Introduction to robotics
10th lab

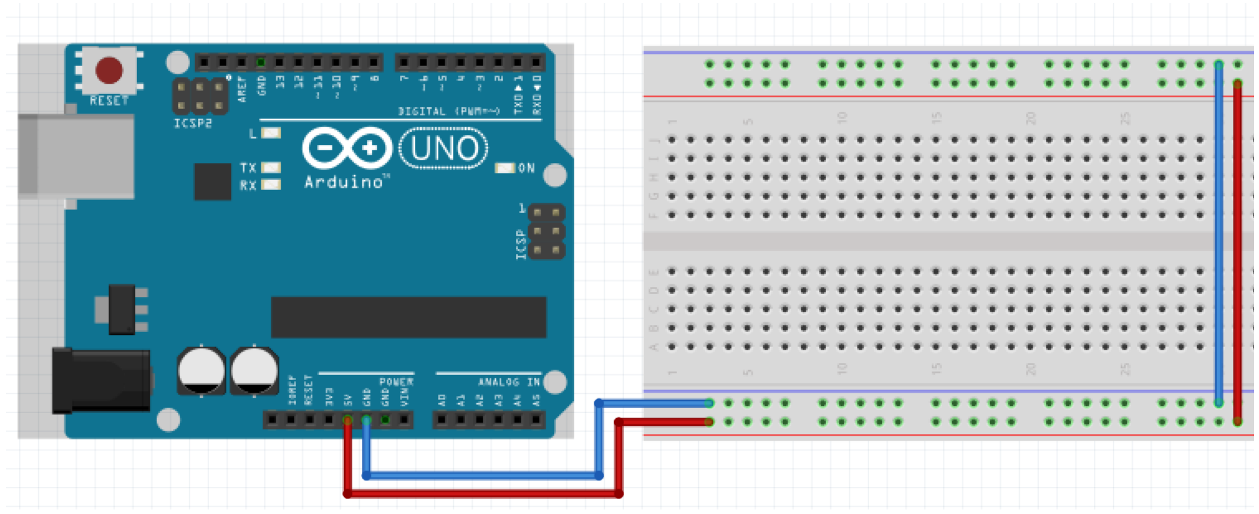
Remember, when possible, choose the wire color accordingly:

- **BLACK** for **GND (dark colors if not available)**
- **RED** for **POWER (3.3V / 5V / VIN) (bright colors if not available)**
- **Bright Colored** for read and write signal (use **red** when none available and **black** only as a last option)
- We know it is not always possible to respect this due to lack of wires, but the first rule is **DO NOT USE BLACK FOR POWER OR RED FOR GND!**

Now, let's pick it up where we left off…

Pull out your Arduino and breadboard and connect them like in the schematic. This is to "power up" the breadboard so we can easily have access to **5V** and **GND**.

Attention! Remember how the breadboard works. Use correct wire colors.

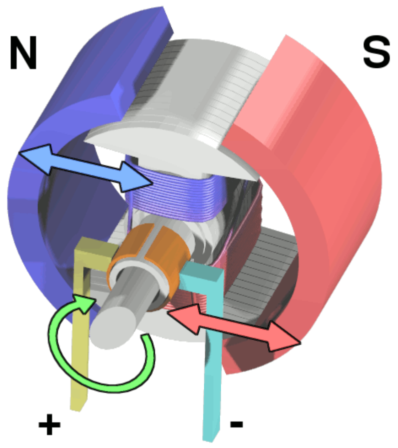


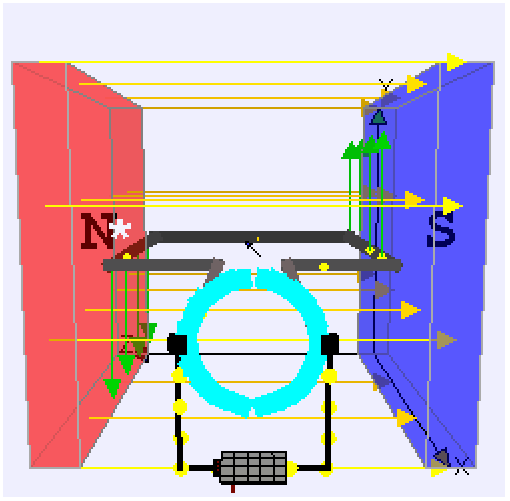
0. Homework check

1. DC Motor

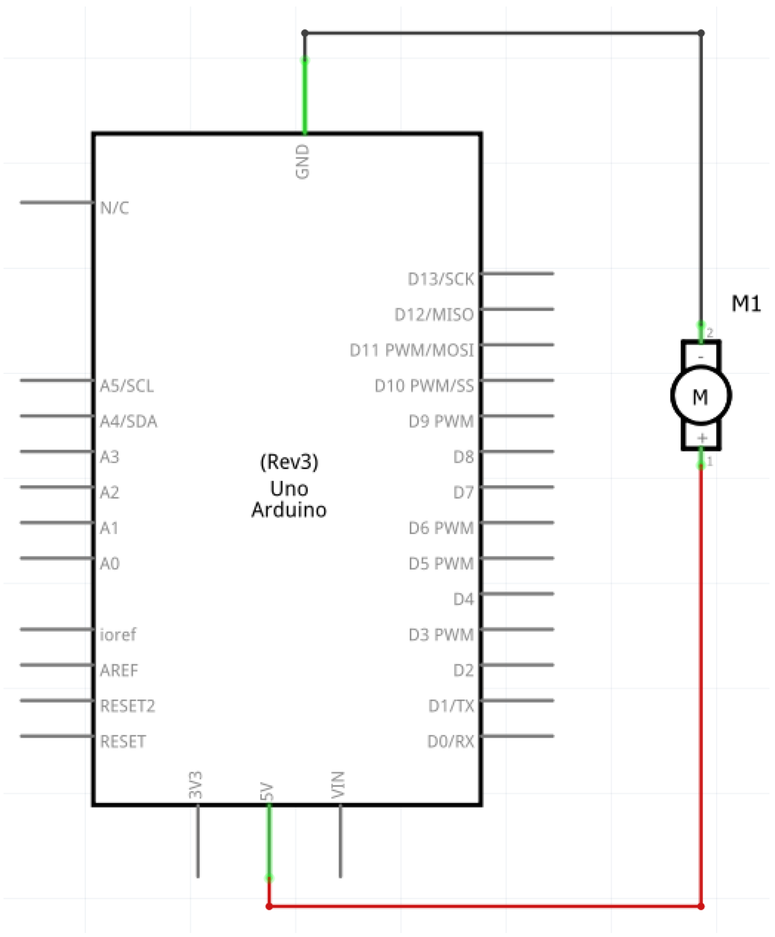
A **DC motor** is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

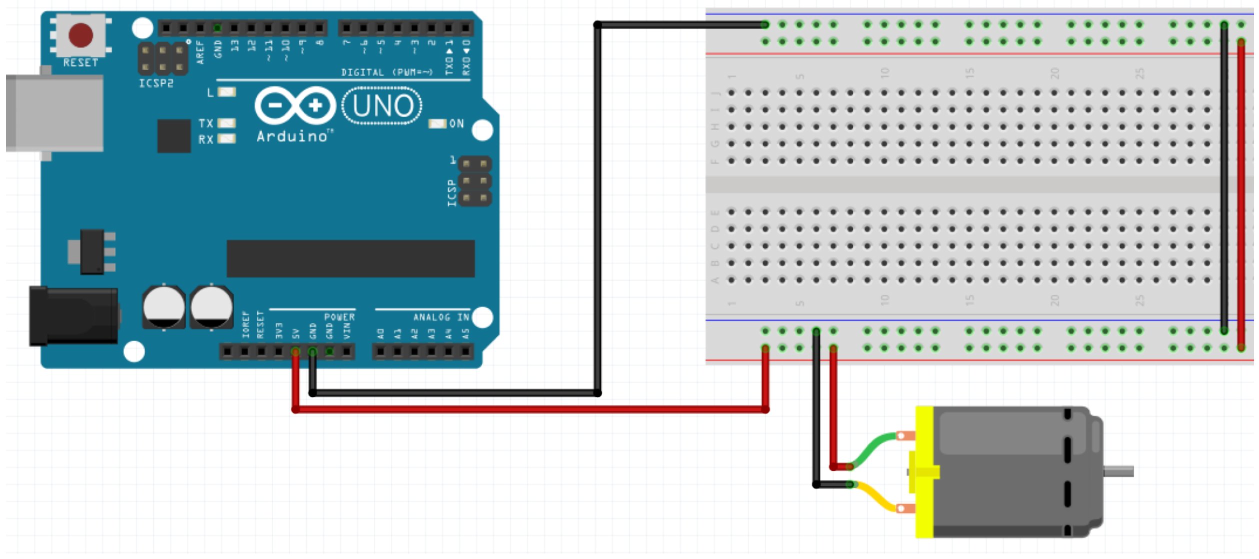
(source: https://en.wikipedia.org/wiki/DC_motor)





Before we get an Arduino board to control the motor, let's experiment with it a bit: connect the 2 motor's pins to **5V** and **GND**, respectively.





Note which way the motor is spinning. You can do this by pinching the motor shaft between your fingers. Swap over the motor leads so that the motor lead that was going to **5V** now goes to **GND** and vice-versa. The motor will turn in the opposite direction.

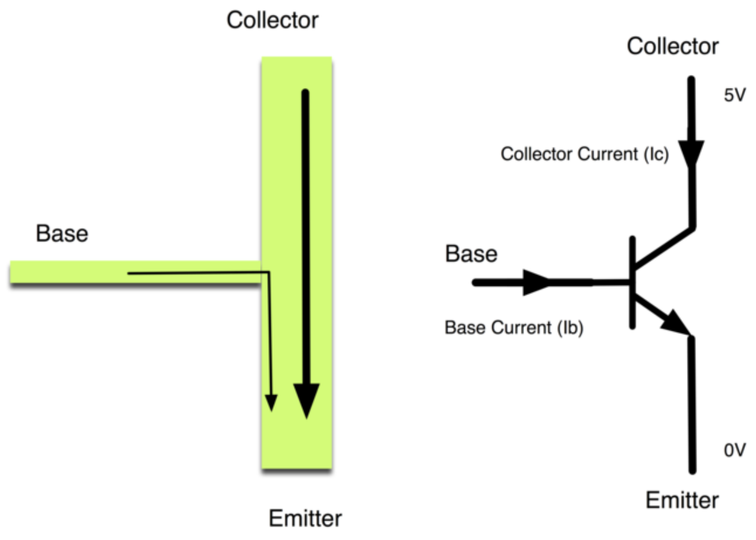
1.2 Home challenge 1: Connecting a DC Motor without a motor driver

https://learn.adafruit.com/adafruit-arduino-lesson-13-dc-motors
We will control a small DC motor using an Arduino and a transistor. You will use an Arduino analog output (PWM) to control the speed of the motor by sending a number between 0 and 255 from the Serial Monitor.
Required items:

- Arduino
- Breadboard
- PN2222 transistor
- 1N4001 diode
- 270 ohm Resistor
- Wires

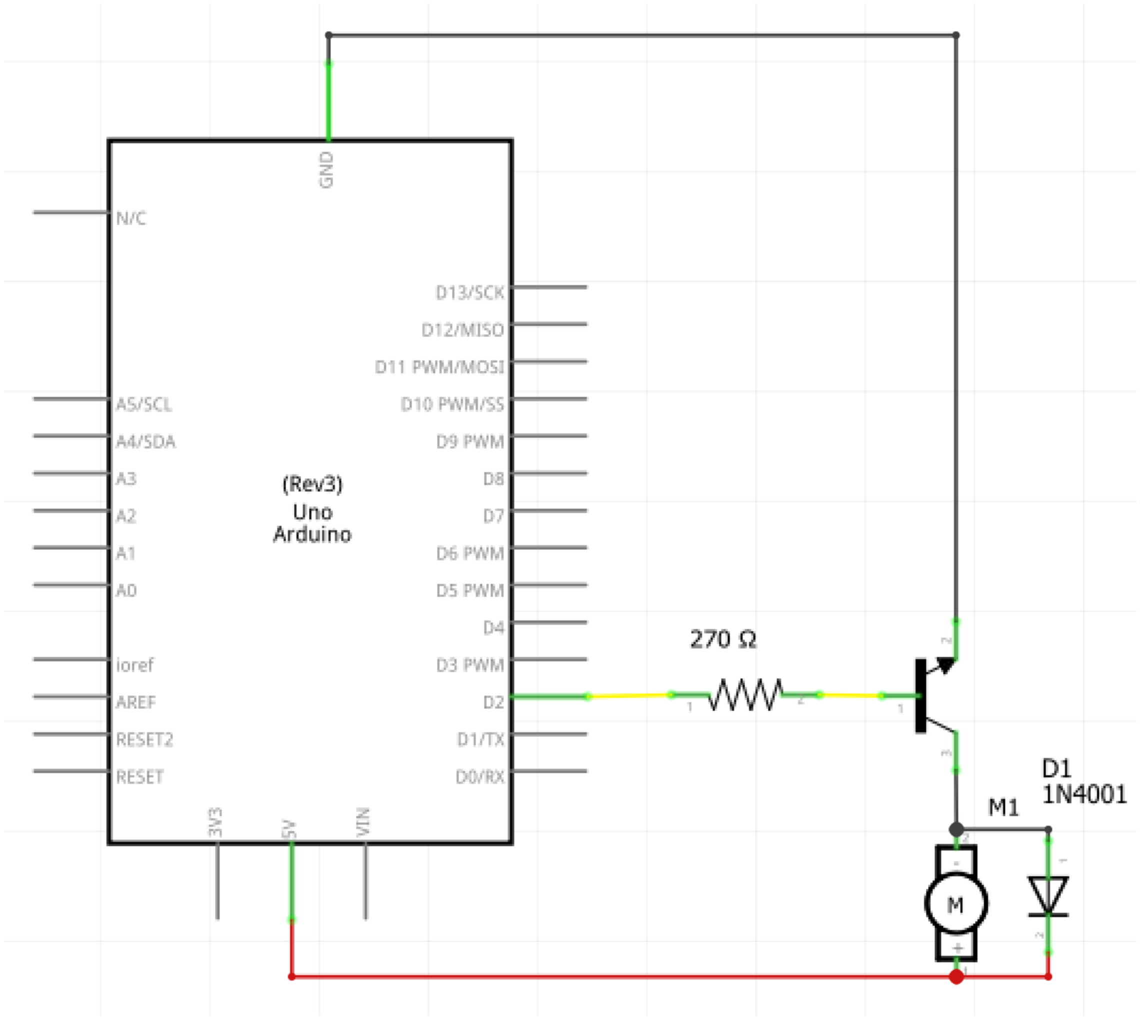
The small DC motor is likely to use more power than an Arduino digital output can handle directly on an output pin. If we tried to connect the motor straight to an Arduino digital output pin, there is a good chance that it could damage the Arduino. A small transistor like the PN2222 can be used as a switch that uses just a little current from the Arduino digital output to control the much bigger current of the motor.

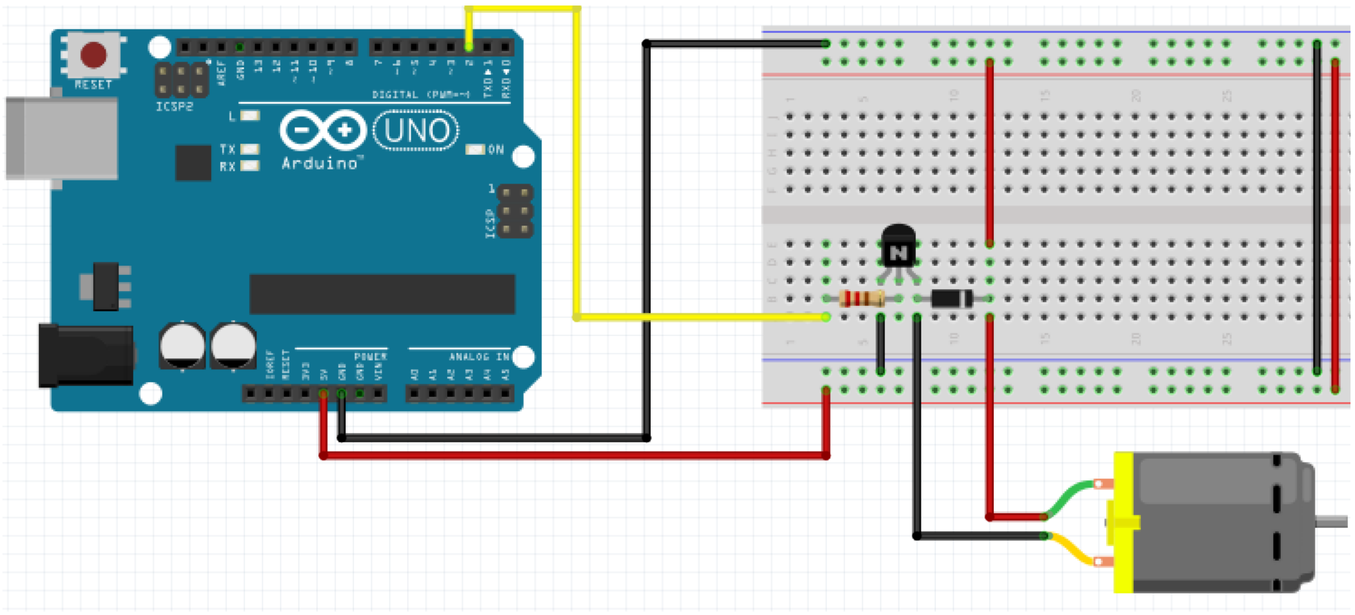
The transistor has three leads. Most of the electricity flows from the Collector to the Emitter, but this will only happen if a small amount is flowing into the Base connection. This small current is supplied by the Arduino digital output.



When you put together the breadboard, there are two things to look out for:

- Firstly, make sure that the transistor is the right way around. The flat side of the transistor should be on the right-hand side of the breadboard.
- Secondly the striped end of the diode should be towards the +5V power line - see the image below!


GND
N/C
D13/SCK
D12/MISO
D11 PWM/MOSI
D10 PWM/SS
D9 PWM
D8
D7
D6 PWM
D5 PWM
D4
D3 PWM
D2
D1/TX
D0/RX
A5/SCL
A4/SDA
A3
A2
A1
A0
ioref
AREF
RESET2
RESET
(Rev3)
Uno
Arduino
3V3
5V
VIN
270 Ω
M1
D1
1N4001
M


RESET
ICSP2
DIGITAL (PWM=~)
UNO
Arduino
ON
ICSP
POWER
ANALOG IN
N

```
int motorPin = 3;

void setup()
{
 pinMode(motorPin, OUTPUT);
 Serial.begin(9600);
 while (!Serial);
 Serial.println("Speed 0 to 255");
}


void loop()
{
 if (Serial.available())
 {
   int motorSpeed = Serial.parseInt();
   if (motorSpeed  >= 0 && motorSpeed <= 255)
   {
     analogWrite(motorPin, motorSpeed);
   }
 }
}
```

“A transistor is a semiconductor device used to amplify or switch electronic signals and electrical power.”
The transistor acts like a switch, controlling the power to the motor, Arduino pin 3 is used to turn the transistor on and off and is given the name 'motorPin' in the sketch.
When the sketch starts, it prompts you to remind you that to control the speed of the motor you need to enter a value between 0 and 255 in the Serial Monitor.

In the 'loop' function, the command 'Serial.parseInt' is used to read the number entered as text in the Serial Monitor and convert it into an 'int'. You could type any number here, so the 'if' statement on the next line only does an analog write with this number if the number is between 0 and 255.

Try reversing the connections to the motor. What happens?
Try entering different values (starting at 0) into the Serial Monitor and notice at what value the motor starts to actually turn. You will find that the motor starts to 'sing' as you increase the analog output.
Try pinching the drive shaft between your fingers. Don't hold it like that for too long, or you may cook the transistor, but you should find that it is fairly easy to stop the motor. It is spinning fast, but it does not have much torque.

What are some limitations to this design?

1.3 Connecting a DC Motor with L293D motor driver

https://learn.adafruit.com/adafruit-arduino-lesson-15-dc-motor-reversing/overview

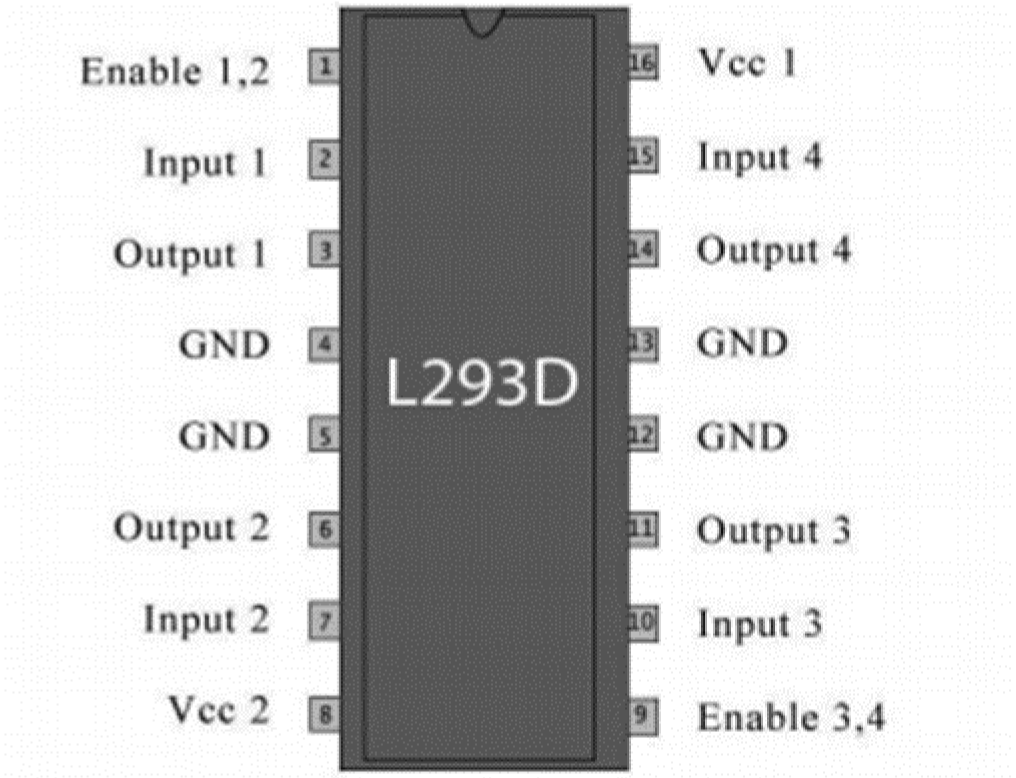
Now that we've seen how to make our own circuit to control the DC motor, let's use the L293D motor driver, a H-bridge that lets us control the rotation speed, direction of two motors at the same time.

Required items:
- Arduino
- Breadboard
- Wires
- DC Motor
- L293D motor driver

1.3.1 L293D Motor Driver

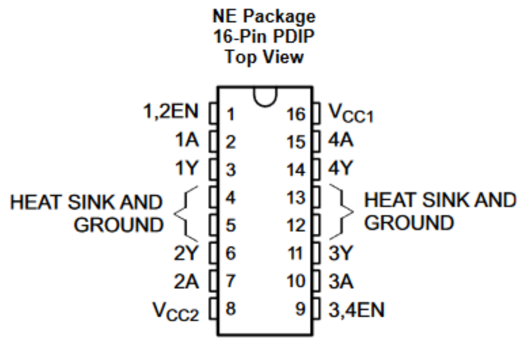
http://www.ti.com/lit/ds/symlink/l293d.pdf

This is a very useful chip. It can actually control two motors independently. We are just using half the chip in this lesson, most of the pins on the right hand side of the chip are for controlling a second motor.

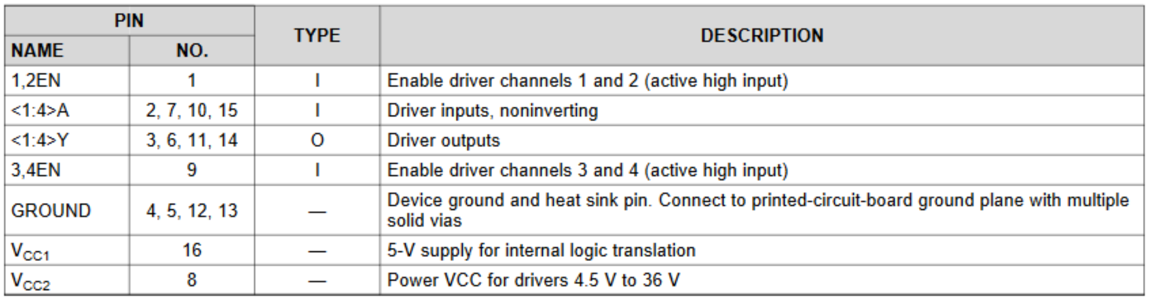


The L293D has two +V pins (8 and 16). The pin '+Vmotor (8) provides the power for the motors, and +V (16) for the chip's logic. We have connected both of these to the Arduino 5V pin. However, if you were using a more powerful motor, or a higher voltage motor, you would provide the motor with a separate power supply using pin 8 connected to the positive power supply and the ground of the second power supply is connected to the ground of the Arduino

5 Pin Configuration and Functions



Pin Functions



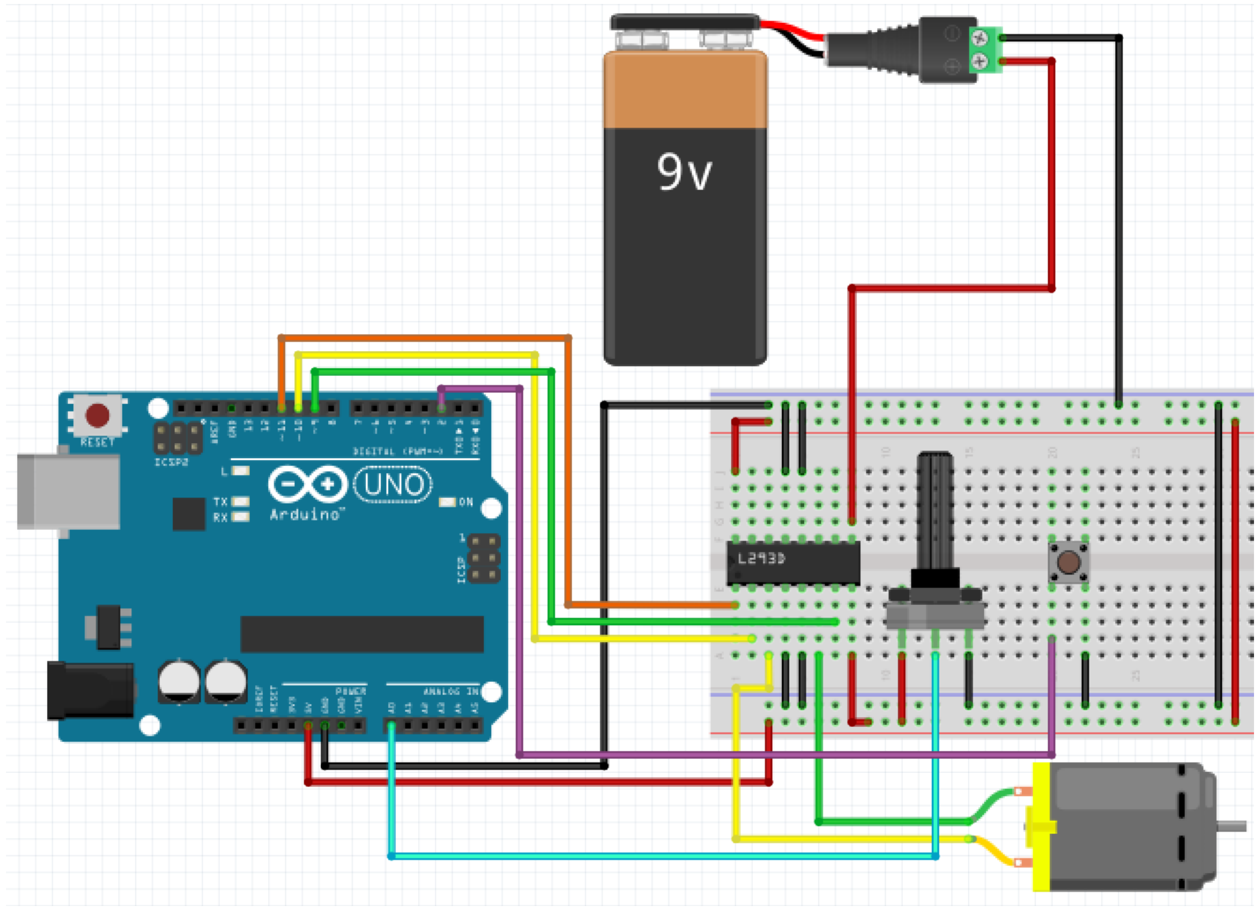
PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to printed-circuit-board ground plane with multiple solid vias
V_{CC1}	16	—	5-V supply for internal logic translation
V_{CC2}	8	—	Power VCC for drivers 4.5 V to 36 V

.(source: http://www.ti.com/lit/ds/symlink/l293d.pdf)

Now that we have got the hang of controlling the motor directly, we can let the Arduino manage the Enable, In1 and In2 pins. When you build the breadboard, you need to ensure that the IC is the right way around. The notch should be towards the top of the breadboard.

Motor direction based on Input 1 and Input 2 states

Input 1	**Input 2**	**DC Motor**
GND	GND	Stopped
5V	GND	Turns in Direction A
GND	5V	Turns in Direction B
5V	5V	Stopped



Note: this was supposed to be the schematic without the potentiometer and button (as they are used in the next exercise) but I mixed them and now don't have access to it this week.

```
int enablePin = 11;
int in1Pin = 10;
int in2Pin = 9;
int motorSpeed= 0;
boolean reverse = LOW;
void setup()
{
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(enablePin, OUTPUT);
}

void loop()
```

```
{
  setMotor(motorSpeed , reverse);
}

void setMotor(int motorSpeed, boolean reverse)
{
  analogWrite(enablePin, motorSpeed);
  digitalWrite(in1Pin, !reverse);
  digitalWrite(in2Pin, reverse);
}
```

(Modify speed and reverse to control the motor. These will be controlled by potentiometer and button)

Firstly, the speed is set, by using an analogWrite to the enable pin. The enable pin of the L293 just turns the motor on or off irrespective of what the in1 and in2 pins of the L293 are set to.

To control the direction of the motor, the pins in1 and in2 must be set to opposite values.

If in1 is HIGH and in2 is LOW, the motor will spin one way, if on the other hand in1 is LOW and in2 HIGH then the motor will spin in the opposite direction.

The '!' command means 'not'. So the first digitalWrite command for in1 sets it to the opposite of whatever the value of 'reverse' is, so if reverse is HIGH it sets it to LOW and vice versa.

The second digitalWrite for 'in2' sets the pin to whatever the value of 'reverse' is. This means that it will always be the opposite of whatever in1 is.

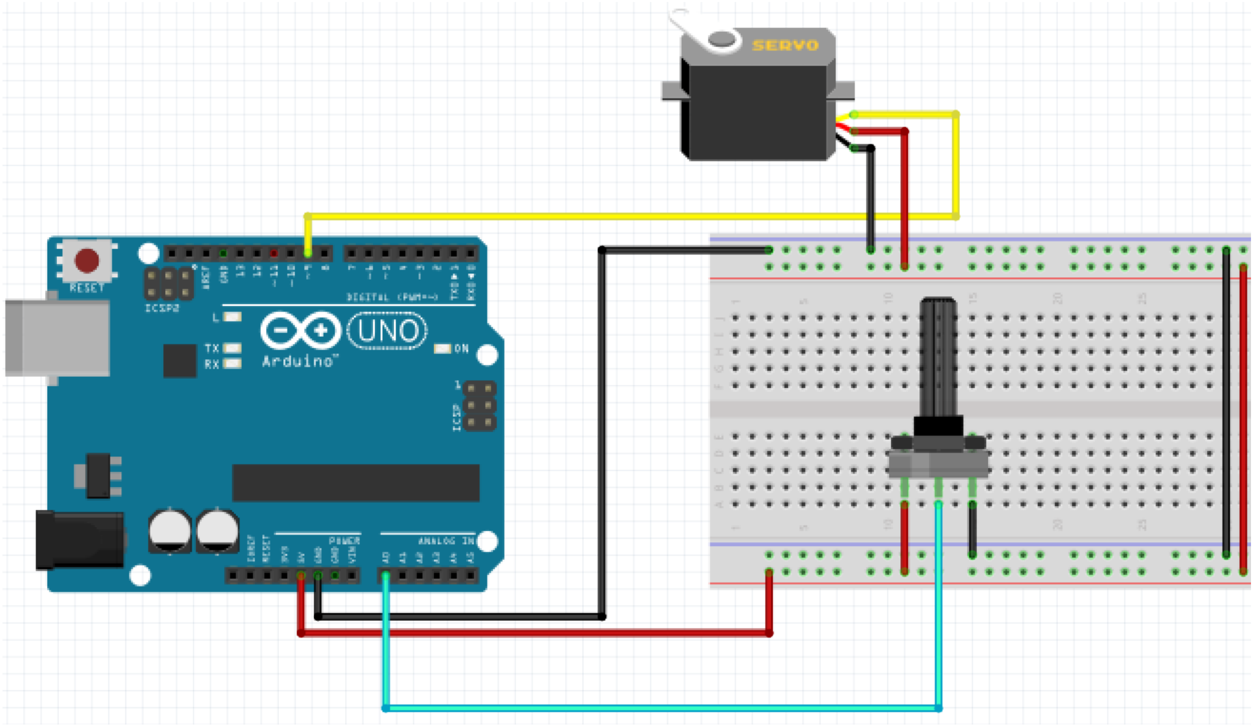
2. Servo motor

We will learn how to control a servo motor using an Arduino.

Firstly, you will get the servo to sweep back and forth automatically, and then you will add a pot to control the position of the servo.

Required items:

- Arduino
- Breadboard
- Wires
- Servo motor
- Potentiometer
- ~~100 µF capacitor~~ (only if it misbehaves)



(do not connect it directly to Arduino's GND and 5V. Use the breadboard)

The servo motor has three leads. The color of the leads varies between servo motors, but the red lead is always 5V and GND will either be black or brown. The other lead is the control lead, which is usually orange or yellow. This control lead is connected to digital pin 9.

If the servo misbehaves

Your servo may behave erratically, and you may find that this only happens when the Arduino is plugged into certain USB ports. This is because the servo draws quite a lot of power, especially as the motor is

starting up, and this sudden high demand can be enough to drop the voltage on the Arduino board, so that it resets itself.
If this happens, then you can usually cure it by adding a high value capacitor between GND and 5V on the breadboard.
The capacitor acts as a reservoir of electricity for the motor to use, so that when it starts,
/ it takes charge from the capacitor as well as the Arduino supply.
Source: https://learn.adafruit.com/adafruit-arduino-lesson-14-servo-motors/if-the-servo-misbehaves

Attention! The longer lead of the capacitor is the positive lead, and this should be connected to 5V. The negative lead is also often marked with a '-' symbol.

```
#include <Servo.h>

int servoPin = 9;
Servo servo;

int angle = 0;   // servo position in degrees

void setup()
{
  servo.attach(servoPin);
}


void loop()
{
  // scan from 0 to 180 degrees
  for(angle = 0; angle < 180; angle++)
  {
          servo.write(angle);
          delay(15);
  }
  // now scan back from 180 to 0 degrees
  for(angle = 180; angle > 0; angle--)
  {
          servo.write(angle);
          delay(15);
  }
}
```
Servo motors are controlled by a series of pulses, and to make it easy to use them, an Arduino library has been created so that you can just instruct the servo to turn at a particular angle.

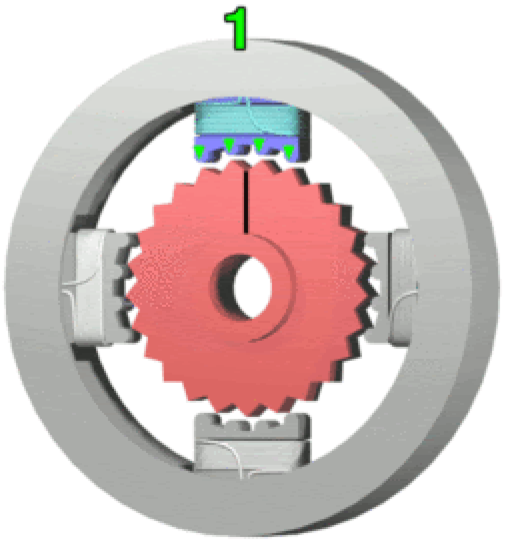
3. Stepper Motor

What is a stepper motor?

Stepper motors, due to their unique design, can be controlled to a high degree of accuracy without any feedback mechanisms. The shaft of a stepper, mounted with a series of magnets, is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence, precisely moving it forward or backward in small "steps".

(source: https://www.arduino.cc/en/tutorial/stepperSpeedControl)

Stepper motors fall somewhere in between a regular DC motor and a servo motor. They have the advantage that they can be positioned accurately, moved forward or backwards one 'step' at a time, but they can also rotate continuously.



In our it we have and will be using the 28BYJ-48 Stepper Motor

28BYJ-48 Stepper Motor Technical Specifications

- Rated Voltage: 5V DC
- Number of Phases: 4
- Stride Angle: 5.625°/64
- Pull in torque: 300 gf.cm
- Insulated Power: 600VAC/1mA/1s
- Coil: Unipolar 5 lead coil
- Datasheet: http://robocraft.ru/files/datasheet/28BYJ-48.pd
- Step angle
    - Half step mode (recommended): 0.0879°
    - Full step mode: 0.176°
- Steps per revolution

- Half step mode: 4096 (**see note**)
- Full step mode: 2048

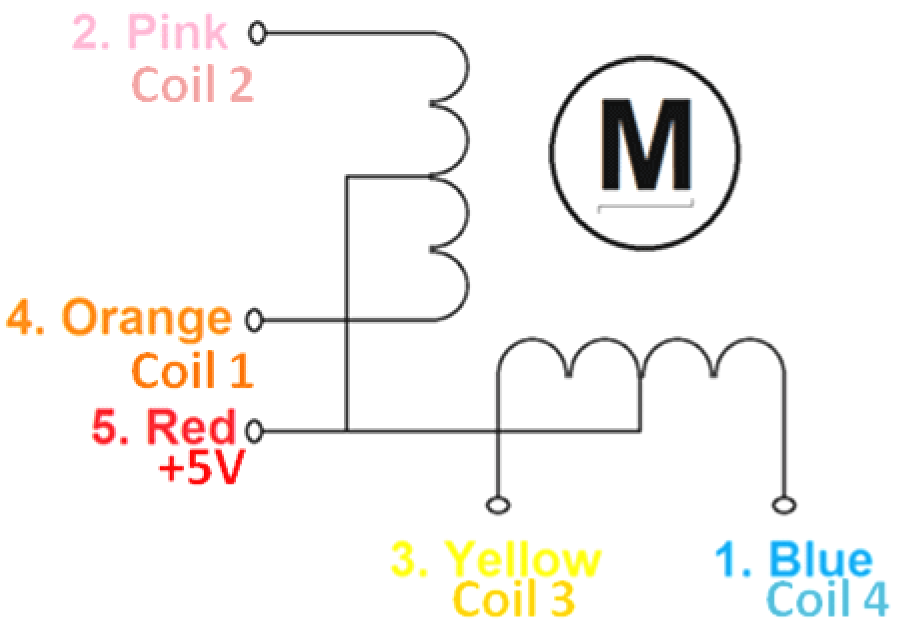
!Important: Manufacturers usually specify that the motors have a 64:1 gear reduction. Some members of the Arduino Forums noticed that this wasn't correct and so they took some motors apart to check the actual gear ratio. They determined that the exact gear ratio is in fact **63.68395:1**, which results in approximately **4076** steps per full revolution (in half step mode).

Where to use 28-BYJ48 Stepper Motor

The most commonly used stepper motor is the **28-BYJ48 Stepper Motors**. You can find this (or similar) motors in your DVD drives, Motion camera and many more places. The motor has a 4 coil unipolar arrangement and each coil is rated for +5V hence it is relatively easy to control with any basic microcontrollers. These motors has a stride angle of 5.625°/64, this means that the motor will have to make 64 steps to complete one rotation and for every step it will cover a 5.625° hence the level of control is also high. However, these motors run only on 5V and hence cannot provide high torque, for high torque application you should consider the **Nema17 motors**. So if you are looking for a compact easy to use stepper motor with decent torque then this motor is the right choice for you.

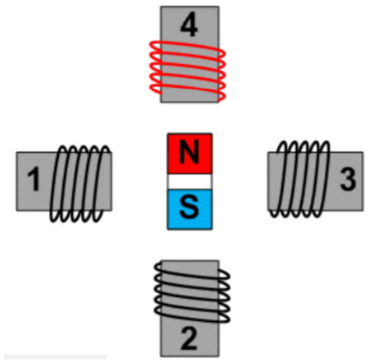
How to use 28-BYJ48 Stepper Motor

These stepper motors consume high current and hence a driver IC like the ULN2003 is mandatory. To know how to make this motor rotate we should look into the coil diagram below.



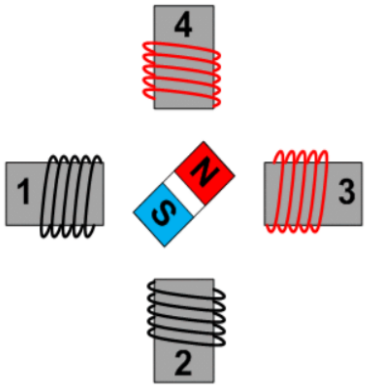
As we can see there are four coils in the motor and one end of all the coil is tied to +5V (Red) and the other ends (Orange, Pink, Yellow and Blue) are taken out as wires. The Red wire is always provided with a constant +5V supply and this +5V will be across (energize) the coil only if the other end of the coil is grounded. A stepper motor can be made to rotate only if the coils are energized (grounded) in a logical sequence. This logical sequence can be programmed using a microcontroller or by designing a digital circuit. The sequence in which each coil should be triggered is shown in the table below. Here “1” represents the coil is held at +5V, since both the ends of the coil are at +5V (red and other end) the coil will not be energized. Similarly “0” represents the coil is held to ground, now one end will be +5V and the other one is grounded so the coil will be energized.

Full Stepping



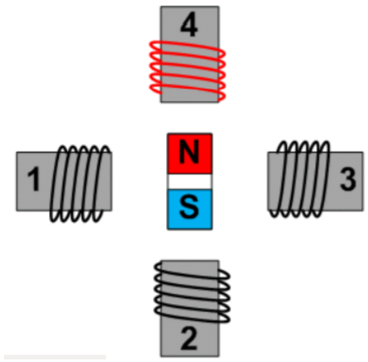
Step Number	Coil 1	Coil 2	Coil 3	Coil 4
1	OFF	OFF	OFF	ON
2	OFF	OFF	ON	OFF
3	OFF	ON	OFF	OFF
4	ON	OFF	OFF	OFF

Double stepping



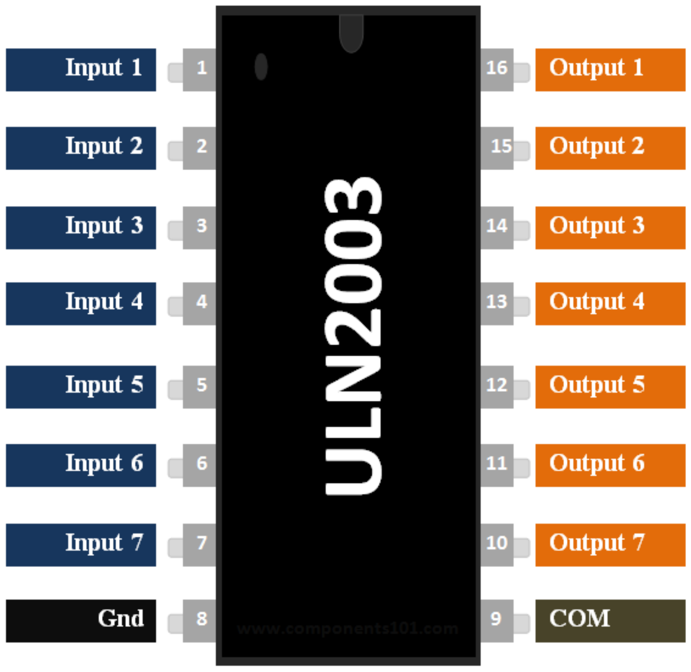
Step Number	Coil 1	Coil 2	Coil 3	Coil 4
1	ON	ON	OFF	OFF
2	OFF	ON	ON	OFF
3	OFF	OFF	ON	ON
4	ON	OFF	OFF	ON

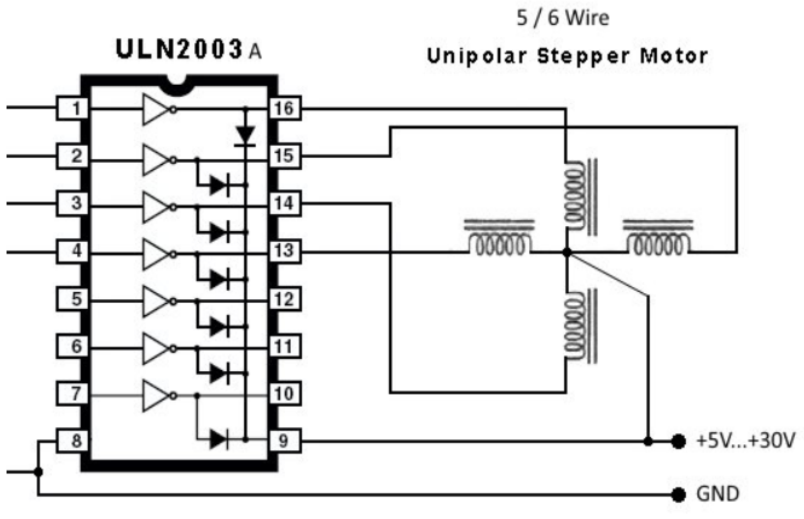
Half Stepping



Step Number	Step Position	Coil 1	Coil 2	Coil 3	Coil 4
1	0.0	ON	OFF	OFF	OFF
2	0.5	ON	ON	OFF	OFF
3	1.0	OFF	ON	OFF	OFF
4	1.5	OFF	ON	ON	OFF
5	2.0	OFF	OFF	ON	OFF
6	2.5	OFF	OFF	ON	ON
7	3.0	OFF	OFF	OFF	ON
8	3.5	ON	OFF	OFF	ON

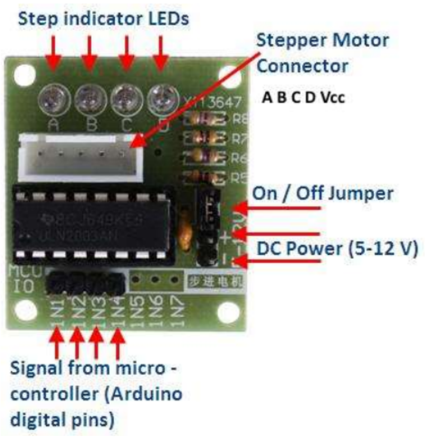
5.1 ULN2003





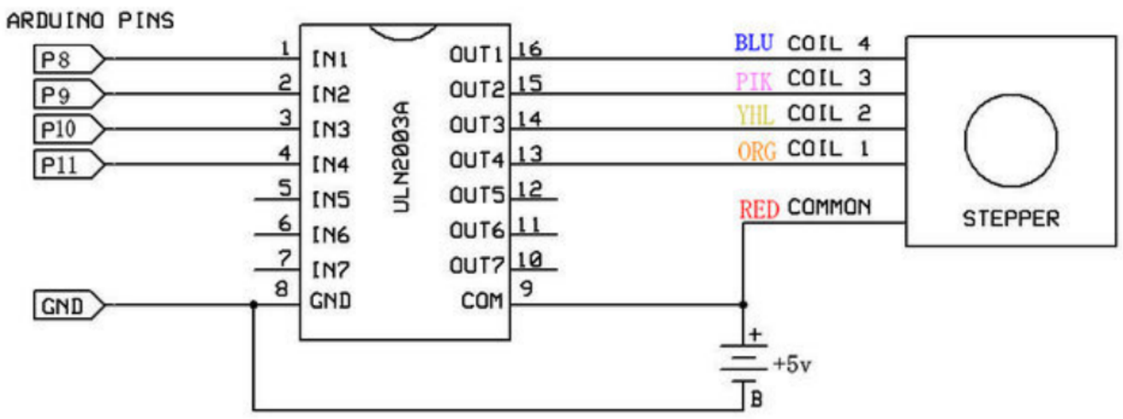
https://en.wikipedia.org/wiki/ULN2003A

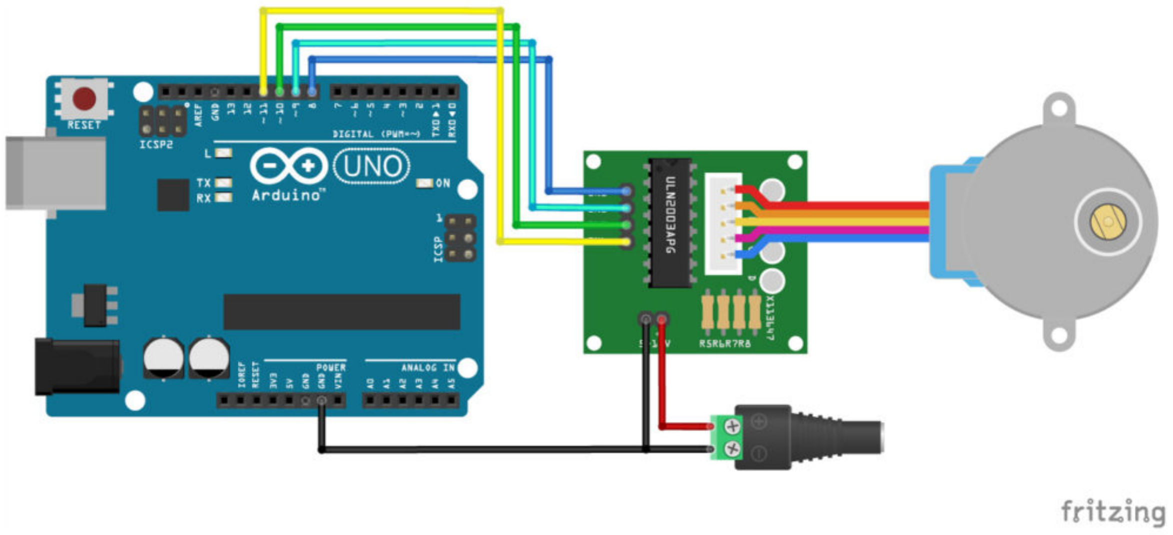
The **ULN2003A** is an array of seven NPN Darlington transistors capable of 500 mA, 50 V output. It features common-cathode flyback diodes for switching inductive loads



!important Note that the stepper motor can also be controlled using a H-bridge, such as L293D.

5.2 Connecting and controlling the stepper motor





5.2.1 Full stepping example

```
const int stepperPin1 = 8;
const int stepperPin2 = 9;
const int stepperPin3 = 10;
const int stepperPin4 = 11;

// if the delay is to big, you won't see the rotor moving
int delayTime = 100;

void setup() {
 pinMode(stepperPin1, OUTPUT);
 pinMode(stepperPin2, OUTPUT);
 pinMode(stepperPin3, OUTPUT);
 pinMode(stepperPin4, OUTPUT);
}

void loop() {
 digitalWrite(stepperPin1, HIGH);
 digitalWrite(stepperPin2, LOW);
 digitalWrite(stepperPin3, LOW);
 digitalWrite(stepperPin4, LOW);
 delay(delayTime);

 digitalWrite(stepperPin1, LOW);
 digitalWrite(stepperPin2, HIGH);
 digitalWrite(stepperPin3, LOW);
 digitalWrite(stepperPin4, LOW);
 delay(delayTime);

 digitalWrite(stepperPin1, LOW);
 digitalWrite(stepperPin2, LOW);
 digitalWrite(stepperPin3, HIGH);
 digitalWrite(stepperPin4, LOW);
 delay(delayTime);

 digitalWrite(stepperPin1, LOW);
 digitalWrite(stepperPin2, LOW);
 digitalWrite(stepperPin3, LOW);
 digitalWrite(stepperPin4, HIGH);
 delay(delayTime);

}
```

5.2.2 Half Stepping example

```
const int stepperPin1 = 8;
const int stepperPin2 = 9;
const int stepperPin3 = 10;
const int stepperPin4 = 11;

int delayTime = 100;

void setup() {
  pinMode(stepperPin1, OUTPUT);
  pinMode(stepperPin2, OUTPUT);
  pinMode(stepperPin3, OUTPUT);
  pinMode(stepperPin4, OUTPUT);
}

void loop() {
  digitalWrite(stepperPin1, HIGH);
  digitalWrite(stepperPin2, LOW);
  digitalWrite(stepperPin3, LOW);
  digitalWrite(stepperPin4, LOW);
  delay(delayTime);

  digitalWrite(stepperPin1, HIGH);
  digitalWrite(stepperPin2, HIGH);
  digitalWrite(stepperPin3, LOW);
  digitalWrite(stepperPin4, LOW);
  delay(delayTime);

  digitalWrite(stepperPin1, LOW);
  digitalWrite(stepperPin2, HIGH);
  digitalWrite(stepperPin3, LOW);
  digitalWrite(stepperPin4, LOW);
  delay(delayTime);

  digitalWrite(stepperPin1, LOW);
  digitalWrite(stepperPin2, HIGH);
  digitalWrite(stepperPin3, HIGH);
  digitalWrite(stepperPin4, LOW);
  delay(delayTime);

  digitalWrite(stepperPin1, LOW);
  digitalWrite(stepperPin2, LOW);
  digitalWrite(stepperPin3, HIGH);
  digitalWrite(stepperPin4, LOW);
  delay(delayTime);
  digitalWrite(stepperPin1, LOW);
  digitalWrite(stepperPin2, LOW);
  digitalWrite(stepperPin3, HIGH);
  digitalWrite(stepperPin4, HIGH);
  delay(delayTime);

  digitalWrite(stepperPin1, LOW);
  digitalWrite(stepperPin2, LOW);
  digitalWrite(stepperPin3, LOW);
  digitalWrite(stepperPin4, HIGH);
  delay(delayTime);

  digitalWrite(stepperPin1, HIGH);
  digitalWrite(stepperPin2, LOW);
  digitalWrite(stepperPin3, LOW);
  digitalWrite(stepperPin4, HIGH);
  delay(delayTime);
}
```

5.4 Using stepper.h

This library allows you to control unipolar or bipolar stepper motors.

```
/* Example sketch to control a 28BYJ-48 stepper motor with ULN2003 driver board and Arduino
UNO. More info: https://www.makerguides.com */
// Include the Arduino Stepper.h library:
#include <Stepper.h>
// Define number of steps per rotation:
const int stepsPerRevolution = 2048;
// Wiring:
// Pin 8 to IN1 on the ULN2003 driver
// Pin 9 to IN2 on the ULN2003 driver
// Pin 10 to IN3 on the ULN2003 driver
// Pin 11 to IN4 on the ULN2003 driver
// Create stepper object called 'myStepper', note the pin order:
Stepper myStepper = Stepper(stepsPerRevolution, 8, 9, 10, 11);
void setup() {
  // Set the speed to 5 rpm:
  myStepper.setSpeed(5);

  // Begin Serial communication at a baud rate of 9600:
  Serial.begin(9600);
}
void loop() {
  // Step one revolution in one direction:
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
  delay(500);

  // Step one revolution in the other direction:
  Serial.println("counterclockwise");
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

5.3 Home challenge 2

Control a highly accurate stepper motor using a potentiometer. Keep in mind that this is not the same as the servo where you can map the same position. Here, you will use the **change** in the potentiometer and move the stepper according to it.
Advice: Play with the speed as well.

Visit https://www.makerguides.com/28byj-48-stepper-motor-arduino-tutorial/. Scroll down to the **"Example codes for 28BYJ-48 stepper motor with Arduino and AccelStepper library"** and continue from there

The **AccelStepper** is a powerful library that can be used for controlling many types of steppers in different ways.
If you ever have to work with steppers, we recommend this library. Also, check out the MultiStepper library as well for **simultaneous** multi-motor drive without multi-threading. This is a challenging problem to do without it.

5.4 Home Challenge 3

Write a function that receives an angle and direction as input and moves the motor.