Introduction to robotics

9th lab

What we'll do today

1. (optional) Play with a DC motor, simulating our control circuit using a transistor
2. Play with a DC motor using a L293D motor driver (H-bridge)
3. Play with a servo motor
4. Play with a stepper motor

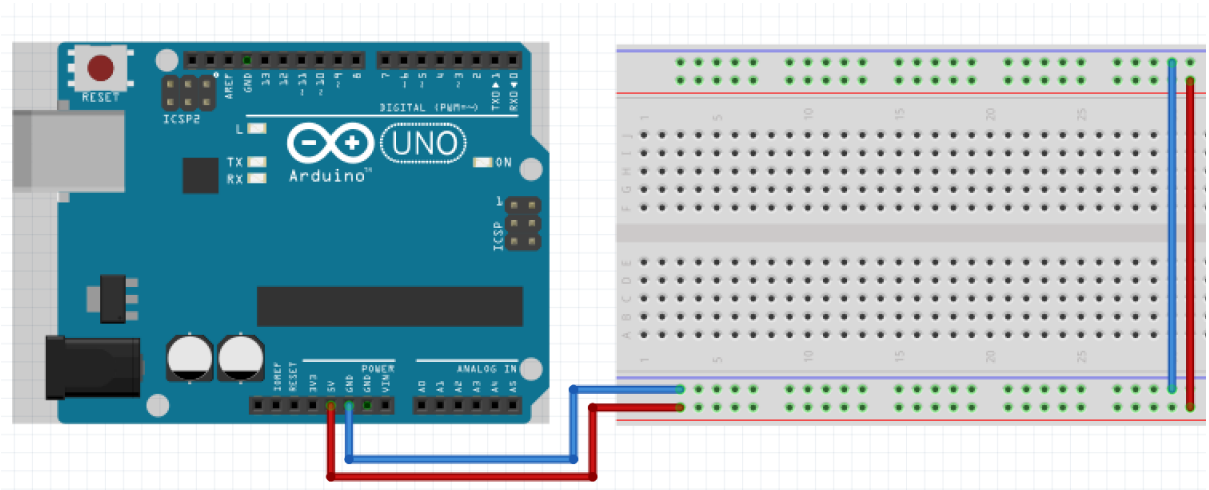
Remember, when possible, choose the wire color accordingly:

- **BLACK** (or dark colors) for **GND**
- **RED** (or colored) for **POWER (3.3V / 5V / VIN)**
- **Remember** than when you use digitalWrite or analogWrite, you actually send power over the PIN, so you can use the same color as for **POWER**
- **Bright Colored** for read signal
- We know it is not always possible to respect this due to lack of wires, but the first rule is **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.

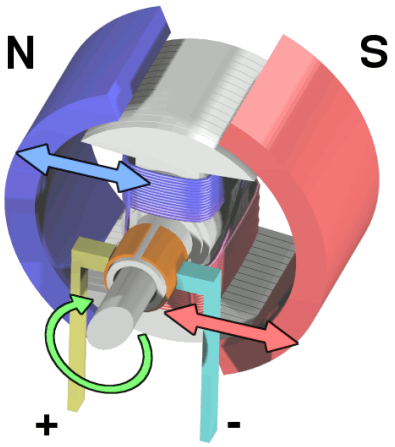


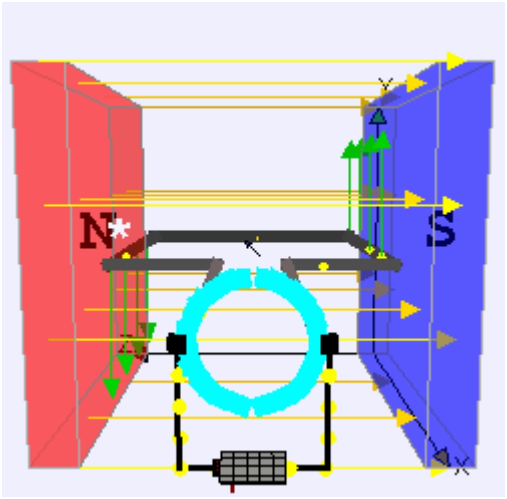
1. DC Motor

Sure, but what is a dc motor?

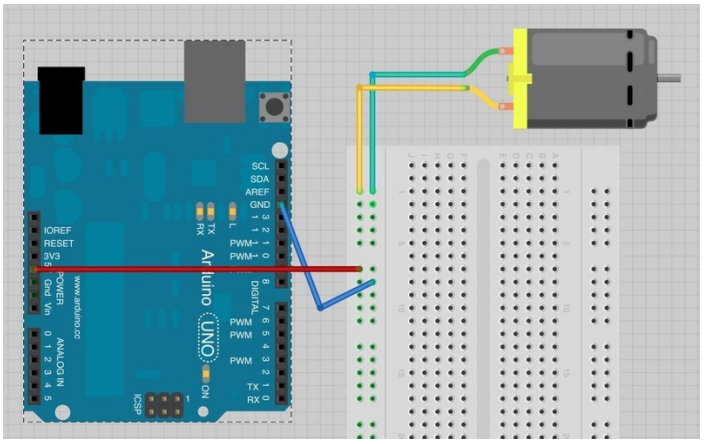
A **DC motor** is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic,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

2. (OPTIONAL) Connecting a DC Motor without a motor driver

https://learn.adafruit.com/adafruit-arduino-lesson-13-dc-motors

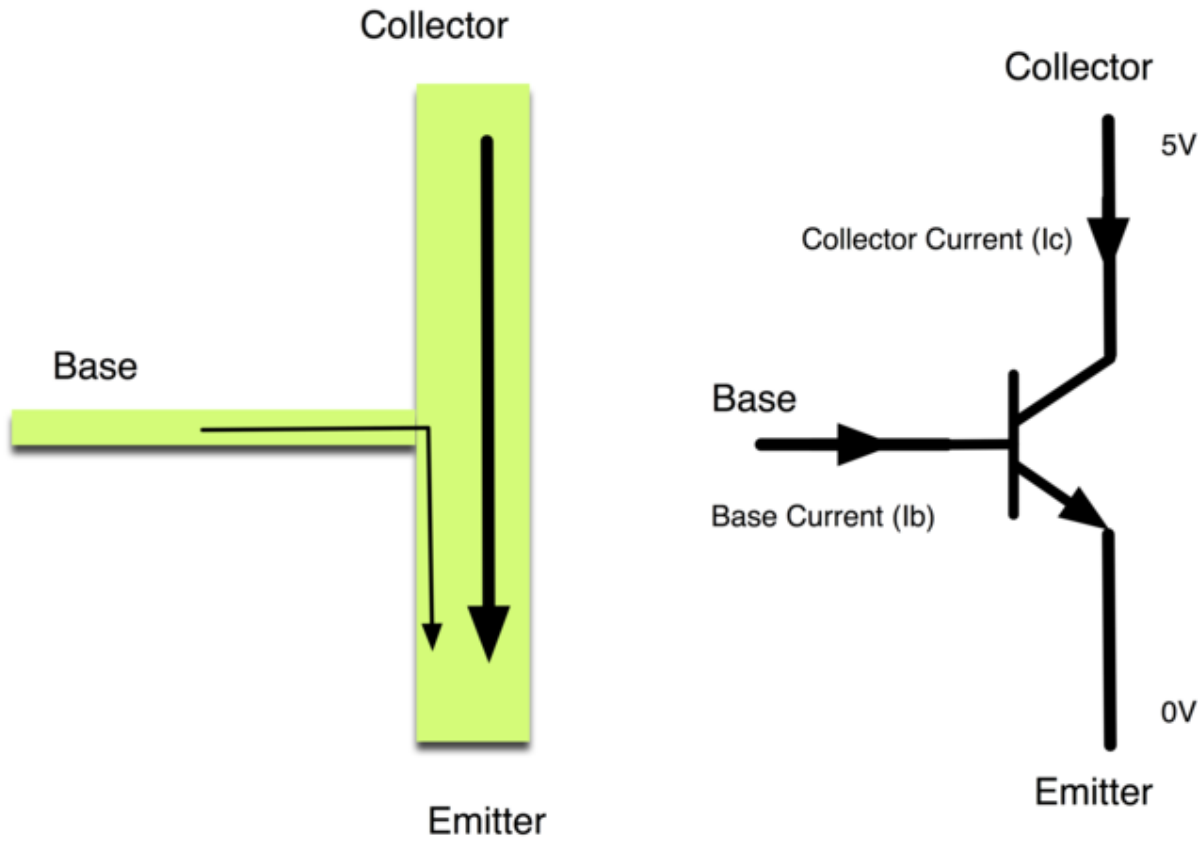
We will control a small DC motor using an Arduino and a transistor. You will use an Arduino analog output (PWM) to control the speed of the motor by sending a number between 0 and 255 from the Serial Monitor.

Required items:

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

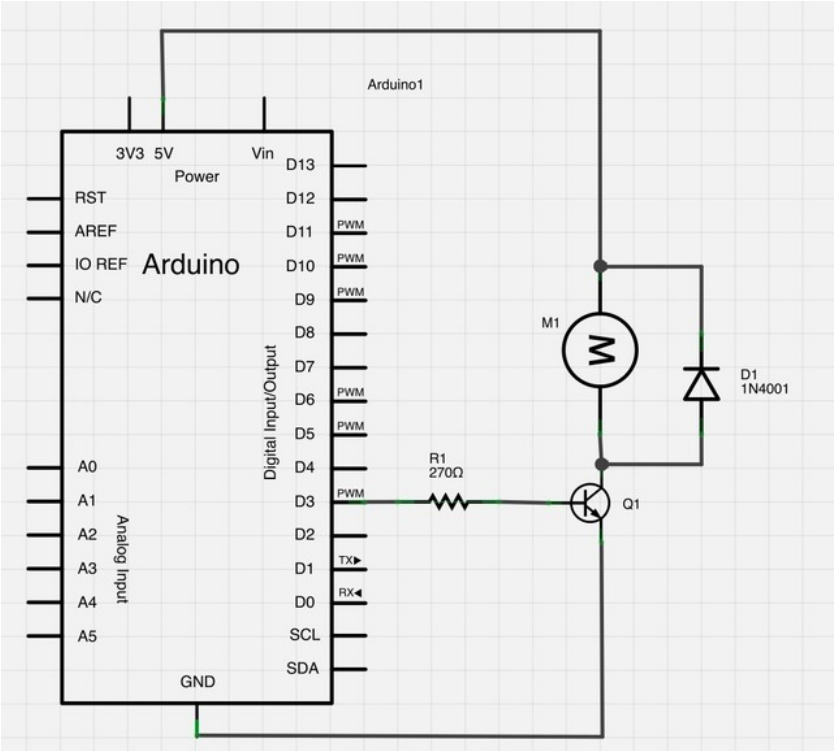
The small DC motor is likely to use more power than an Arduino digital output can handle directly on an output pin. If we tried to connect the motor straight to an Arduino digital output pin, there is a good chance that it could damage the Arduino. A small transistor like the PN2222 can be used as a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

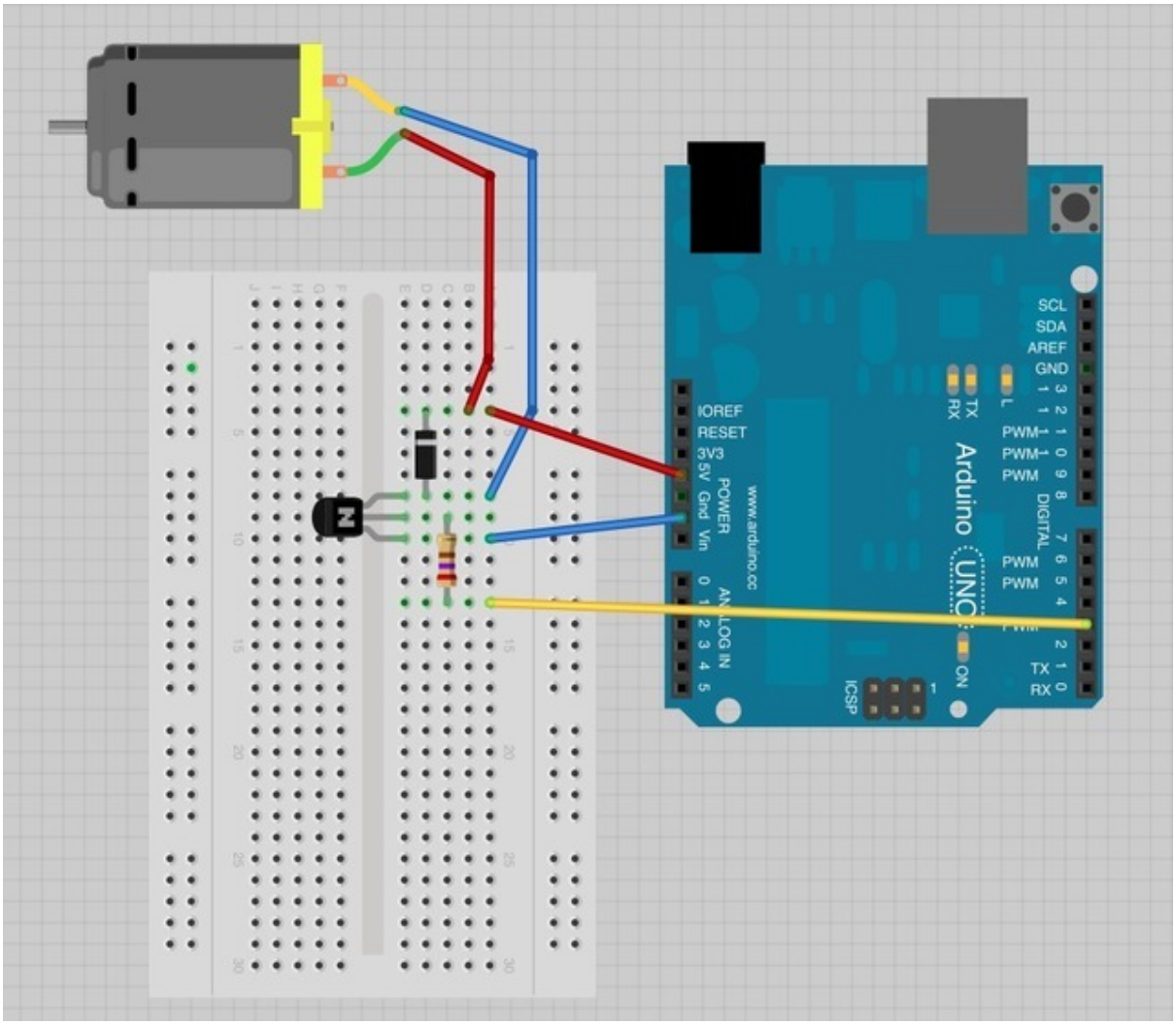
The transistor has three leads. Most of the electricity flows from the Collector to the Emitter, but this will only happen if a small amount is flowing into the Base connection. This small current is supplied by the Arduino digital output.



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!


Arduino1
3V3 5V
Power
Vin
RST
AREF
IO REF
N/C
Arduino
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
SCL
SDA
PWM
Digital Input/Output
A0
A1
A2
A3
A4
A5
Analog Input
GND
M1
D1
1N4001
R1
270Ω
Q1
TX
RX


IOREF
RESET
3V3
5V
Gnd
Vin
POWER
ANALOG IN
www.arduino.cc
Arduino
UNO
DIGITAL
PWM
SCL
SDA
AREF
GND
TX
RX
ICSP
ON

```
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?

3. Connecting a DC Motor with L293D motor driver

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

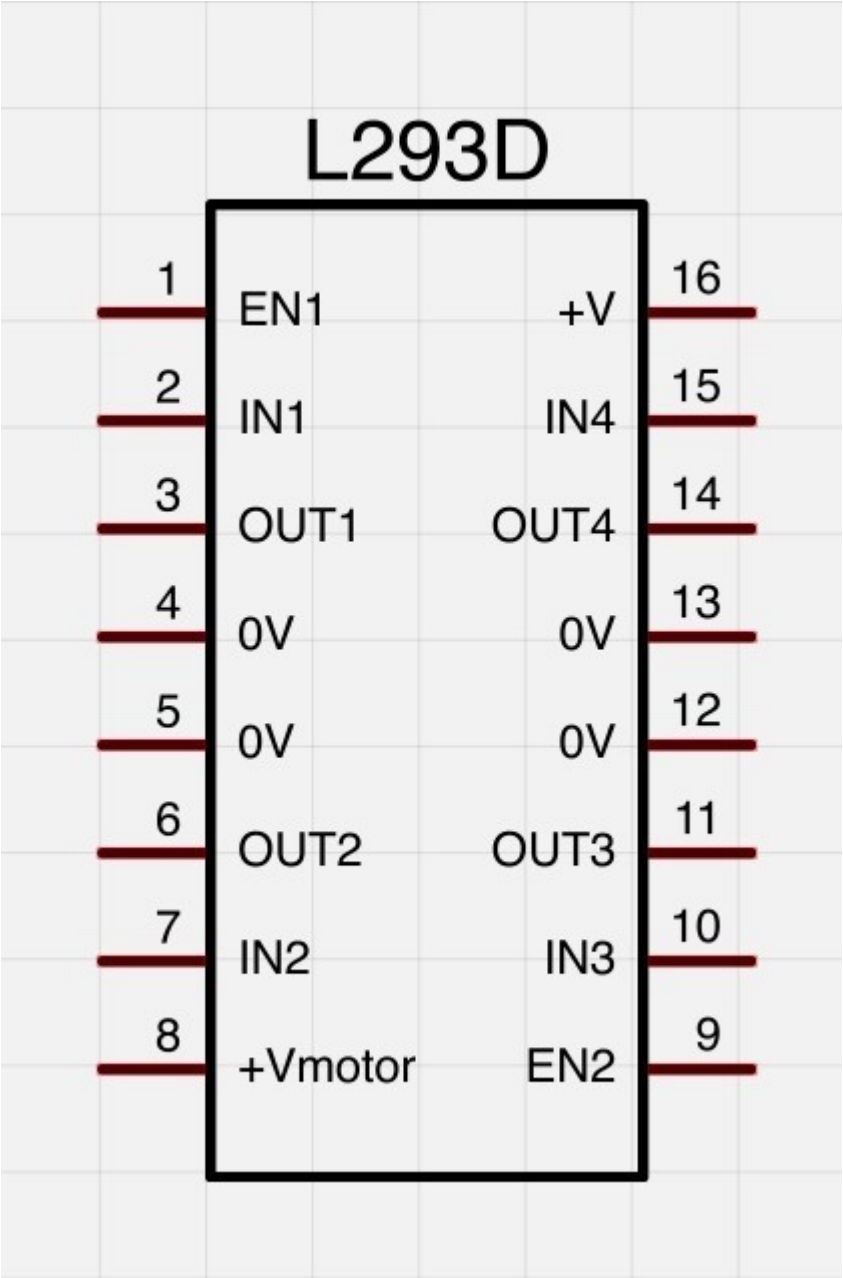
Now that we've seen how to make our own circuit to control the DC motor, let's use the L293D motor driver, a H-bridge that lets us control the rotation speed, direction of two motors at the same time.

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

3.1 L293D Motor Driver

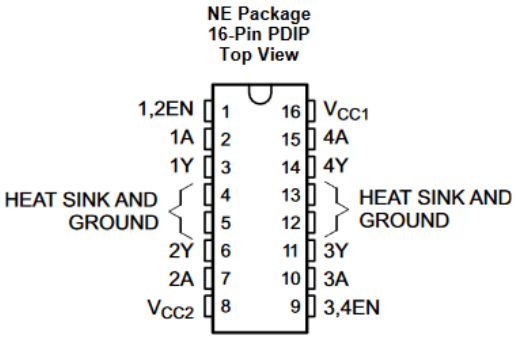
http://www.ti.com/lit/ds/symlink/l293d.pdf

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

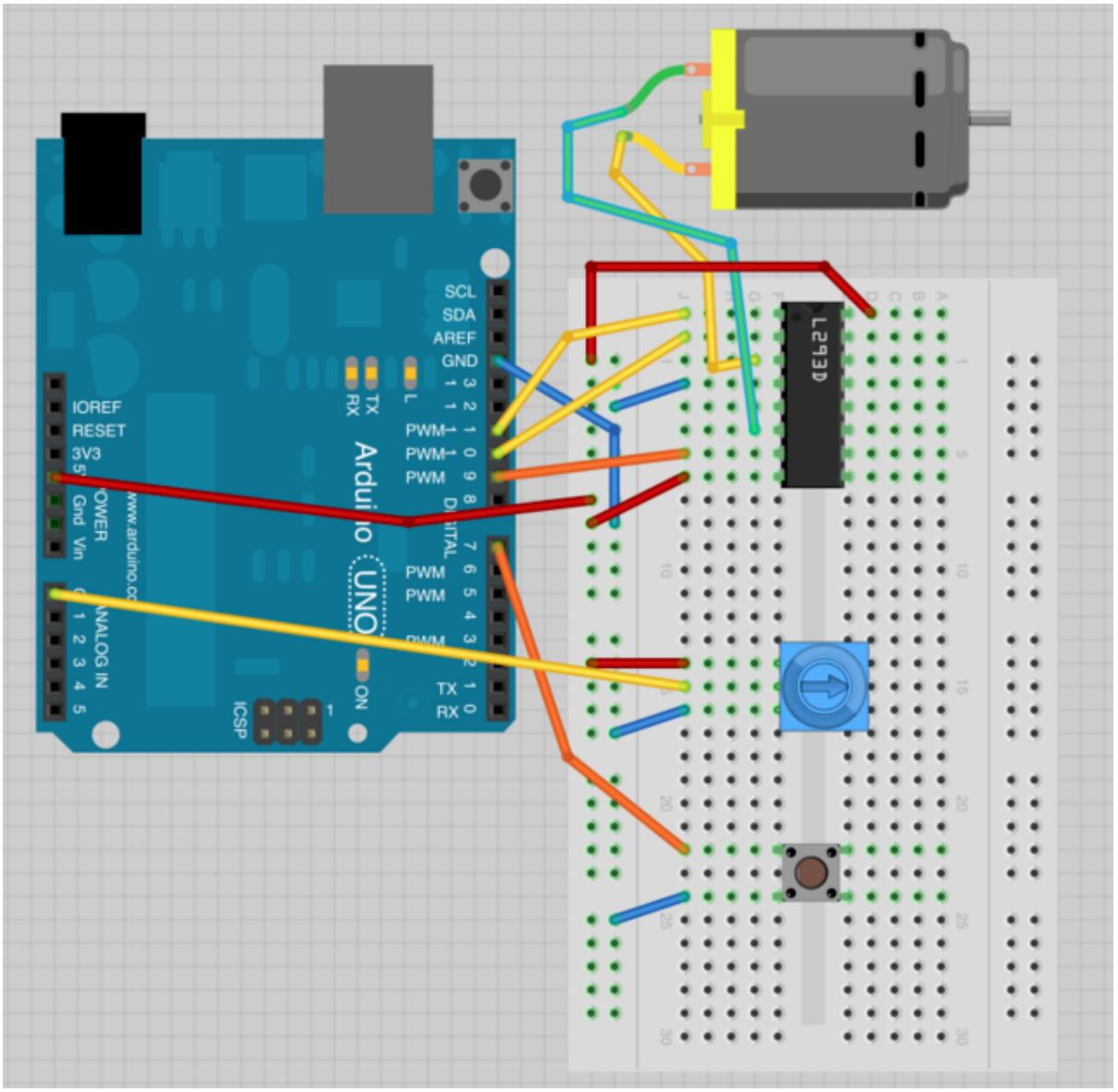
PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect 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.

In1	**In2**	**DC Motor**
GND	GND	Stopped
5V	GND	Turns in Direction A
GND	5V	Turns in Direction B
5V	5V	Stopped



EN1 -> pin11

IN1 -> pin10

IN2 -> pin9

V+ si V+motor -> 5v

Pin4(GND) -> GND

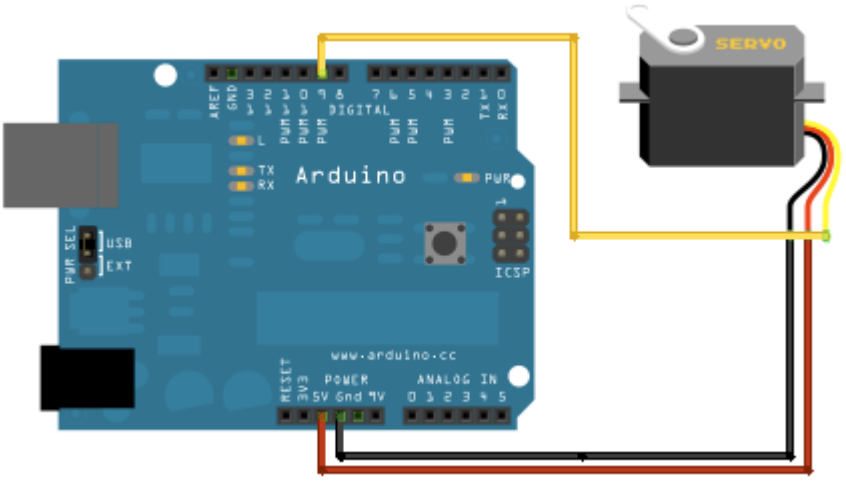
4. Servo motor

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



(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 and this 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.

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.

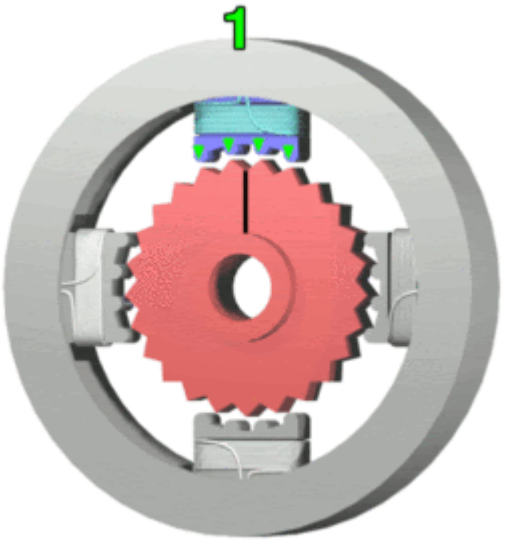
5. Stepper Motor

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

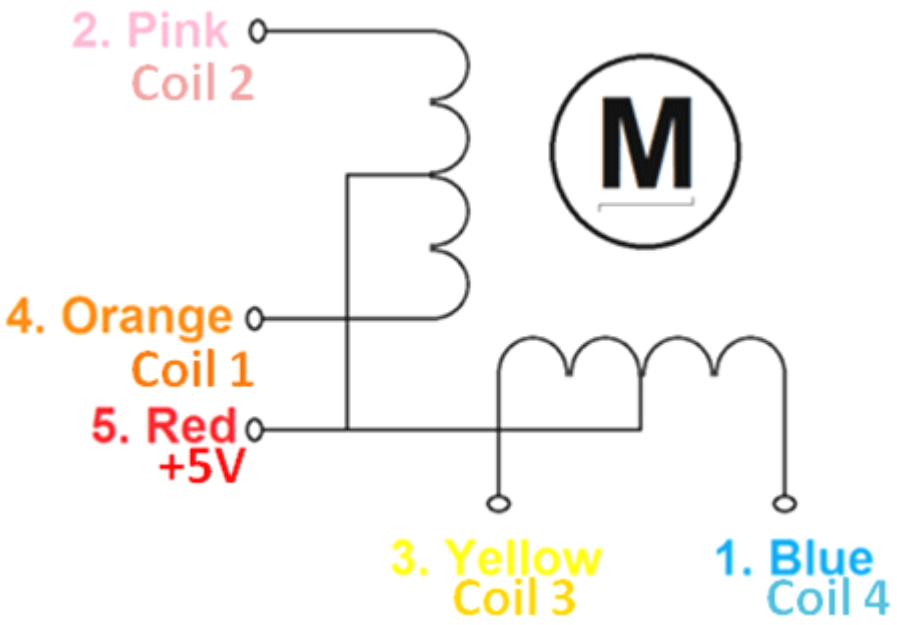
!Important: Manufacturers usually specify that the motors have a 64:1 gear reduction. Some members of the Arduino Forums noticed that this wasn't correct and so they took some motors apart to check the actual gear ratio. They determined that the exact gear ratio is in fact **63.68395:1**, which results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

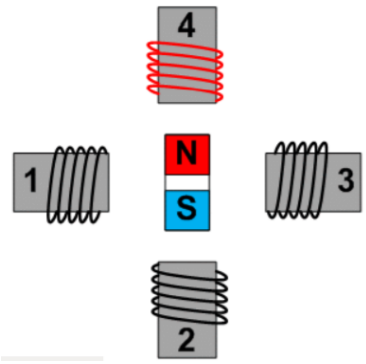
How to use 28-BYJ48 Stepper Motor

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.



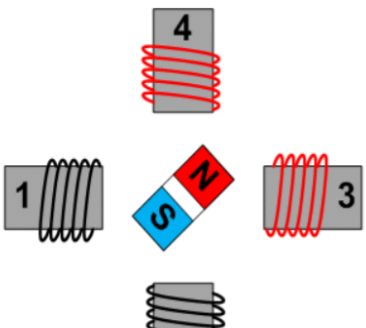
As we can see there are four coils in the motor and one end of all the coil is tied to +5V (Red) and the other ends (Orange, Pink, Yellow and Blue) are taken out as wires. The Red wire is always provided with a constant +5V supply and this +5V will be across (energize) the coil only if the other end 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” represent the coil is held at +5V, since both the ends of coil is at +5V (red and other end) the coil will not be energised. 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 energised.

Full Stepping



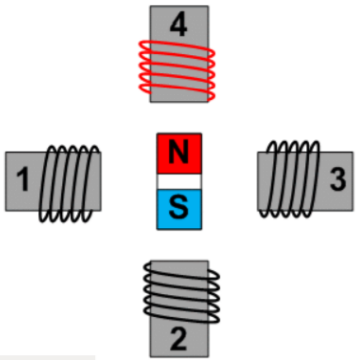
Step Number	Coil 1	Coil 2	Coil 3	Coil 4
1	OFF	OFF	OFF	ON
2	OFF	OFF	ON	OFF
3	OFF	ON	OFF	OFF
4	ON	OFF	OFF	OFF

Double stepping



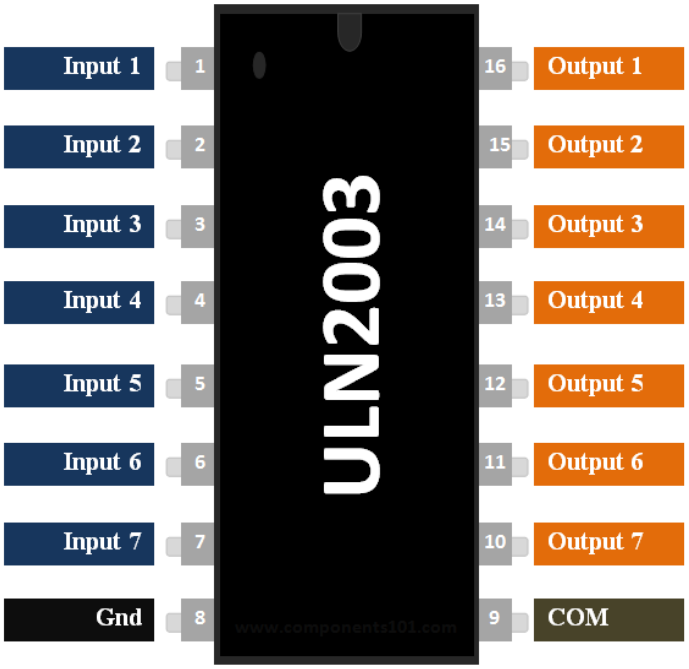
Step Number	Coil 1	Coil 2	Coil 3	Coil 4
1	ON	ON	OFF	OFF
2	OFF	ON	ON	OFF
3	OFF	OFF	ON	ON
4	ON	OFF	OFF	ON

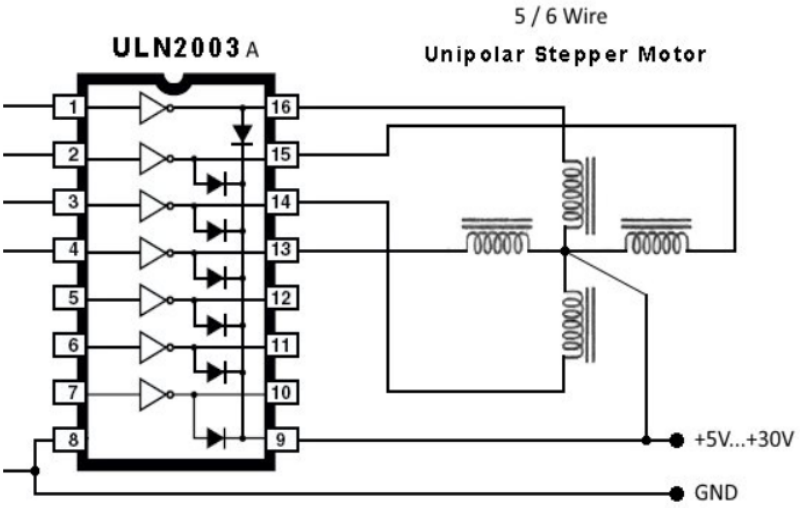
Half Stepping



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



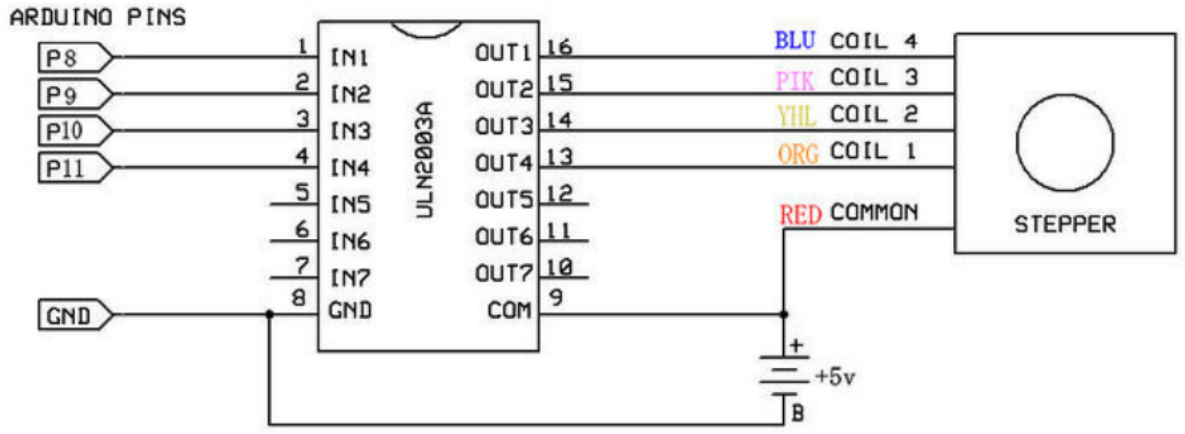


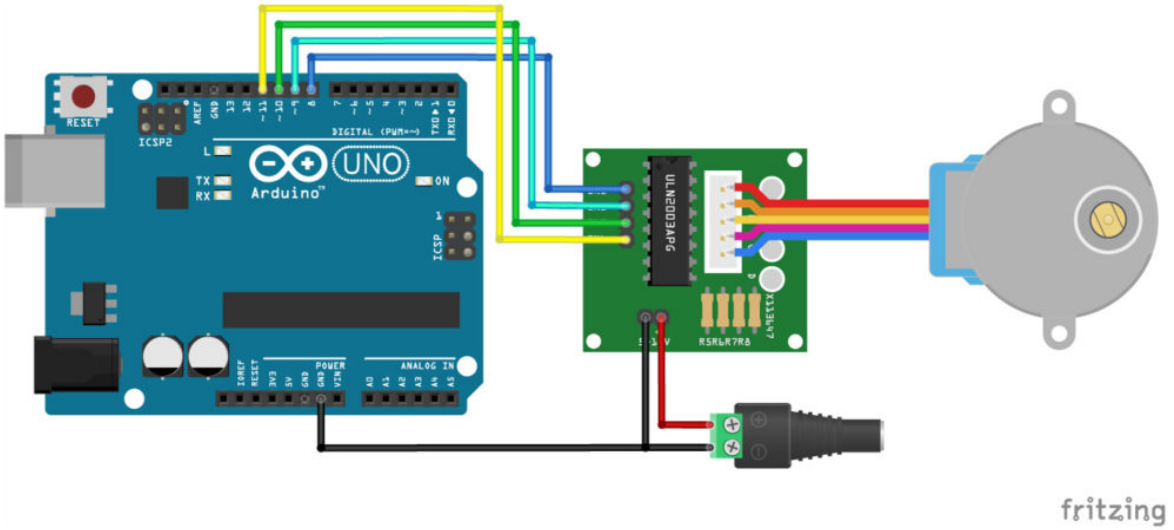
https://en.wikipedia.org/wiki/ULN2003A

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





We won’t be using an external power source for these examples, **although that is usually desirable.**