

0 items  
login

SparkFun Electronics



## [Home](#) | [Tutorials](#) | **Ardumoto Quickstart Guide**

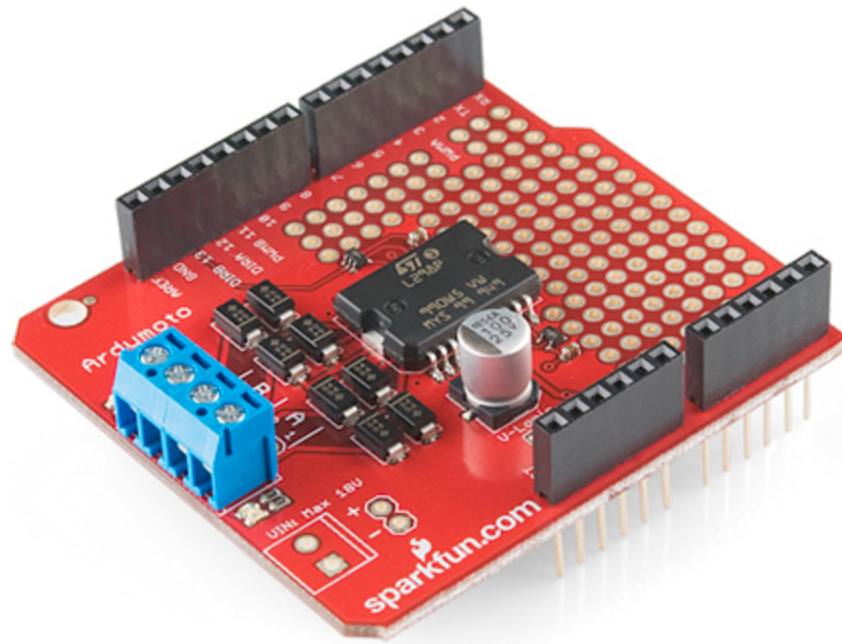
[Ardumoto Quickstart Guide](#)

by [Pete-O](#) | June 29, 2010 | [10 comments](#) Skill Level: ☆ Beginner

---

### **Ardumoto Motor Driver Shield**

Thanks for checking out the Ardumoto motor driver shield for Arduino! Combined with An Arduino controller board, the Ardumoto makes a fantastic controller platform for small scale robotics or RC. This page will help to get you going quickly.

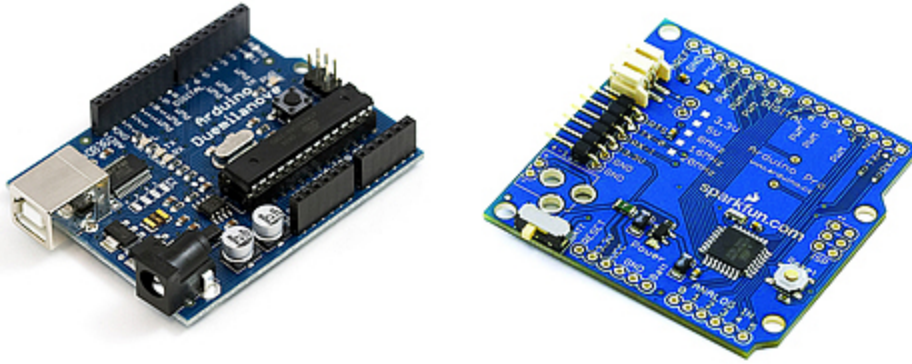


## What is it?

The Ardumoto is a board that has an L298 H-bridge on it, used primarily for driving small DC motors. Why do you need a driver board? Well, motors tend to draw a lot of current, and trying to drive a motor straight from your Arduino output pins will make your Arduino quite cross with you. The Ardumoto lets you control a whole bunch of current (good for motors) with an itty-bitty signal (good for Arduinos).

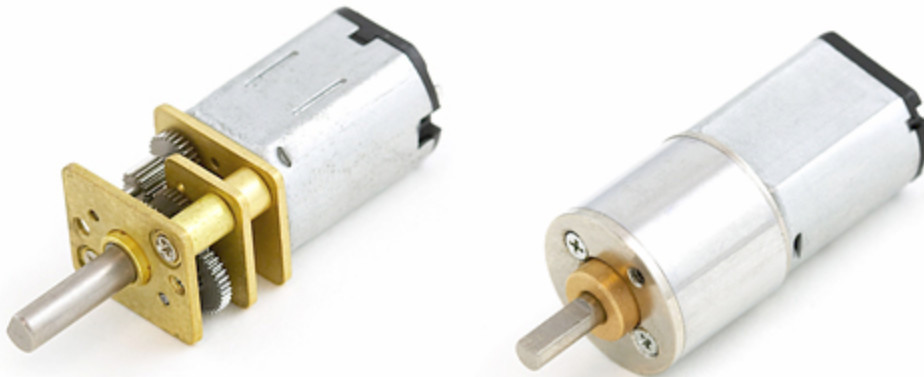
## What else do I need?

There's a good chance you've already got most of the following covered, but for the sake of being complete (or you're a noob, which is totally cool) we'll list this all out. Besides the Ardumoto Shield, you'll need a 5V Arduino board:



- Arduino Duemilanove (e.g. [DEV-00666](#))
- Arduino Pro 328 - 5V/16MHz (e.g. [DEV-09219](#))

For the purpose of this tutorial, we'll assume you've got an Arduino Duemilanove to work with. You'll also need some motors to drive. These should probably be motors that draw no more than 1-2 amps when stalled (when it's under power, but you're holding it still). If you're still shopping for those, we have a few smaller motors to choose from:



- Micro Metal Gearmotor 100:1 (e.g. [ROB-08910](#))
- Micro Metal Gearmotor 30:1 (e.g. [ROB-08911](#))
- Mini Metal Gearmotor 100:1 (e.g. [ROB-08912](#))
- Mini Metal Gearmotor 24:1 (e.g. [ROB-08913](#))

For my purposes, I'm going to use a couple of the 30:1 micros, the second one down in the list above. We'll also need a soldering iron, some solder, some wire to hook up our motors and wire strippers to strip the wire:



- Soldering iron (e.g. [TOL-00085](#))
- Solder (e.g. [TOL-09161](#))
- Black wire (e.g. [PRT-08867](#))
- Red wire (e.g. [PRT-08865](#))
- Wire strippers (e.g. [TOL-08696](#))

You'll also need a computer to do the programming... no, I'm not going to list computer sources. We'll assume that if you're here you've most likely gone through the basic Arduino tutorials and have programmed one once or twice before this. If you haven't, check out our [beginning Arduino tutorial](#).

## Sweet. Now how do I make it work?

If you haven't already done so, now would be a good time to [assemble the Ardumoto](#). Once you've done that...

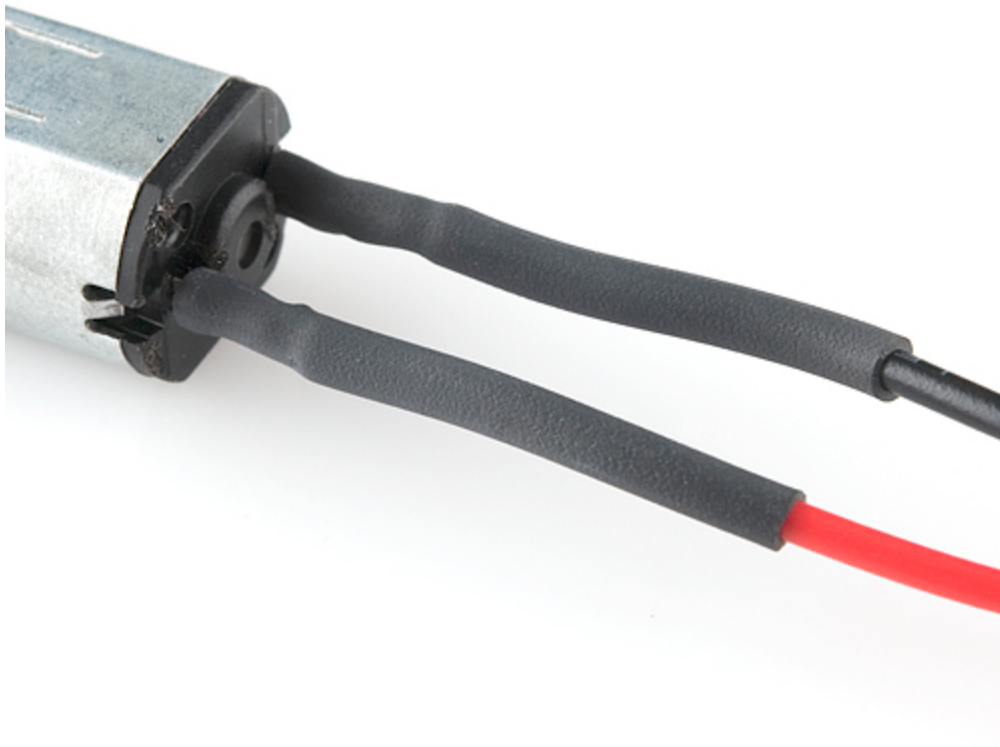
First, estimate how long you think you need the motor leads to be. We'll assume you're going to be snaking these leads through a chassis of some kind, so estimate the distance and multiply by about 1.3, then cut a length of red and black wire. You've probably got to do this for each motor separately as the distance from the driver to one motor will likely be different from the distance to the other. The reason for the 1.3 multiplier is 2-fold; first, if you cut too short you're going to kick yourself, and second, it allows you to twist the wires up real nice and professional, right? Makes it easier for snaking, too.

These little motors I've used for an example can be a pain to solder. Not because of solder/iron/heat issues, but because these little suckers slide all over the place as soon as they see you coming with an iron in your hand. Our Third Hand vise ([TOL-09317](#)) could be useful here, but I normally do something low-tech like hold it down with the spool of solder. The Mini-motors listed above actually have round cases, so they'll be even more wiley.

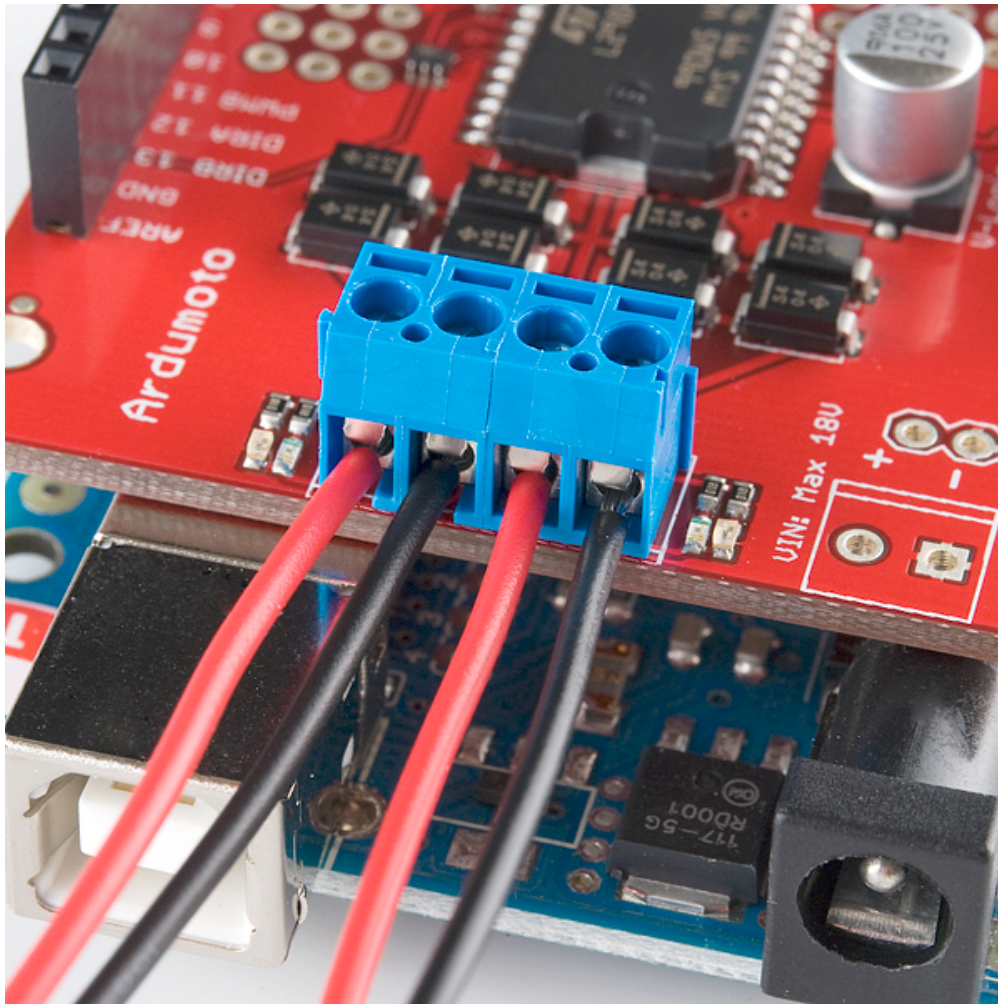


If you want to look really pro, use some heat shrink ([PRT-08831](#)) on your soldered connections.



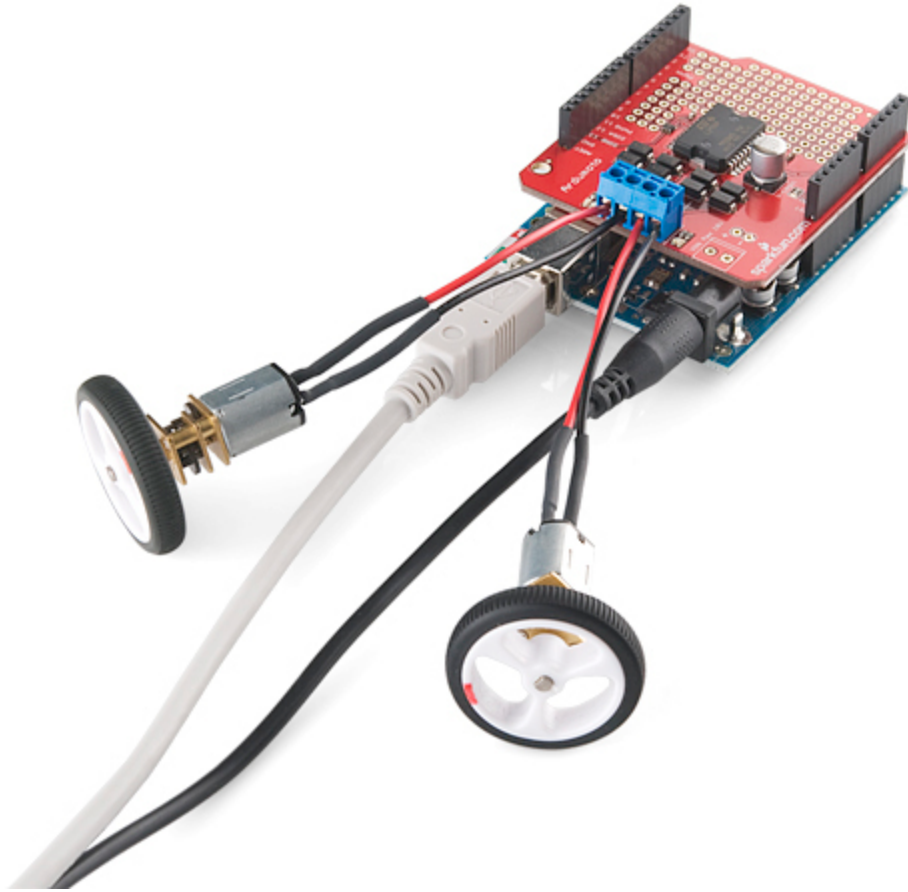


When you've got both motors set up with leads, you can mount them as you like, and route the wiring how you like to the place where you're Ardumoto/Arduino is going to live (those steps are going to be specific to the user's application, so we'll gloss over that a little bit and focus on the meat of what we're here to do). When you've got that worked out, you can attach the motors to the Ardumoto/Arduino pair at the screw terminals with a small Phillips screwdriver. Strip off about 1/8"-3/16" of the wires, stick one into each screw terminal and tighten them down (one of the motors to lines 1-2, and the other motor to lines 3-4).



Getting close now. The Ardumoto takes power from its host Arduino board from Vin to drive the motors and +5V (regulated) to drive the logic on the L298 driver chip. This means two things to you: 1) you can't drive the Ardumoto using the USB connection from an Arduino Duemilanove (only the logic will power up), and 2) you might want to use a power source that's capable of sourcing a couple of amps (depending on your motor choice). We've got a few really good LiPo battery packs that would work really awesome, like [PRT-09704](#) (7.4V, 1000mAh) and [PRT-09703](#) (7.4V, 2200mAh), but for our demonstration we'll just use a 9V, 650mA wall-wart power supply ([TOL-00298](#)) that will just plug into the Duemilanove's barrel jack (we have small motors). Lastly, plug in your USB to your computer and fire up your Arduino development suite.





Here's an example sketch for the Ardumoto: [Link](#). Save the content of this zip file to your Arduino projects folder, open the sketch and load it to your Arduino. When finished, your two motors will run sequentially in one direction at 2 speeds, then the opposite direction at two speeds (that may sound confusing, but it should become obvious when you see it do its' thing).

## Explanation of the Sketch (Sketchplanation?)

The example sketch has been designed to be as simple as possible. Painfully so, in fact. Here - I'll prove it. First, we identify what pins will get what signals.

```
//int pwm_a = 10; //PWM control for motor outputs 1 and 2 is on digital pin 10
int pwm_a = 3;   //PWM control for motor outputs 1 and 2 is on digital pin 3
int pwm_b = 11;  //PWM control for motor outputs 3 and 4 is on digital pin 11
int dir_a = 12;  //dir control for motor outputs 1 and 2 is on digital pin 12
int dir_b = 13;  //dir control for motor outputs 3 and 4 is on digital pin 13
```

```
void setup()
```

```
{
  pinMode(pwm_a, OUTPUT); //Set control pins to be outputs
  pinMode(pwm_b, OUTPUT);
  pinMode(dir_a, OUTPUT);
  pinMode(dir_b, OUTPUT);

  analogWrite(pwm_a, 100);
  //set both motors to run at (100/255 = 39)% duty cycle (slow)
  analogWrite(pwm_b, 100);
}

void loop()
{
  digitalWrite(dir_a, LOW); //Set motor direction, 1 low, 2 high
  digitalWrite(dir_b, LOW); //Set motor direction, 3 high, 4 low

  delay(1000);

  analogWrite(pwm_a, 255);
  //set both motors to run at 100% duty cycle (fast)
  analogWrite(pwm_b, 255);

  delay(1000);

  digitalWrite(dir_a, HIGH); //Reverse motor direction, 1 high, 2 low
  digitalWrite(dir_b, HIGH); //Reverse motor direction, 3 low, 4 high

  delay(1000);

  analogWrite(pwm_a, 100);
  //set both motors to run at (100/255 = 39)% duty cycle
  analogWrite(pwm_b, 100);

  delay(1000);
}
```

The function of each line is likely self explanatory, but the way the jobs get done may not be if you're new to Arduino. The digitalWrite function sets a pin either high or low, and is used here for controlling the direction of the motors (dir\_a corresponds to the motor strapped between outputs 1 and 2, and dir\_b corresponds to the motor strapped between outputs 3 and 4). The delay functions are in milliseconds and are there to provide a little wait-time so the user can more easily see the operation of the Ardumoto. The analogWrite function is a PWM (**Pulse Width Modulation**) function, and it's this value that sets the speed of the motors (pwm\_a goes with dir\_a and outputs 1 and 2, pwm\_b goes with dir\_b and outputs 3 and 4). This function will accept values from 0 (motor's stopped) to 255 (full throttle).

And...that's it! You're off and running (into walls, chairs, cats...) Please drop us

a line and let us know about your projects!

*Have a suggestion for how we can improve this quickstart guide? Concepts not explained clearly? Need more example code? Please let us know. You can leave a comment below or email us [spark@sparkfun.com](mailto:spark@sparkfun.com). Also let us know if this is the most awesome Quickstart guide you have ever encountered and we will stop trying to improve it.*

## ▼ Comments 10 comments

**Login to post comments.**

-  **lyweilian** | about 5 months ago \* 2

Don't know if there is a library for this already out there, but I wrote a library for my project. Hope it helps someone that is stuck.  
Ardumoto.h

```
/*
Ardumoto.h - Library for driving the Ardumotor Driver code.
Created by Will Ly based upon the example on Sparkfun Ardumoto Quick Start Tutorial
Release into the public domain.

*/

#ifndef Ardumoto_h
#define Ardumoto_h
#include "WProgram.h"
class Ardumoto
{
public:
    Ardumoto(int pmwAPin, int pmwBPin, int dirAPin, int dirBPin);
    void stopMotors();
    void setSpeed(int speedMotorA, int speedMotorB);
    void setMotorDirection(bool isForward);
private:
    int _pwm_a;
    int _pwm_b;
    int _dir_a;
    int _dir_b;
};
#endif
```

-  **lyweilian** | about 5 months ago 2

Ardumoto.cpp

```
/*
Ardumoto.cpp - Library for driving the Ardumotor Driver code.
Created by Will Ly based upon the example on Sparkfun Ardumoto Quick Start Tutorial
```


```
Release into the public domain.
*/
#include "WProgram.h"
#include "Ardumoto.h"
Ardumoto::Ardumoto(int pwmAPin, int pwmBPin, int dirAPin, int dirBPin)
{
    _pwm_a = pwmAPin;
    _pwm_b = pwmBPin;
    _dir_a = dirAPin;
    _dir_b = dirBPin;
    pinMode(_pwm_a, OUTPUT);
    pinMode(_pwm_b, OUTPUT);
    pinMode(_dir_a, OUTPUT);
    pinMode(_dir_b, OUTPUT);
}
void Ardumoto::stopMotors()
{
    digitalWrite(_pwm_a, LOW);
    digitalWrite(_pwm_b, LOW);
    setSpeed(0,0);
}
void Ardumoto::setSpeed(int speedMotorA, int speedMotorB)
{
    if(speedMotorA > 0)
    {
        digitalWrite(_dir_a, LOW); //Forward
    }
    else
    {
        digitalWrite(_dir_a, HIGH); //Reverse
    }
    if(speedMotorB > 0)
    {
        digitalWrite(_dir_b, LOW); //Forward
    }
    else
    {
        digitalWrite(_dir_b, HIGH); //Reverse
    }
    analogWrite(_pwm_a, abs(speedMotorA)); //Left Motor
    analogWrite(_pwm_b, abs(speedMotorB)); //Right Motor
}
void Ardumoto::setMotorDirection(bool isForward)
{
    stopMotors();
    if(isForward)
    {
        digitalWrite(_dir_a, LOW); //Set motor direction, 1 low, 2 high
        digitalWrite(_dir_b, LOW); //Set motor direction, 3 high, 4 low
    }
    else
    {
        digitalWrite(_dir_a, HIGH); //Reverse motor direction, 1 high, 2 low
        digitalWrite(_dir_b, HIGH); //Reverse motor direction, 3 low, 4 high
    }
}
```

}

- o  **MobileWill** | about 3 months ago \* 1

I made the following changes for individual motor control since I needed it for my tank robot.

```
void Ardumoto::setMotorDirection(bool isMotorL, bool isMotorR)
{
    stopMotors();
    if(isMotorL)
    {
        digitalWrite(_dir_a, LOW); //Set motor direction, 1 low, 2 high
    }
    else
    {
        digitalWrite(_dir_a, HIGH); //Reverse motor direction, 1 high, 2 low
    }
    if(isMotorR)
    {
        digitalWrite(_dir_b, LOW); //Set motor direction, 3 high, 4 low
    }
    else
    {
        digitalWrite(_dir_b, HIGH); //Reverse motor direction, 3 low, 4 high
    }
}
```

-  **3.1415926535897932384626** | about a year ago 1

The sample code throws the motors from full forwards to full reverse. The motors should have capacitors connected across the terminals, especially when used in PWM mode, else every time the bridge switches off the inductor (aka “electric motor”) dumps its energy across the power rails.

-  **hfkids** | about a year ago 1

I only require a single motor, therefore I do not need the second motor “control”. May I simply ignore the connections to motor 2 – but not cut the wire traces – and use those pins for other project needs (like communicating with the color LCD shield)?

- o  **BERTO** | about 10 months ago 1

Sure you could use this ports for something else. The motor controller output has a current amplifier and will drive more current



than the arduino pin can if you dont limit it. It can also be commanded to reverse polarity (i.e. the +5V can also be commanded to -5V). IF you are not sure what that means. The LED tutorial should explain why these can sizzle your bulbs if you don't pay attention.

-  **ivan** | **about 9 months ago 1**

Newbie here. I ran the code above using the Ardumoto shield on a Uno. As long as I use the USB Serial connection, it runs great. When I remove the USB Serial Connection and am powering it from a 4 AA batteries, it will run forward, but will not reverse directions.

Can you help me?

Thanks,

Ivan

-  **Member #248567** | **about 5 months ago \* 1**

Hi all,

Could I use a FPGA to control the Ardumoto Motor Driver Shield or must I have an Arduino board to be able to control mine Ardumoto Motor Driver Shield?

Thanks

-  **Paul\_d** | **about 4 months ago 1**

Newbie q: Is this essentially a 6v motor running off a 9v PS? If so, can I run a little 3V motor off the same arrangement?

-  **tonio4001** | **last week 1**

Hello, with the shield you don't need resistor between arduino and entries of control component? Is it normal ? Thank you for your help.