CLIL PARTE 1

STEPPER MOTOR

There are several types of electric motors, each with their own advantages and disadvantages.
If you're needing a motor to provide accurate positioning, your options are somewhat limited.
You may consider servo motors, which typically only have 180° of rotation.

Another option may be some sort of DC motor with feedback via an encoder or even a Hall effect sensor.
However, today we'll evaluate the pros and cons of the almighty hybrid stepper motor and see how it works.

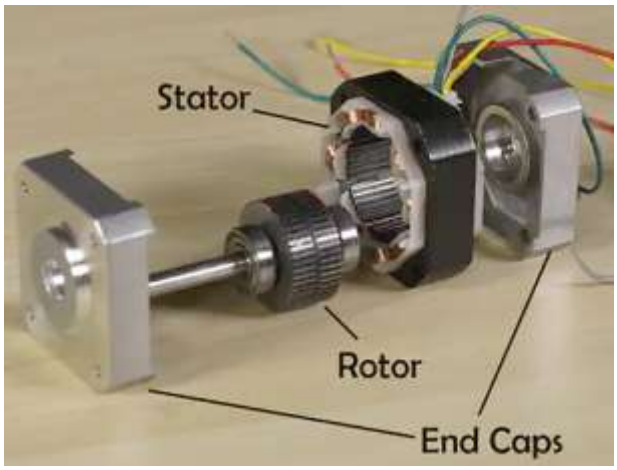
A stepper motor literally takes small steps to advance its rotation. A common configuration for stepper motors is 200 steps per revolution, but you'll likely see motors with 400 or even more steps per revolution.

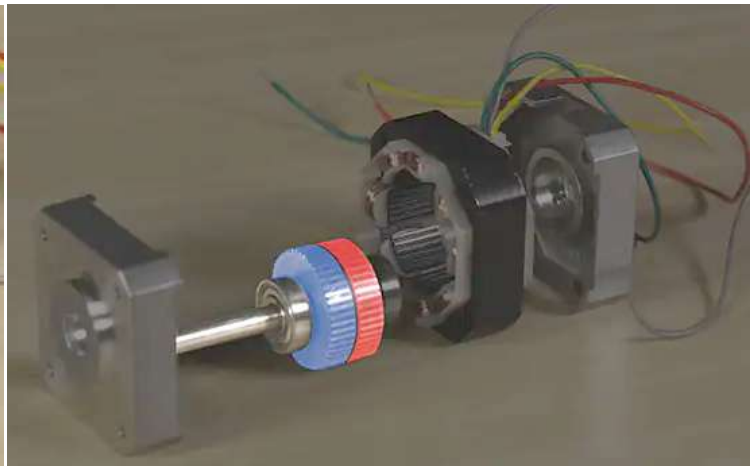
If you divide 360° by the number of steps per revolution, you'll find the number of degrees each step will rotate.
For example, a 200 step motor will rotate 1.8° per step. Stepper motors may be listed for sale by this figure rather than the number of steps/revolution.

Stepper motors are also able to prevent rotation by holding their position.
If you've seen stepper motors in action, you may have noticed they can get warm to the touch even while not rotating. This is because the motor is likely being held in position which does consume some power.

Being able to hold position and rotate precisely are the two main advantages of stepper motors, which is why they're found in devices such as CNC machines and 3D printers.

Next, let's take a look inside a stepper motor.



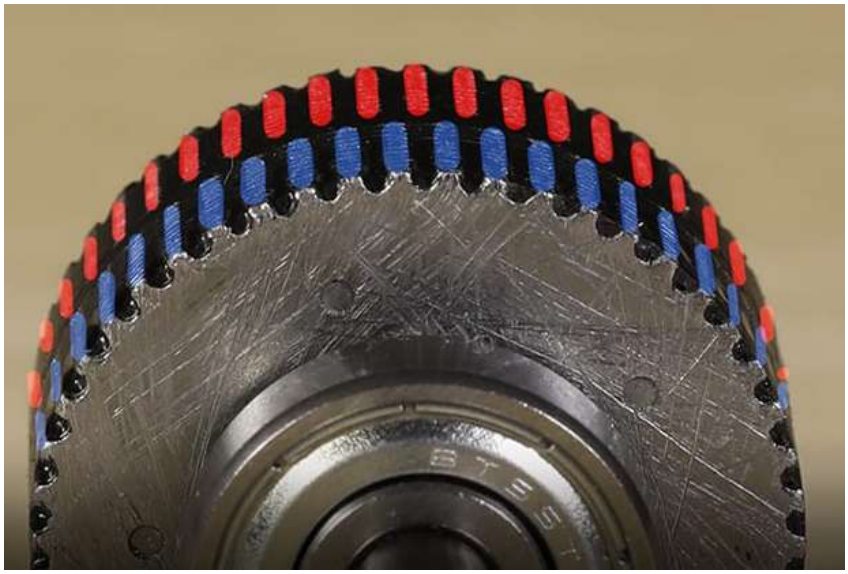


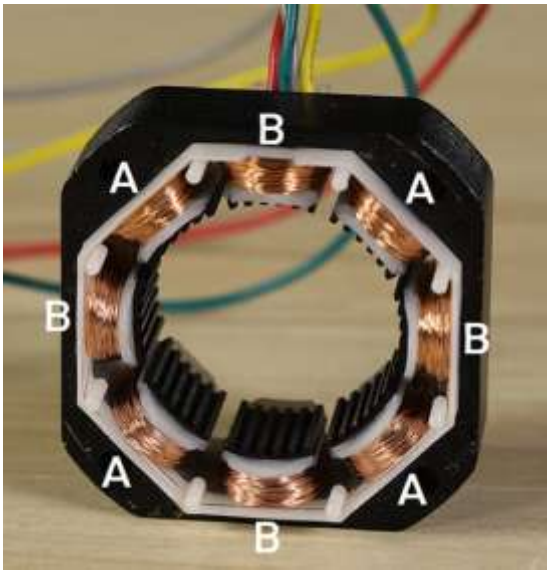
Inside the motor, we'll see two main components: the rotor and the stator.

As you may have guessed, the rotor is the part of the motor that rotates, while the stator is stationary and attached to the motor housing.

The rotor assembly consists of a permanent magnet rotor and the motor shaft.
Like all magnets, the rotor has a north and south pole.

Between these poles the rotor is divided into two halves called rotor cups; on each side of this division there are teeth that alternate. North teeth line up with the troughs between the South teeth.





There are teeth in the motor's stator as well. Looking at the stator more closely, we'll see that it consists of eight electromagnetic coils.
These eight coils are divided between two electrical phases, typically named phase A and B.

The image shows how the phases are divided within the stator.

Normally each phase will have two wires exiting the motor, some motors will have an additional center tap wire on each phase.

Energizing a phase of the motor will cause the rotor to turn to the next "step".

The teeth of the energized stator poles will attract rotor teeth of opposing polarity. The rotor will be held in this position until power is removed.

To rotate the motor continuously, the phases need to be energized in a particular order: typically +A, +B, -A, -B.

It's impractical to make these changes manually, which is why stepper motors are usually accompanied by a microcontroller and a stepper motor driver IC (integrated circuit) or driver board.

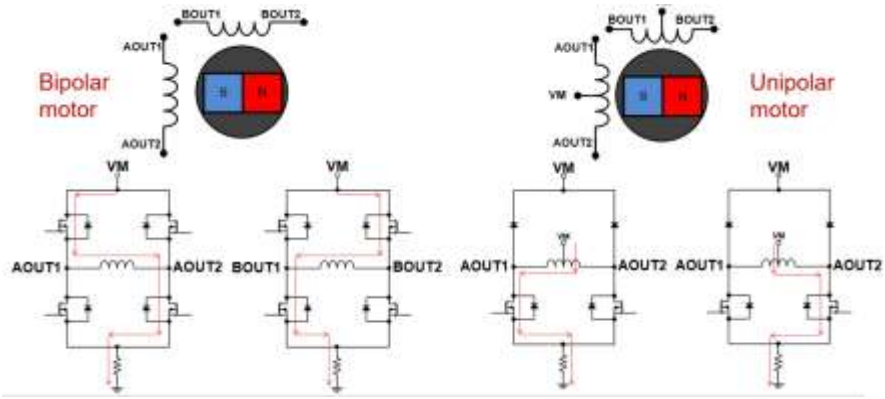
Driving a motor in the way described above is called "wave stepping".

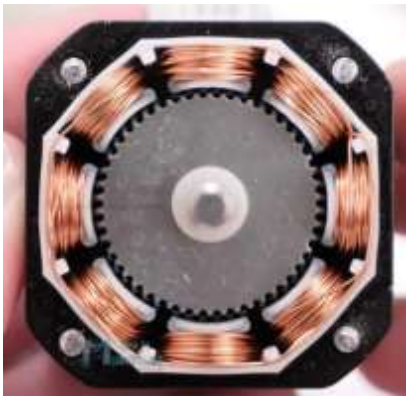
There are several different methods of driving stepper motors. Most commonly found are full stepping, which delivers the highest torque, and many degrees of microstepping, which allow for smoother operation.

Stepper motors have revolutionized automation, enabling precision operations even without sensors for feedback.

Types of stepper motor

Broadly speaking there are two types of stepper motor - unipolar and bipolar.

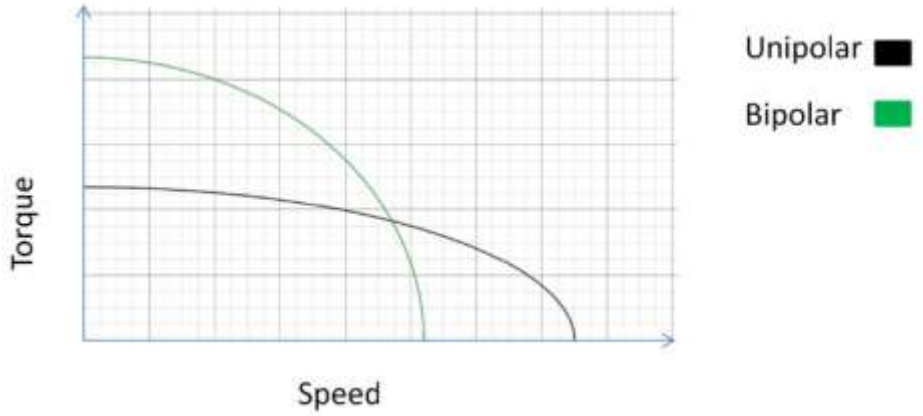




The main difference between “unipolar” and “bipolar” stepper motors is the availability of the center-tap wire, which splits the full coils of the winding in half.
This splitting can be done with one connection wire for the pair or two wires (one for the adjacent ends of each coil). By deleting the center tap, the unipolar connection becomes a bipolar-series connection.

Motor-lead colors are somewhat standardized in the industry and are fully consistent within the product line of an individual vendor, so many wiring diagrams show color rather than numbering the leads.

However, there is more to the situation than just choosing unipolar or bipolar configuration. It also means a change in the electrical characteristics of the windings inside the motor and thus affects voltage, resistance, and inductance, velocity, acceleration, and torque characteristics (see Figure).

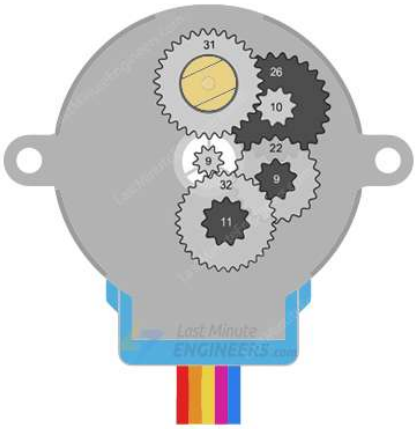


Unipolar and bipolar pole arrangements have different speed versus torque rolloff curves

Bipolar motors have 4 wires connecting to the two separate coils inside the motor - one pair for each coil.
There are also two types of unipolar motor, those with 5 wires and those with 6 wires.

The 6-wire motors can also be referred to as hybrid motors.
They are similar to the 4-wire bipolar motors and just have an extra wire connected to the centre of each of the coils. If you want to use a 6-wire motor in bipolar mode just ignore the wires that connect to the centres of the coils.

The 5-wire motors cannot be driven by a driver designed for a bipolar motor.
An example of a 5-wire motor is the small **28BYJ-48** motor which can be seen in many Arduino projects and usually uses a **ULN2003** chip as its driver.



Gear Ratios:

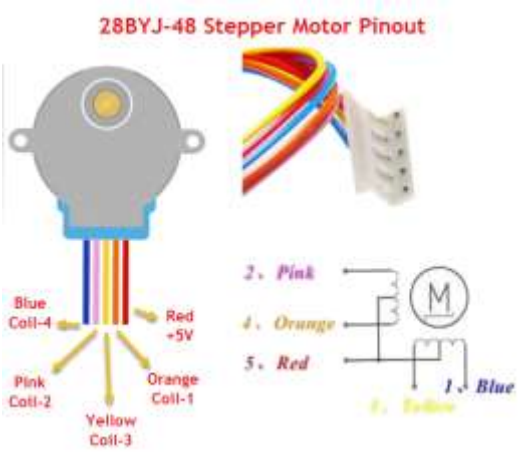
- 32 / 9
- 22 / 11
- 26 / 9
- 31 / 10

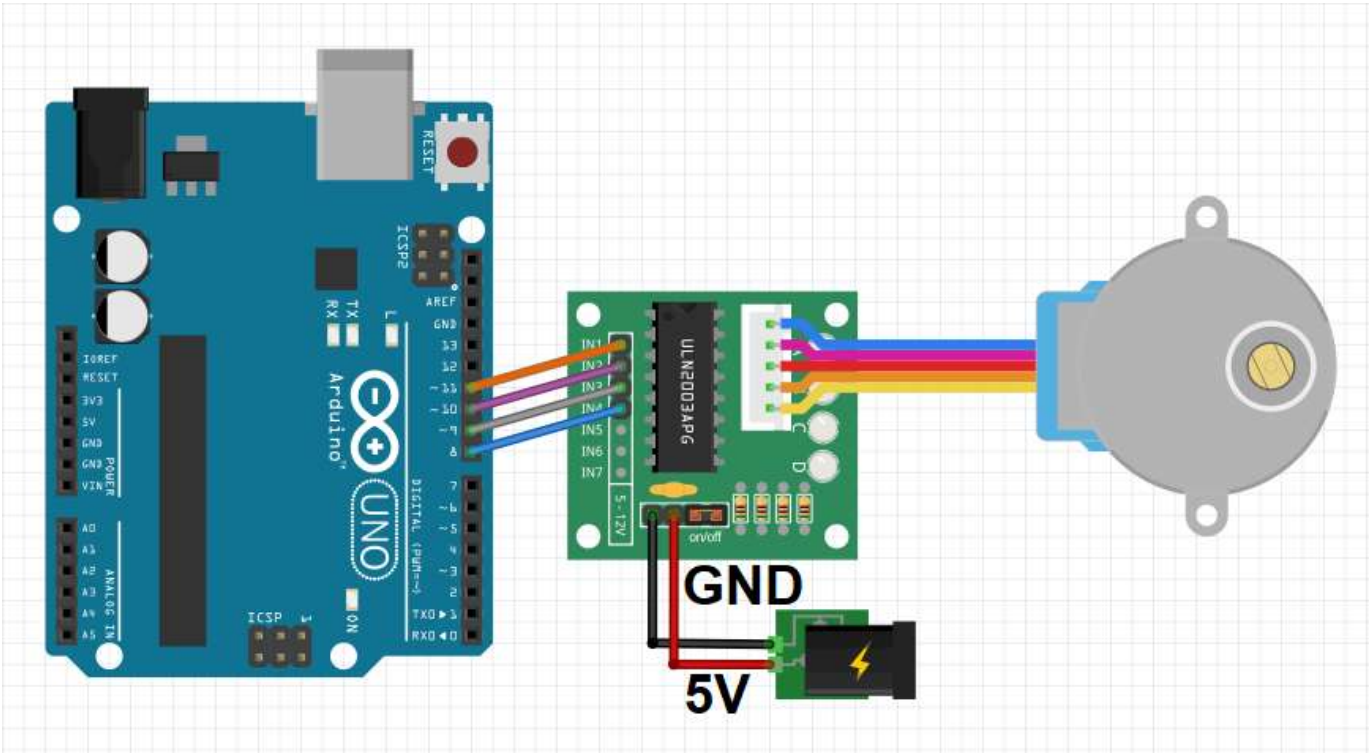
Multiplying the gear ratios:

$$\frac{32}{9} \times \frac{22}{11} \times \frac{26}{9} \times \frac{31}{10} = 63.68395$$

Round 63.68395 up: 64

This gives us a 64:1 gear ratio over all





Move from A to B by a specific number of steps

We will define the speed and acceleration at which we want the stepper motor to rotate and then make it rotate by a specified number of steps, stop, then rotate in the opposite direction in a specified number of steps.

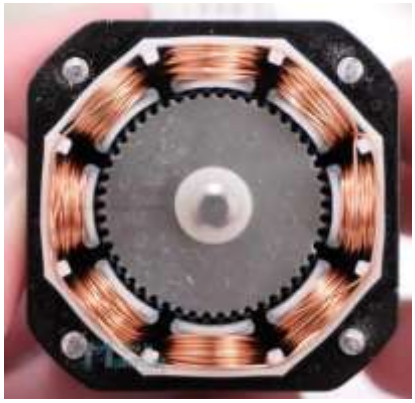
Specifically, we will do 3 rotations in one direction (meaning total steps will be the steps per revolution multiplied by 3 revolutions), stop and complete 3 rotations in the opposite direction.

CODE:

```
//Arduino Code - Move from A to B by a specific number of steps with set speed and acceleration

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

// Define the motor pins:
#define MP1  8 // IN1 on the ULN2003
#define MP2  9 // IN2 on the ULN2003
#define MP3  10 // IN3 on the ULN2003
#define MP4  11 // IN4 on the ULN2003

#define MotorInterfaceType 8 // Define the interface type as 8 = 4 wires * step factor (2 for half step)

//Define the pin sequence (IN1-IN3-IN2-IN4)
AccelStepper stepper = AccelStepper(MotorInterfaceType, MP1, MP3, MP2, MP4);

const int SPR = 2048;//Steps per revolution

void setup() {
  stepper.setMaxSpeed(1000);   //Set the maximum motor speed in steps per second
  stepper.setAcceleration(200); //Set the maximum acceleration in steps per second^2
}

void loop() {
  stepper.moveTo(3*SPR); //Set the target motor position (i.e. turn motor for 3 full revolutions)
  stepper.runToPosition(); // Run the motor to the target position

  delay(1000);

  stepper.moveTo(-3*SPR);//Same as above: Set the target motor position (i.e. turn motor for 3 full revolutions)
  stepper.runToPosition(); // Run the motor to the target position

  delay(1000);
}
```

Motor Specifications

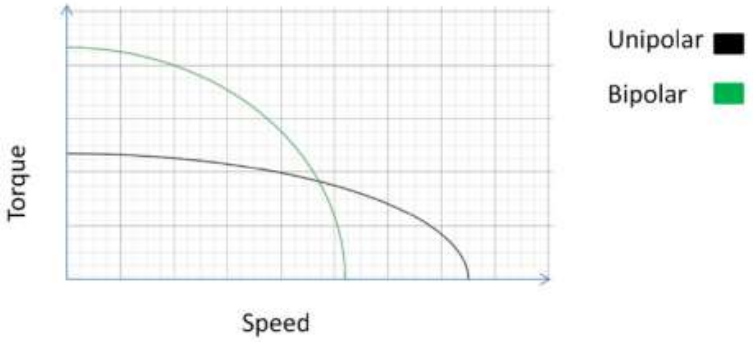
Datasheets normally quote the coil current, coil resistance, nominal voltage and holding torque and steps per revolution. For example, for this motor the values are 1 Amp, 2.7 Ohms, 2.7volts, 1.4Kg-cm and 200 steps/rev.



The nominal voltage is irrelevant for all practical purposes. The important figure is the rated current.

The rated current is normally the current per-coil and when currents are quoted for stepper motor driver boards that is normally also a per-coil figure.

The holding torque is the torque available to resist rotation while the motor is stationary. The available torque will decline as speed increases.
Some manufactures provide graphs showing how the torque varies with speed.



Operating Voltage

Stepper motors are very different from regular DC motors.

With a DC motor you control the current in order to control the speed of the motor.
The usual way to control the current is to vary the voltage - perhaps using the Arduino analogWrite() function to control a ***Pulse Width Modulated*** power supply to the motor.

Stepper motors pretty much draw their full current all the time, even when they are stationary - that is how they resist being moved from their present position. This means they are very inefficient.

For all practical purposes the nominal voltage of a stepper motor is irrelevant.

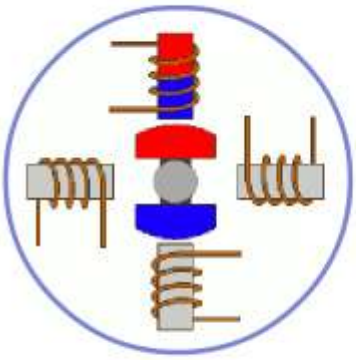
It is the voltage which would drive the rated current through the coil when the motor is stationary based on Ohms law e.g. 2.7v = 1A * 2.7 Ohms.

However, as soon as the motor starts moving the combination of the inductance of the coils and the back-emf generated by the movement will prevent the nominal voltage from producing the rated current.

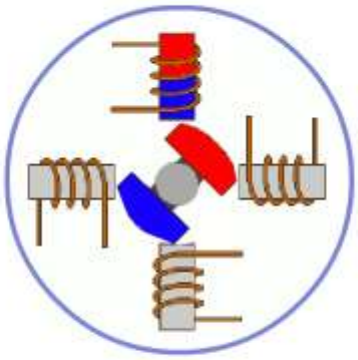
For this reason stepper motors are normally driven with a much higher voltage.
This, in turn, means that a specialized stepper motor driver board is needed which can limit the current to whatever the motor can take. If the current is not limited the high voltage would quickly destroy the motor.

Stepper motor STEPS

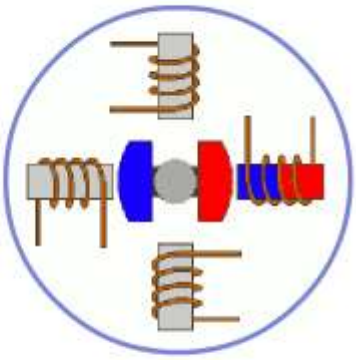
Each of those rotations is called a "step", with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle.



To make the motor shaft turn, first, one electromagnet is given power, which magnetically attracts the gear's teeth.



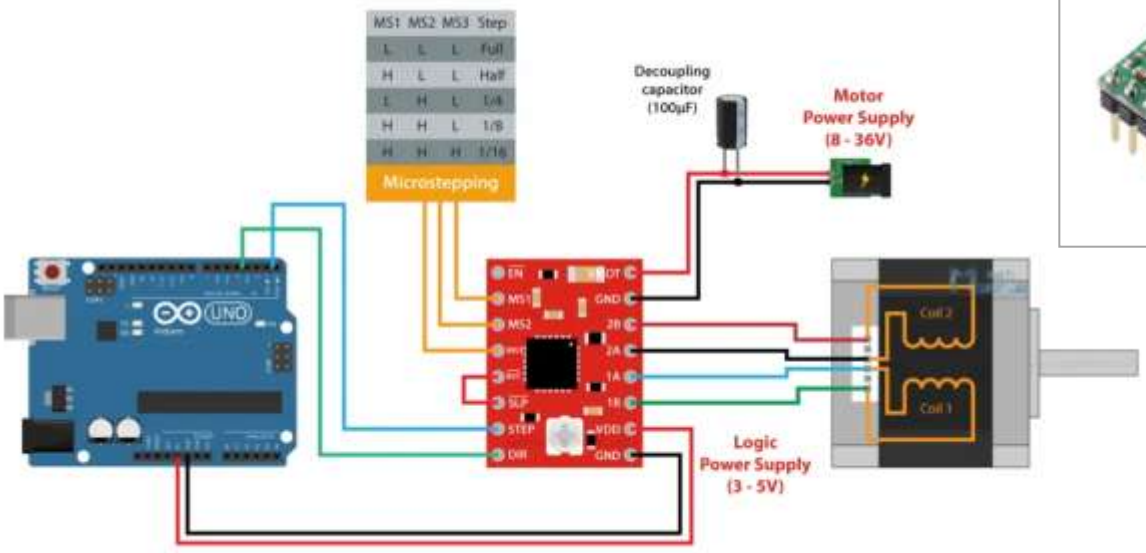
When the gear's teeth are aligned to the first electromagnet, they are slightly offset from the next electromagnet.



So when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated.

Stepper Motor Driver Boards

These are specialized components designed to control stepper motors conveniently and efficiently.
The Pololu A4988 is a typical example that is often used with Arduino.



Generally speaking specialized stepper motor driver boards only require two connections (plus GND) to the Arduino for step and direction signals.
Normally specialized stepper motor driver boards have the ability to ***limit the current*** in the motor which allows them to drive the motor with a high voltage (up to 35v for the Pololu A4988) for better high speed performance.

Each stepper motor has a maximum current (supplied by the manufacturer) which must not be exceeded. Otherwise, the engine will heat up and be damaged!

All drivers, usually, have the ability to do microstepping: *microstepping is a method of controlling stepper motors, typically used to achieve higher resolution or smoother motion at low speeds.*

The Pololu A4988 can do 1/2, 1/4, 1/8 and 1/16 microsteps. It defaults to full steps: 200 full steps per revolution.

Tuning motor current with A4888

For the A4988 the calculation for the maximum trip current is:

→ I_max= Vref/(8*Rs)

with

- ***Vref****= reference voltage we set on driver*
- ***Rs****= sensing resistors on driver*

With official “Pololus A4988 driver”, the sensing resistors are Rs=0.05 ohm, so a Vref of 0.4 should produce a maximum current

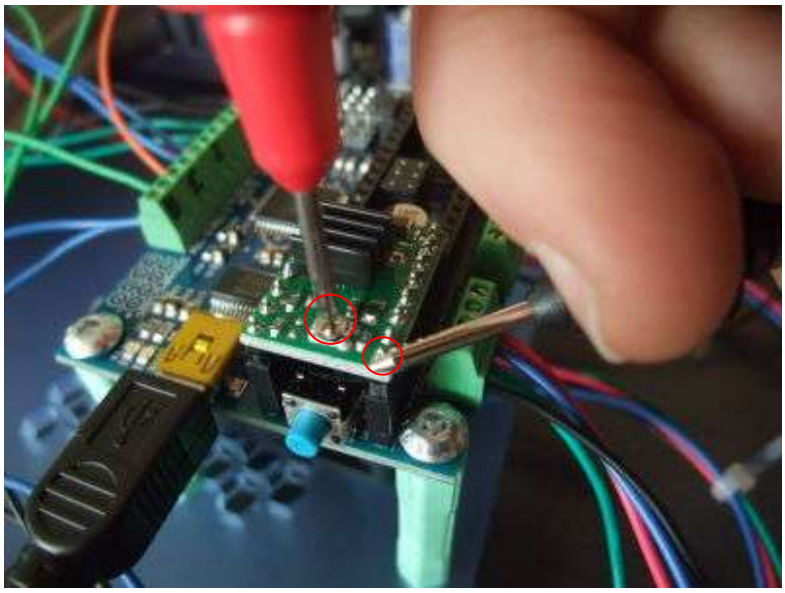
I_max= 0.4/(8*0.05)=1A.

Clockwise increases the current which will make the motor run hotter and counterclockwise reduces it which will cool it down.

As another example, aiming for 50% temperature rise on 1A rated steppers by using max 0.7A, so rearrange it as:

Vref = I_max * 8 * Rs → Vref = 0.7A * 8 * 0.05 = 0.280V

The Vref signal is accessible as the "VREF" pin on the carriers with voltage regulators, as the through-hole via on the carriers without, and also as the wiper on the trim pot itself on both carriers.

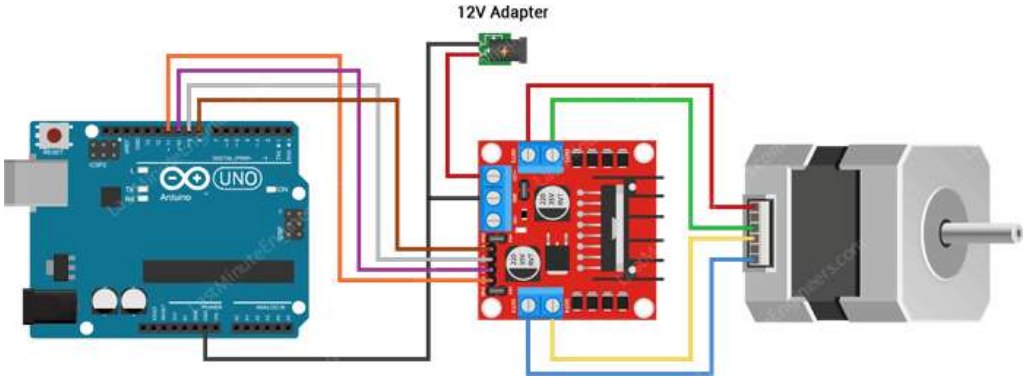


example of a v-ref checking, + probe on the turnpot and - on a ground pin

Note: most "Made in China" A4988 Pololu driver “ knock-offs have Rs=0.1 ohm.

H-bridge driver - e.g. LN298

These can be made to control a stepper motor but they are a very poor choice - mainly because they have no method for limiting the current and therefore cannot use high voltages.
They are also more trouble to connect to an Arduino (they require more pins) and more trouble to control with an Arduino (more calculations for the Arduino to do).



The L298N Motor Driver Board is built around the L298 dual full-bridge driver, made by STMicroelectronics.

With this motor driver you can control DC motors, stepper motors, relays, and solenoids. It comes with two separate channels, called A and B, that you can use to drive 2 DC motors, or 1 stepper motor when combined.

The L298N is usually mounted on a (red) breakout board, which makes wiring a lot easier. The breakout board also includes a 78M05 5 V power regulator.

Why is my stepper motor getting HOT?

One thing that is very important to remember is that the L298 does not have an easy way to set a current limit unlike other stepper motor drivers like the A4988. This means that the current draw depends on the relationship between the inductance and resistance (L/R) of the stepper motor that you connect to it.

When the motor draws too much current, you can damage the driver and the motor will get hot!

What this means for you, is that you need to be careful when selecting the stepper motor and power supply to use with this motor driver. Not all stepper motors will work! The L298N operating voltage is between 4.8 and 46 volts (max 35 V when mounted on the breakout board).
Since the driver can supply a maximum of 2 amperes per channel, you need to find a stepper motor that can be used in this voltage range and doesn't exceed the maximum current rating.

Check the datasheet of your stepper motor and look for the voltage/current draw of the motor.
If you can't find the datasheet, you can measure the resistance of one of the windings and use the following formula to get an estimation of the current draw:

I = U ÷ R or Current draw (A) = Supply voltage (V) ÷ Winding resistance (Ω)

I would try to find a motor that draws less than 2 A at the voltage that you want to use.

Choosing a motor and motor driver

First choose the motor

The important specification is the torque of the motor.
For example for a small NEMA 17 it is 0.59 Nm.
The available useful torque will decline as the speed increases and at no-load maximum speed it will be zero.
Some (probably the more expensive) motor manufactures provide graphs showing how the torque varies with speed.

To figure out what motor you need you will have to measure or estimate the torque required.
It would be a good idea to choose a motor with a good margin of surplus torque.

It is not too difficult to make a rough measurement of the torque required but it is beyond the scope of this note.

Then choose the stepper motor driver

When you have selected a motor and know what current it requires you can choose a stepper motor driver that can comfortably supply the required current.

You should be aware that the economical single-chip stepper drivers (such as the A4988 and the DRV8825) can only supply about 2 amps. If your motor requires more than that, you will need to get one of the more expensive commercial stepper drivers. However the working principle will be practically identical to the A4988.

NEMA 17 – 23 – 34 – 42

These standards only define the size of the front face of the motor and the location and size of the mounting screw holes. They say nothing about the power of the motor. The 17 is an abbreviation of 1.7 inches.

Microsteps

Most (but certainly not all) stepper motors do 200 full steps per revolution.
By appropriately managing the current in the coils it is possible to make the motor move in smaller steps.
The Pololu A4988 can make the motor move in 1/16th steps per revolution.

It defaults to full steps: 200 full steps per revolution.

The main advantage of microstepping is to reduce the roughness of the motion.
The only fully accurate positions are the full-step positions.
The motor will not be able to hold a stationary position at one of the intermediate positions with the same position accuracy or with the same holding torque as at the full step positions.

Generally speaking when high speeds are required full steps should be used.

It is possible with most drivers including the Pololu A4988 to use the Arduino program to change the microstep setting.

Stepper Motor Speed

By comparison with regular DC motors stepper motors are very slow devices.

Typical speeds might be 1000 to 4000 steps per second and for a 200 step motor that would represent 5 to 20 rps (300 to 1200 rpm).

Generally speaking the motors with low coil resistance and high currents (and low nominal voltages) will be most suitable for higher speeds. A high voltage will also be needed for high speed.

Acceleration

If the stepper motor is required to move a heavy load it will normally be necessary to start the movement slowly (as with any motor) and accelerate to the desired speed and, equally, to decelerate when it is necessary to stop.

This is quite different from a DC motor which will accelerate and decelerate automatically.

If you try to start or stop a stepper motor too quickly it will simply skip steps with no damage to motor.
However The Arduino has no means to know whether or how many steps have been missed and all of the position control will be lost.

For this reason, in particular, it is essential to choose a motor with sufficient torque for the job and to use acceleration and deceleration when necessary.

Position Feedback

Stepper motors do not have the ability to tell the Arduino what position they are at, nor do they have the ability (like a servo) to go to a particular position. All they can do is move N steps from where they are now.

If it is essential to have position feedback a rotary encoder can be attached to the motor shaft - but that is beyond the scope of this notes.

Initial Position

When it starts up the Arduino has no means of knowing where the stepper motor is positioned - for example somebody might have moved it manually when the power was off.

The usual way to establish a datum for counting steps is with a limit switch.
At startup the Arduino will move the motor until it triggers the switch.
The Arduino will then regard that step position as step zero for the purpose of future position keeping.

Stepper Motor Applications

Step motors are used every day in both industrial and commercial applications because of their low cost, high reliability, high torque at low speeds. For making more advanced technologies we use stepper motors rather than simple motors.

Computer-controlled stepper motors are a type of motion-control positioning system.

Step Motors are typically digitally controlled motors, for use in holding or precise positioning applications.

These motors are used at

- -Industrial Machines : CNC Machines, Milling Machines, Laser Cutters etc,
- -Computer Technology: CD-Roms, DVD Players, Floppy Disk Drives, Scanners etc.
- -Printing : Printers, Plotters, 3D Printers etc.
- -Intelligent Lighting Systems : Lasers, Optical Devices, Mirror Mounts

Arduino Libraries

When using an Arduino with a specialized stepper motor driver board such as the Pololu A4988 there is little to be gained from using an Arduino library unless you need the acceleration feature of the AccelStepper library.

The AccelStepper library provides an object-oriented interface for 2, 3 or 4 pin stepper motors and motor drivers.

The standard Arduino IDE includes the Stepper library (http://arduino.cc/en/Reference/Stepper) for stepper motors. It is perfectly adequate for simple, single motor applications.

AccelStepper significantly improves on the standard Arduino Stepper library in several ways:

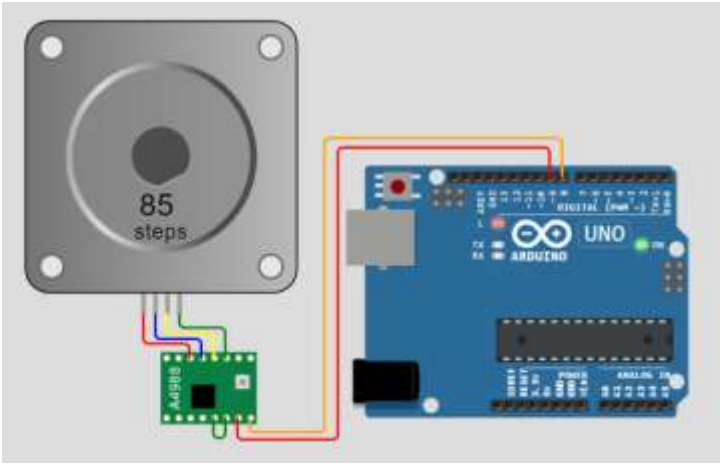
- Supports acceleration and deceleration
- Supports multiple simultaneous steppers, with independent concurrent stepping on each stepper
- Most API functions never delay() or block (unless otherwise stated)
- Supports 2, 3 and 4 wire steppers, plus 3 and 4 wire half steppers.
- Supports stepper drivers such as the Sparkfun EasyDriver (based on 3967 driver chip)
- Very slow speeds are supported

Arduino code

The code is intended as a first step to getting your motor working.
It also shows how easy it is to control a motor without a library when a specialized stepper motor driver such as the Pololu A4988 is used.

Simulate the circuit in wokwi.

SAMPLE 1




```
// test a stepper motor with a Pololu A4988 driver board
// onboard led will flash at each step
// this version uses delay() to manage timing

byte directionPin = 8;
byte stepPin = 9;
int numberOfSteps = 200;
byte ledPin = 13;
int pulseWidthMicros = 20;  // microseconds
int millisbetweenSteps = 20; // milliseconds - or try 100
for slower steps


void setup() {

  Serial.begin(9600);
  Serial.println("Starting StepperTest");
  digitalWrite(ledPin, LOW);

  delay(2000);

  pinMode(directionPin, OUTPUT);
  pinMode(stepPin, OUTPUT);
  pinMode(ledPin, OUTPUT);


  digitalWrite(directionPin, HIGH);
  for(int n = 0; n < numberOfSteps; n++) {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(pulseWidthMicros);
    digitalWrite(stepPin, LOW);

    delay(millisbetweenSteps);

    digitalWrite(ledPin, !digitalRead(ledPin));
  }

  delay(3000);


  digitalWrite(directionPin, LOW);
  for(int n = 0; n < numberOfSteps; n++) {
    digitalWrite(stepPin, HIGH);
    // delayMicroseconds(pulseWidthMicros); // probably not needed
    digitalWrite(stepPin, LOW);

    delay(millisbetweenSteps);

    digitalWrite(ledPin, !digitalRead(ledPin));
  }
}

void loop() {
}
```

CLIL PARTE 2

STEPPER MOTOR

Read the attached code and then create the circuit capable of realizing it in the wokwi simulator.

SAMPLE2

```
// test a stepper motor with a Pololu A4988 driver board or equivalent
// this version uses millis() to manage timing rather than delay()
// and the movement is determined by a pair of momentary push switches (push button)
// press one and it turns CW, press the other and it turns CCW

byte directionPin = 9;
byte stepPin = 8;
byte buttonCWpin = 10;
byte buttonCCWpin = 11;
boolean buttonCWpressed = false;
boolean buttonCCWpressed = false;
byte ledPin = 13;

unsigned long curMillis;
unsigned long prevStepMillis = 0;
unsigned long millisBetweenSteps = 25; // milliseconds

void setup() {
     Serial.begin(9600);
     Serial.println("Starting Stepper Demo with millis()");
     pinMode(directionPin, OUTPUT);
     pinMode(stepPin, OUTPUT);
     pinMode(ledPin, OUTPUT);
     pinMode(buttonCWpin, INPUT_PULLUP);
     pinMode(buttonCCWpin, INPUT_PULLUP);
}

void loop() {
    curMillis = millis();
    readButtons();
    actOnButtons();
}

void readButtons() {
    buttonCCWpressed = false;
    buttonCWpressed = false;

    if (digitalRead(buttonCWpin) == LOW) {
        buttonCWpressed = true;
    }
    if (digitalRead(buttonCCWpin) == LOW) {
        buttonCCWpressed = true;
    }
}

void actOnButtons() {
    if (buttonCWpressed == true) {
        digitalWrite(directionPin, LOW);
        singleStep();
    }
    if (buttonCCWpressed == true) {
        digitalWrite(directionPin, HIGH);
        singleStep();
    }
}

void singleStep() {
    if (curMillis - prevStepMillis >= millisBetweenSteps) {
            // next 2 lines changed 28 Nov 2018
        //prevStepMillis += millisBetweenSteps;
        prevStepMillis = curMillis;
        digitalWrite(stepPin, HIGH);
        digitalWrite(stepPin, LOW);
    }
}
```

Briefly answer the following questions

1. What are the main differences between bipolar and unipolar steppers?

2. What type of stepper motor provides the most torque at the same speed?

3. What type of engine is the 28BYJ-48?

4. How much current does a small stepper motor typically consume?

5. What is the typical working voltage of stepper motors?

6. What is a "driver"?

7. What is meant by "microstepping"?

8. How do you choose a stepper motor?

9. What does the abbreviation NEMA 23 mean?

10. With a 1/16 microsteps what angular precision is obtained?

11. What is the maximum speed of a stepper motor?

12. What torque does a small NEMA 17 motor typically have?

13. What does the Arduino code in “sample 1” do?

VIDEO ABOUT STEPPER MOTOR

https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/

https://howtomechatronics.com/tutorials/arduino/stepper-motors-and-arduino-the-ultimate-guide/