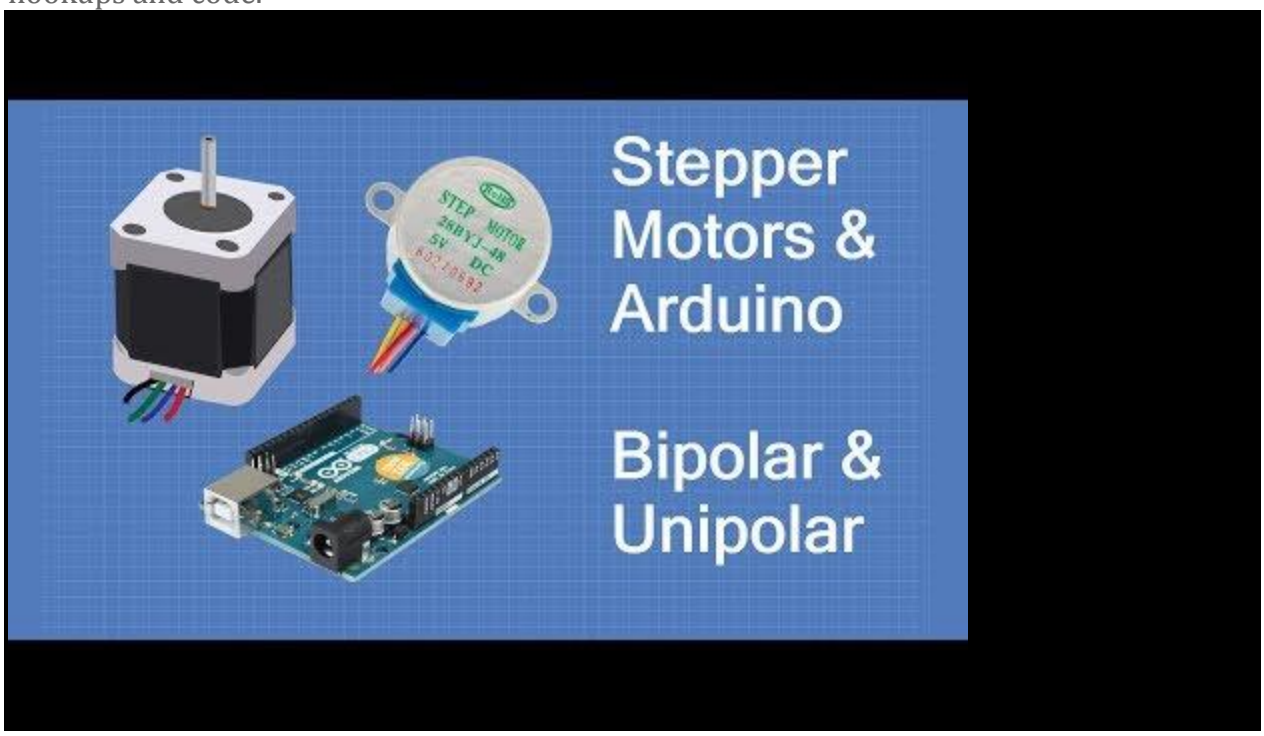
Stepper Motors with Arduino – Getting Started with Stepper Motors

Learn how to control bipolar and unipolar stepper motors with an Arduino using drivers like ULN2003, L298N and A4988. In this article I'll show you all you need to know to get started with stepper motors.

Introduction

Stepper Motors are used in a wide variety of devices ranging from 3D printers and CNC machines to DVD drives, heating ducts and even analog clocks. Yet despite their popularity many experimenters shy away from using stepper motors as they seem to require complex hookups and code.



In this article I hope to dispel that myth by showing you just how easy it is to use a stepper motor with an Arduino. So follow along, I promise to take you through all of this "complex" stepper theory one step at a time!

Stepper Motors

Stepper motors are DC motors that rotate in precise increments or "steps". They are very useful when you need to position something very accurately. They are used in 3D printers to position the printhead correctly and in CNC machines where their precision is used to

position the cutting head. If your digital camera has an autofocus or remote zoom feature chances are a stepper motor is being employed to do that.

Unlike DC motors stepper motors are controlled by applying pulses of DC electricity to their internal coils. Each pulse advances the motor by one step or by a fraction of a step, the latter is known as “microstepping” and will be explained shortly.

Some users confuse stepper motors with servo motors but they are actually two different beasts. A servo motor is unique in that it’s motor shaft can be moved to a precise angle, most servos only rotate 180 or 270 degrees although there are modified servos that can spin a full 360 degrees. A servo motor is “aware” of its position and can be moved to a specific angle even if an external force moves the motor shaft.

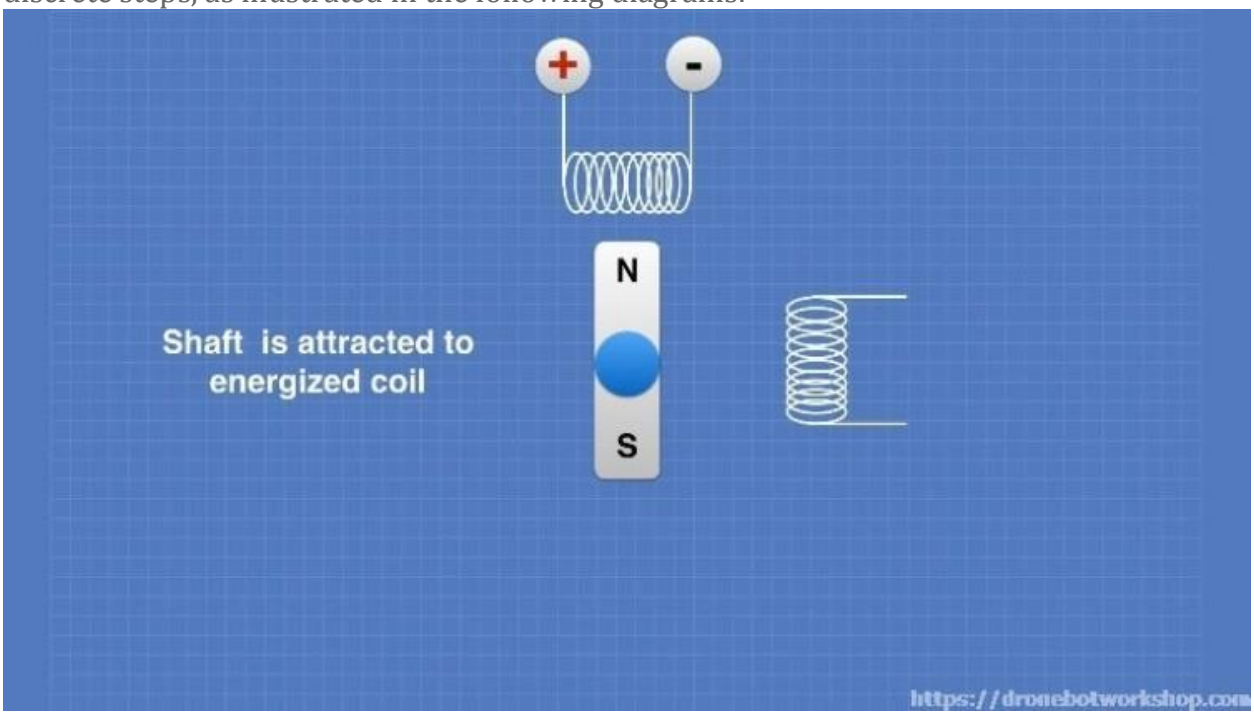
Steppers, on the other hand, are “unaware” of their position. They can be moved to an exact position in reference to where they start stepping (i.e 36 degrees clockwise) but unlike servos they can be misaligned if their shaft is moved by an external force. In many applications a servo is first moved to a “homing” or reference position before being controlled, printers commonly do this when they are first initialized.

Because the move in discrete steps a stepper motor is not often used where a smooth continuous rotation is required, However with the use of gearing and microstepping they can approach a smooth rotation and their ability to be very accurately positioned often outweighs the roughness of their movement.

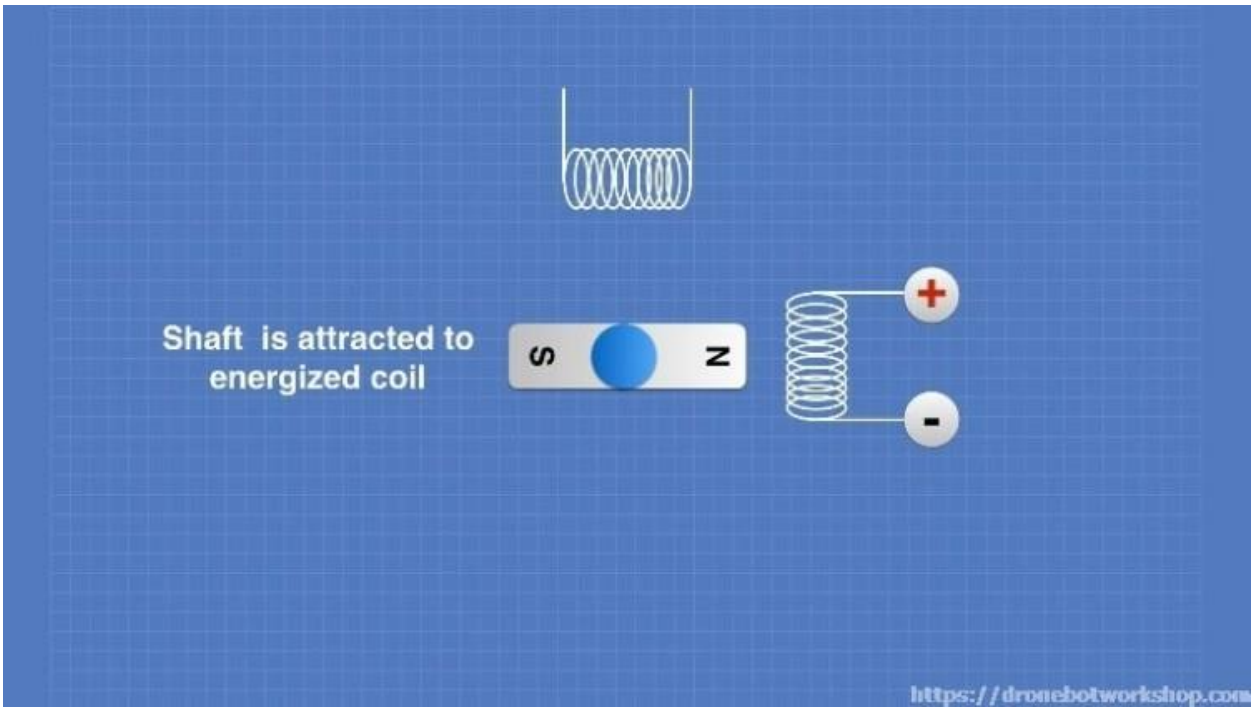
Another advantage stepper motors have over DC motors is the ability to move art very slow speeds without stalling, in fact stalling really isn't a concept with stepper motors. They also pack a lot of torque into a comparably small package.

How Stepper Motors Work

Stepper motors have a magnetized geared core that is surrounded by a number of coils which act as electromagnets. Despite the number of coils electrically there really are usually only two coils in a stepper motor, divided into a number of small coils.
By precisely controlling the current in the coils the motor shaft can be made to move in discrete steps, as illustrated in the following diagrams:



In the first diagram the coil at the top is energized by applying electricity in the polarity shown. The magnetized shaft is attracted to this coil and then locks into place.
Now look what happens when the electricity is removed from the top coil and applied to the other coil. The shaft is attracted to the second coil and locks into place there.

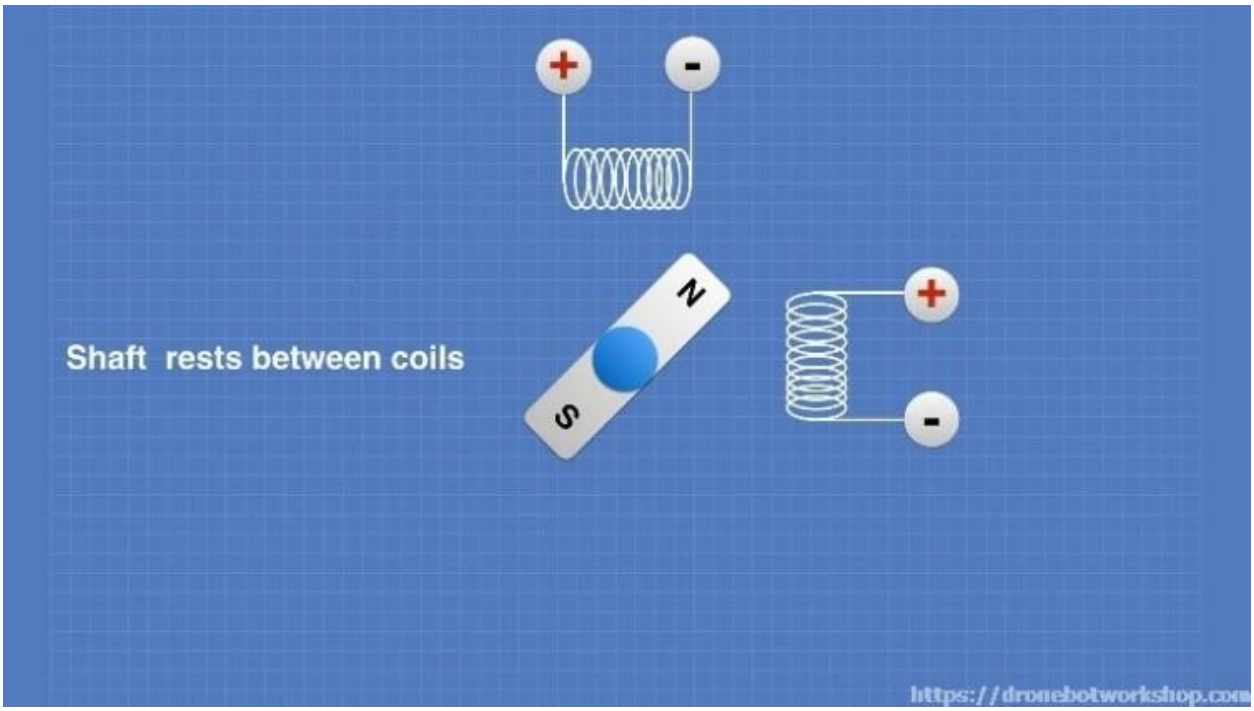


The jump between the two positions is one step (in this illustration a step is 90 degrees, in actual fact a stepper motor usually steps just a fraction of this. The diagrams are simplified for clarity).

Microstepping

We have seen how the motor shaft moves to lock itself into place in front of an attracting electromagnet, each magnet represents one step. It is, however, possible to move the motor shaft into positions between steps. This is known as “microstepping”.

In order to understand how microstepping works look at the next diagram:



In this illustration the current has been applied to BOTH coils in an equal amount. This causes the motor shaft to lock into place halfway between the two coils. This would be known as a “half step”.

The principle can be extended to include quarter steps, eight steps and even sixteenth steps. This is done by controlling the ratio of the current applied to both coils to attract the motor shaft to a position between the coils but closer to one coil than the other.

By using microstepping it is possible to move the shaft of a stepper motor a fraction of a degree, allowing for extremely precise positioning.

Types of Stepper Motors

Internally there are a number of ways to design a stepper motor, such as Variable Reluctance, Permanent Magnet and Hybrid stepper motors. These design differences primarily deal with the method employed to create the magnetic field within the motor.

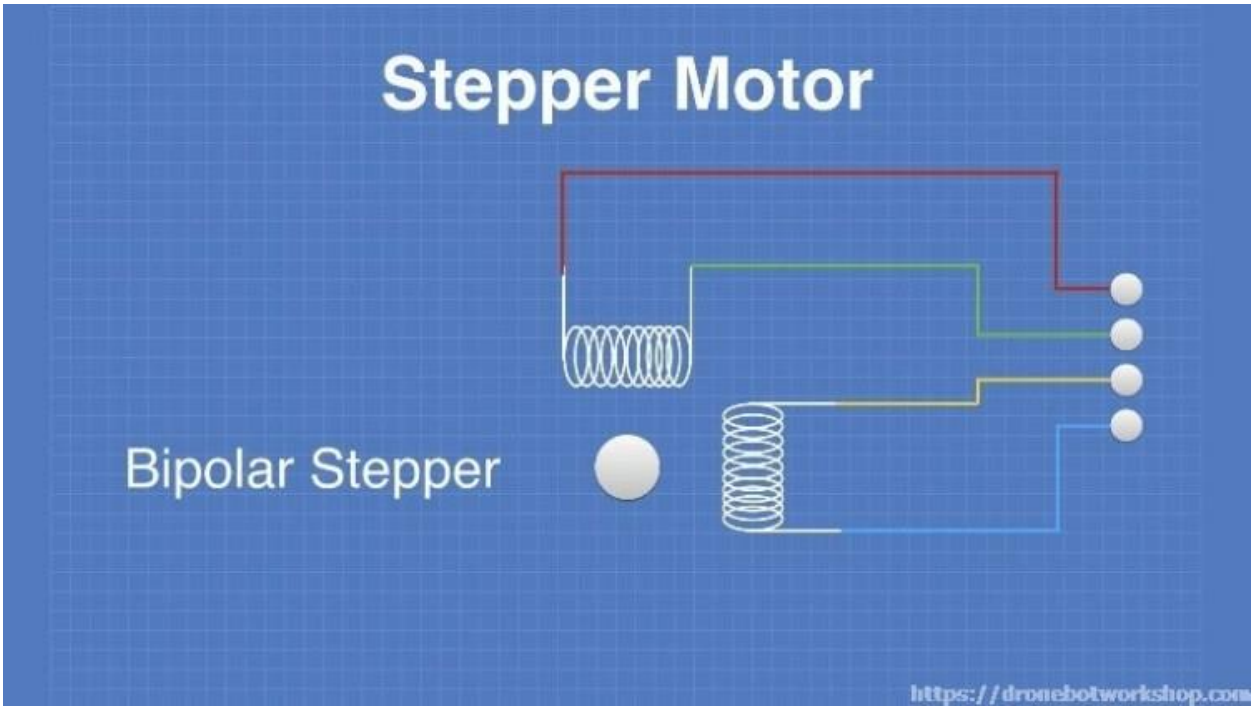
For most experimenters these differences will be merely academic but if you are choosing a stepper motor for a very specific design you may want to look into this more.

For most users the main difference between stepper motor design boils down to the way the coils are wired within the motor. There are two methods employed – Bipolar and Unipolar. These two types of stepper motors are not interchangeable (although it is possible to “hack” a Unipolar motor to create a Bipolar motor).

Let’s look at these two types of stepper motors.

Bipolar Stepper Motors

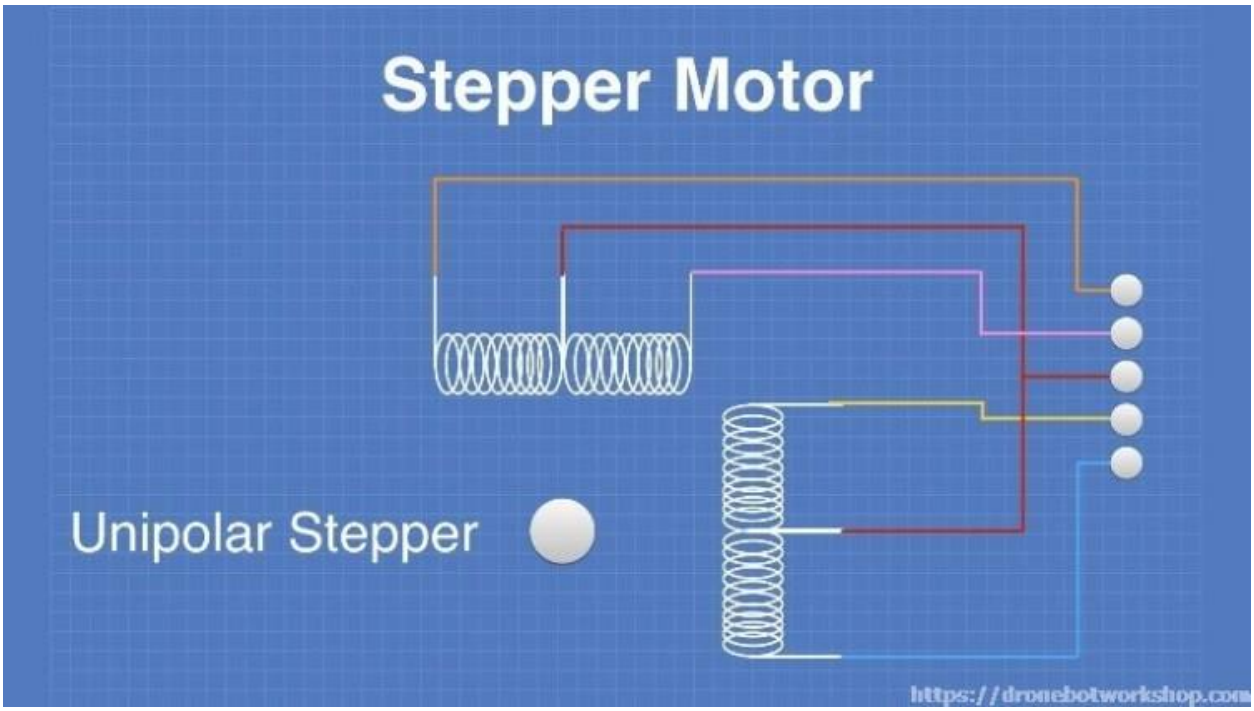
Bipolar stepper motors consist of two coils of wire (electrically, actually split into several physical coils) and generally have four connections, two per coil. The simplified diagrams of stepper operation that you just looked at in the previous section are all bipolar stepper motors.



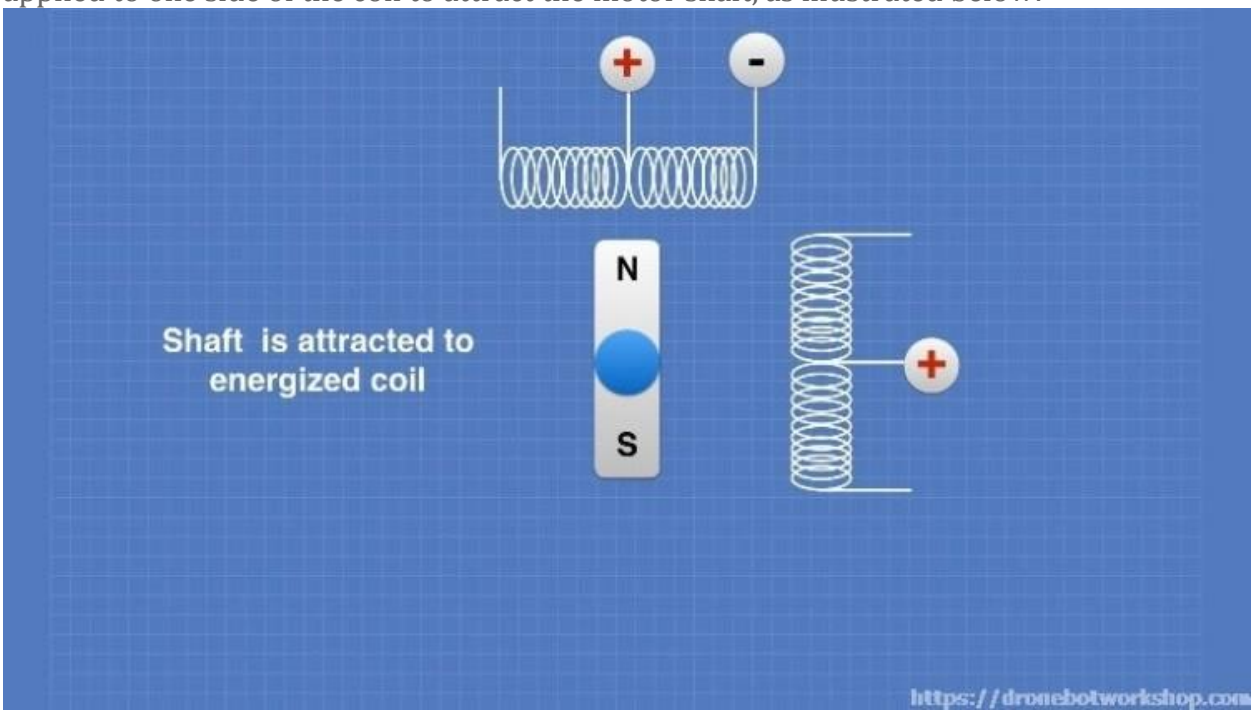
An advantage of bipolar stepper motors is that they make use of the entire coil winding so they are more efficient. However they require a more complex controller or driver to operate as to reverse direction the polarity of the voltage applied to the coils needs to be reversed.

Unipolar Stepper Motors

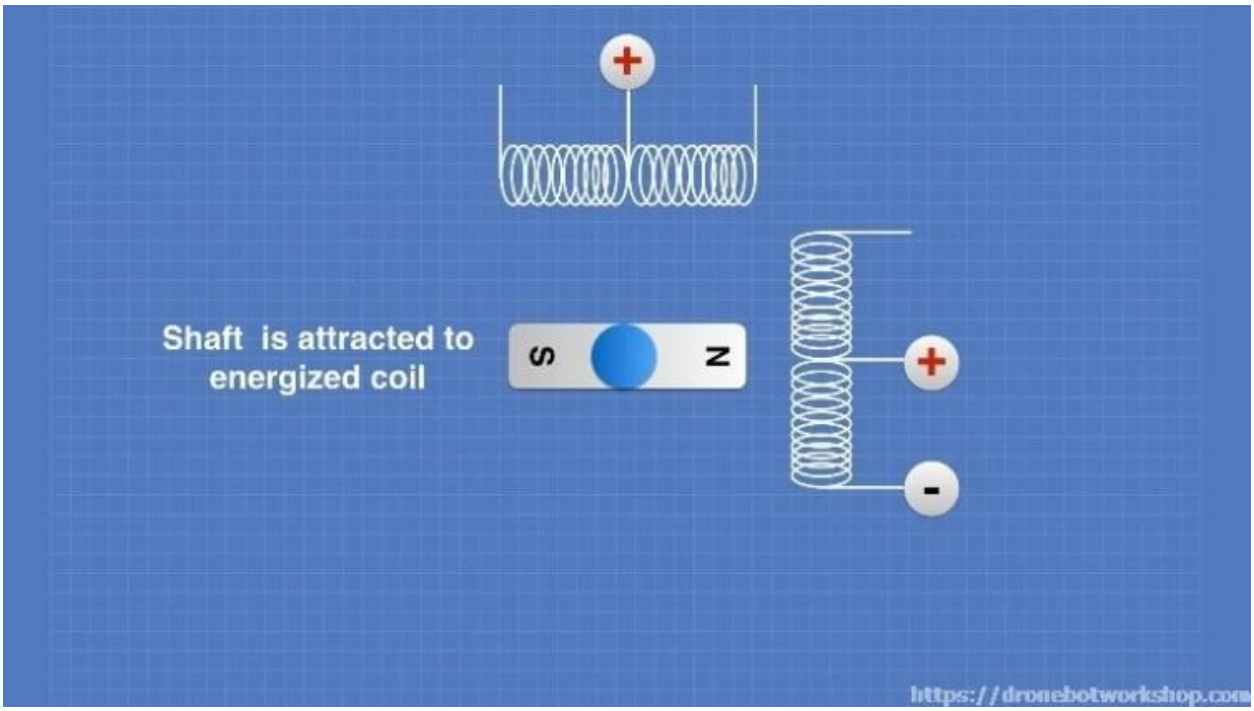
A unipolar stepper motor also consists of two coils (electrically) but each coil has a center tap so there are three connections on each coil. This results in six connections, however many unipolar stepper motors have only five connections as the two center taps are internally connected.



In a unipolar stepper motor only half of each coil is used at one time. In most configurations a positive voltage is applied to the center tap and left there. A negative voltage is then applied to one side of the coil to attract the motor shaft, as illustrated below:

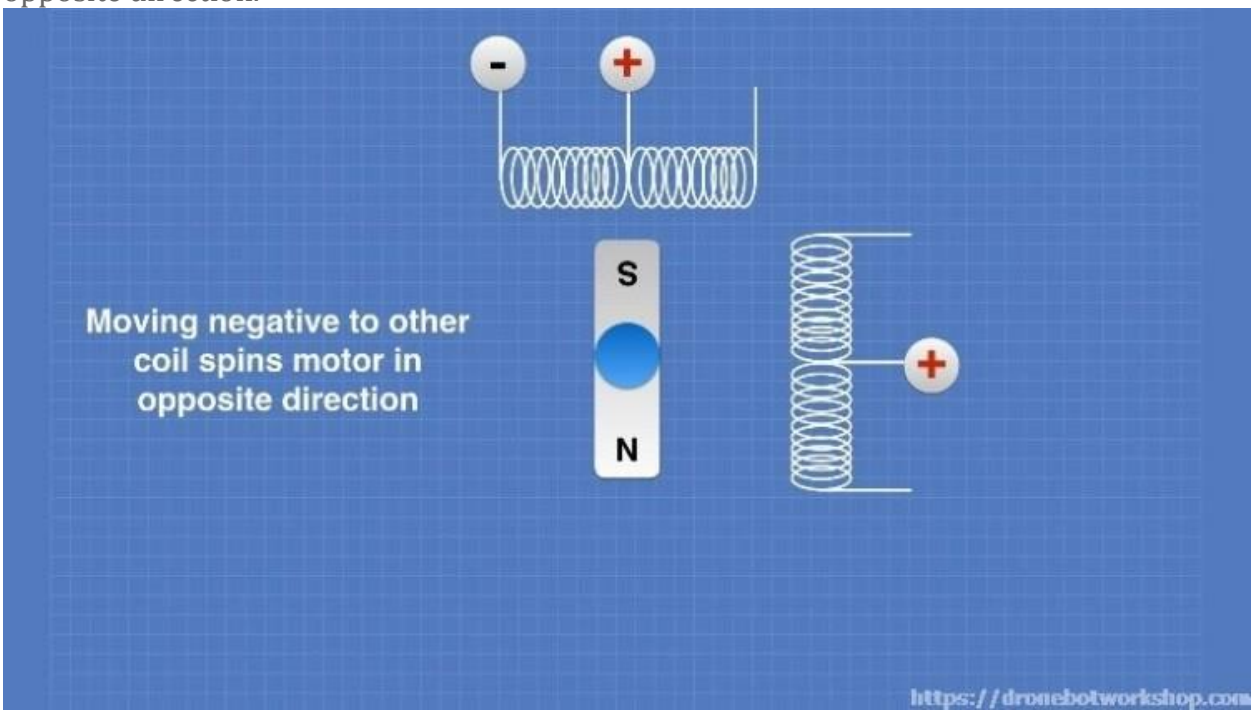


As with the bipolar motor the unipolar stepper motor can be made to advance one step when current is removed from the top coil and applied to one side of the second coil:



You can also microstep a unipolar stepper motor by using the same technique that we used with bipolar steppers, applying current to both coils.

Now to reverse the direction of a unipolar motor you don't need to reverse polarity. Instead the negative voltage is applied to the OTHER side of the coil. This causes the current to flow in the opposite direction within the coil and this in turn moves the motor shaft in the opposite direction.



Unipolar stepper motors are easier to control as there is no requirement to reverse current polarity to change direction. However as the unipolar stepper motor only makes use of half of the coil windings at any given moment they are not as efficient as half of the wiring is essentially wasted.
We will work with both unipolar and bipolar stepper motors in the experiments we are about to do.
It should be noted that there are also stepper motors that can be wired as both bipolar and unipolar. These motors have four coils which can be joined to make either two center tapped coils (for a unipolar configuration) or just two big coils (in a bipolar configuration). These stepper motors will have eight wires, two per coil.

Reading Stepper Motor Specifications

Choosing a stepper motor can be a somewhat daunting task but it doesn't have to be. Many first time users are scared off by the vast number of specifications included with some stepper motors. In actual fact they are not that difficult to understand.
Here are a few of the key specifications you'll find included with stepper motors, along with a short definition of them:
Phase: This refers to the groupings of the individual coils in the stepper motor. A stepper motor may have several coils but they are wired together and controlled in phases. Two, Four and Five phase stepper motors are common. There will often be a phase diagram included with a stepper motor that indicates the sequence that the motor phases are driven in.
Step Angle: This is the amount that the shaft of the motor will spin for each individual full step, measured in degrees, In some stepper motors this is referred to as **Steps Per Revolution** and the two figures are just different ways of expressing the same thing.
As an example a common rating for a stepper motor is a 1.8 degree step angle. As there are 360 degrees in a full rotation this is equivalent to 200 steps per revolution (1.8 x 200 = 360).
Voltage: Simply the voltage rating of the motor coils. It is also a function of the current rating and the coil resistance and you can use Ohm's Law to calculate one from the other.
Current: The maximum current at the rated voltage. This is a useful specification as it will allow you to select a suitable driver and power supply for your stepper motor.
Resistance: The coil resistance, measure in ohms.
Inductance: The inductance of each motor coils, measured in millihenries. This is an important specification as inductance will limit the maximum speed you'll be able to

efficiently drive your stepper at. Typically unipolar stepper motors have an advantage here as they only use half a coil and thus have lower inductance than their bipolar equivalents.

Holding Torque: This will be the amount of force that is created when the stepper motor is energized.

Detent Torque: This is the amount of holding torque that can be expected when the motor is NOT energized.

Shaft Style: The physical shape of the motor shaft. You will need to know this in order to mate your stepper motor with gears, pulleys and other external connections such as shaft couplers. There are several common shapes used, in addition the shaft length can be important for obvious reasons.

Some common shaft types are as follows:

- **Round Shaft** – pretty well says it all!
- **"D" Shaft** – a "D-shaped" shaft, useful for mounting gears with set screws.
- **Geared Shaft** – a shaft with a gear etched into it.
- **Lead-Screw Shaft** – A shaft shaped like a screw, used in constructing linear actuators.

Another obvious specification of a stepper (or any motor) is its physical size. There are a group of stepper motors that have standard sizes, we will look at these now.

NEMA Motor sizes

NEMA is an abbreviation for the National Electrical Manufacturers Association. Although based in the United States this is actually an international standards committee, although being American the specifications were all originally created using the imperial system instead of the metric system.

In 1984 the NEMA committee set out some standards for motor sizes, based upon the face plate size of the motor. This standard is still in use today and results in motors designated "NEMA 17" or "NEMA 23".

The NEMA 17 sized stepper motor has become extremely popular, especially in the construction of 3D printers. It also creates a lot of confusion as you often hear people refer to a motor simply as a "NEMA 17", which really only designates the size of the motor and not it's other specifications such as voltage, current, step angle or even if it is bipolar or unipolar.

The "17" in "NEMA 17" is the face plate size, in the NEMA standard the face plate is the NEMA "number" divided by 10 in inches. So a NEMA 17 motor has a face plate approximately 1.7 inches wide while a NEMA 23 is 2.3 inches wide.

Techref has a good description of NEMA motor sizes.

Experimenting with Stepper Motors

OK enough theory! Time to dig out our Arduino and start experimenting with stepper motors.

There are four experiments we will do today, two of them using a unipolar stepper motor and two of them with the unipolar variety. In addition we will make use of a couple of Arduino libraries, one of which is already included in the Arduino IDE.

Although these experiments have been illustrated using an Arduino Uno any Arduino will work. You can also feel free to change the pin numbers if you need to as there are no special requirements there, just be sure to alter the sketch to reflect those changes if you decide to do that.

Let's get started!

Demo 1 – 28BYJ-48 Unipolar Stepper with ULN2003

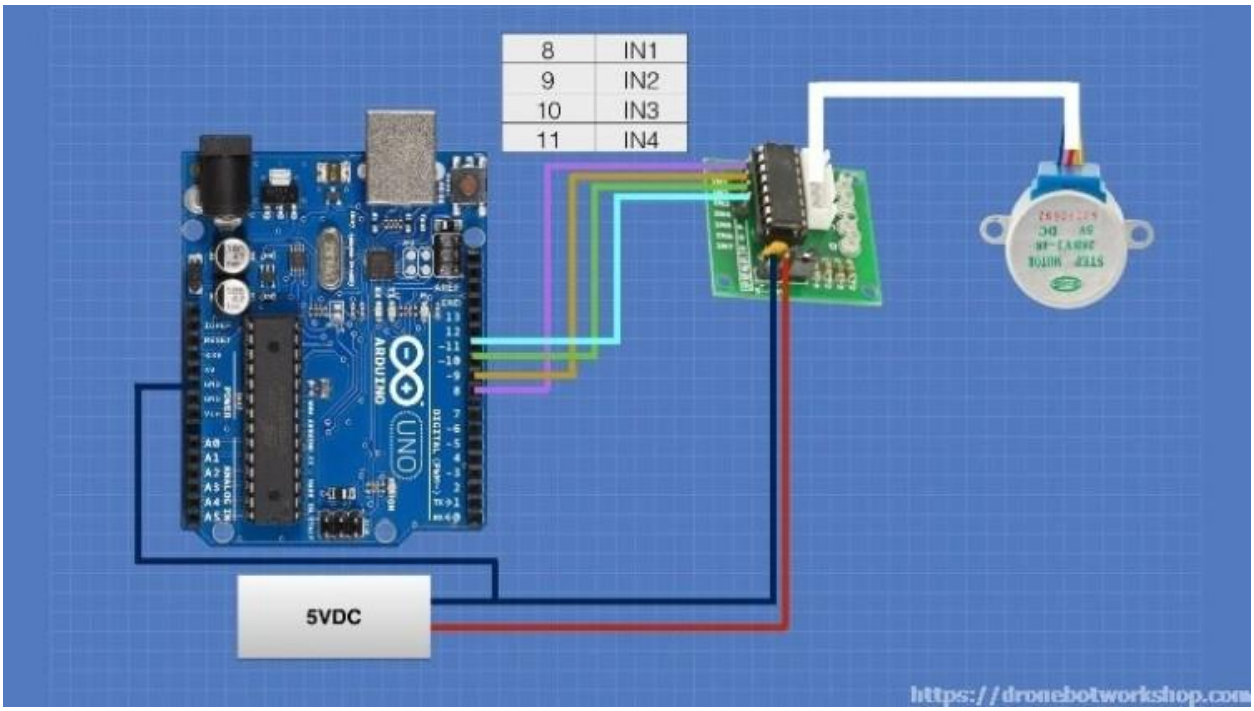
Our first demonstration will make use of an extremely popular stepper motor and driver combination. These motors have been manufactured for decades and are made by the millions so they are very inexpensive, the driver and motor should run you less than five dollars in total.

The 28BYJ-48 is a 5-wire unipolar stepper motor that moves 32 steps per rotation internally but has a gearing system that moves the shaft by a factor of 64. The result is a motor that spins at 2048 steps per rotation. It should be noted that some of these motors may have a different gearing system so the number of steps per rotation of your motor may not be the same. The 28BYJ-48 runs on 5 volts.

The motor is commonly packaged with a tiny driver board based around the ULN2003 darlington transistor array. The board has a connector that mates perfectly with the motor wires so it is very easy to use. There are also connections for four 5-volt digital inputs as well as power supply connections.

On the subject of power supplies one very important thing to note is that you should NEVER use the 5-volt power from your Arduino to power this (or any) stepper motor no matter how tempting it is. Even though the 28BYJ-48 doesn't draw much current it will induce electrical "noise" onto its power supply lines and this could damage your Arduino. Always use a seperate power supply to power your stepper motors!

We will hookup our motor, driver and Arduino as follows:



Now that we have everything hooked up we will need to program the Arduino. Here is the sketch that we will use to do that:

```
/*
  Stepper Motor Demonstration 1
  Stepper-Demo1.ino
  Demonstrates 28BYJ-48 Unipolar Stepper with ULN2003 Driver
  Uses Arduino Stepper Library

  DroneBot Workshop 2018
  https://dronebotworkshop.com
*/

//Include the Arduino Stepper Library
#include <Stepper.h>

// Define Constants

// Number of steps per internal motor revolution
const float STEPS_PER_REV = 32;

//  Amount of Gear Reduction
const float GEAR_RED = 64;

// Number of steps per geared output rotation
const float STEPS_PER_OUT_REV = STEPS_PER_REV * GEAR_RED;

// Define Variables
```

```
26
27 // Number of Steps Required
28 int StepsRequired;
29
30 // Create Instance of Stepper Class
31 // Specify Pins used for motor coils
32 // The pins used are 8,9,10,11
33 // Connected to ULN2003 Motor Driver In1, In2, In3, In4
34 // Pins entered in sequence 1-3-2-4 for proper step sequencing
35
36 Stepper steppermotor(STEPS_PER_REV, 8, 10, 9, 11);
37
38 void setup()
39 {
40 // Nothing  (Stepper Library sets pins as outputs)
41 }
42
43 void loop()
44 {
45   // Slow - 4-step CW sequence to observe lights on driver board
46   steppermotor.setSpeed(1);
47   StepsRequired  =  4;
48   steppermotor.step(StepsRequired);
49   delay(2000);
50
51    // Rotate CW 1/2 turn slowly
52   StepsRequired  =  STEPS_PER_OUT_REV / 2;
53   steppermotor.setSpeed(100);
54   steppermotor.step(StepsRequired);
55   delay(1000);
56
57   // Rotate CCW 1/2 turn quickly
58   StepsRequired  =  - STEPS_PER_OUT_REV / 2;
59   steppermotor.setSpeed(700);
60   steppermotor.step(StepsRequired);
61   delay(2000);
62
63 }
```

In this sketch we make use of the Arduino Stepper Library which comes packaged with your Arduino IDE. The stepper library takes care of sequencing the pulses we will be sending to our stepper motor and it can be used with a wide variety of motors, both unipolar and bipolar.

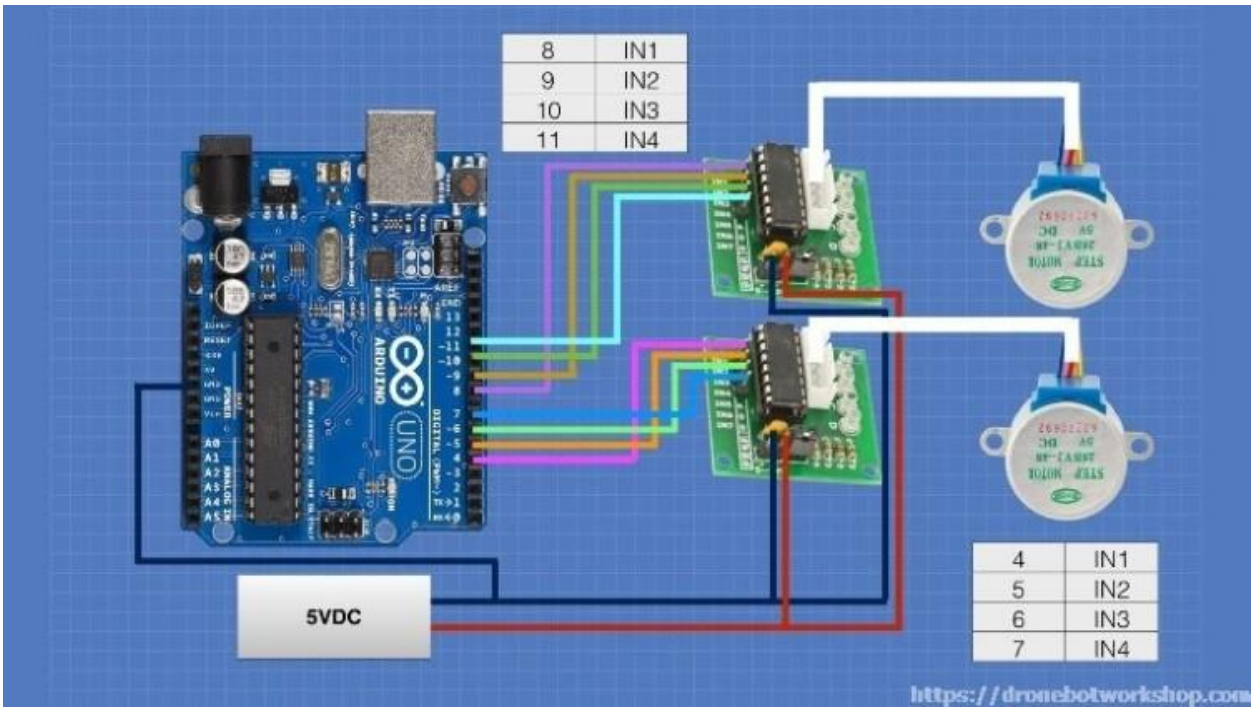
The 28BYJ-48 stepper motors have internal gearing which reduces the output rotation by a factor of 64 (as noted above some are different). So we define three constants to handle motor rotation:

- STEPS_PER_REV is the number of steps the actual motor takes per revolution. This is set at 32.
- GEAR_RED is the amount of gear reduction. I set mine to 64 but you may need to adjust this if your motor is different
- STEPS_PER_OUT_REV is the final output of the motor shaft after gear reduction. It is the multiple of the above two numbers.

In case you are wondering why a float was used instead of an integer for the above constants it's because the gear reduction is sometimes a number like 63.5. If yours is 64 you could always use integers.
The variable "StepsRequired" will be used to define the number of steps we want our motor to rotate.
The 28BYJ-48 Unipolar stepper motor has a step sequence as follows: 1-3-2-4 . This information will be used to drive the motor by creating an instance of the Stepper class called "steppermotor" with the pin sequence of 8,10, 9, 11 . Make sure you get this right or the motor will not operate properly.
There is nothing to set in the setup routine as the Arduino Stepper library class already sets the four I/O pins as outputs.
In the loop we have three demonstration runs, you can feel free to add as many more as you wish or to change the existing ones.
The first run steps the motor four steps very slowly. It is interesting to observe the LEDs on the UNL2003 as this runs.
The second run turns the motor clockwise a half turn very slowly. And the final run returns the motor a half turn at a much faster speed. You can experiment with the "setSpeed" parameter to determine what the top speed for your motor is.

Demo 2 – Two 28BYJ-48 Unipolar Steppers with ULN2003

For the next demonstration we will add a second 28BYJ-48 stepper and ULN2003 driver set to the picture to drive two motors simultaneously. We will also make use of an advanced stepper motor library that you will need to install in your Arduino IDE.
Before we get to the code let's hook up an additional motor and driver to our Arduino. Leave the connections you made in Demo 1 as they are and wire the new devices as follows:



Once again we will power the driver and motor from an external 5-volt power supply. This sketch will make use of the AccelStepper library which is an advanced library written by Mike McCauley. You will need to install this library using the Arduino IDE Library Manager as it is not included in the Arduino IDE.

```
/*
  Stepper Motor Demonstration 2
  Stepper-Demo2.ino
  Demonstrates Two 28BYJ-48 Unipolar Steppers with ULN2003 Driver
  Uses Accelstepper Library

  DroneBot Workshop 2018
  https://dronebotworkshop.com
*/

// Include the AccelStepper Library
#include <AccelStepper.h>

// Define Constants

// Define step constants
#define FULLSTEP 4
#define HALFSTEP 8

// Define Motor Pins (2 Motors used)

```

```
#define motorPin1  8     // Blue   - 28BYJ48 pin 1
#define motorPin2  9     // Pink   - 28BYJ48 pin 2
#define motorPin3  10    // Yellow - 28BYJ48 pin 3
#define motorPin4  11    // Orange - 28BYJ48 pin 4


#define motorPin5  4     // Blue   - 28BYJ48 pin 1
#define motorPin6  5     // Pink   - 28BYJ48 pin 2
#define motorPin7  6     // Yellow - 28BYJ48 pin 3
#define motorPin8  7     // Orange - 28BYJ48 pin 4

// Define two motor objects
// The sequence 1-3-2-4 is required for proper sequencing of 28BYJ48
AccelStepper stepper1(HALFSTEP, motorPin1, motorPin3, motorPin2, motorPin4);
AccelStepper stepper2(FULLSTEP, motorPin5, motorPin7, motorPin6, motorPin8);

void setup()
{
  // 1 revolution Motor 1 CW
  stepper1.setMaxSpeed(1000.0);
  stepper1.setAcceleration(50.0);
  stepper1.setSpeed(200);
  stepper1.moveTo(2048);

  // 1 revolution Motor 2 CCW
  stepper2.setMaxSpeed(1000.0);
  stepper2.setAcceleration(50.0);
  stepper2.setSpeed(200);
  stepper2.moveTo(-2048);

}


void loop()
{
  //Change direction at the limits
  if (stepper1.distanceToGo() == 0)
    stepper1.moveTo(-stepper1.currentPosition());
    if (stepper2.distanceToGo() == 0)
    stepper2.moveTo(-stepper2.currentPosition());

  stepper1.run();
  stepper2.run();

}
```

In this demonstration we will drive one motor at full steps and the second one at half steps. We will define two constants at the beginning of the sketch for this. We'll also define eight constants, one for each motor output pin on the Arduino.

Next we setup two motor objects, one for each motor. We use our pin definitions and the step definitions to set these up.

In the setup routine we setup the maximum speed, acceleration factor, initial speed and the number of steps we will move to (I used 2048 as the motor spins 2048 steps per

rotation). Note that the second motor has a negative number of steps, this indicates it is to move counterclockwise when it is initialized.

In the loop we use an If statement to check how far the motors need to travel until they get to the "moveTo" position using the AccelStepper library "distanceToGo" parameter. Once that reaches zero we change the "moveTo" position to the negative of the current position, which will result in the motor moving in the opposite direction to the other end of travel. The motors are actually set into motion using the AccelStepper library "run" function.

The movement of the two motors is interesting to watch as they accelerate and decelerate in opposite directions. Note that one motor is running at full steps while the other uses half steps, observe the lights on the UNL2003 controller when the motors start and stop and you'll notice a difference in the step patterns.

Demo 3 – Bipolar Stepper with L298N H-Bridge

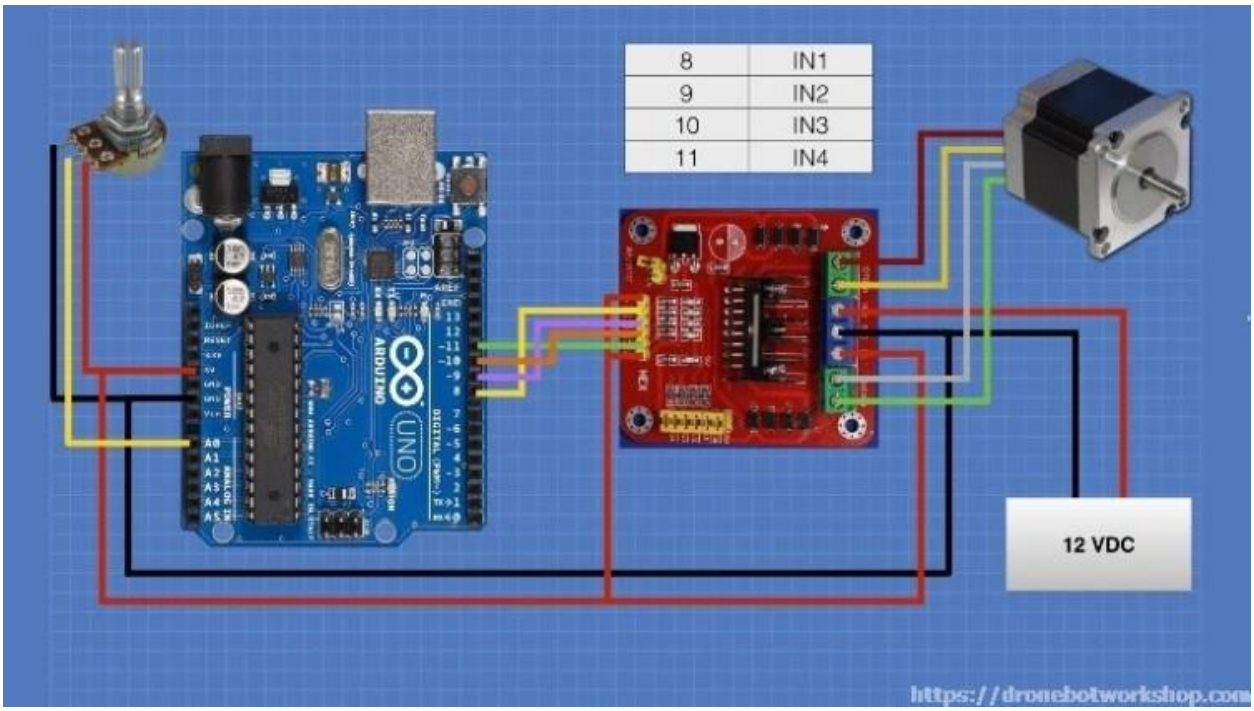
Now that we have worked with a unipolar stepper motor it's time to switch to a bipolar stepper. For this experiment I used a NEMA 17 sized bipolar stepper rated at 12 volts but any bipolar stepper motor can be used as long as you observe the voltage ratings and use a suitable power supply. Once again please don't attempt to power the motor from the Arduino power supply.

As you recall a bipolar stepper motor requires a driver that can reverse polarity to the motor coils in order to reverse the motor direction. A good component to accomplish this with is an "H-Bridge".

We have discussed using an H-Bridge before when we talked about controlling brushed DC Motors. Essentially this is a device that contains four internal power transistors that allow control of the direction of current through a motor coil.

We will be using the same H-Bridge controller that we used in the previous article, the L298N module. These modules are very inexpensive and are very reliable and they can be used to control either two DC motors or one stepper motor. Obviously we will be using it to control a stepper motor today, the L298N is a dual H-Bridge so each H-Bridge will drive one of the coils in our bipolar stepper motor.

Here is how I have hooked up my L298N H-Bridge, bipolar stepper and Arduino Uno:



Note that you may not need to make all of these connections, this depends upon how you configure your L298N module. And also note that the motor power supply you use should match your motor requirements.

The L298N module has a jumper to set its internal 5-volt logic circuits to use either an external power supply (jumper off) or to use a built-in voltage regulator and derive the 5-volts from the motor power supply (jumper on). If you choose to use your motor supply make sure it is at least 7.5 volts and eliminate the power connection from the Arduino 5 volt output.

Some L298N modules also have a set of jumpers that allow you to tie the two Enable lines high so the the motors are always enabled, which is what we want here. If you have these you can also eliminate the connection from the Arduino 5-volt output to ENA and ENB and just set the jumpers instead.

We will also use a potentiometer to act as a speed control. Any value from 10k up will work, lower values will put a lot of load onto the Arduino 5-volt output.

Once you get everything hooked up it's time to load the code up to the Arduino. Here is the sketch:

```
1 /*
```

```
  Stepper Motor Demonstration 3
  Stepper-Demo3.ino
  Demonstrates NEMA 17 Bipolar Stepper with L298N Driver
  Uses Potentiometer on Analog Input A0
  Uses Arduino Stepper Library

  DroneBot Workshop 2018
  https://dronebotworkshop.com
*/

// Include the Arduino Stepper Library
#include <Stepper.h>

// Define Constants

// Number of steps per output rotation
const int STEPS_PER_REV = 200;
const int SPEED_CONTROL = A0;

// Create Instance of Stepper Class
// Specify Pins used for motor coils
// The pins used are 8,9,10,11
// Connected to L298N Motor Driver In1, In2, In3, In4
// Pins entered in sequence 1-2-3-4 for proper step sequencing

Stepper stepper_NEMA17(STEPS_PER_REV, 8, 9, 10, 11);

void setup() {
  // nothing to do inside the setup
}

void loop() {
  // read the sensor value:
  int sensorReading = analogRead(SPEED_CONTROL);
  // map it to a range from 0 to 100:
  int motorSpeed = map(sensorReading, 0, 1023, 0, 100);
  // set the motor speed:
  if (motorSpeed > 0) {
    stepper_NEMA17.setSpeed(motorSpeed);
    // step 1/100 of a revolution:
    stepper_NEMA17.step(STEPS_PER_REV / 100);
  }
}
```

The sketch uses the Arduino Stepper library again. If you feel like a challenge you can rewrite it to use the AccelStepper library instead.

After including the library we define a couple of constants:

- STEPS_PER_REV is the number of steps per revolution that our motor is rated at. Mine was rated at 200, which is the same as 1.8 degrees per step. Change this to match your motor.
- SPEED_CONTROL is the analog port we connect the potentiometer. It is set to A0.

The sequence for our bipolar stepper is 1-2-3-4 so we create our instance of the stepper class with this in mind. Our L298N is connected to pins 8, 9, 10 and 11.

Again the stepper library sets up the pins as outputs so there is no need to do that in the setup routine.

In the loop we read the potentiometer position by measuring the input voltage on the analog pin using the Arduino analogRead function. We then map it to a range of 0 to 100 using the useful Arduino map function.

The value derived from the map function is then used to set the motor speed. As long as it is over zero we set the motor speed and then step it one one hundredth of a revolution, which in the case of my motor will move it two steps or 3 degrees.

After that we do it all again. The result is that the motor speed will be controlled by the potentiometer.

Note that no attempt has been made to control the motor direction in tis design. If you wish you can do this by setting the motor speed to a negative number to spin the motor counterclockwise. The H-Bridge will do the job of reversing the motor voltage polarity to reverse the motor.

As you can see an L298N makes a great stepper motor controller as well as a DC motor controller. But like the UNL2003 it still require the Arduino to do all the motor sequencing. In our next experiment we will use a dedicated motor controller.

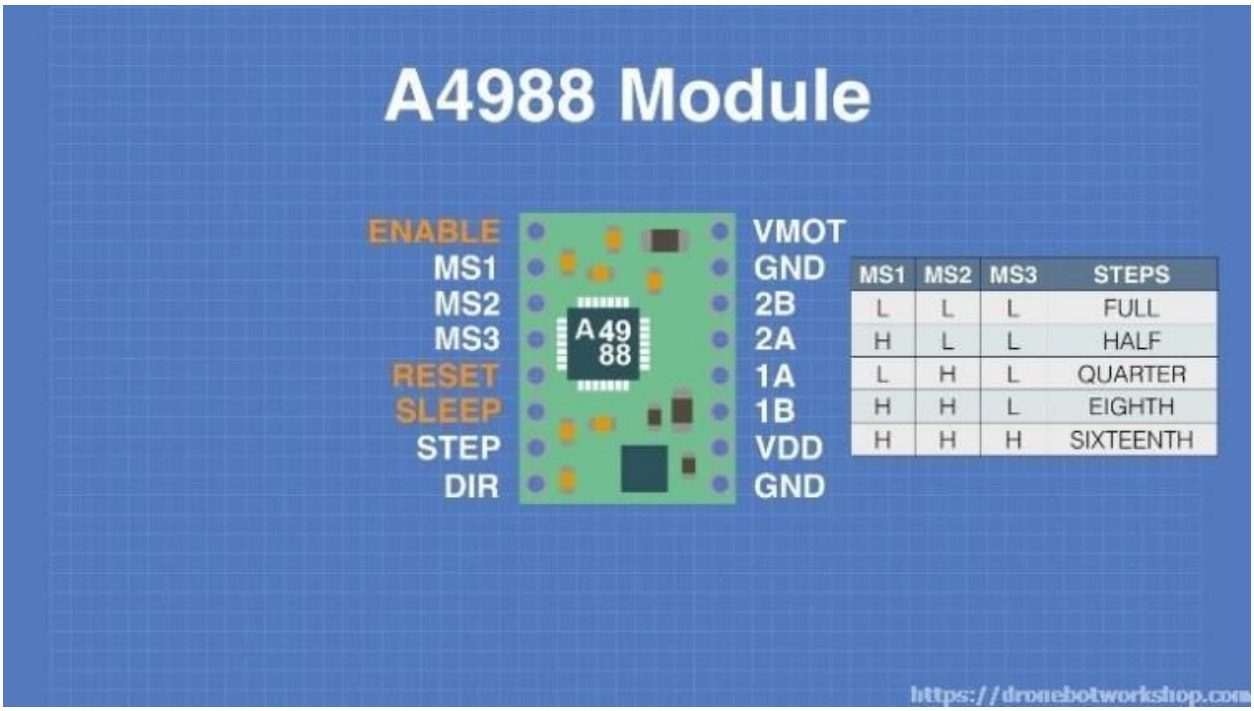
Demo 4 – Bipolar Stepper with A4988

The final demonstration will make use of a dedicated bipolar stepper motor controller called the A4988. Using a dedicated controller has many advantages:

- The logic for stepping the motor is all contained in the controller, freeing up the Arduino (or other microcontroller or microcomputer) to do other things.
- The number of connections to the microcontroller or microcomputer is reduced, making it much easier to control multiple stepper motors
- Doing advanced things like microstepping is easy, without using any special library or tying up computing resources
- You actually can control the motor without a microcontroller, a simple square wave oscillator can suffice in many situations.

The A4988 is a very common and inexpensive stepper motor controller that is used a lot in 3D printers and CNC machines where several stepper motors need to be managed. Other than the controller and motor it only requires one other part, a decoupling capacitor that is mounted physically close to the controller. With a heatsink the device can handle up to 2 amperes.

Let's take a look at the pinout of the A4988 module before we put it to use:



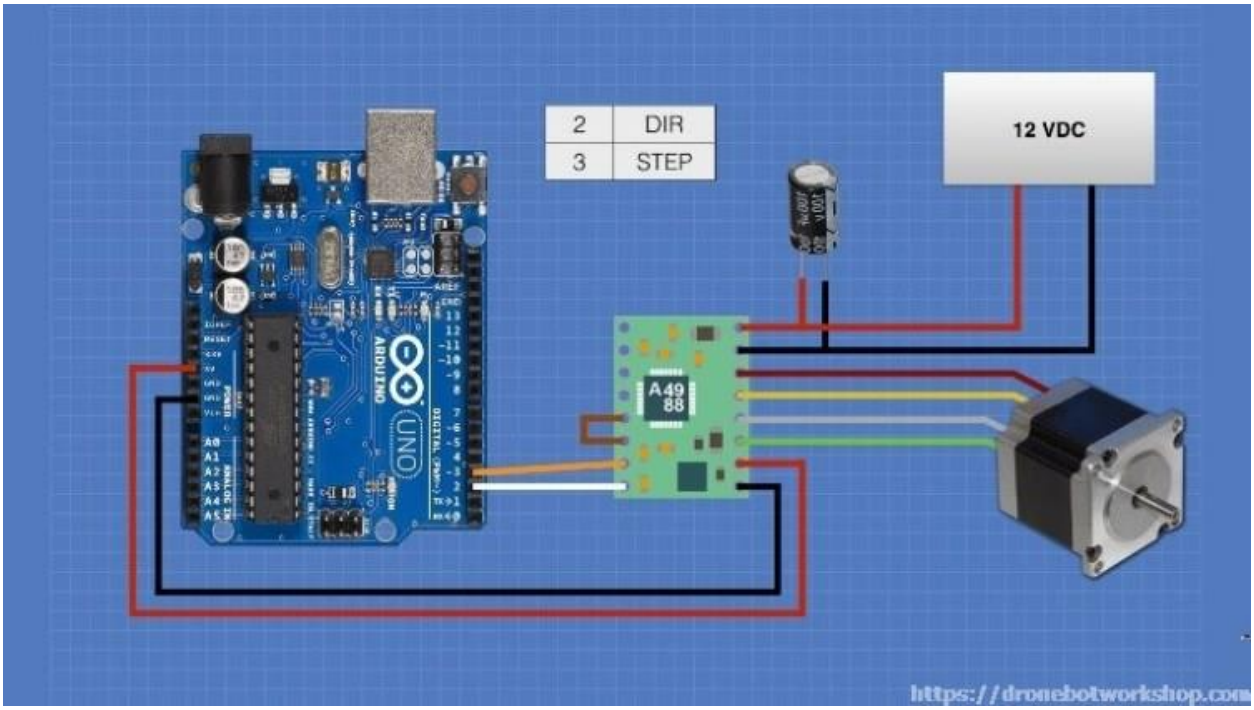
Starting from the top right and working down we see the following pins:

- **VMOT** – The motor DC supply voltage (positive). The maximum voltage is 35 volts.
- **GND** – The motor supply voltage ground.
- **2B, 2A** – The connections to coil 2 of the bipolar stepper motor.
- **1A, 1B** – The connections to coil 1 of the bipolar stepper motor.
- **VDD** – The logic supply DC voltage (positive) . This can range from 3 to 5.5 volts.
- **GND** – The logic supply ground.

Now looking down the other side of the A4988 module:

- **ENABLE** – This is an active low connection, when brought low (ground) the A4988 module is enabled. By default this is pulled low so the module is always enabled unless you apply a logic high here.
- **MS1, MS2, MS3** – These three connections determine the microstepping mode of the A4988 module. By setting the logic levels here you can set the motor to Full, Half, Quarter, Eighth or Sixteenth steps. See the chart on the connection diagram for details.
- **RESET** – This is an active low line that will reset the module. By default it is pulled high.
- **SLEEP** – If this line is set low the module will enter a low-powered sleep mode and consume minimal current. By tying this line to the Reset pin the module will always be on at full power consumption.
- **STEP** – This is how you drive the motor from an external microcontroller or square wave oscillator. Each pulse sent here steps the motor by whatever number of steps or microsteps that has been set by MS1, MS2 and MS3 settings. The faster you pulse this the faster the motor will travel.
- **DIR** – The direction control A high input here drives the motor clockwise, a low will drive it counterclockwise.

The key thing to note here is that the A4988 only requires two inputs from the Arduino to control the stepper motor and does not need the Arduino to “figure out” the stepping logic. This makes it a lot easier to control multiple stepper motors for advanced projects.
Now that we have seen how the A4988 is laid out we will connect one to our Arduino.



Note that there is an additional component, a 100uf capacitor, in this circuit. This is essential to decouple the power supply. Any value from 47uf up will suffice, try and mount the capacitor as close to the A4988 VMOT and GND pins as possible.
Otherwise the connections are quite simple. Make sure to observe the motor connections, the A4988 is conveniently laid out to match the 4-pin connector that is common on several bipolar motors but you should check your motor connections to be sure they are correct.

A4988 Current Adjustment

Before we load our sketch there is one thing that needs to be done. We need to set the current that flows through our motor coils using a small potentiometer on the A4988 module.
One method of doing this is to measure the voltage at a testpoint (labeled “+”) near the potentiometer while you adjust it . Use the following formula to derive the current:
I= Vref*2.5 – “Vref” is the voltage you measure an “I” is the current.
Another way is to tie the “STEP” input to high (5-volts) and place an ammeter in series with one of the motor coils. This Is the method I used in the video.
Once you have the motor current adjusted it’s time to load the sketch:

```
/*
  Stepper Motor Demonstration 4
  Stepper-Demo4.ino
  Demonstrates NEMA 17 Bipolar Stepper with A4988 Driver

  DroneBot Workshop 2018
  https://dronebotworkshop.com
*/

// Define Constants

// Connections to A4988
const int dirPin = 2;  // Direction
const int stepPin = 3; // Step

// Motor steps per rotation
const int STEPS_PER_REV = 200;

void setup() {

  // Setup the pins as Outputs
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
}
void loop() {

  // Set motor direction clockwise
  digitalWrite(dirPin,HIGH);

  // Spin motor one rotation slowly
  for(int x = 0; x < STEPS_PER_REV; x++) {
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(2000);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(2000);
  }

  // Pause for one second
  delay(1000);

  // Set motor direction counterclockwise
  digitalWrite(dirPin,LOW);

  // Spin motor two rotations quickly
  for(int x = 0; x < (STEPS_PER_REV * 2); x++) {
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(1000);
  }

  // Pause for one second
  delay(1000);
}
```

In this sketch we won't be using any stepper libraries as all we need to do is send a pulse out to the A4988 and let it do all the "heavy lifting".
We start by defining constants to represent the pins we have connected the A4988 STEP and DIR pins to. We also define STEPS_PER_REV as we did in the previous sketch, the number of steps our motor needs to complete one rotation. Again you should set this to match your stepper motor specifications.
In the setup we set our two defined A4988 pins as outputs.
Now to the loop. We will do two things here, spin the motor slowly clockwise one turn and then spin it counterclockwise two turns. We will insert a one second delay between each spin.
To set the direction of the motor we set the DIR pin either HIGH or LOW depending upon which way we want to go. A HIGH here will cause the motor to spin clockwise.
The speed is set by the frequency of the pulses we send on the STEP pin. The pulses are manually generated in a very similar fashion as the Arduino Blink sketch, by bringing the output HIGH, waiting a bit then Bringing it LOW and waiting again. This is repeated as many times as necessary to rotate our motor oin the amount we desire, one full rotation for the first routine and two rotations for the second one.
Of course you can add as many routines as you wish to make your motor move in the speed and direction you like.
As you can see the A4988 makes it very easy to drive a bipolar stepper motor with a minimum of code. You can also get a shield for your Arduino that allows you to drive multiple A4988 modules, which would be great if you are building a CNC machine or a 3D printer.

Conclusion

Hopefully this article and the accompanying video have shown you that stepper motors are not really that hard to work with after all. If you are designing a project that requires you to be able to position something precisely a stepper motor is an ideal choice.
Please let me know in the comments about any problems or observations you encounter using stepper motors. I'd really love to hear how you incorporate them into your own designs.
Now get out there and start building with stepper motors!