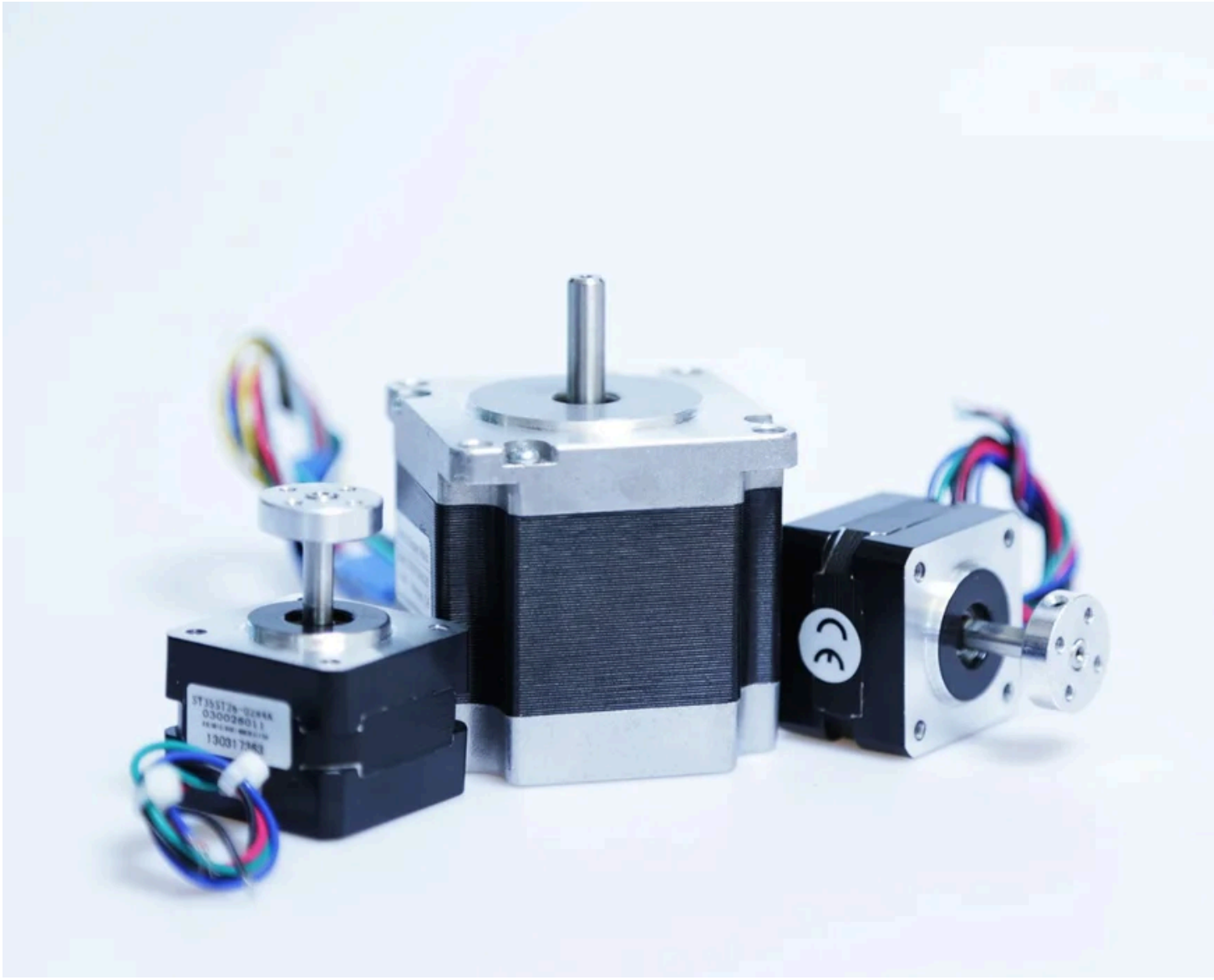
Intro to Stepper Motors

By Aleator777 in CircuitsElectronics

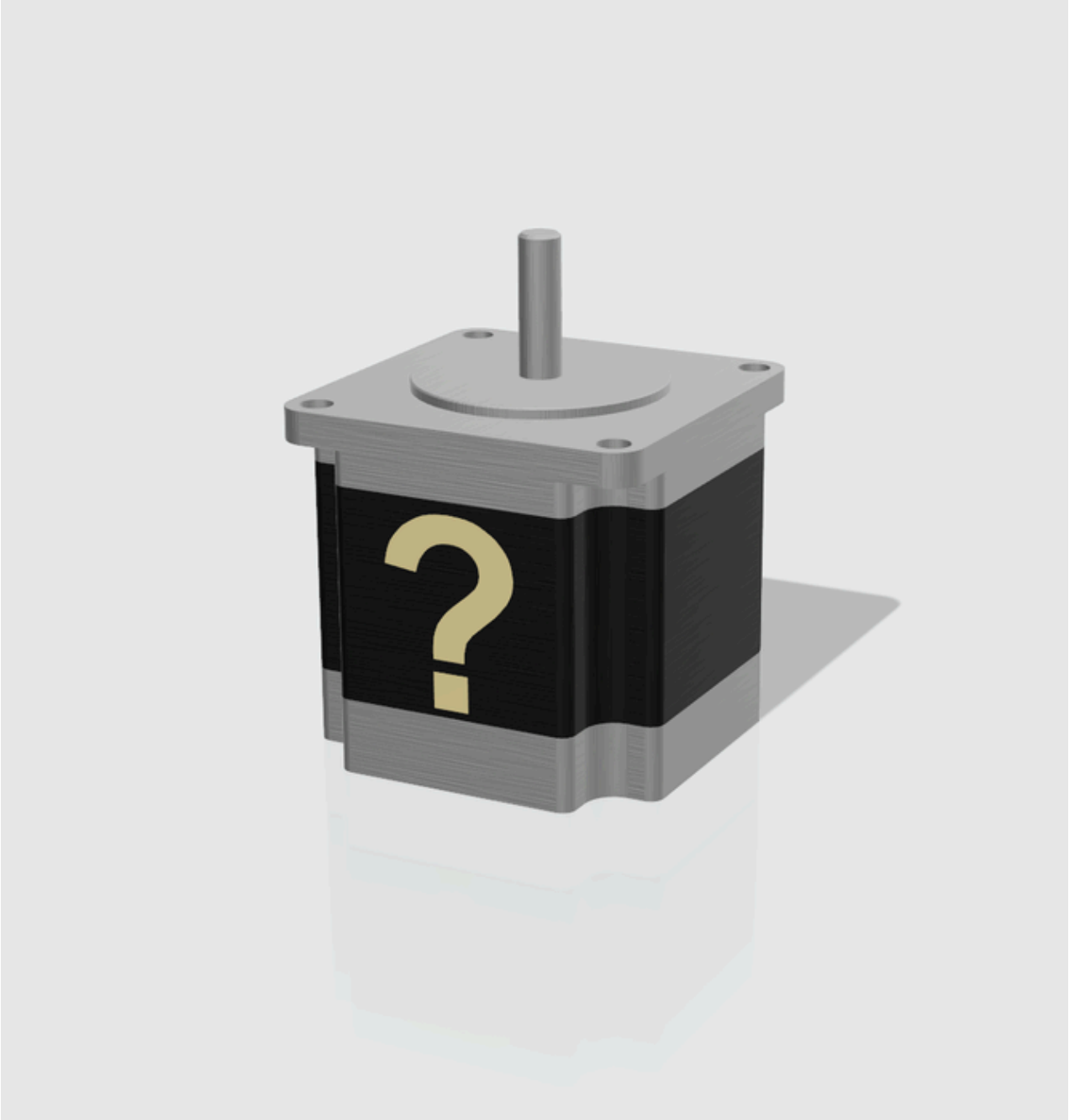
Introduction: Intro to Stepper Motors



If you've ever had the pleasure of dismantling an older printer to salvage electronic parts (I highly recommend this weekend activity if you haven't before!) you may have come across a slew of cylindrical mystery motors with 4 or more wires jutting out the side. Perhaps you've heard the characteristic buzz and whine of a desktop 3D printer or a glitchy electro-mechanical symphony of disk drives playing classic songs? If so, then you've encountered a stepper motor!

Stepper motors make the electro-mechanical world go round (and with higher torque!), but unlike their regular DC motor counterparts, controlling a stepper takes a bit more than a current across a mere two wires. In this Instructable, Ill break down some of the theory behind stepper motor design and operation. Once we've covered the basics, I'll show you how to build a simple circuits for controlling stepper motors and then we'll move on to using dedicated driver chips. Once we've unlocked the mysteries of these magical motors with the Intel Edison, I'll discuss applications for putting them to use!

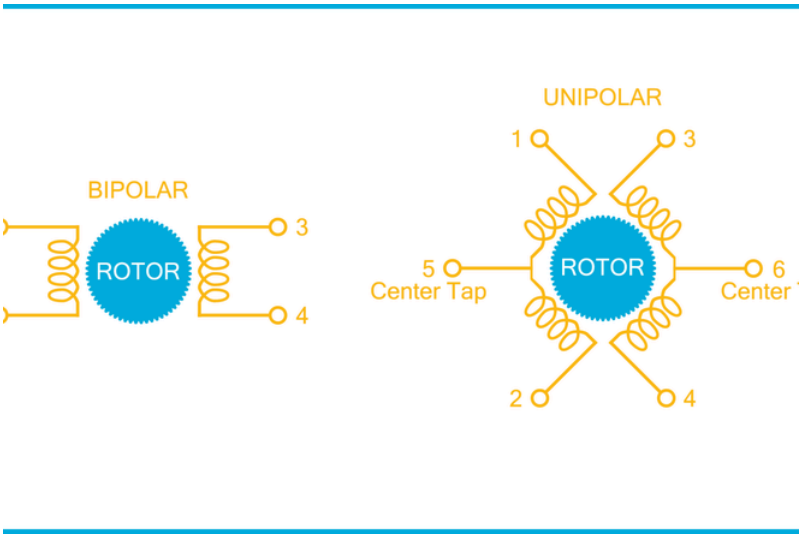
Step 1: What Makes a Motor a Stepper?

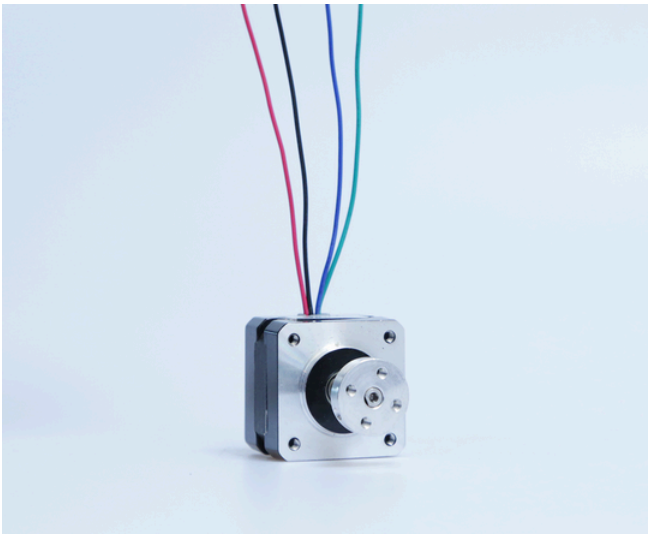


Who could need more than two wires and an H-bridge? Why? Well, unlike common brushedDC motors built for maximum RPM (or kV for the RC folk), stepper motors are brushless motors designed for high torque (subsequently lower speed) and higher precision rotational movement. Whereas a typical DC motor is great for spinning a propeller at high speed for maximum thrust, a stepper is better suited to rolling a sheet of paper in sync with an inkjet mechanism inside of a printer, or carefully spinning the shaft of a linear rail in a CNC mill.

Internally, stepper motors are more complex than a simple DC motor, having multiple coils around a permanent magnet core, but with this added complexity comes greater control. Through a careful arrangement of coils built into the *stator*, the *rotor* of a stepper motor can be made to rotate a set *step* by varying the polarity across the coils and switching their polarity in a set *firing pattern*. Steppers aren't all made alike, and the internal designs require unique (yet basic) circuitry to run them properly. We'll discuss the most common types of stepper motors on the following step.

Step 2: Types of Stepper Motors





There are a handful of different stepper motor designs; these include unipolar, bipolar, universal, and variable reluctance. We'll be discussing the design and operation of bipolar and unipolar motors, as these are the most common motor type sold, and the kinds you're most likely to find if you're salvaging motors from other electronics.

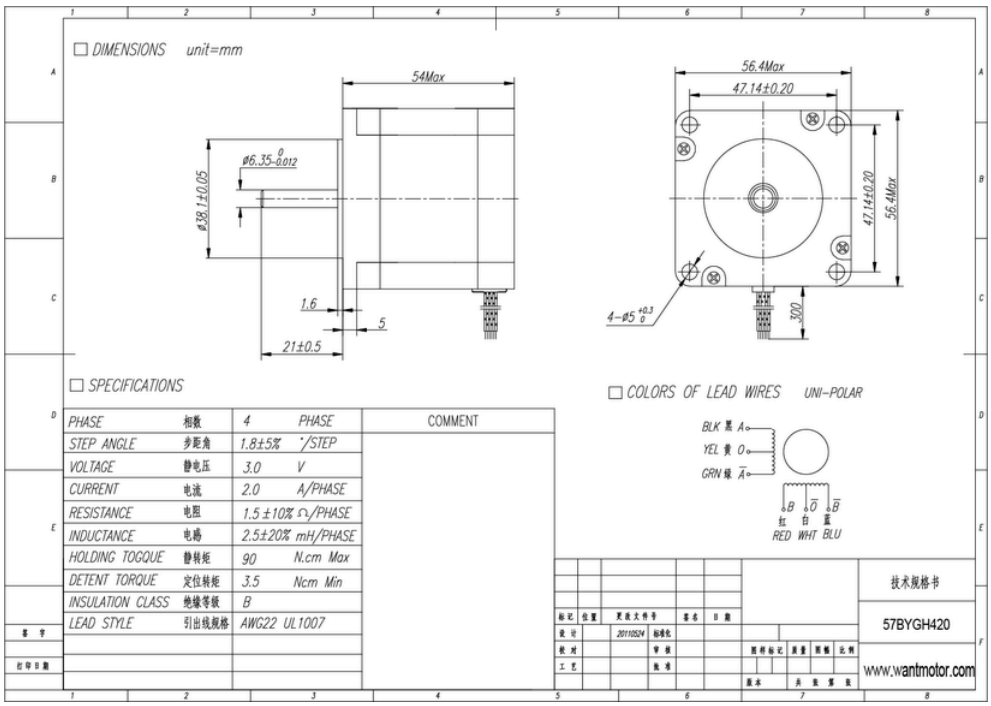
Unipolar Motors

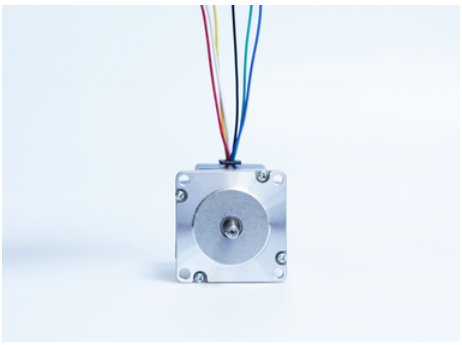
Unipolar motors typically have five, six, or eight wire leads extending from the base and one coil per phase. In the case of a five wire motor, the fifth wire is the joined center taps of the coil pairs. In a six wire motor, the coil pairs each have their own center tap. In a motor with eight wires, each coil pair is completely separated from the others, allowing it to be wired in different configurations. These extra wires allow unipolar motors to be driven directly by an external controller with simple transistors to drive each coil individually. The firing pattern in which each coil is driven determines which direction the shaft of the motor will rotate. Unfortunately, given that only one coil is energized at a time, the holding torque of a unipolar motor will always be less than that of a bipolar motor of the same size. Bypassing the center taps of a unipolar motor, it can now be driven as a bipolar motor, but will require a more complicated control scheme. We'll actually drive a unipolar motor on step four of this instructable, which should clarify some of the concepts introduced above.

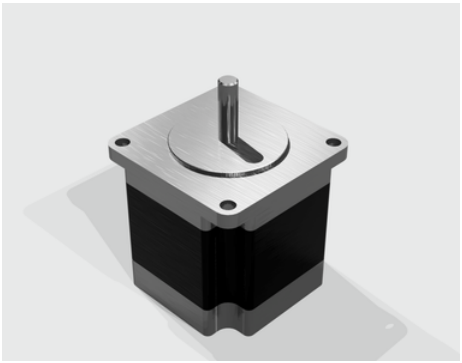
Bipolar Motors

Bipolar motors usually have four wires and are stronger than a comparably sized unipolar motor, but since we only have one coil per phase, we'll need to reverse the current through the coils in order to step. Our need to reverse the current means we'll no longer be able to directly drive the coils with a single transistor, but instead with a full h-bridge circuit. Building a proper h-bridge is tedious (let alone two!), which is why we'll be using a dedicated bipolar motor driver (see step 5).

Step 3: Understanding Stepper Motor Specifications







Let's talk about how to determine the specifications of a motor. If you've come across a square-ish motor with a definite three piece assembly (see picture three), chances are this is a NEMA motor. The National Electrical Manufacturers Association has a defined standard for motor specifications using a simple letter-based code for defining a motor's faceplate diameter, mount type, length, phase current, operating temperature, phase voltage, steps per revolution, and wiring. The RepRap organization has a full breakdown if you'd like to read more about the NEMA standard.

Reading a Motor Data Sheet

For the next step, I'll be using this unipolar motor. I've attached the datasheet above, and although it is brief, it provides us with everything we'll need to drive it properly. Let's breakdown what is listed:

Phase: This a four phase unipolar motor. Internally, the motor may have any number of actual coils, but in this case they are grouped into four phases which can be independently driven.

Step Angle: With an approximate of resolution of 1.8° per step, we'll get 200 steps per revolution. Although this is the mechanical resolution, with *microstepping* we can squeeze more resolution without any modification to the motor (more on this in step 5).

Voltage: The rated voltage of this motor is 3 volts. This is a function of the current and resistance ratings of the motor (Ohms Law V=IR thus 3V = 2A * 1.5Ω)

Current: How much juice does this motor want? Two amps per phase! This figure will be important when selecting our power transistors for the basic driver circuit.

Resistance: 1.5 ohms per phase will limit how much current we can apply to each phase.

Inductance: 2.5 millihenries. The inductive nature of the motor coils will place a limit on how quickly we can charge the coils,

Holding Torque: This will be how much actual force we can create when the stepper is energized.

Detent Torque: This is how much holding torque we can expect from the motor when it is *not* energized.

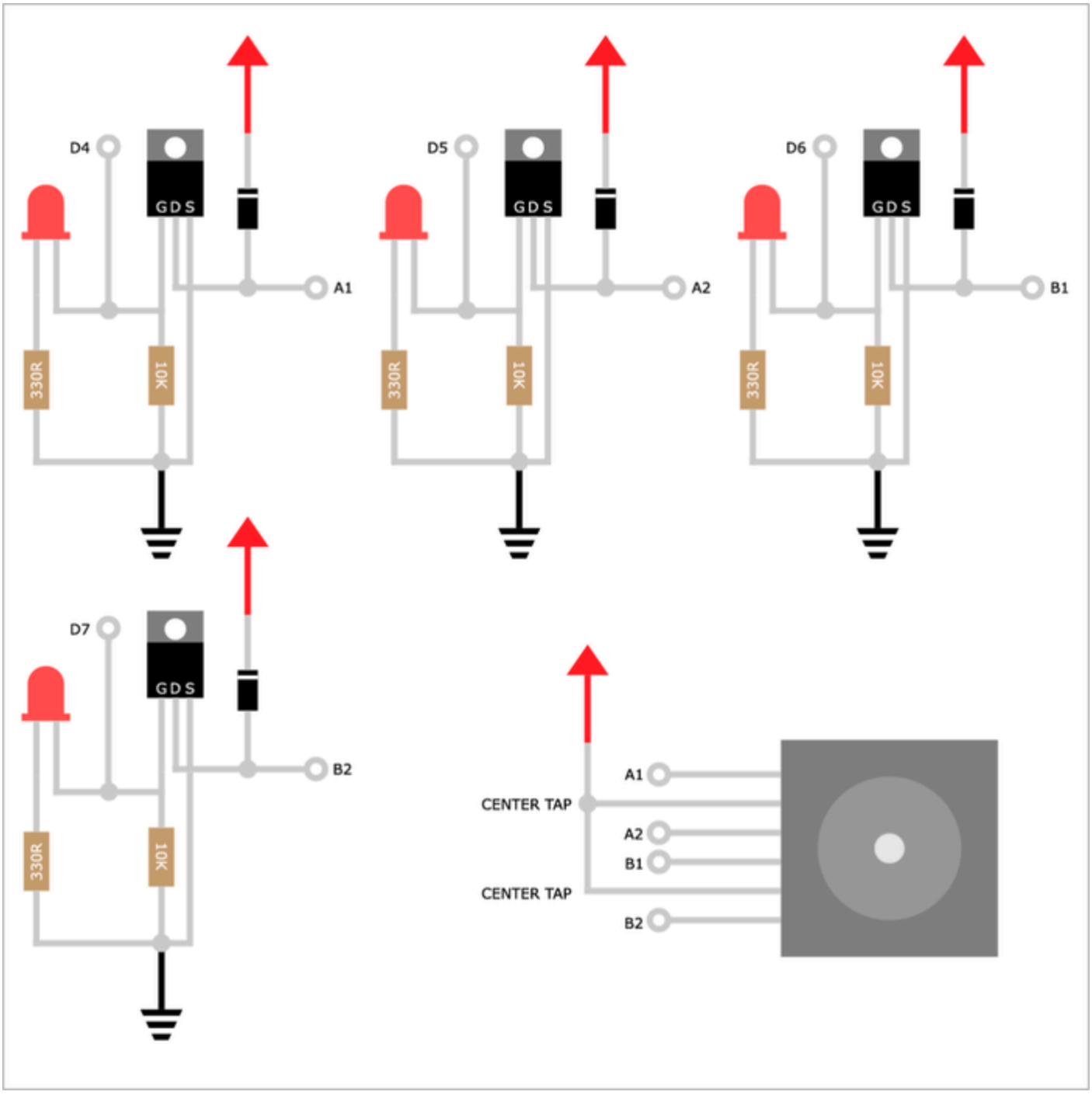
Insulation Class: Class B is part of the NEMA standard and gives us a rating of 266 °F. Steppers are not terribly efficient, and drawing maximum current all of the time means they'll get quite hot during normal operation.

Lead Style: 22 American wire gauge standard.

Determining Coil Pairs

Although the resistance of the coil windings can vary from motor to motor, if you've got a multimeter, you can measure the resistance across any two wires, if the resistance is <10 ohms, you've likely found a pair! This is mostly a process of trial an error, but should work for most motors when you don't have a part number/data sheet.

Step 4: Directly Driving Stepper Motors

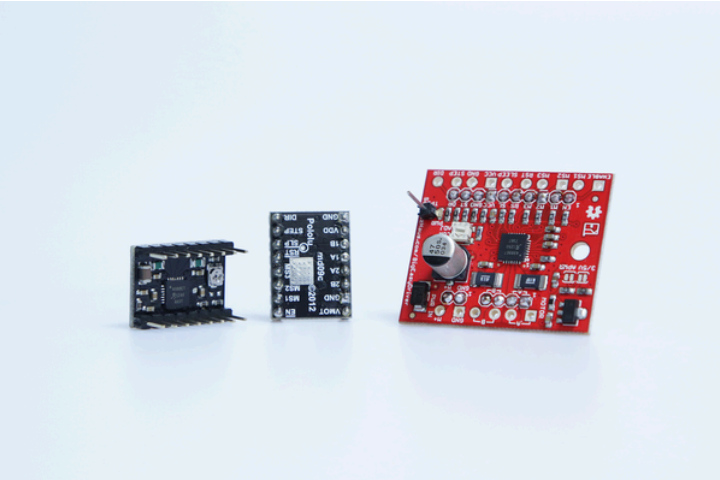


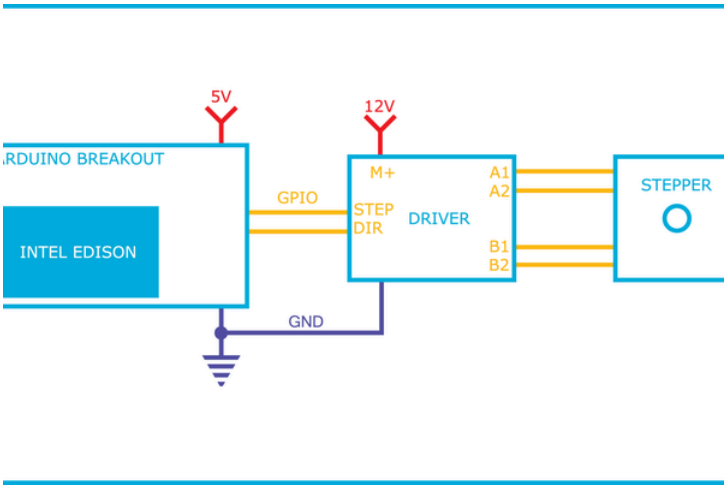
Due to the arrangement of wires within a unipolar motor, we can energize the coils in sequence using only simple power MOSFETs. The illustration above shows a simple MOSFET driven circuit. This arrangement allows simple logic level control with an external microcontroller. In this case I'm using the Intel Edison board with an Arduino-style breakout board to give me easy access to the GPIO (any micro with four GPIO will do however). For this circuit I'm using the IRF510 N-channel power MOSFET. With the ability to sink up to 5.6 amps, the IRF510 will have plenty of current head space to meet the motor's 2 amp needs. The LEDs aren't necessary, but will give you a nice visual confirmation of the firing sequence. It's important to note that the IRF510 must be driven with a logic level of at least 5 volts in order for it to be able to sink enough current for the motor. The power to the motor in this circuit will be 3V.

Firing Sequence

Controlling the unipolar motor at full steps with this setup is dead easy. In order to rotate the motor, we'll have to engage the phases in a set *firing pattern* in order for it to spin properly. In order to rotate the motor clockwise, we'll drive the phases as follows: A1, B1, A2, B2. That's it! In order to spin counter clockwise, we'll just reverse the direction of the sequence to B2, A2, B1, A1. This is fine for basic control, but what if you want more precision and less work? Let's talk about using a dedicated driver to make things much easier!

Step 5: Stepper Motor Driver Boards





If you'd like to get down to controlling bipolar motors (or unipolar motors in a bipolar configuration), you'll want to grab a dedicated driver breakout board. Pictured above are the Big Easy Driver and the A4988 Stepper Motor Driver Carrier board. Both of these boards are breakout PCBs for the Allegro A4988 microstepping bipolar stepper motor driver, which is by far one of the most common chips for driving smaller stepper motors. Aside from having the requisite dual h-bridges for driving a bipolar motor, these boards pack a lot of punch for a tiny low-cost package. You can read a solid breakdown of features on the Pololu webpage, but we'll talk about the most important features here.

Hookup

These all-in-one boards have a wondrously low hookup connection. You can begin driving a motor with only three connections (only two GPIO) to your main controller: common ground, step, and direction. The step and direction pins are left floating, so you'll need to tie them to a reference voltage with a pull-up resistor. A pulse sent to the STEP pin will move the motor one step at a resolution according to the microstep reference pins.The logic level of the DIR pin determines whether the motor will spin clockwise or counterclockwise.

Microstepping

Depending on how the the M1, M2, and M3 pins are set, you can achieve increased motor resolution through microstepping. Microstepping involves sending varied pulses to pull the motor in between the electromagnetic resolution of the physical magnets in the rotor allowing for very precise control. The A4988 can go from full-step down to sixteenth step resolution. With our 1.8° motor, this would provide up to 3200 steps per revolution. Talk about fine detail!

Code / Libraries

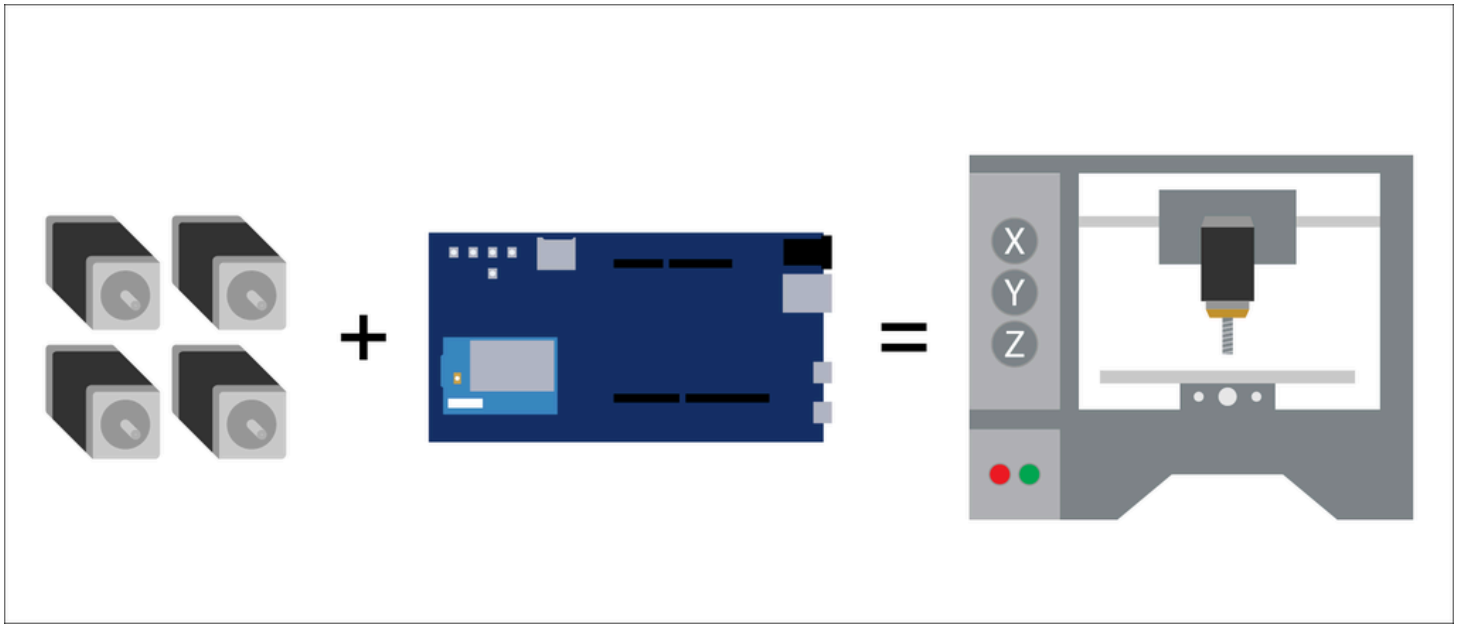
Connecting the motors might be a breeze, but what about controlling them? Check out these ready-made code libraries for driving stepper motors:

Stepper- A classic that comes built in to the Arduino IDE, allows for basic stepping and RPM control.

AccelStepper- A much more fully feature library that allows for better control of multiple motors and proper motor acceleration and deceleration.

Intel C++ MRAA Stepper - A lower level library for those who want to venture into raw C++ control of a stepper motor using the Intel Edison.

Step 6: Stepper Motor Applications



I hope you've found this instructable both informative and enjoyable. This should be enough to get your feet wet in the electro-mechanical world of working with stepper motors, but this is only the beginning. Now that you've got a foothold, where should you go from here? Check out these great projects for stepper-centric inspiration:

Subtractively manufacture your dreams with your own CNC Mill or precisely harness the power of lasers to do your bidding with your own engraver/cutter. If you prefer more constructive means of fabrication, why not build your own3D printer? You can start from open source plans, or build it from scratch! If you're a bigger fan of fewer dimensions, check out these 2D plotter projects for producing pleasant portraits (or anything really). Perhaps you'd like none of these things, and simply wish to observe and harness the raw energy of the sun to power your DIY projects.