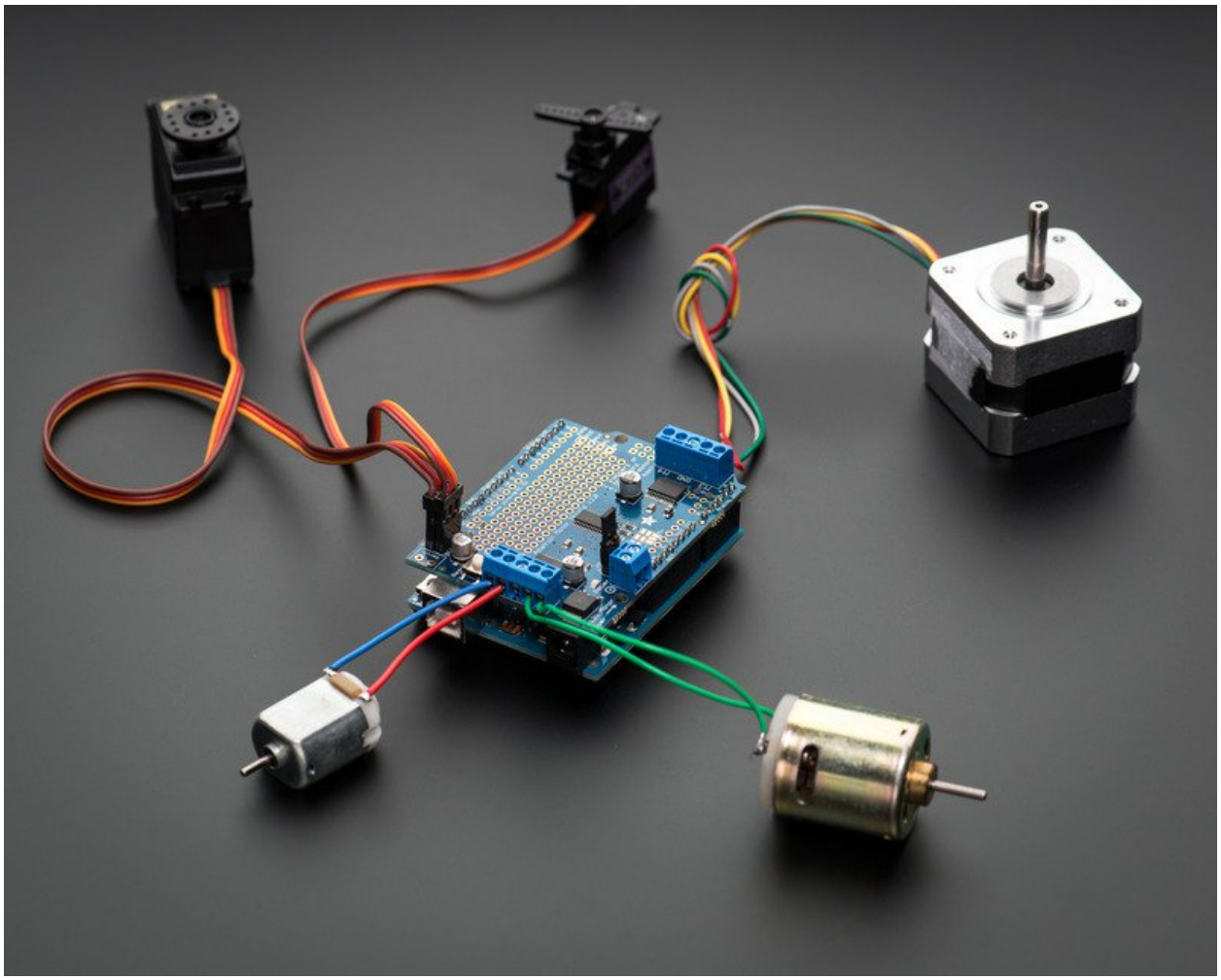
ADAFRUIT MOTOR SHEILD V2

FOR ARDUINO



The original motor shield by Adafruit has been upgraded to make the best, easiest way to drive DC and Stepper motors. The ability to drive up to 4 DC motors or 2 stepper motors has been kept, with many added improvements:

Instead of a L293D Darlington driver, we now have the TB6612 MOSFET drivers with 1.2A per channel current capability (you can draw up to 3A peak for approx 20ms at a time). It also has much lower voltage drops across the motor so you get more torque out of your batteries, and there are built-in flyback diodes as well.

Instead of using a latch and the Arduino's PWM pins, it has a **fully-dedicated PWM driver chip** onboard. This chip handles all the motor and speed controls over I2C. Only two GPIO pins (SDA & SCL) plus 5v and GND are required to drive the multiple motors, and since it's I2C you can also connect

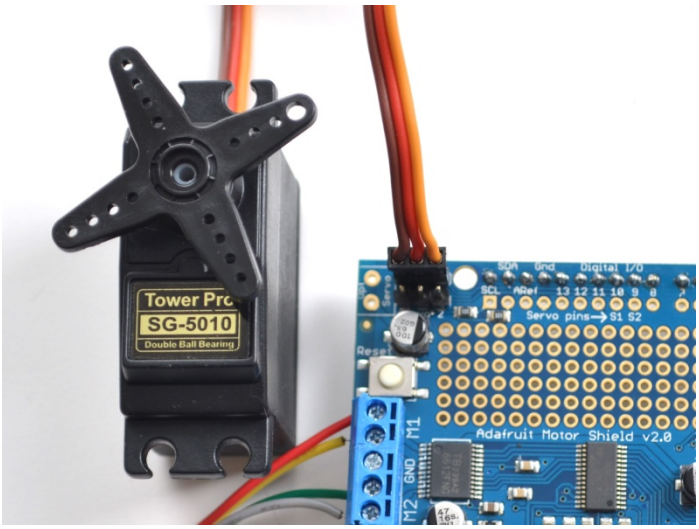
any other I2C devices or shields to the same pins. This also makes it drop-in compatible with any Arduino, such as the Uno, Leonardo, Due and Mega R3.

Completely stackable design: 5 address-select pins means up to 32 stackable shields: that's 64 steppers or 128 DC motors!

Specifications:

- **2 connections for 5V 'hobby' servos** connected to the Arduino's high-resolution dedicated timer.
- 4 H-Bridges: TB6612 chipset provides **1.2A per bridge** (3A for brief 20ms peaks) with thermal shutdown protection, internal kickback protection diodes. Can run motors on 4.5VDC to 13.5VDC.
- **Up to 4 bi-directional DC** motors with individual 8-bit speed selection (so, about 0.5% resolution)
- **Up to 2 stepper motors** (unipolar or bipolar) with single coil, double coil, interleaved or micro-stepping.
- Motors automatically disabled on power-up
- Big terminal block connectors to easily hook up wires (18-26AWG) and power
- Arduino reset button brought up top
- Polarity protected 2-pin terminal block and jumper to connect external power, for separate logic/motor supplies
- Tested compatible with Arduino UNO, Leonardo, ADK/Mega R3, Diecimila & Duemilanove. Works with Due with 3.3v logic jumper. Works with Mega/ADK R2 and earlier with 2 wire jumpers.
- **5v** or **3.3v** compatible logic levels - jumper configurable.

Using RC Servos

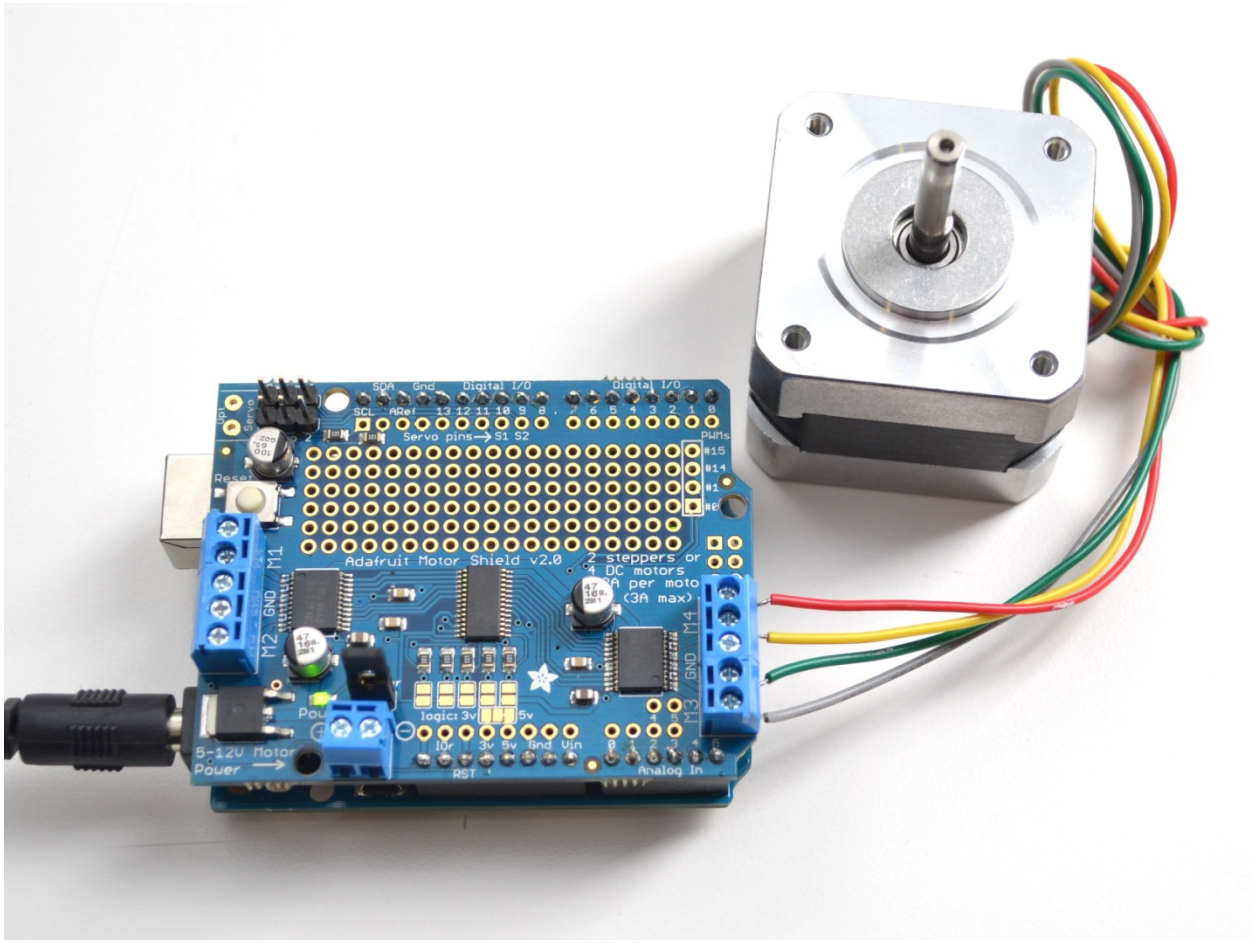


Hobby servos are the easiest way to get going with motor control. They have a 3-pin 0.1" female header connection with +5V, ground and signal inputs. The motor shield simply brings out the PWM output lines from Arduino pins 9 and 10 to two 3-pin headers so that it's easy to plug in and go. They can take a lot of power so a 9V battery won't last more than a few minutes!
The nice thing about using the onboard PWM is that it's very precise and goes about its business in the background. You can use the built in **Servo** library.

Powering Servos

Power for the Servos comes from the Arduino's on-board 5V regulator, powered directly from the USB or DC power jack on the Arduino. If you need an external supply, cut the 5v trace on the bottom of the board and connect a 5V or 6V DC supply directly to the **Opt Servo** power input. Using an external supply is for *advanced users* as you can accidentally destroy the servos by connecting a power supply incorrectly!

Using Stepper Motors



For unipolar motors: to connect up the stepper, first figure out which pins connected to which coil, and which pins are the center taps. If it's a 5-wire motor then there will be 1 that is the center tap for both coils.
The center taps should both be connected together to the GND terminal on the motor shield output block.
Then coil 1 should connect to one motor port (say M1 or M3) and coil 2 should connect to the other motor port (M2 or M4).
For bipolar motors: it's just like unipolar motors except there's no 5th wire to connect to ground. The code is exactly the same.
Running a stepper is a little more intricate than running a DC motor but it's still very easy.

Include the required libraries

Make sure you **#include** the required libraries

```
#include <Wire.h>

#include <Adafruit_MotorShield.h>

#include "utility/Adafruit_PWMServoDriver.h"
```

Create the Adafruit_MotorShield object

```
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
```

Create the stepper motor object

Request the **Stepper** motor from the **Adafruit_MotorShield**:

```
Adafruit_StepperMotor *myMotor = AFMS.getStepper(200, 2);
```

...with getStepper(steps, stepper#).

Steps indicate how many steps per revolution the motor has. A 7.5 degree/step motor has 360/7.5 = 48 steps.

Stepper# is which port it is connected to if you're using M1 and M2, its port **1**. If you're using M3 and M4 indicate port **2.**

Set default speed

Set the speed of the motor using **setSpeed(rpm)** where rpm is how many revolutions per minute you want the stepper to turn.

Run the motor

Then every time you want the motor to move, call the step(#steps, direction, steptype) procedure. **#steps** is how many steps you'd like it to take. direction is either **FORWARD** or **BACKWARD** and the step type is **SINGLE**, **DOUBLE**, **INTERLEAVE** or **MICROSTEP**.

- "Single" means single-coil activation
- "Double" means 2 coils are activated at once (for higher torque)
- "Interleave" means that it alternates between single and double to get twice the resolution (but of course its half the speed).
- "Microstepping" is a method where the coils are PWM'd to create smooth motion between steps.

You can use whichever stepping method you want, changing it "on the fly" to as you may want minimum power, more torque, or more precision.

By default, the motor will 'hold' the position after its done stepping. If you want to release all the coils, so that it can spin freely, call **release()**

The stepping commands are 'blocking' and will return once the steps have finished.