Introduction to robotics

8th lab

What we'll do today

1. 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

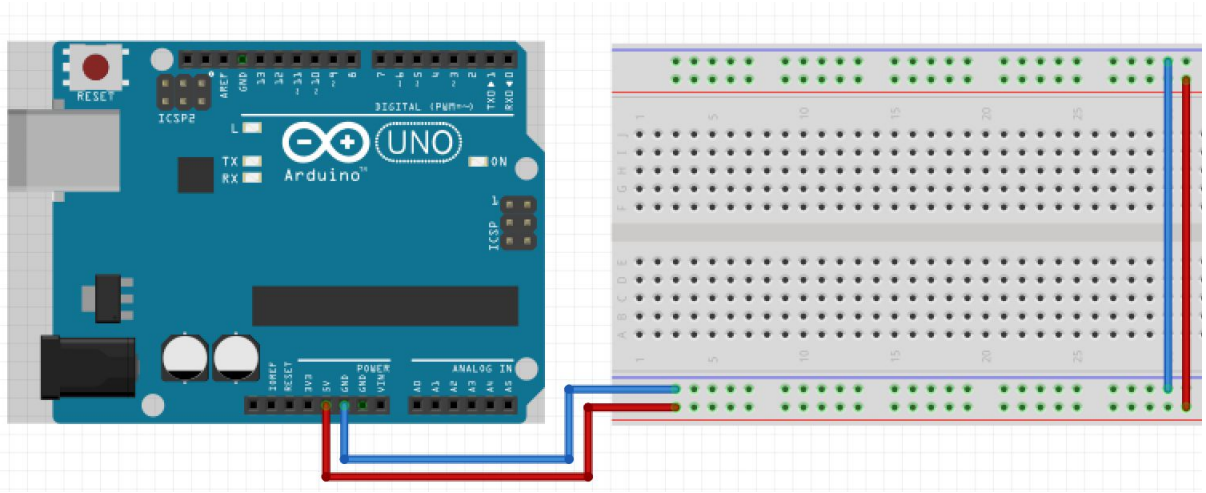
Remember, when possible, choose the wire color accordingly:

- **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 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.

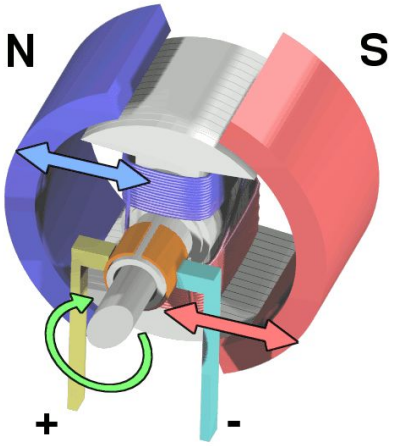


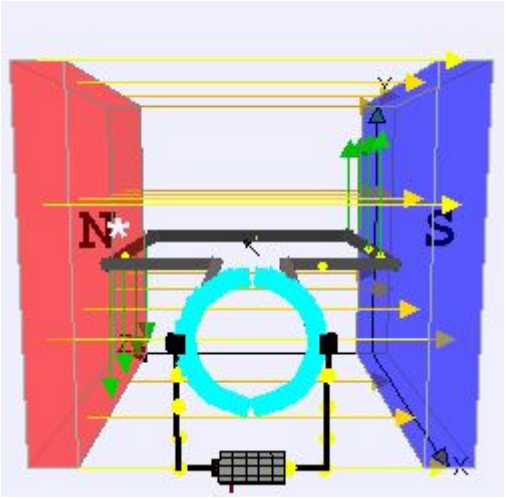
1. DC Motor

Sure, but what is a dc motor?

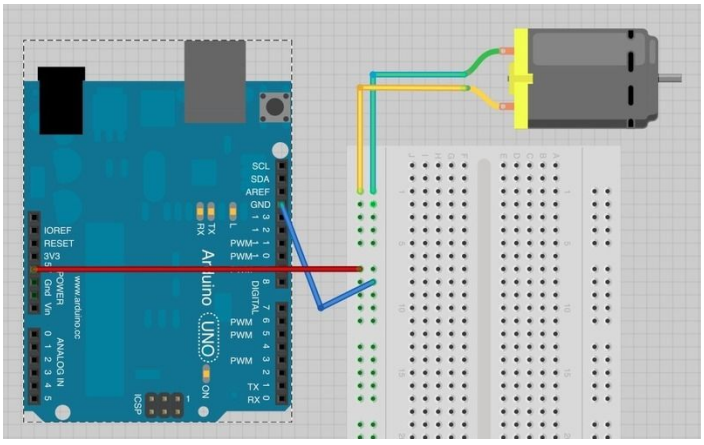
A **DC motor** is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, 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. Connecting a DC Motor without a motor driver

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

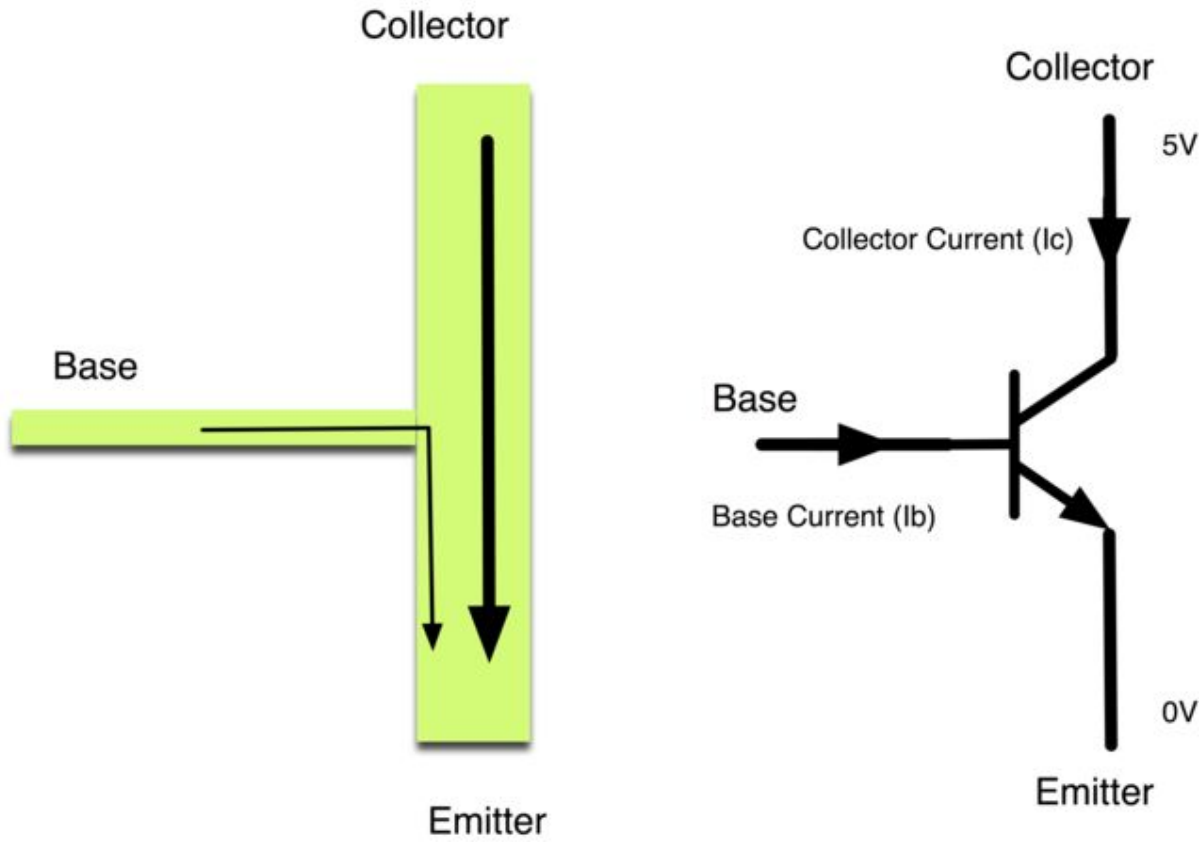
We will control a small DC motor using an Arduino and a transistor. You will use an Arduino analog output (PWM) to control the speed of the motor by sending a number between 0 and 255 from the Serial Monitor.

Required items:

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

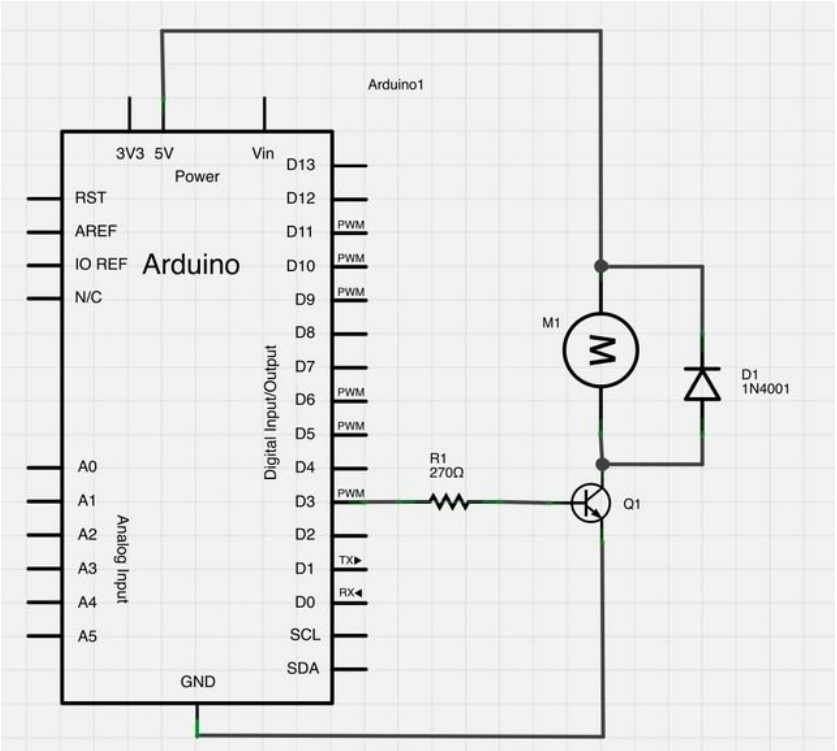
The small DC motor, is likely to use more power than an Arduino digital output can handle directly on an output pin. If we tried to connect the motor straight to an Arduino digital output pin, there is a good chance that it could damage the Arduino. A small transistor like the PN2222 can be used as a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

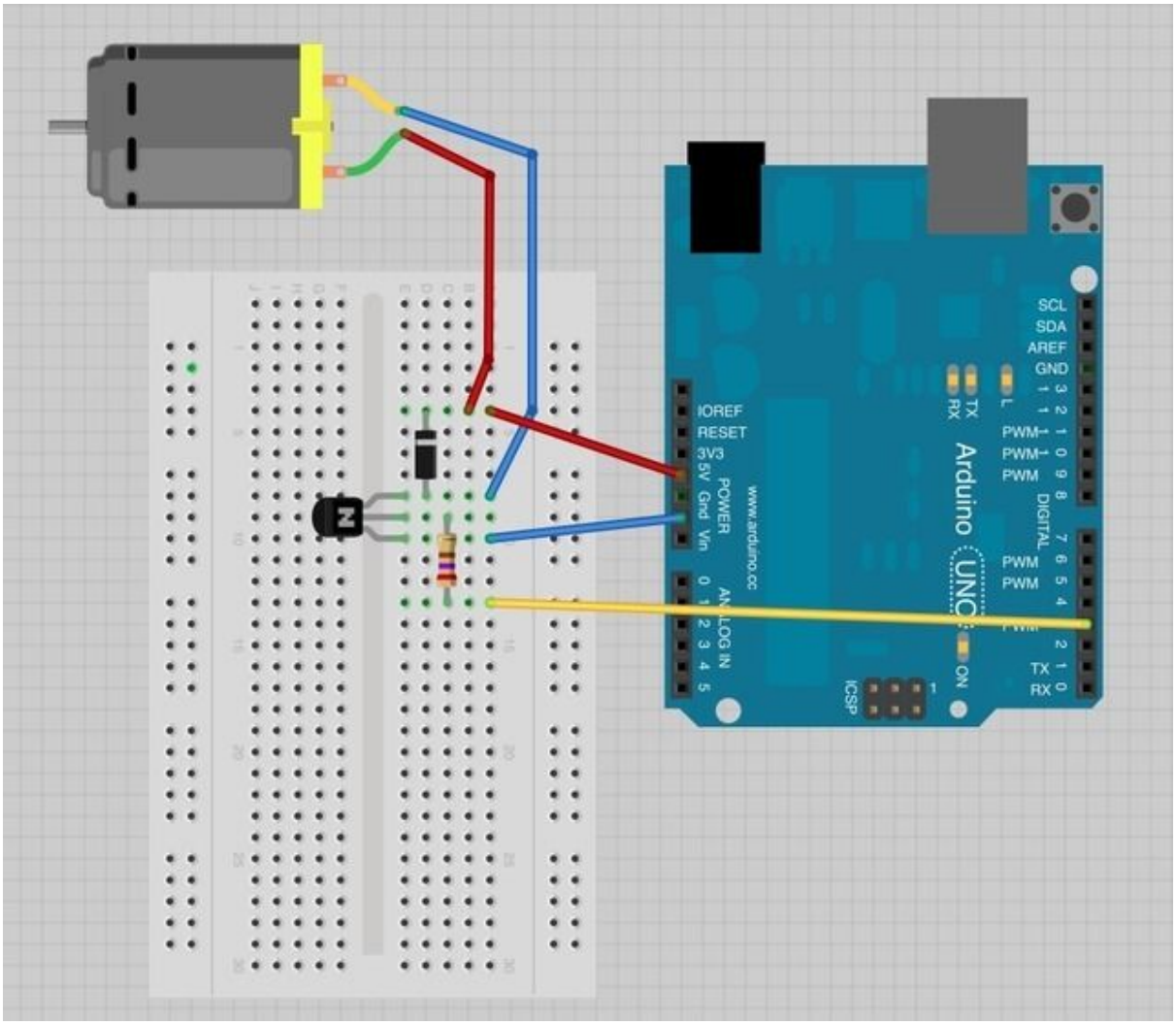
The transistor has three leads. Most of the electricity flows from the Collector to the Emitter, but this will only happen if a small amount is flowing into the Base connection. This small current is supplied by the Arduino digital output.



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 Vin
Power
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


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

```
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

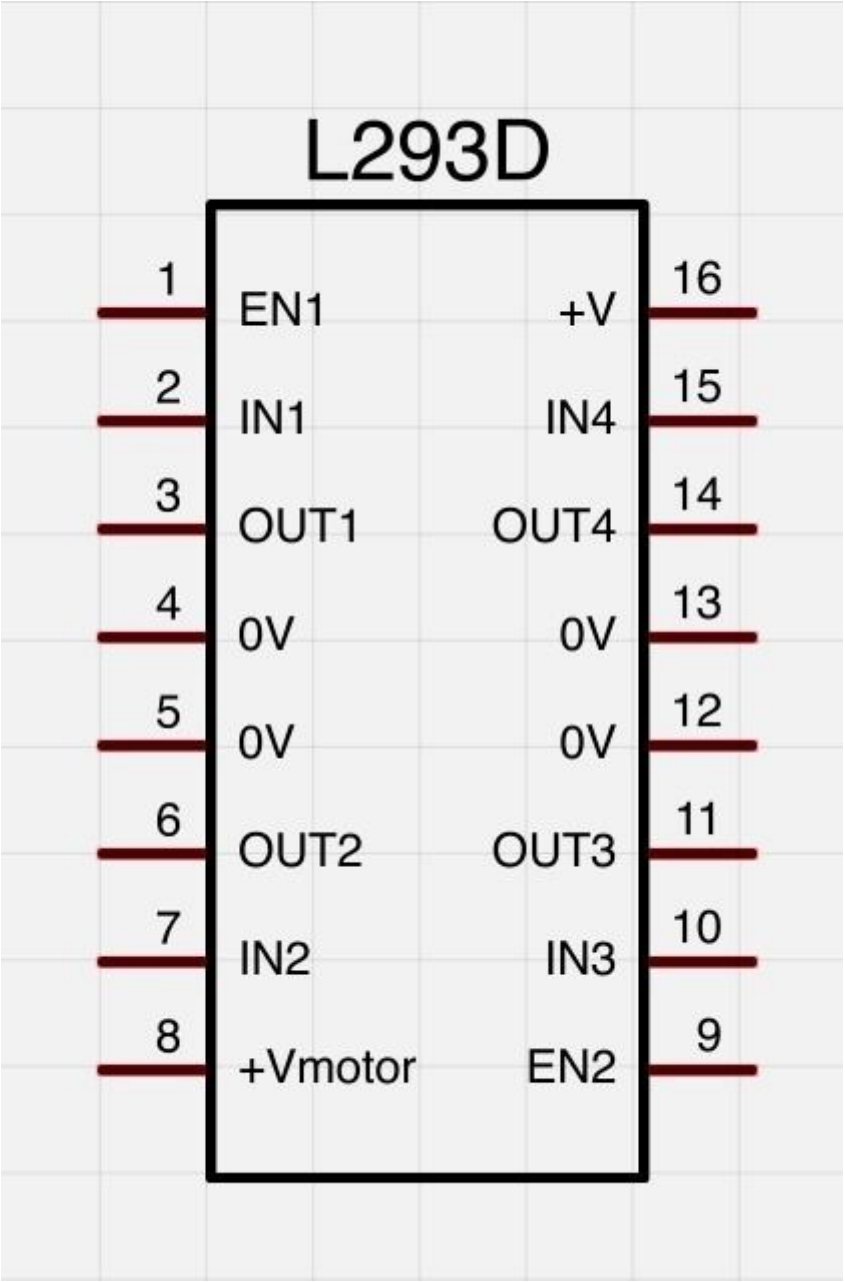
Now that we've seen how to make our own circuit to control the DC motor, let's use L293D motor driver, a H-bridge that lets us control the rotation speed, direction of two motors at the same time.

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

3.1 L293D Motor Driver

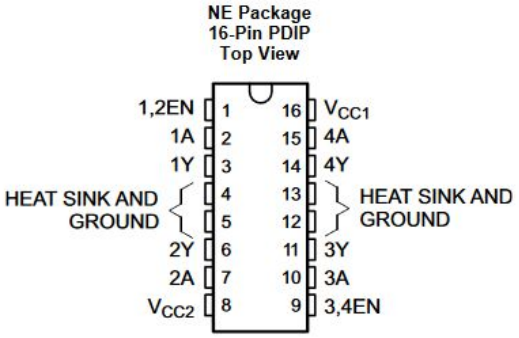
http://www.ti.com/lit/ds/symlink/l293d.pdf

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

```
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.

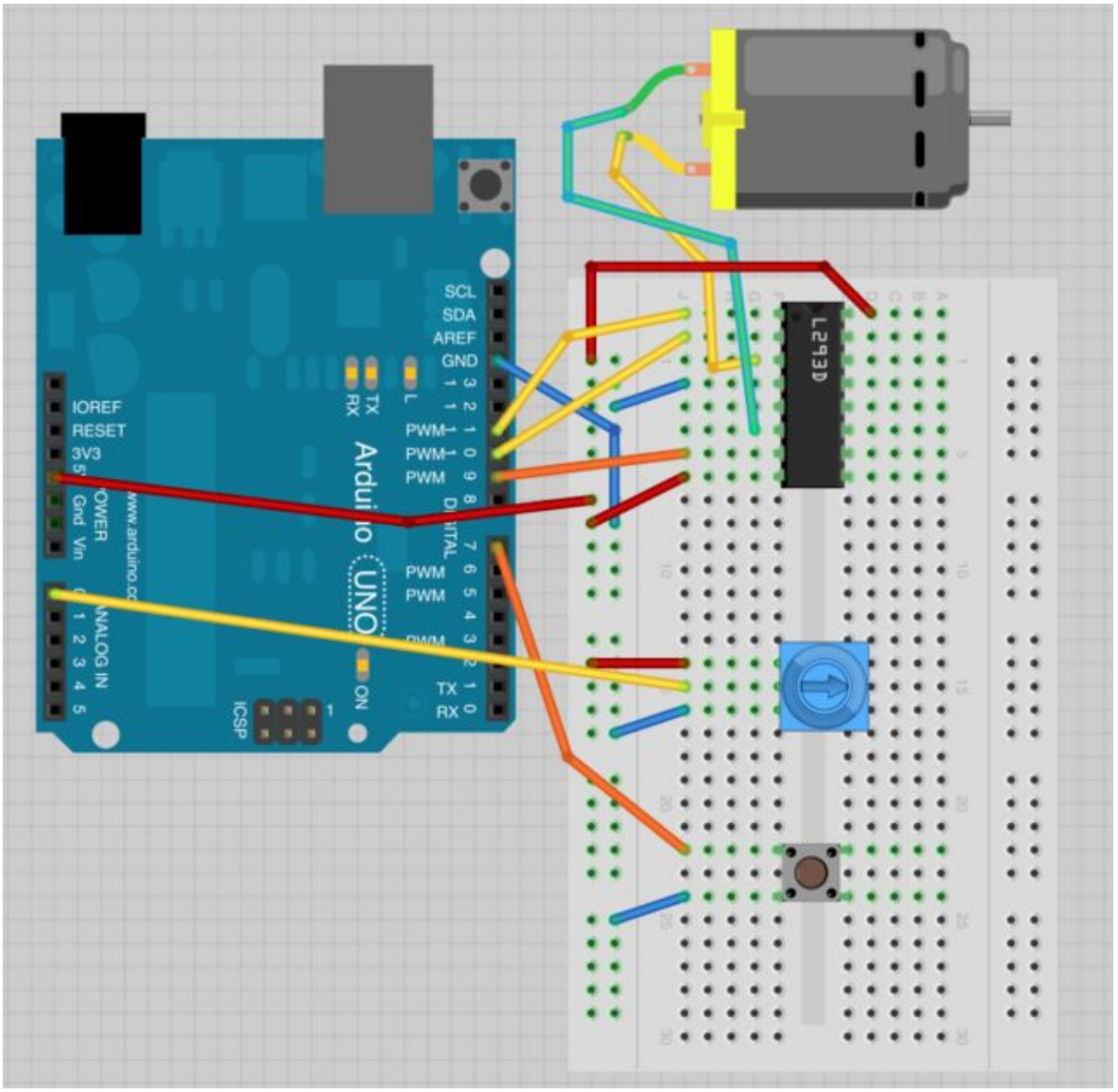
To control the direction of the motor, the pins in1 and in2 must be set to opposite values.

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.

To Do Alone: control the motor speed with a potentiometer and the motor direction with a button




```
int enablePin = 11;
int in1Pin = 10;
int in2Pin = 9;
int switchPin = 7;
int potPin = 0;

void setup()
{
 pinMode(in1Pin, OUTPUT);
```

```
 pinMode(in2Pin, OUTPUT);
 pinMode(enablePin, OUTPUT);
 pinMode(switchPin, INPUT_PULLUP);
}

void loop()
{
 int motorSpeed = analogRead(potPin) / 4;
 boolean reverse = digitalRead(switchPin);
 setMotor(motorSpeed , reverse);
}

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

Part 2: connect another motor to L293D (provided on the table)

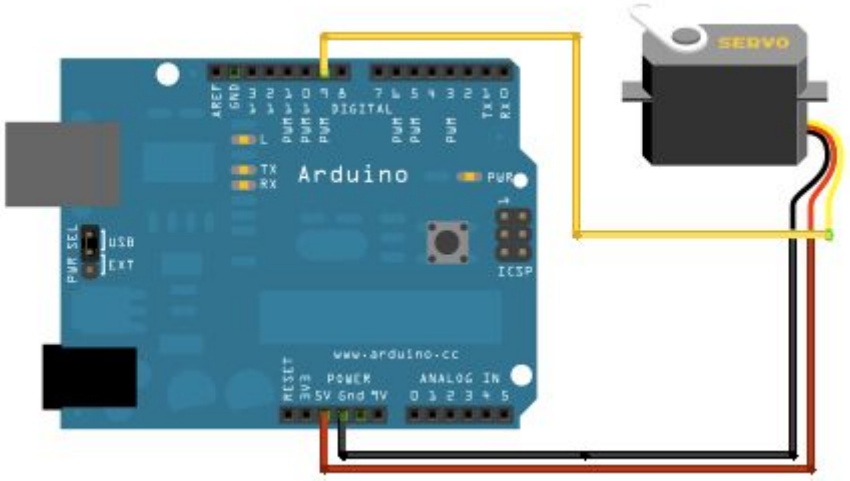
4. Servo motor

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.

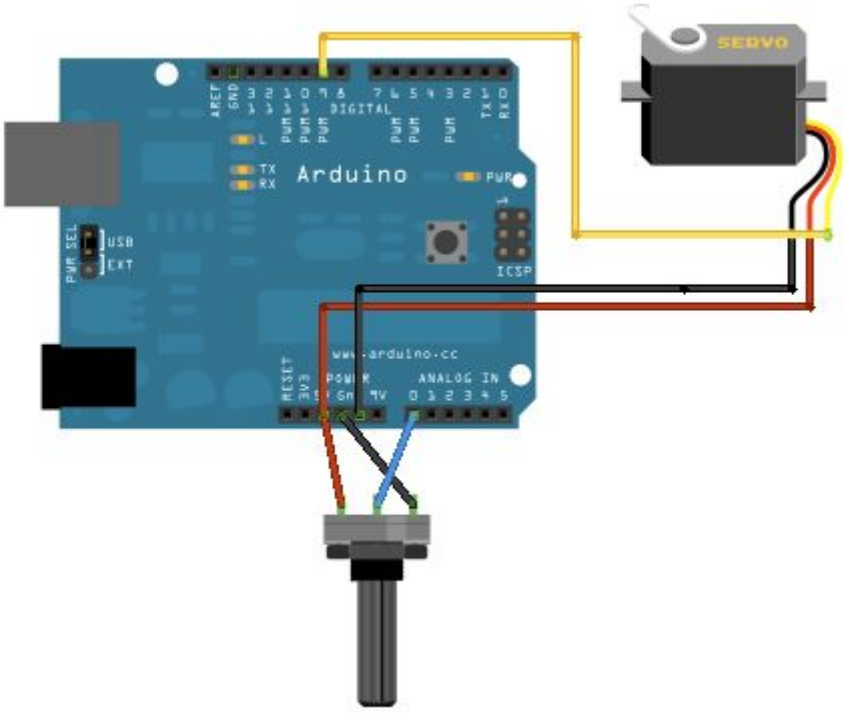
Attention! The longer lead of the capacitor is the positive lead and this should be connected to 5V. The negative lead is also often marked with a '-' symbol.

```
#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 to a particular angle.

To Do Alone: control the servo motor angle with a potentiometer

Solutin:
https://www.arduino.cc/en/Tutorial/Knob




```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin);            // reads the value of the
potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);  // scale it to use it with the servo
(value between 0 and 180)
  myservo.write(val);                  // sets the servo position according
to the scaled value
  delay(15);                           // waits for the servo to get there
}
```