MOTOR DRIVER

A motor driver is a little current amplifier; the function of motor drivers is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor. This can be done using a microcontroller or a module. Motor Driver ICs are primarily used in autonomous robotics only. Also most microprocessors operate at low voltages and require a small amount of current to operate while the motors require a relatively higher voltages and current. Thus current cannot be supplied to the motors from the microprocessor. This is the primary need for the motor driver IC.

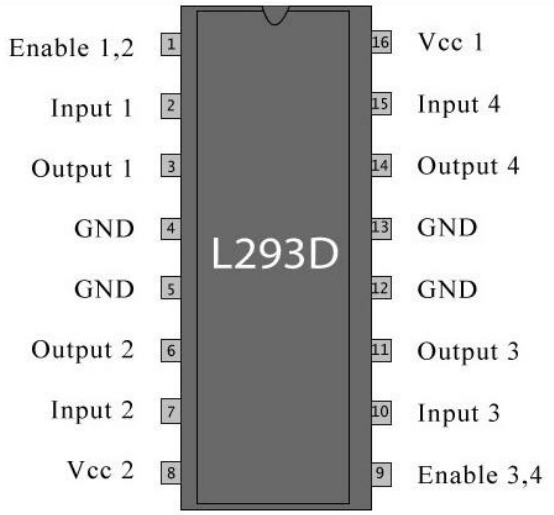
L293D Motor Driver

A motor driver IC is an integrated circuit chip which is usually used to control motors in autonomous robots. Motor driver ICs act as an interface between microprocessors in robots and the motors in the robot. The most commonly used motor driver IC's are from the L293 series such as L293D, L293NE, etc. These ICs are designed to control 2 DC motors simultaneously. L293D consist of two H-bridge. H-bridge is the simplest circuit for controlling a low current rated motor.

The L293D IC receives signals from the microprocessor and transmits the relative signal to the motors. It has two voltage pins, one of which is used to draw current for the working of the L293D and the other is used

to apply voltage to the motors. The L293D switches it output signal according to the input received from the microprocessor.

IC Layout of L293D

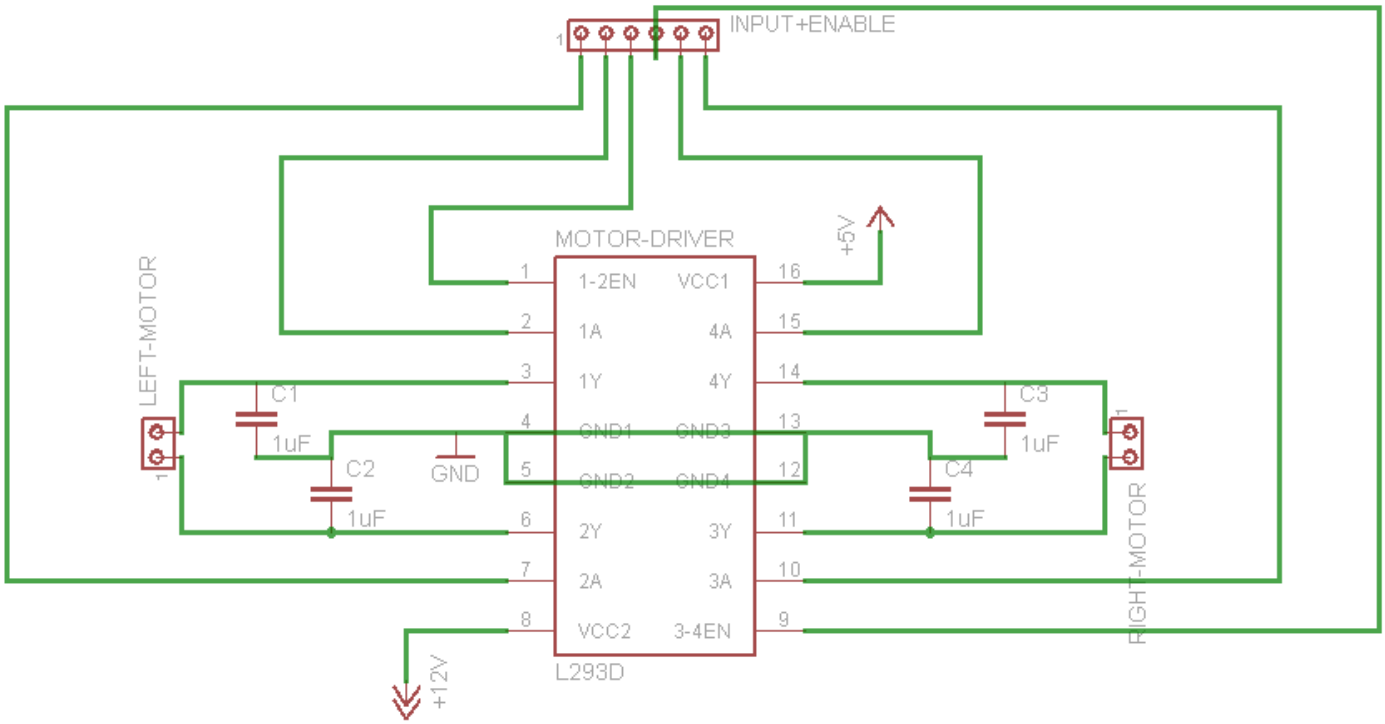


Pin Configuration

Pin No.	Pin Characteristics
1	Enable 1-2, when this is HIGH the left part of the IC will work and when it is low the left part won't work. So, this is the Master Control pin for the left part of IC
2	INPUT 1, when this pin is HIGH the current will flow though output 1
3	OUTPUT 1, this pin should be connected to one of the terminal of motor
4,5	GND, ground pins
6	OUTPUT 2, this pin should be connected to one of the terminal of motor
7	INPUT 2, when this pin is HIGH the current will flow though output 2
8	VC, this is the voltage which will be supplied to the motor. So, if you are driving 12 V DC motors then make sure that this pin is supplied with 12 V
16	VSS, this is the power source to the IC. So, this pin should be supplied with 5 V
15	INPUT 4, when this pin is HIGH the current will flow though output 4
14	OUTPUT 4, this pin should be connected to one of the terminal of motor
13,12	GND, ground pins
11	OUTPUT 3, this pin should be connected to one of the terminal of motor
10	INPUT 3, when this pin is HIGH the current will flow though output 3
9	Enable 3-4, when this is HIGH the right part of the IC will work and when it is low the right part won't work. So, this is the Master Control pin for the right part of IC

IC on PCB:

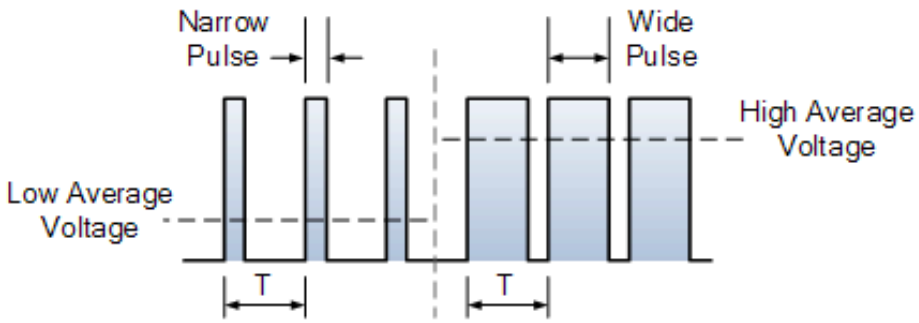
Given below is the circuit diagram for how the IC needs to be soldered on a PCB with the connectors.



Motor Speed Control Using PWM (Pulse Width Modulation)

Pulse width modulation speed control works by driving the motor with a series of “ON-OFF” pulses and varying the duty cycle, the fraction of time that the output voltage is “ON” compared to when it is “OFF”, of the pulses while keeping the frequency constant.

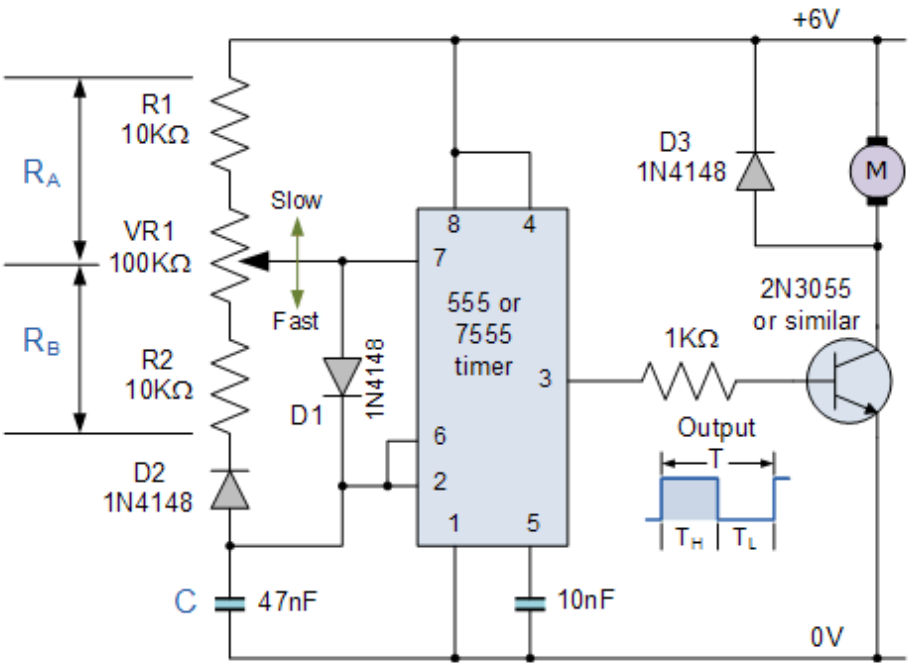
The power applied to the motor can be controlled by varying the width of these applied pulses and thereby varying the average DC voltage applied to the motors terminals. By changing or modulating the timing of these pulses the speed of the motor can be controlled, i.e, the longer the pulse is “ON”, the faster the motor will rotate and likewise, the shorter the pulse is “ON” the slower the motor will rotate.



Pulse Width Modulation Waveform

How Can We Produce a Pulse Width Modulation Signal

Use an Astable 555 Oscillator circuit. This simple circuit based around the familiar NE555 or 7555 timer chip is used to produce the required pulse width modulation signal at a fixed frequency output. The timing capacitor C is charged and discharged by current flowing through the timing networks RA and RB as we looked at in the 555 Timer tutorial.



Circuit Diagram for PWM Signal Generator

The output signal at pin 3 of the 555 is equal to the supply voltage switching the transistors fully “ON”. The time taken for C to charge or discharge depends upon the values of RA, RB.

The capacitor charges up through the network RA but is diverted around the resistive network RB and through diode D1. As soon as the capacitor is charged, it is immediately discharged through diode D2 and network RB into pin 7. During the discharging process the output at pin 3 is at 0 V and the transistor is switched “OFF”.

Then the time taken for capacitor, C to go through one complete charge-discharge cycle depends on the values of RA, RB and C with the time T for one complete cycle being given as:

The time, TH, for which the output is “ON” is: $TH = 0.693(RA).C$

The time, TL, for which the output is “OFF” is: $TL = 0.693(RB).C$

Total “ON”-“OFF” cycle time given as: $T = TH + TL$ with the output frequency being $f = 1/T$

With the component values shown, the duty cycle of the waveform can be adjusted from about 8.3% (0.5V) to about 91.7% (5.5V) using a 6.0V power supply. The Astable frequency is constant at about 256 Hz and the motor is switched “ON” and “OFF” at this rate.

Resistor R1 plus the “top” part of the potentiometer, VR1 represent the resistive network of RA. While the “bottom” part of the potentiometer plus R2 represent the resistive network of RB above.

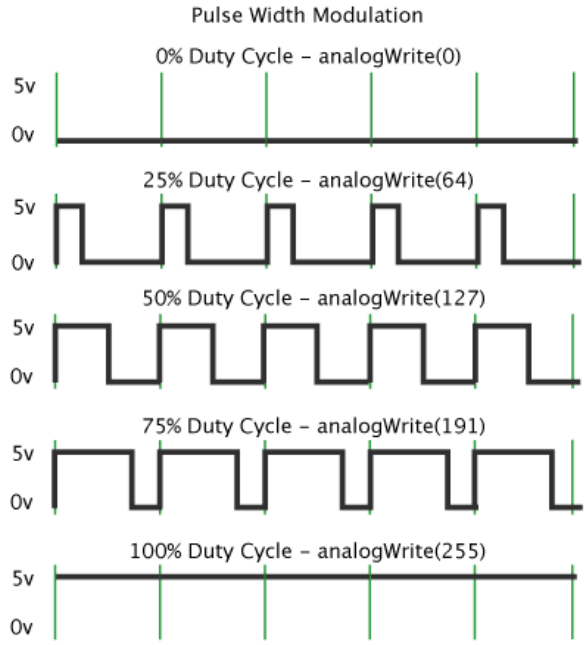
These values can be changed to suite different applications and DC motors but providing that the 555 Astable circuit runs fast enough at a few hundred Hertz minimum, there should be no jerkiness in the rotation of the motor.

Diode D3 is our old favourite the flywheel diode used to protect the electronic circuit from the inductive loading of the motor. Also if the motor load is high put a heatsink on the switching transistor or MOSFET.

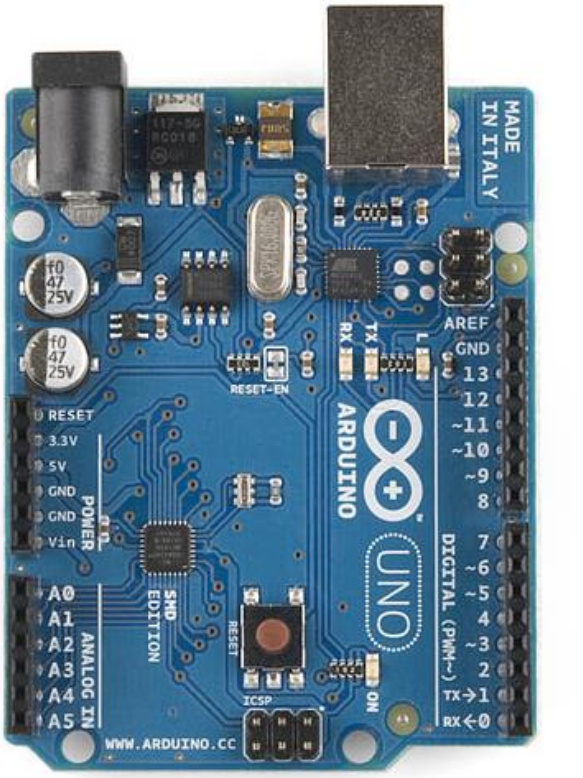
Another way to produce PWM signal is using Arduino

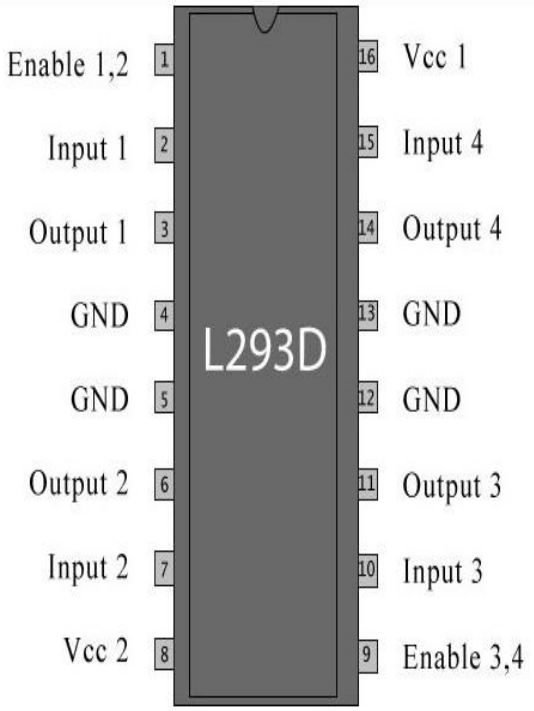
Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.



Operation of L293D Using Arduino





L293D PINS CONNECTION

PIN1(Enable1) -- DigitalPin11(PWM)

PIN2(INPUT1) -- DigitalPin10(PWM)

PIN3(OUTPUT1) -- MOTOR PIN 1

PIN4(GND1) -- Gnd

PIN6(OUTPUT2) -- MOTOR PIN 2

PIN7(INPUT2) -- DigitalPin9(PWM)

PIN8(12v) -- 5v

PIN9(Vss3.3v) -- 5v/// the pin says 3.3v but u can connect it to 5v also.

POTENTIOMETER PINS CONNECTION

Pin1(positive)-- 5v

Pin2(signal) -- analogPIN0

PIn3(negative) -- Gnd

PUSH BUTTON PINS CONNECTION

1PIN= Gnd

2PIN = DigitalPIN7

Arduino Code:

```
int enablePin = 11;

int in1Pin = 10;

int in2Pin = 9;

int switchPin = 7;

int potPin = 0;

int statusPin= 13;

void setup()

{

pinMode(in1Pin, OUTPUT);

pinMode(in2Pin, OUTPUT);

pinMode(enablePin, OUTPUT);

pinMode(switchPin, INPUT_PULLUP);

pinMode(statusPin,OUTPUT);

}

void loop()

{

digitalWrite(13,HIGH);

int speed = analogRead(potPin) / 4;

boolean reverse = digitalRead(switchPin);

setMotor(speed, reverse);
```

```
}

void setMotor(int speed, boolean reverse)

{

analogWrite(enablePin, speed);

digitalWrite(in1Pin, ! reverse);

digitalWrite(in2Pin, reverse);

}
```