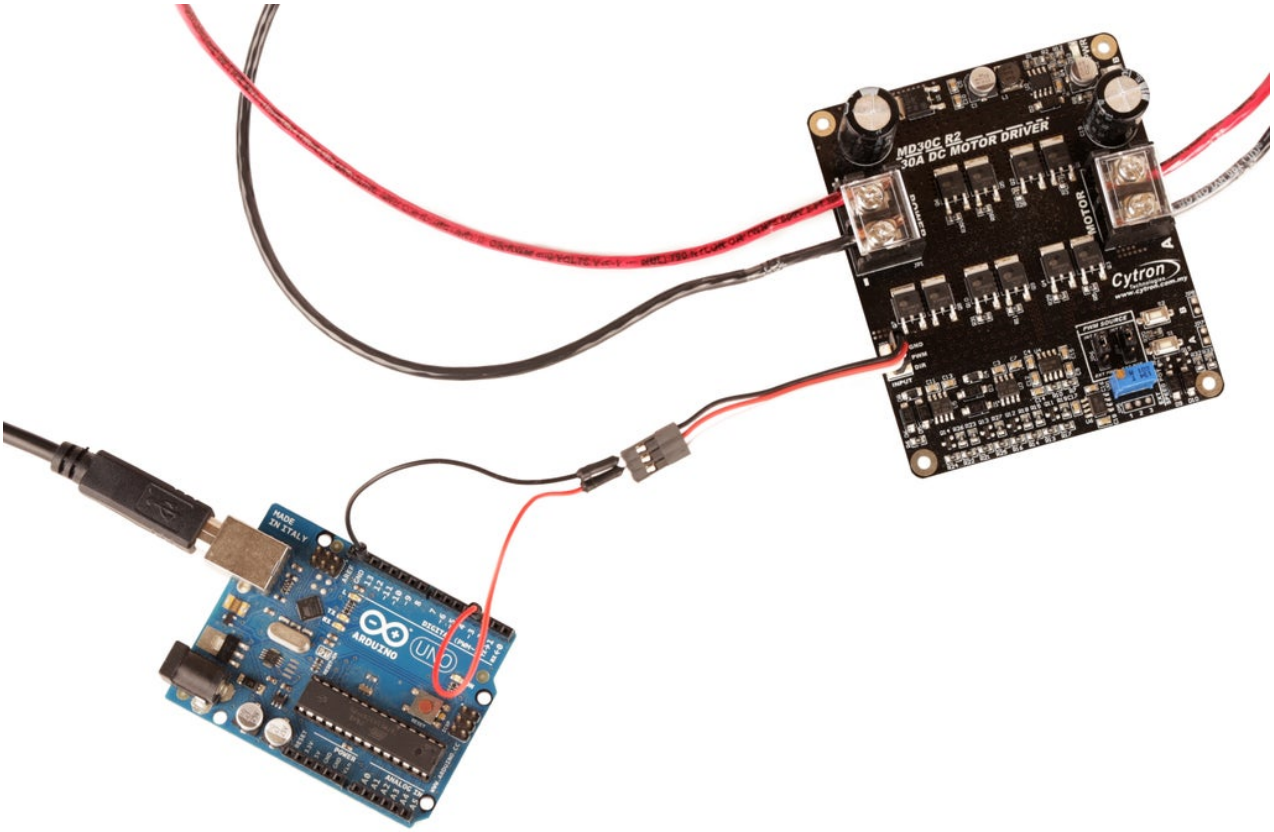


Wiring the motor and battery is very similar to all of the motor controllers we have encountered thus far. The motor gets connected to the large motor terminal block, and the battery gets connected to the other large power supply terminal block.



A servo extension cable is then used for controlling the board via Arduino. On the Cytron MD30C there is a header block with 3 pins for ground, PWM and direction. The PWM pin receives a PWM

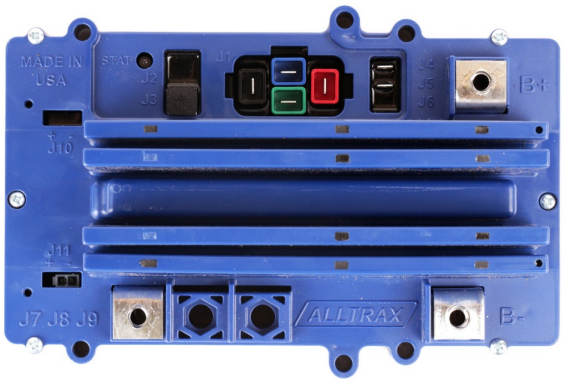
signal from your microcontroller and this simply controls the speed of the motor. The direction pin receives either a high or low signal, and this controls the direction of rotation (more on this in the Reversing a Motor Lesson).

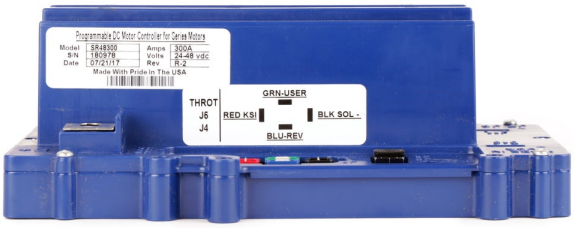


Connecting the board to an Arduino to get it to spin in one direction is rather simple. In this example I have the PWM pin connected to Arduino pin D3, and ground to ground.

The code to get it to move at different speeds is just as straightforward as you can see below:

```
void setup() {

    // Nothing here.

}

void loop() {

    // Turn the motor on at half speed for one second.
    analogWrite(3, 127);
    delay(1000);

    // Turn the motor off for one second.
    analogWrite(3, 0);
    delay(1000);

}
```





The Alltrax 48300 motor speed controller is designed for really large motors, and have a continuous current rating of 210A and a momentary maximum current rating of 350A (for sudden current spikes). This makes it suited for heavy duty applications. If you find this controller does not handle enough current for your application, you can check out one of their other controllers in the SR series , as they make an entire line of high current DC motor controllers. These controllers are typically meant for golf carts, but can be used to build a host of different kinds of electric vehicles.

However, before we can setup the motor controller circuit, we first need to discuss some external components.

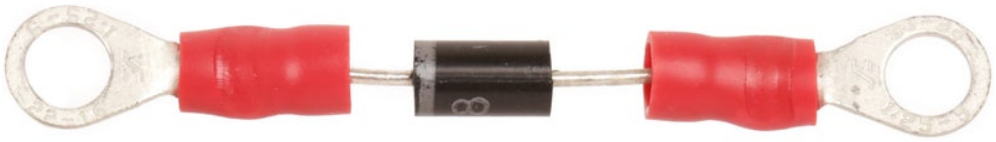
First thing's first, when dealing with high current applications, it is highly recommended to wire a high current fuse in series with your circuit that is rated for slightly less than the maximum current of the motor controller. It is cheaper to replace a $15 250A fuse than a $400 motor controller. As long as the circuit's current is kept below the motor controller's top rating, you should never have to worry about frying your control electronics.

The fuse needs to be connected in series between the battery cable and the cable that is going to provide power to the rest of the circuit.

Once installed, the exposed metal parts of the fuse need to be insulated with heat shrink to prevent possible short circuits. It is recommended to either use clear shrink tube, or to leave a gap that allows you to see the fuses viewing window. Being able to quickly and visually identify a blown fuse will save you a lot of headaches and guesswork.

After the fuse, the next component that should be wired in series is called a main contactor (sometimes also confusingly called a "solenoid"). When dealing with such large motors, it is assumed that power to the motor controller is going to be toggled on and off with a high current main contactor.

This component is essentially just a really large relay. When an electromagnet within the main contactor is energized, a solenoid inside of it toggles a switch which allows high current DC electricity to flow to the motor. When the coil is not energized, the switch is reversed and no electricity can flow.



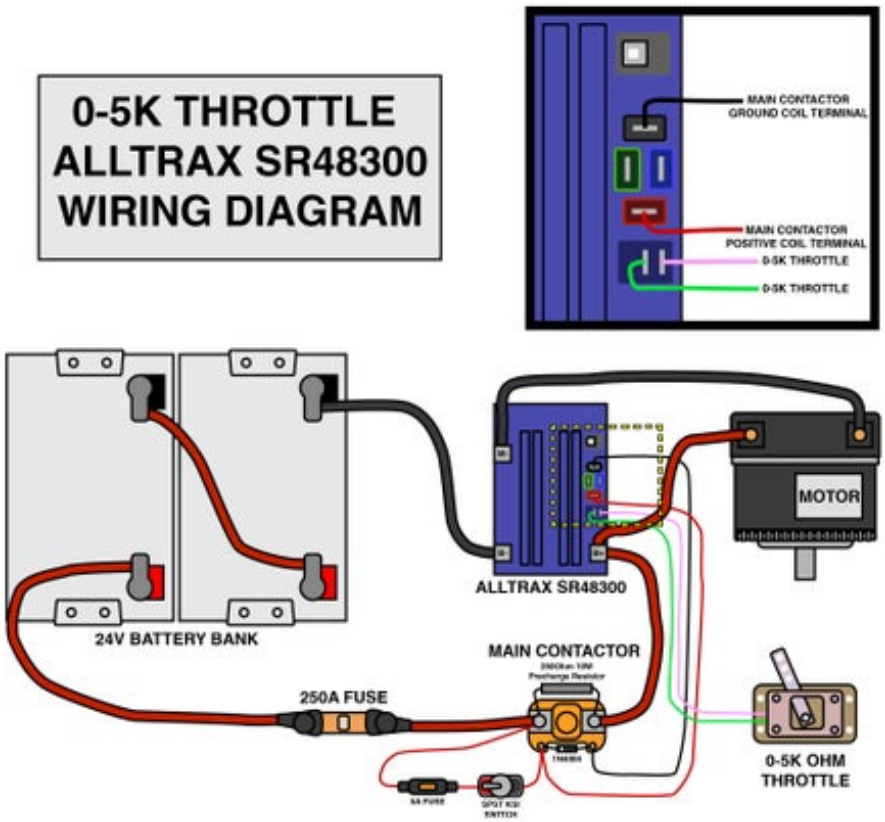
Two additional components are needed to keep the main contactor in tip-top operational order. The first is a 1N5408 diode, which serves as a snubber diode on the coil to protect from reverse voltage spikes, and keeps components further down the line from burning out.

This, of course, assumes that you are using a 200A main contactor. If you are using a larger contactor, you should check the datasheet for the appropriate diode.

The second necessary component is a high current 250 ohm 10W pre-charge resistor connected across the high current contacts. This resistor allows voltage to bypass the contactor and charge the very large capacitors within the motor controller. The reason you want to do this is because if engage the solenoid without pre-charging the capacitors, there will be an inrush of current as capacitors will try to draw as much current as they possibly can. Such a large current across the contactor terminals can potentially spot weld them shut in the 'on' position. In simpler language, your main contactor will not be able to turn off, and power to your motor will be stuck turned on.

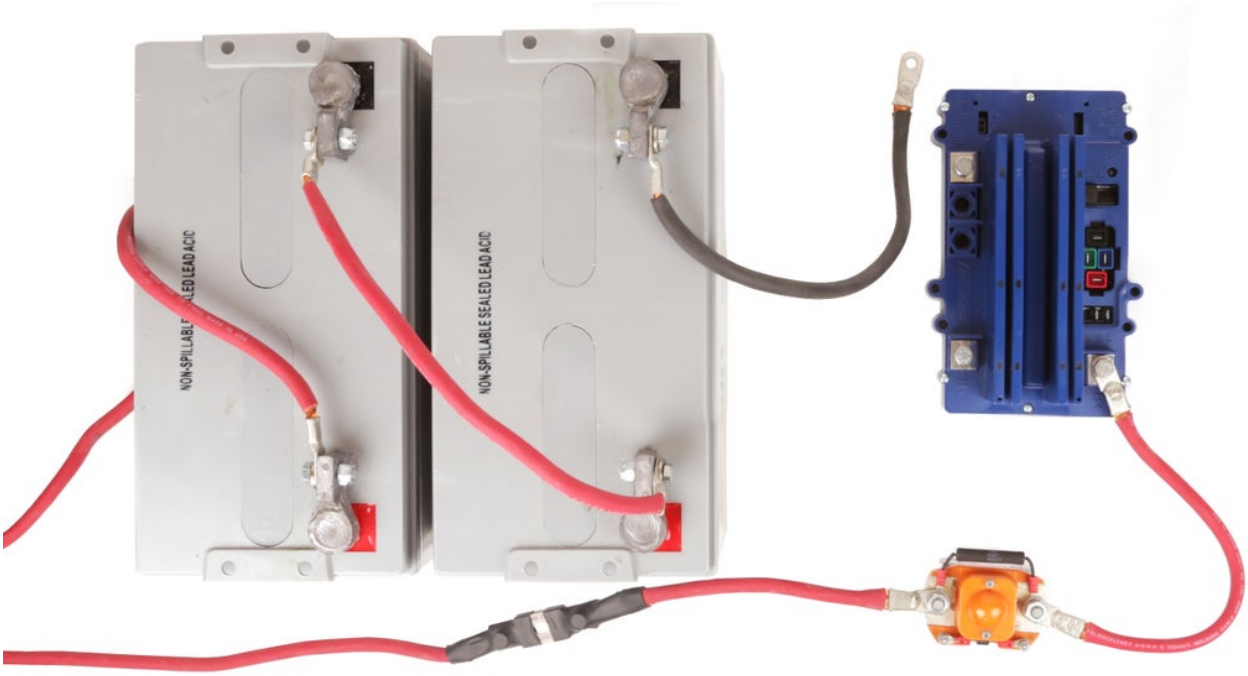
Again, if you are using a larger contactor than the one shown, check the datasheet for the appropriate size pre-charge resistor.

Attach the pre-charge resistor between the high current terminals on the main contactor, and the snubber diode between the low current coil terminals.

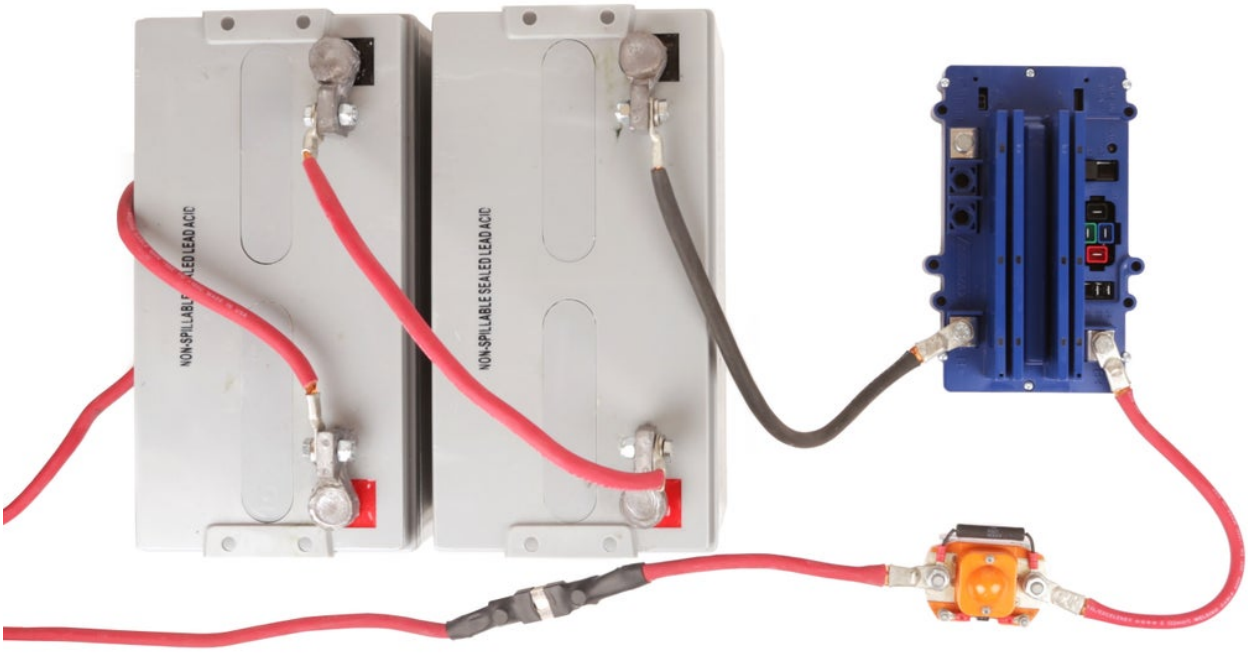


Once the 250A fuse is installed in-line with a power cable and the snubber diode and pre-charge resistor are attached to the main contactor, it is time to build the circuit. Above you will see the complete circuit for controlling a motor using a 0-5K Ohm electromechanical throttle. If you are not sure what that means, don't worry. It will be clear in a little bit.

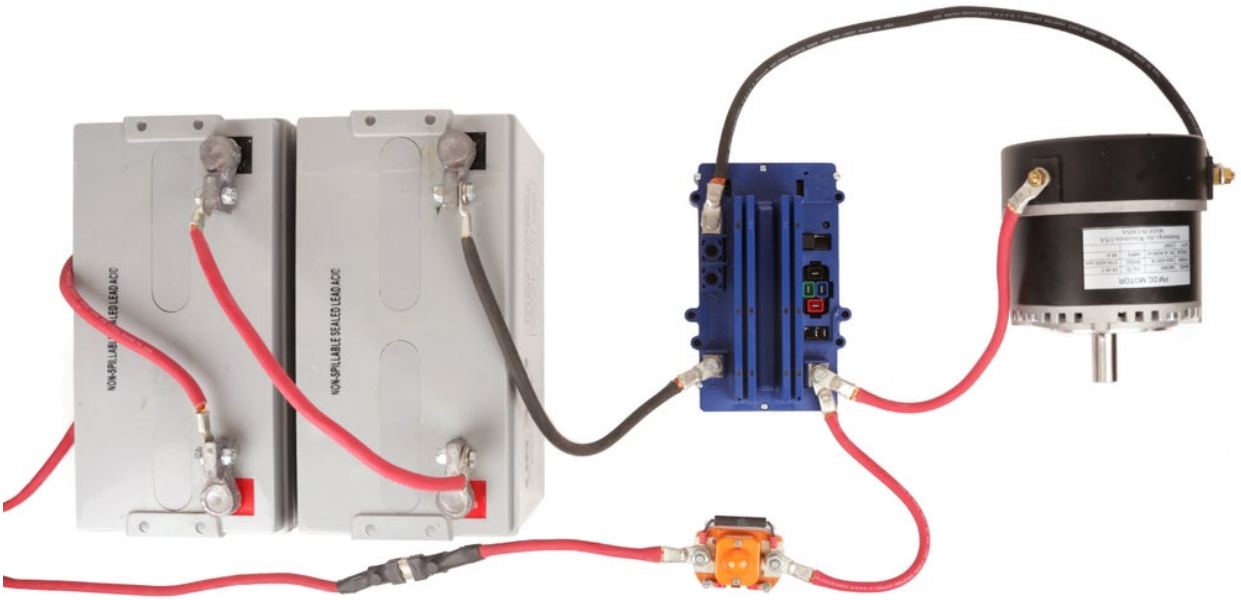
For additional reference, you can also review the official Alltrax SR wiring diagram.



To begin, let's make all of the high current connections in the circuit. The positive terminal of the battery bank, should be wired in series between the 250A fuse and the main contactor. The other side of the main contactor should then be connected to the B+ terminal on the motor controller.



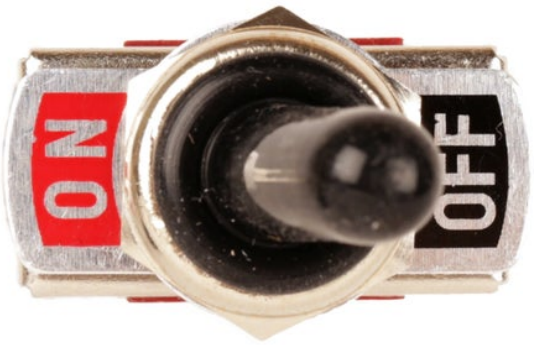
Next, the ground terminal of the battery bank needs to be connected to the B- terminal on the motor controller.



The last high current connection that needs to be made is with the motor. One of the motor's terminals need to go to the M- terminal on the Alltrax motor controller, and the other terminal should go to the B+ terminal of the Alltrax motor controller.

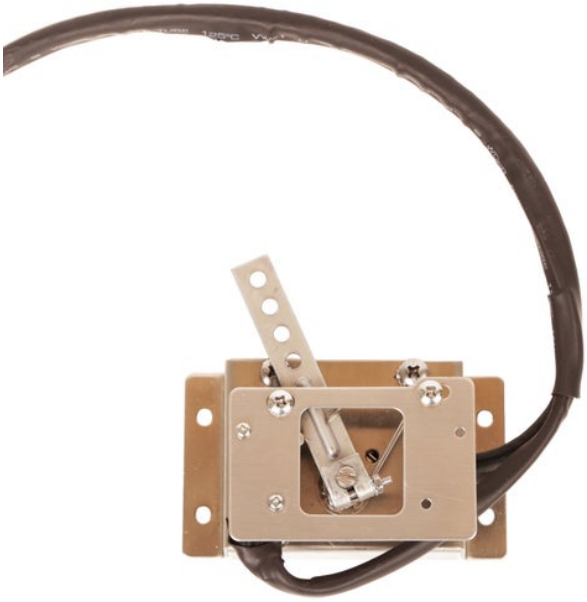
After the high current cables connections are made, it is time to focus on the 24V low-current control circuitry. This circuit also needs to be protected with its own 5A fuse (58V DC) fuse. Both of the fuse values were determined based on the datasheet, but 5A in particular is important because the solenoid drive pin has a 5A (peak) limit. There is no reason for allowing a current over 5A into the control circuitry.

In this example I am also using a mini blade fuse holder.



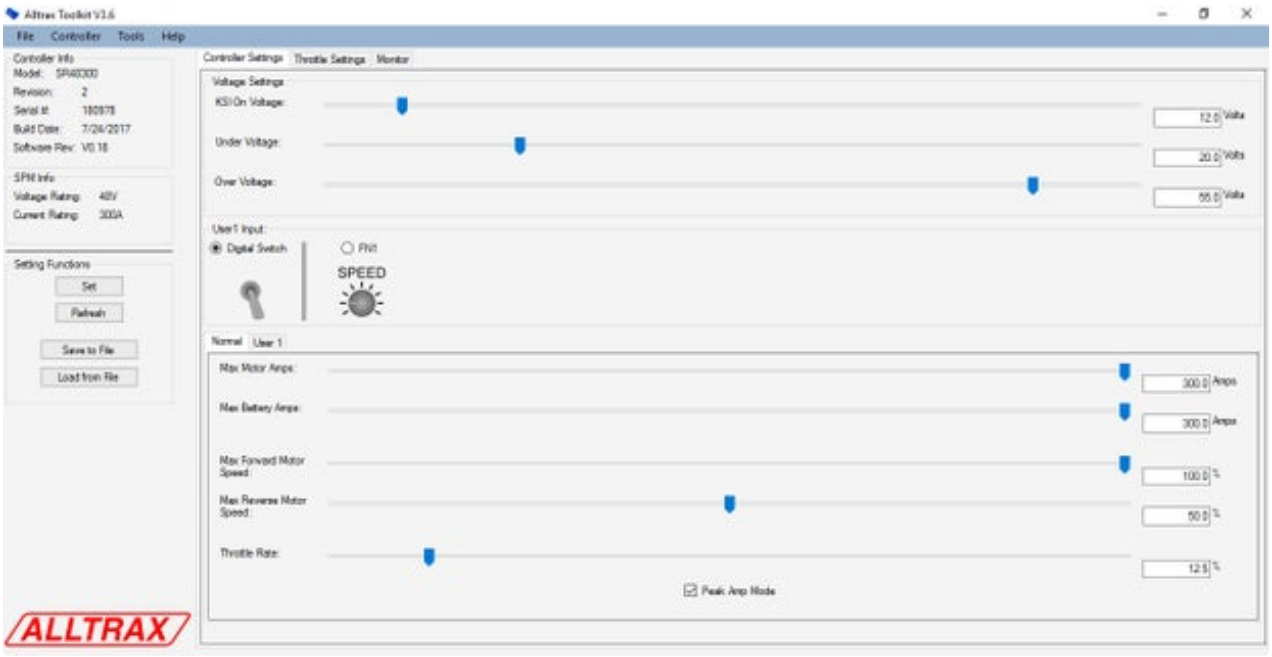
The fuse is wired in series with the KSI, which stands for Key Switch Input. This switch is meant to serve as the power switch (like a key switch in a car). In this example, we will be using an SPST on/off toggle switch, but you could use an actual key switch (more on that in the Further Considerations lesson).

Keep in mind that on account of the pre-charge resistor connected to the solenoid, this is not a power switch for the battery. It's purpose is to toggle on and off the control system.



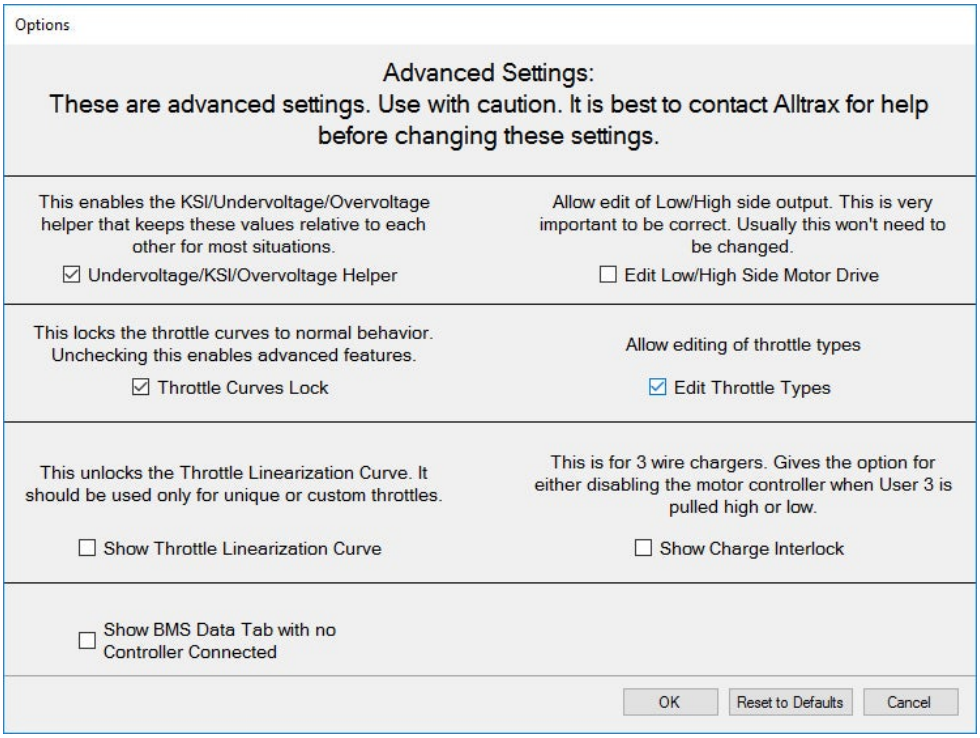
The last order of business is to connect the throttle. A throttle is a mechanical input which controls the speed of the motor, such as a foot pedal. There are many different possible configurations for the throttle, and each needs to be interfaced with the motor controller slightly differently. Therefore,

before we can attach a throttle, we need to first configure the software on the motor controller for the throttle we are using.



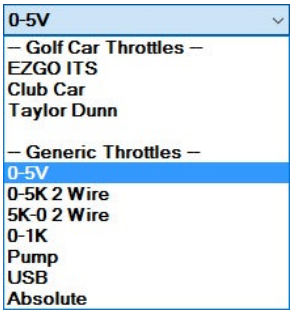
To interface with the Alltrax motor controller, we need to use the Alltrax Toolkit software interface. This software requires a PC running either Windows 8 or 10.

Once the Alltrax Toolkit software is installed, connect the motor controller to your computer using a USB cable and then open the software.



The first thing you are going to want to do is open the "Options" menu from the "Tools" drop-down menu on the top of the page.

Activate the checkbox next to "Edit Throttle Types", and then click "OK" at the bottom of the window. This will enable the throttle selection drop-down menu. This can be found under the "Throttle Settings" tab under the main screen.



Navigate over to the "Throttle Settings" tab if you have not done so already. Select the throttle drop-down menu. On the menu you will see multiple options, but the ones that are most important are the top four options under "Generic Throttles."

Those being:

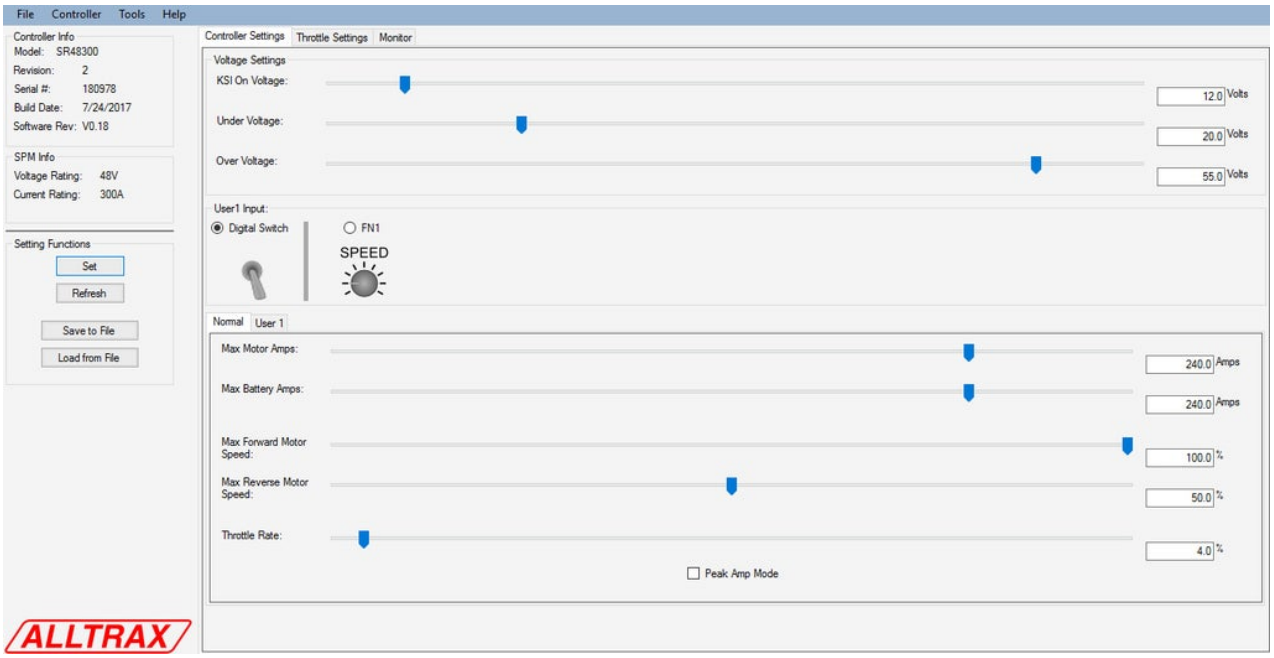
0-5V
A 0-5V throttle input controls the motor speed by using a voltage from 0 up to 5V. This voltage signal can either come from a throttle with an active voltage output or a 5V microcontroller.

0-5K 2 Wire
A 0-5K throttle uses a resistance value starting 0 (slowest speed) up to 5K (fastest speed). This input typically uses a mechanical input which turns a potentiometer to change the resistance.

5K-0 2 Wire
A 5K-0 throttle functions much in the same way as the 0-5K throttle, except it works the opposite. In other words, 5K is the slowest speed, and 0 is the highest speed.

0-1K

A 0-1K throttle is the basically just another resistance-based throttle with the range of 0 to 1,000 ohms.



While we are at it, we may as well return to the main "Controller Settings" tab and configure the remainder of the motor controller settings.

- The **KSI On Voltage** is the lowest voltage the controller will recognize at the KSI pin as its 'on' signal. For a 24V system, a 12V minimum voltage should be more than sufficient.

- The **Under Voltage** is the lowest voltage from the power supply that the controller will operate at. For a 24V system, this should probably be around 20V.

- The **Over Voltage** is the highest voltage the controller will accept from the power supply. This voltage should not exceed the maximum operating voltage of the motor.

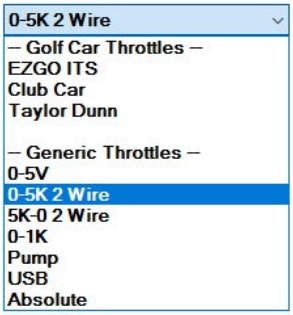
- The **Max Motor Amps** and the **Max Battery Amps** should be the same value. This value should be set to approximately 10A to 20A less than your largest fuse rating. With a 250A fuse in series with the battery bank, this should be set to 230A to 240A.

- The **Max Forward Motor Speed** sets the fastest speed that the motor should spin. This can be adjusted to personal preference.

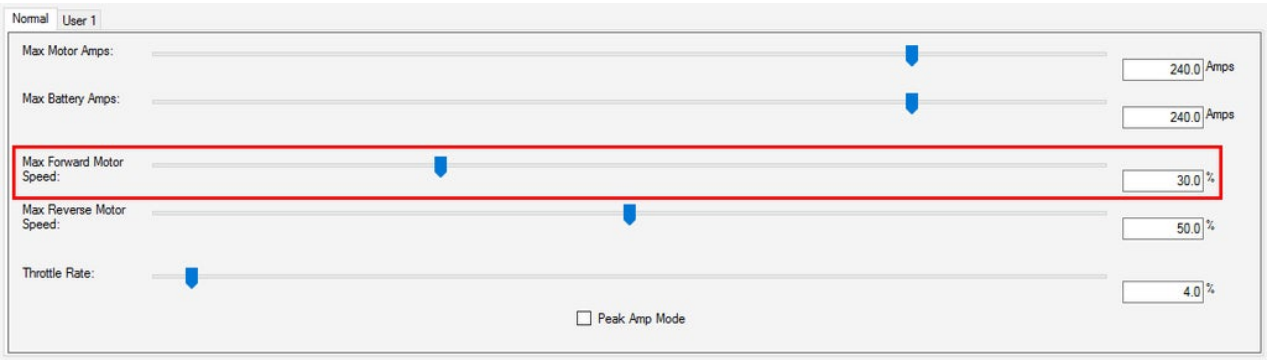
- The **Max Reverse Motor Speed** is a setting we will discuss in the Reversing a Motor Lesson.

- The **Throttle Rate** adjusts how quickly the motor accelerates. For most applications this should be set to a value of 10 or less.

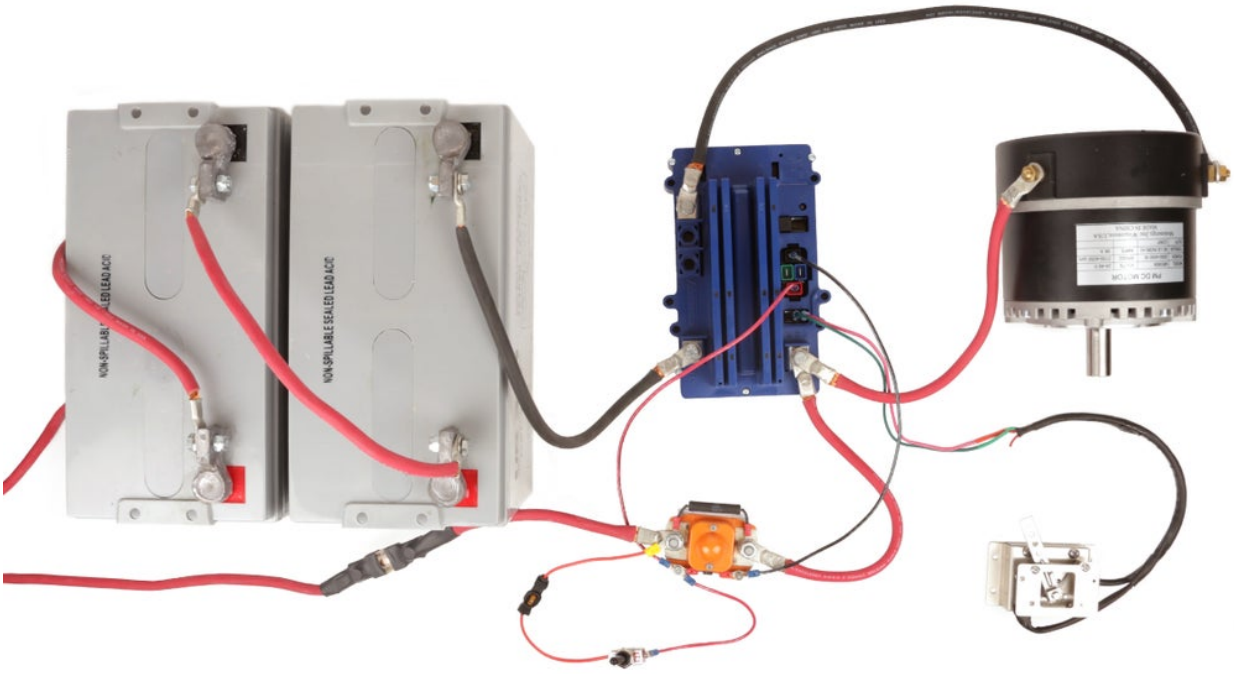
- The **Peak Amp Mode** checkbox on the bottom of the page should be toggled off. In this mode, the controller will provide an additional 15% more current over its maximum rating so long as the operating temperature of the motor controller is below 50C. Enabling this may provide too much current and blow a fuse.



We are going to start by configuring a 0-5K Ohm throttle, which is the most common type that you might encounter. Return to the "Throttle Settings" tab, select "0-5k 2 Wire" from the drop-down menu and click "Set" on the left-hand side of the screen. Wait for it to finish processing.



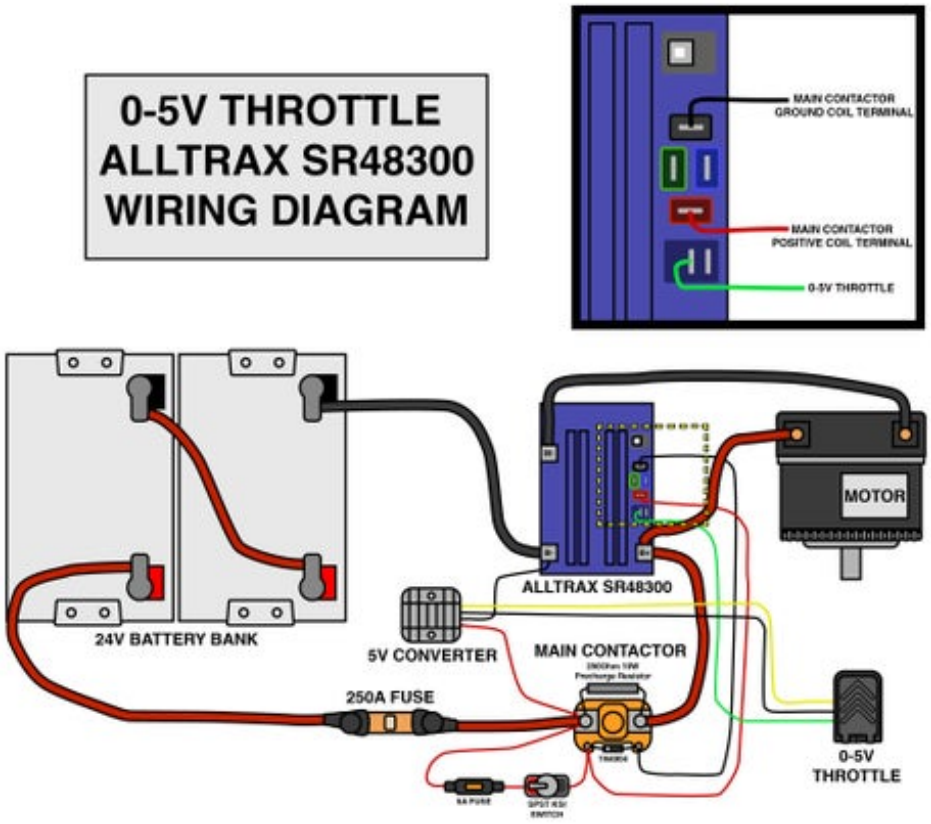
Another thing that you should do before powering up your circuit and testing your throttle is to lower the motor's maximum speed by adjusting the Max Forward Motor Speed dial and setting it around 30%. This will help ensure that you never have any unexpected surprises with such a powerful motor.



Finally, connect the throttle's (internal) potentiometer wires to the J4 and J5 jumpers on the motor controller. If your throttle has more than 2 wires and you are unsure which wires are connected to the 0-5K potentiometer inside of the throttle, you can use a multimeter to figure it out.

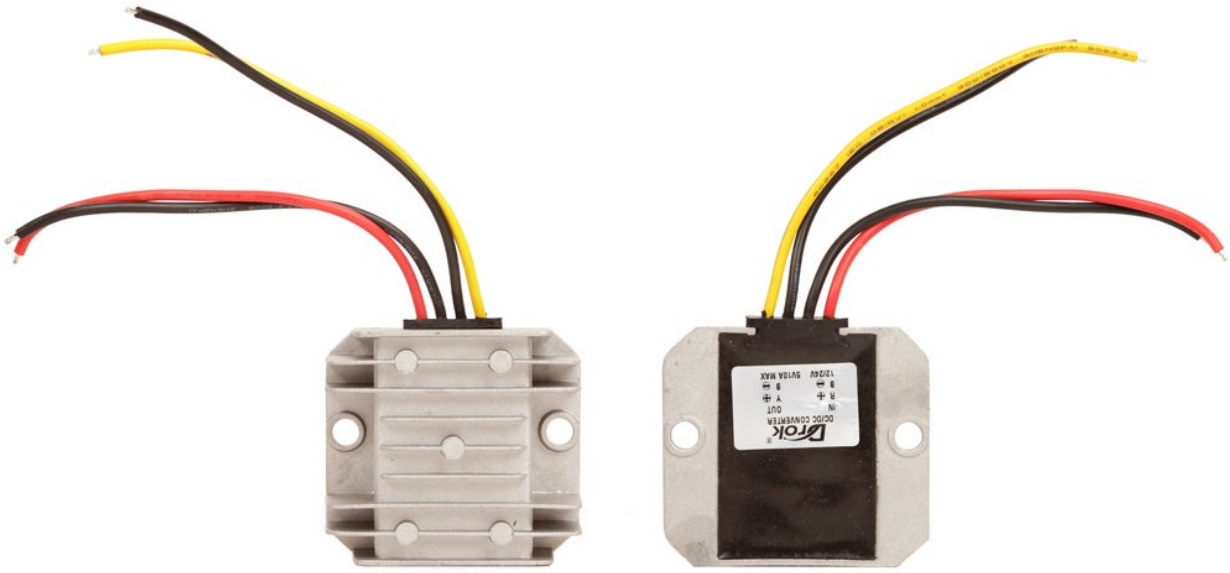
The final circuit should look something like the circuit pictured above.

Before you fire it up, you may want to consider somehow clamping your motor down to your workbench. Remember, a motor at this scale is as powerful as multiple horses, and if the circuit malfunctions in some way, it can cause some serious damage.

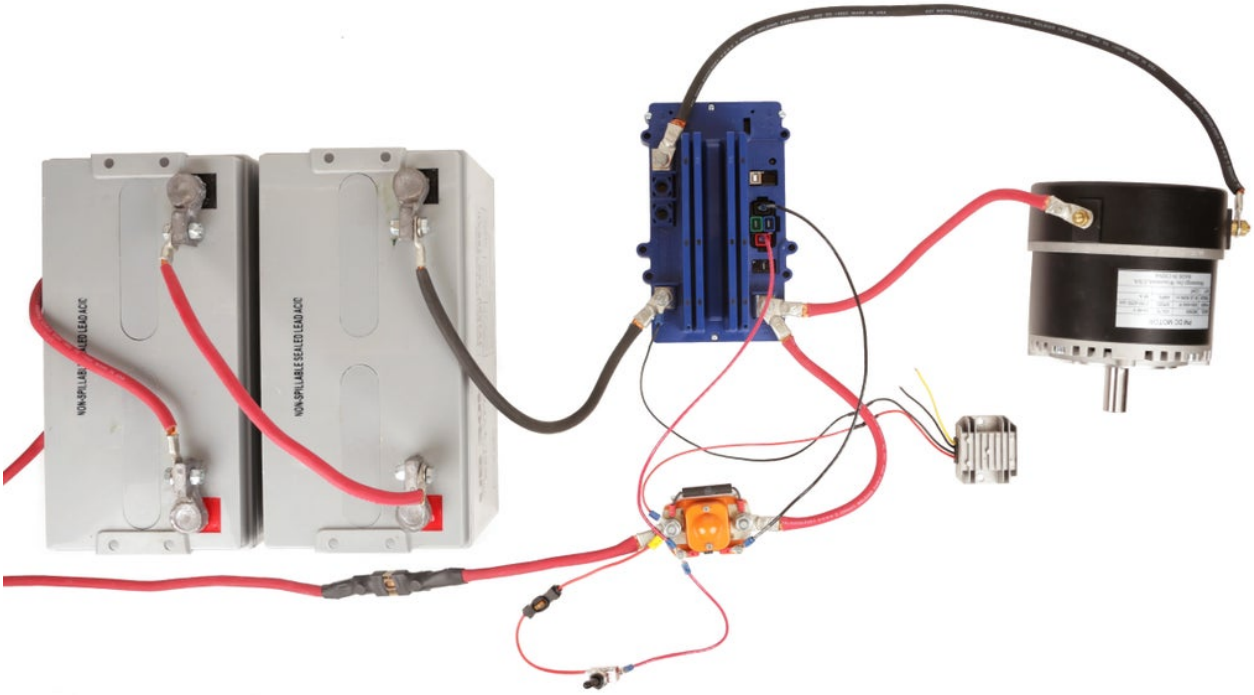


Let's say that instead of a 0-5K throttle, the controller needs to be configured for a 0-5V throttle. This requires a little extra setup, but it's easy enough to handle.

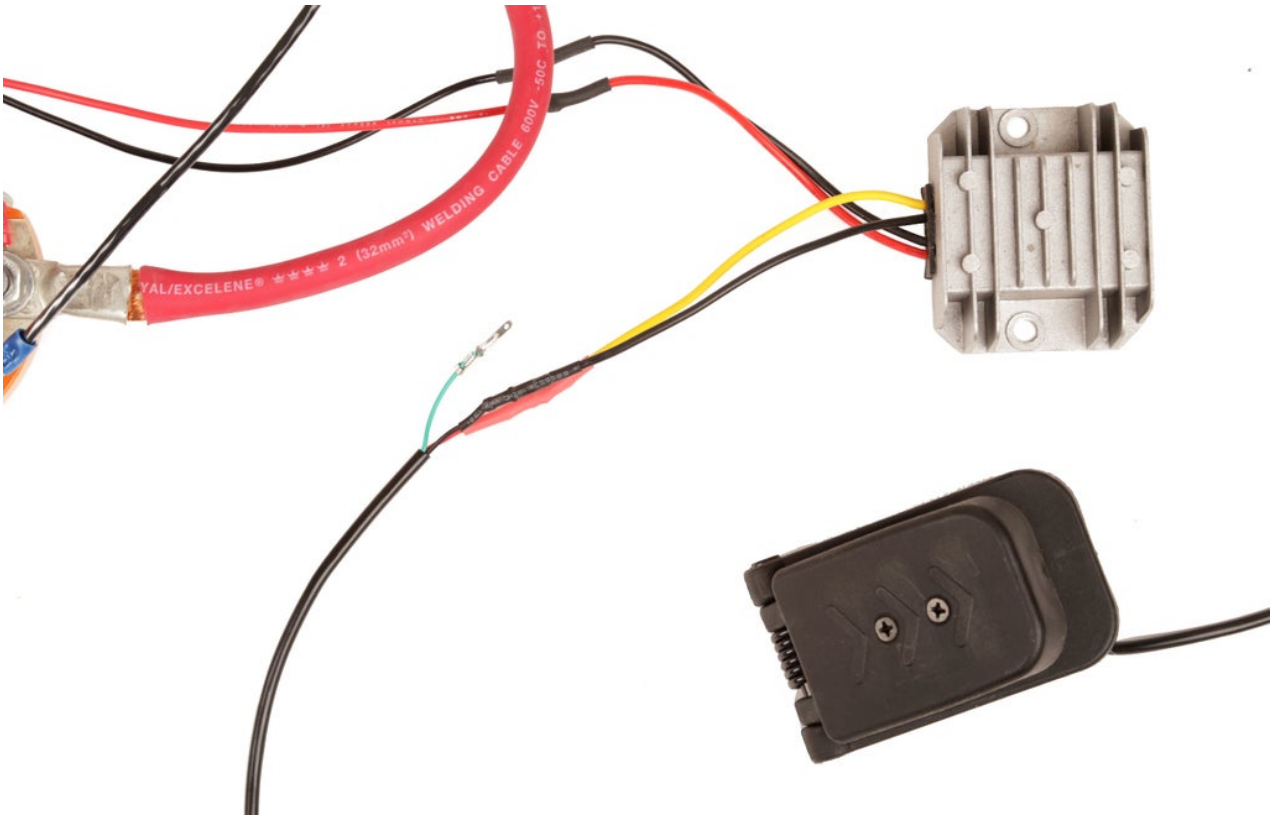
To see a complete list of throttle configurations, check out the official Alltrax SR throttle documentation.



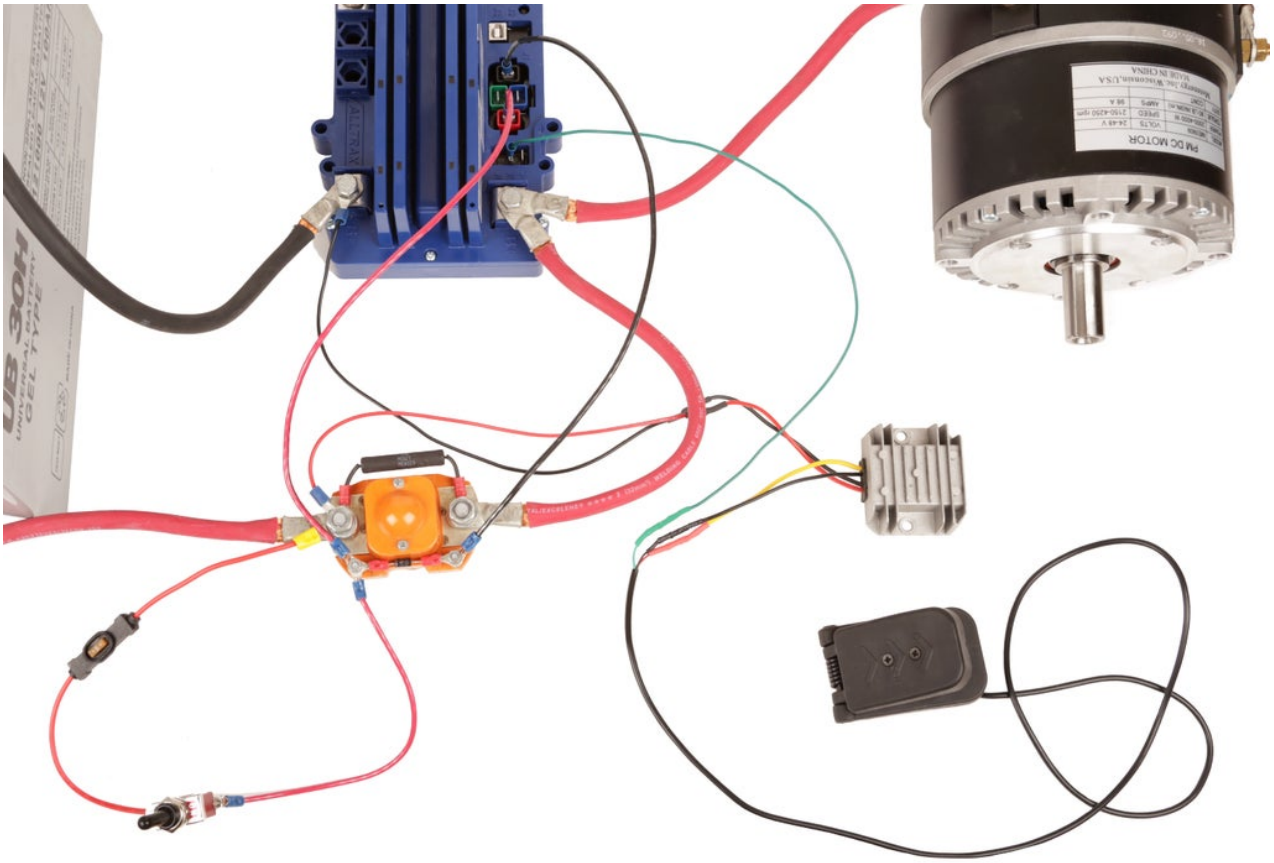
Many 5V throttles require a 5V input. As you probably have noticed, this circuit is operating primarily at 24V. In order to give the throttle a 5V input, it requires the use of a 24V to 5V voltage converter.



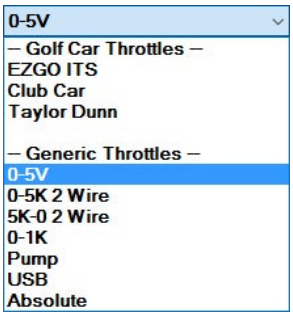
The red wire should be attached to the high current input terminal on the main contactor. The black ground wire (next to the yellow wire) should be connected to the B- (battery ground) connection on the motor controller. It is recommended to make both of these connections using ring terminals.



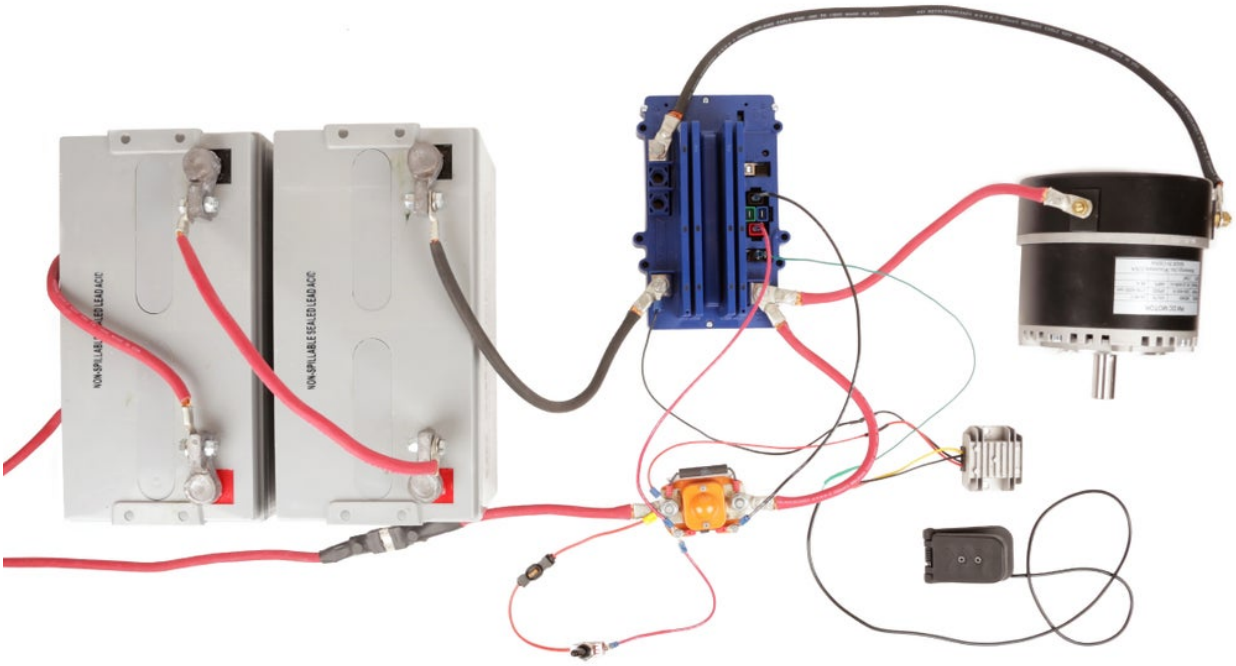
The yellow wire from the voltage converter should go the red input wire on the throttle. The remaining black wire of the voltage converter should go to the black wire from the throttle.



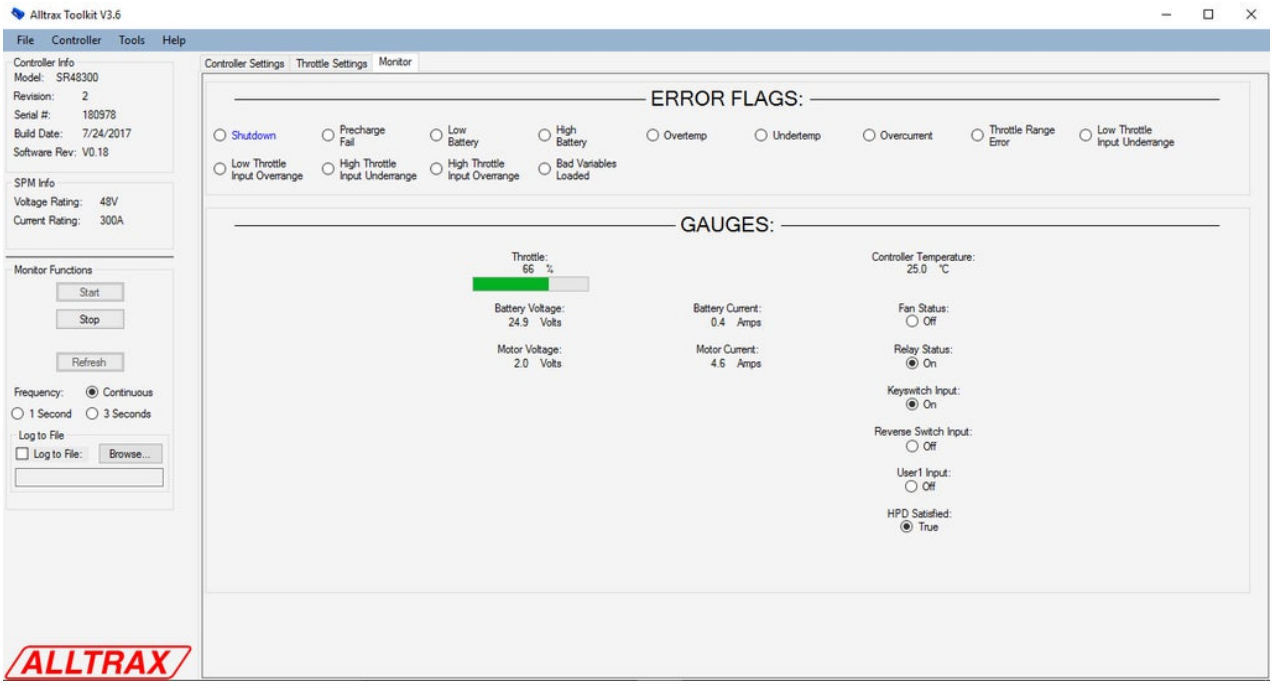
The throttle's green wire should be connected to the J5 terminal on the motor controller.



However, before this throttle will work with the motor controller, the throttle-type needs to be reconfigured in the Alltrax Toolkit software. Go to the "Throttle Settings" tab and change the type of throttle shown in the drop-down menu to 0-5V. Once done, be sure to save the changes by pressing the "Set" button on the left-hand side.



The 0-5V throttle should now be wired up and ready to go.

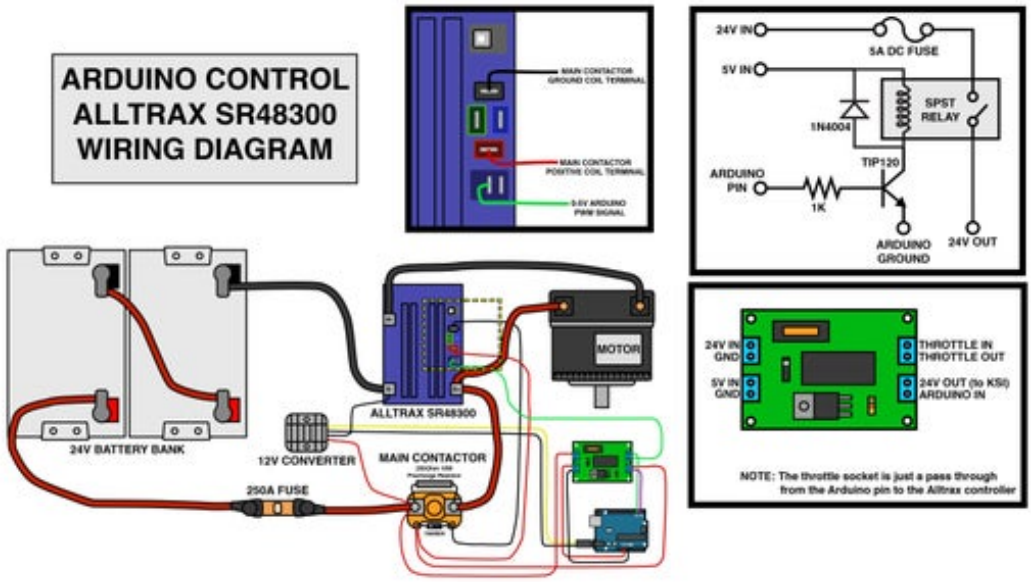


Once your circuit is up and running, you can plug the motor controller back into your computer and use the Alltrax Toolkit "Monitor" tab to get real-time feedback about motor usage, and the status of the controller.

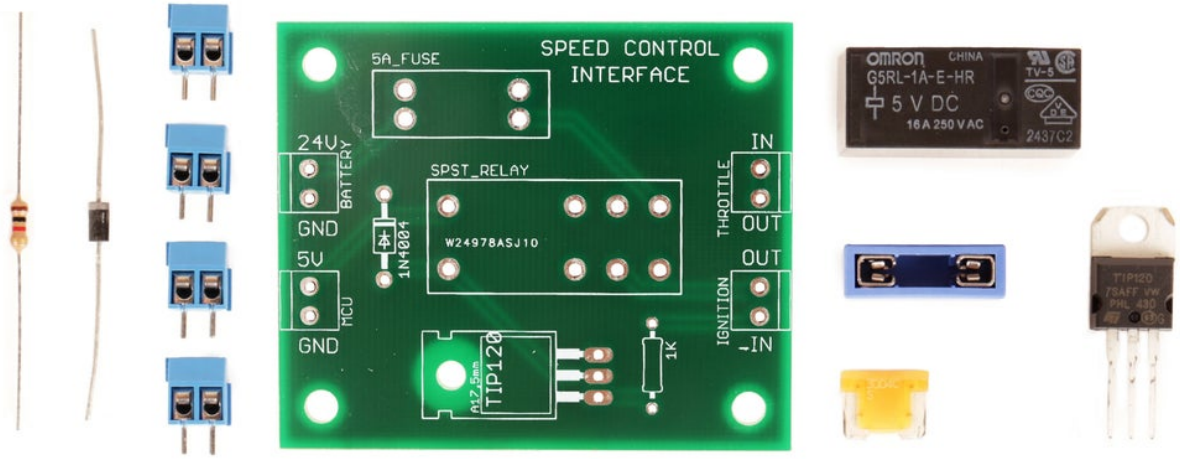
This allows you to actively monitor such useful metrics as the amount of current that is being used by the motor, the status of the battery, the position of the throttle (in percentage), the temperature of the controller, and the state of the solenoid (to name a few).

https://docs.google.com/spreadsheets/d/e/2PACX-1vSPBCm35WGwY0Z9fXXmGkpGZk8qMl1gb9HoeakSNMnuskHE36Y7Yb9yOEnNeKyrv4uktKb8t9gPAZg5/pubhtml?gid=357249423&single=true&widget=true&headers=false

All of this data can even be recorded into timed log files for later review. This is extremely helpful for debugging unexpected behavior and catching errors flags. For help interpreting the error flags, check out the Alltrax Toolkit Manual.



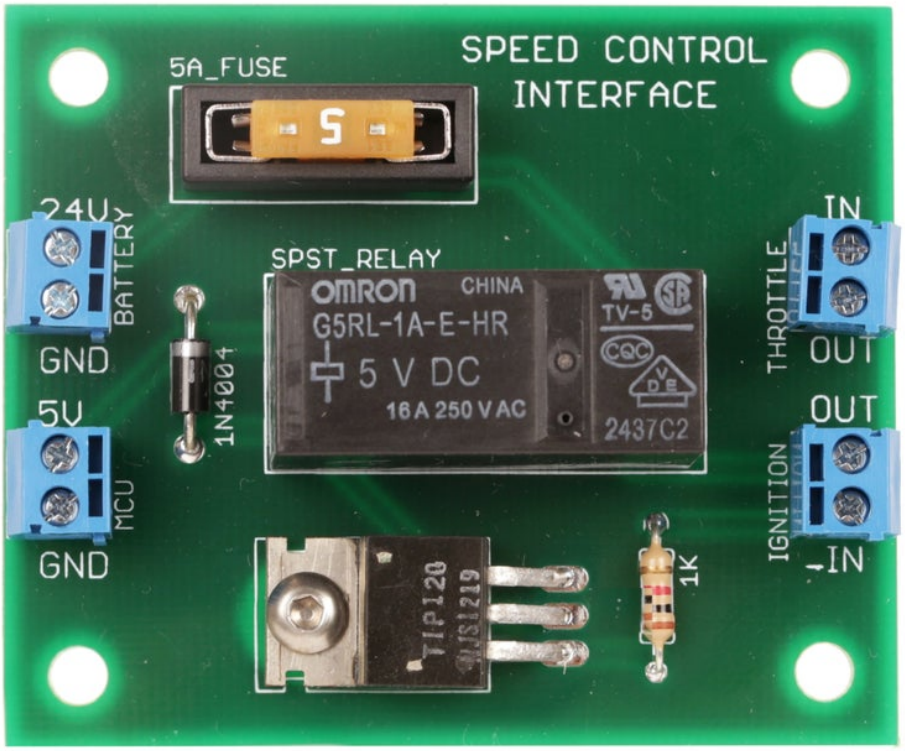
To interface the Alltrax motor controller with an Arduino requires an additional circuit board. Essentially this board replaces the on/off KSI switch with a 5V relay. However, the relay coil requires a little bit more current than the Arduino pins can provide, so it is being triggered with a TIP120 transistor. Additionally, these new components require is a snubber diode on the relay coil, and a 1K resistor in series with the base of the TIP120 to protect the microcontroller pin that is toggling the transistor.



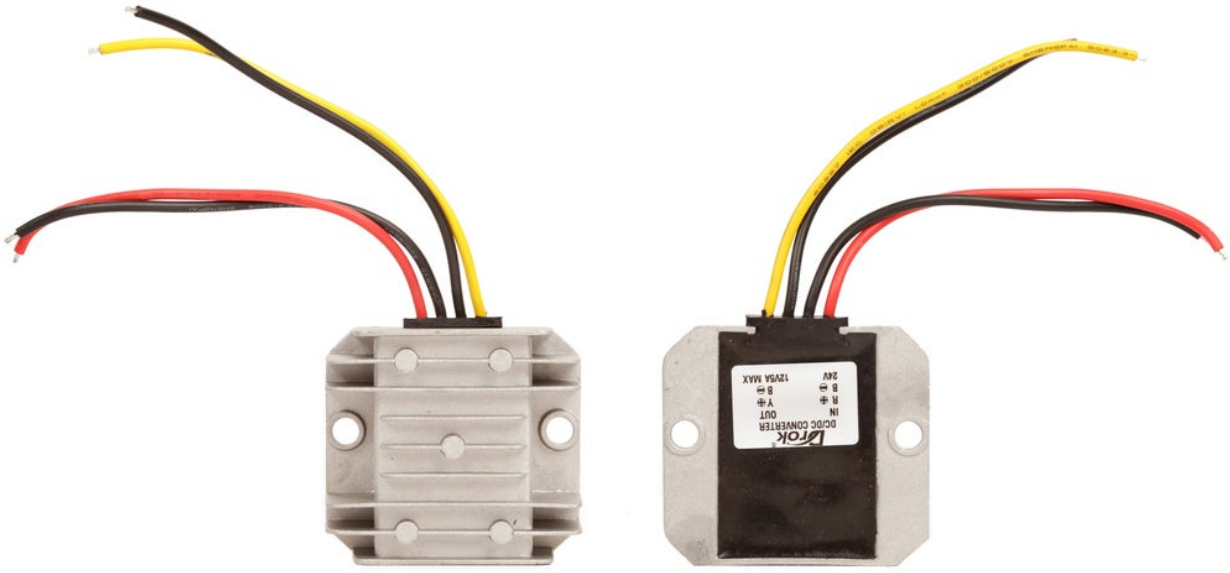
I went ahead and designed my own circuit board to keep all of the wiring nice and tidy. Designing and manufacturing a PCB is not necessary. You can build the circuit on a proto board if pressed for time. However, should you be inspired to design your own PCB, you can use this custom "Large Motor" Eagle library that includes a footprint for a 5A fuse, and one suitable for SPST and SPDT relays. If you have never designed a PCB before, check out my Circuit Board Design Class for instructions (or a refresher).

For this interface circuit interface you will need the following materials:

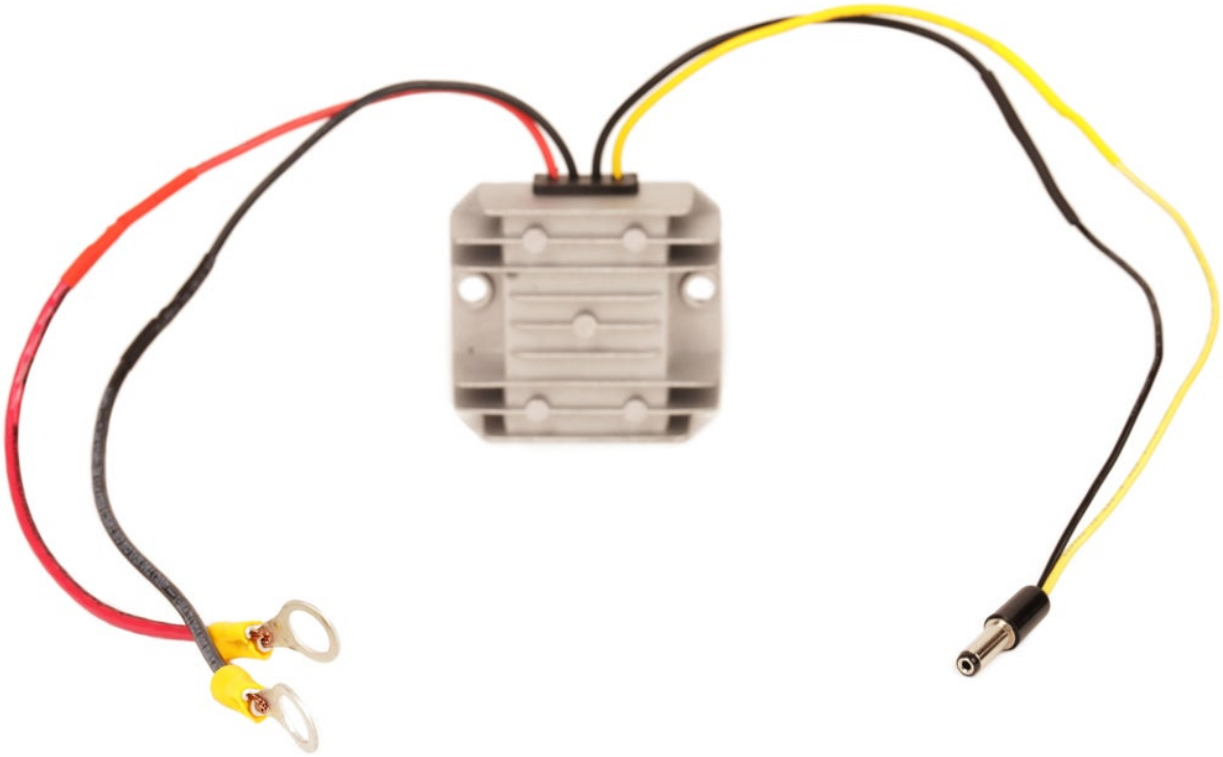
(x1) SPST Relay
(x1) TIP120 transistor
(x1) 1N4004 diode
(x1) 1K resistors
(x1) 5A / 58VDC fuse
(x1) Fuse holder
(x4) 2-pin 3.5mm terminal block



Once that you have all of the materials in hand, it is simply a matter of assembling the interface board as specified in the schematic.

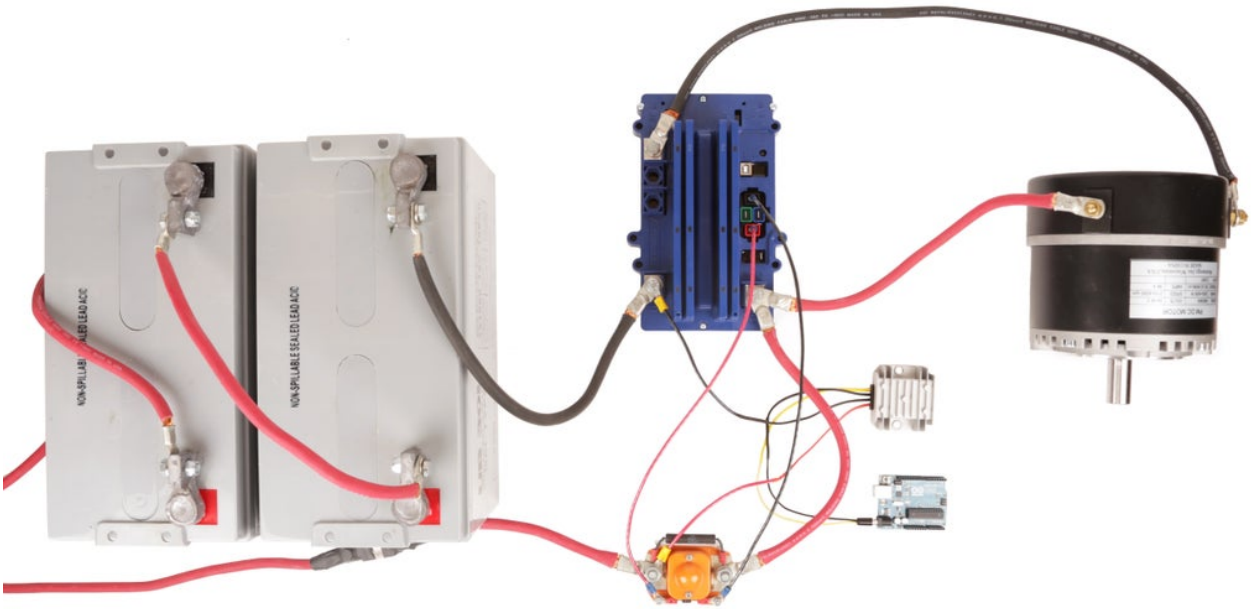


A 24V to 12V converter and barrel plug is also necessary because the Arduino requires a voltage of greater than 7V to be applied at its power jack.



A barrel plug needs to be connected in order to power the Arduino through its input jack. When attaching the plug, the barrel should be connected to the 12V side of the converter with the tip of the plug being positive and the outer jacket being ground.

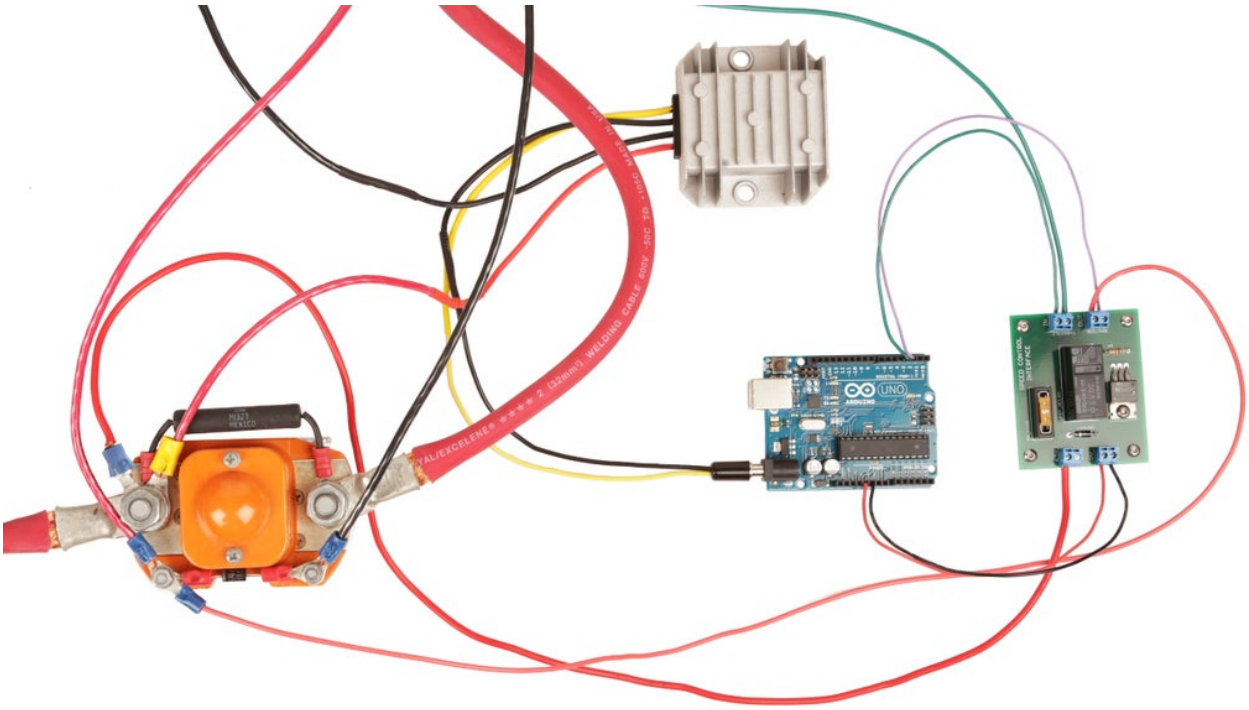
It also helps to attach ring terminals to the 24V side of the converter.



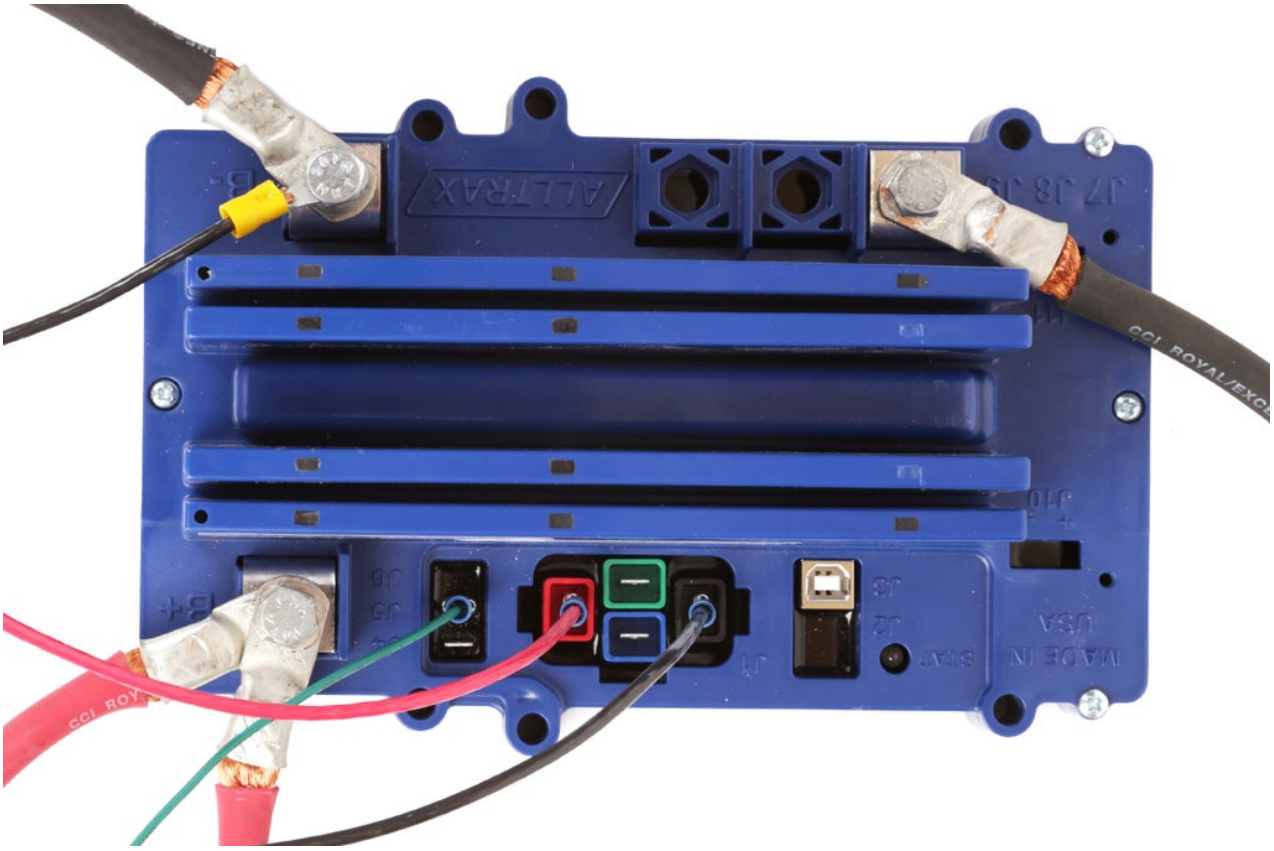
The voltage converter should be connected to the "live" terminal on the mains contactor. This is the terminal that the fuse cable is connected to. The reason for attaching it here is that it is connected after the fuse (protecting it from surges), but is always powered up.

In other words, if you place the voltage converter on the other side of the main contactor that is not connected, the contactor would need to be activated in order for the voltage converter to get powered up. This would be a problem because the voltage converter needs to be on in order to

power the Arduino which is used for activating the contactor.



Now is time to make all of the connections from the controller circuit to the Arduino and Alltrax motor controller. In this example we are using Digital Pin 3 as the throttle and Digital Pin 2 as the ignition switch which activates the relay (which, in turn, activates the solenoid).



The connections to the Alltrax controller are the same as you would make for a KSI switch and 0-5V throttle.

To see it in action, upload the following code:

```
void setup() {

    // Establish pin 2 as an output for the KSI solenoid
    pinMode(2, OUTPUT);

    // Engage the KSI solenoid and wait a second
    digitalWrite(2, HIGH);
    delay(1000);

    // This delay gives you 5 seconds to prepare yourself for the motor starting
    delay(5000);

}

void loop() {

  ////////////////////////////
  // LET'S ENGAGE THE MOTOR! //
  ////////////////////////////

    // Turn on the motor on at approximately 1/2 speed for two seconds
    analogWrite(3, 120);
    delay(2000);

    // Turn off the motor for 5 seconds
    analogWrite(3, 0);
    delay(5000);

}
```

This code is about as simple as Arduino code gets. Digital Pin 2 is set to be either high (relay/contactor active) or low (relay/contactor off). This is taking the place of the KSI switch.

When pin 2 is high, Digital Pin 3 can be used to make a PWM signal which controls the throttle voltage (0 being off, and 255 being full throttle).

It is as simple as that. Even if you want to reverse the motor, as you will see in the next lesson, all of this gets only a little bit more complicated.