Interface L298N DC Motor Driver Module with Arduino

Last Minute Engineers

12–15 minutes

While you'll eventually need to learn to control DC motors in order to build your own robot, you'll probably need something a little easier to get started – which is where the L298N motor driver comes in. It can control the speed and spinning direction of two DC motors.

In addition, it can control a bipolar stepper motor, such as the NEMA 17. If you want to learn more about it, check out this tutorial.

Controlling a DC Motor

We can only have full control over a DC motor if we can control its speed and spinning direction. This is possible by combining these two techniques.

- **PWM** – to control speed
- **H-Bridge** – to control the spinning direction

Let's learn more about these techniques.

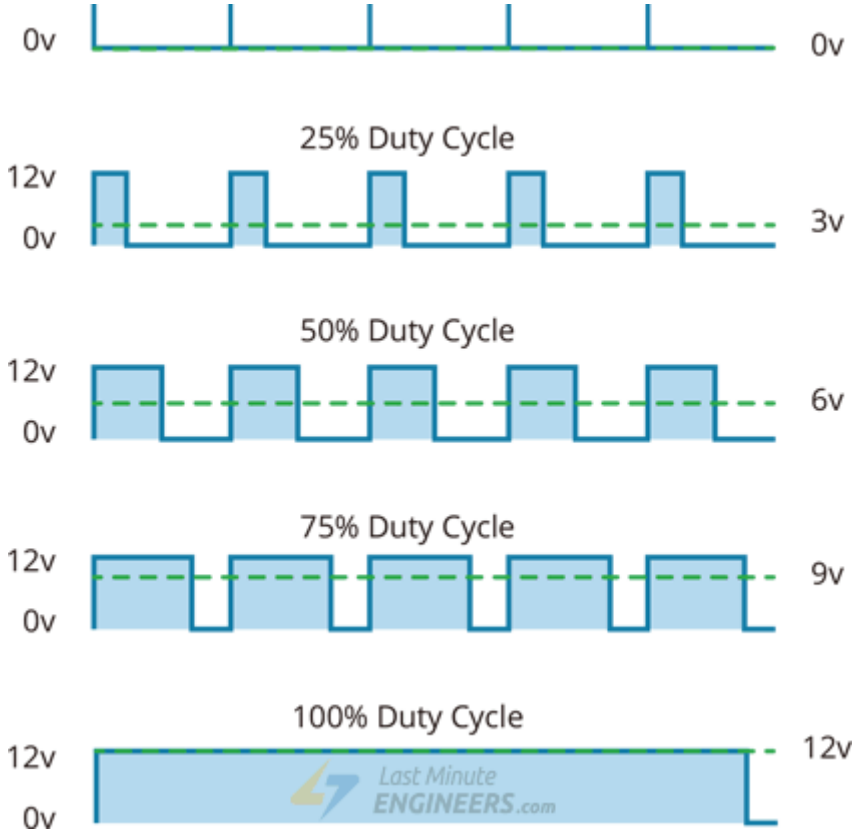
PWM – to control speed

The speed of a DC motor can be controlled by changing its input voltage. A widely used technique to accomplish this is Pulse Width Modulation (PWM).

PWM is a technique in which the average value of the input voltage is adjusted by sending a series of ON-OFF pulses. This average voltage is proportional to the width of the pulses, which is referred to as the **Duty Cycle**.

The higher the duty cycle, the higher the average voltage applied to the DC motor, resulting in an increase in motor speed. The shorter the duty cycle, the lower the average voltage applied to the DC motor, resulting in a decrease in motor speed.

The image below shows PWM technique with various duty cycles and average voltages.



Pulse Width Modulation(PWM) Technique

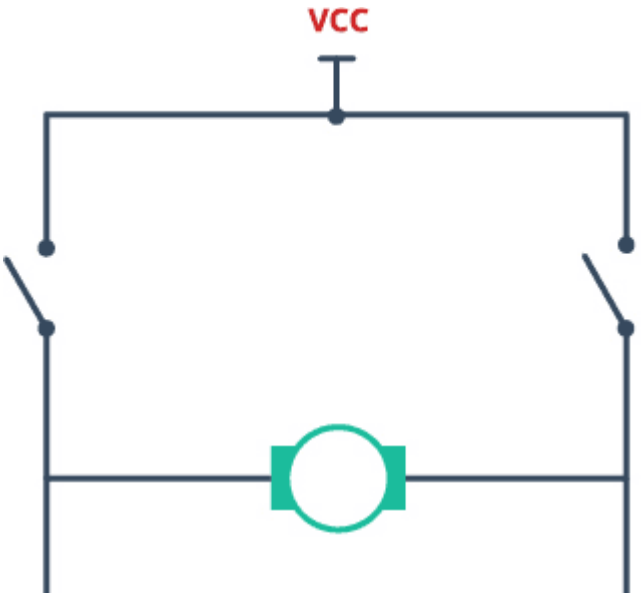
H-Bridge – to control the spinning direction

The spinning direction of a DC motor can be controlled by changing the polarity of its input voltage. A widely used technique to accomplish this is to use an H-bridge.

An H-bridge circuit is made up of four switches arranged in a H shape, with the motor in the center.

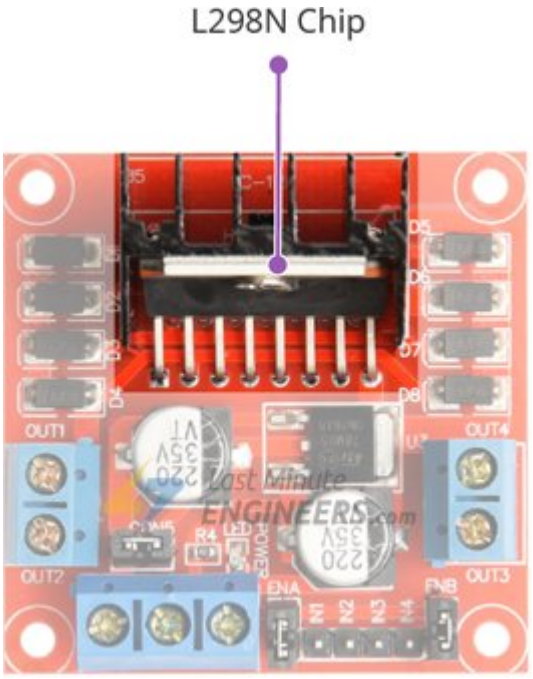
Closing two specific switches at the same time reverses the polarity of the voltage applied to the motor. This causes a change in the spinning direction of the motor.

The following animation shows the working of the H-bridge circuit.



Working of H-Bridge

At the center of the module is a big, black chip with a chunky heat sink – the L298N.



The L298N chip contains two standard H-bridges capable of driving a pair of DC motors, making it ideal for building a two-wheeled robotic platform.

The L298N motor driver has a supply range of 5V to 35V and is capable of 2A continuous current per channel, so it works very well with most of our DC motors.

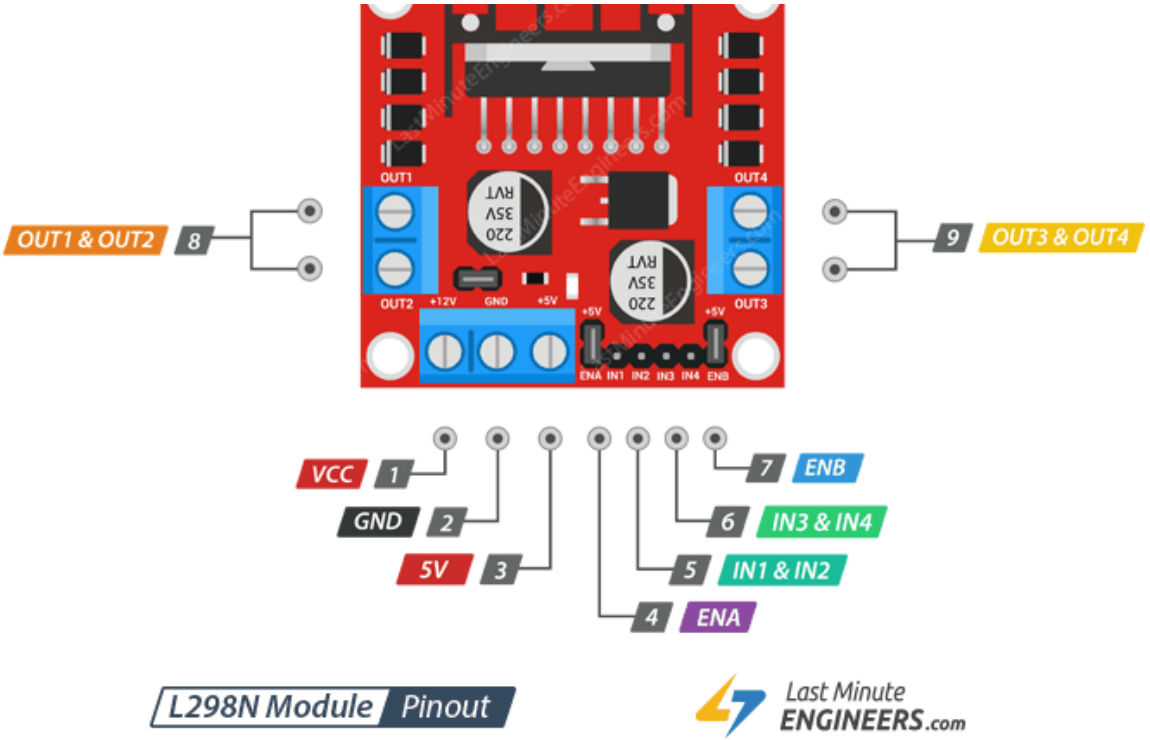
Technical Specifications

Here are the specifications:

For more details, please refer below datasheet.

L298N Motor Driver Module Pinout

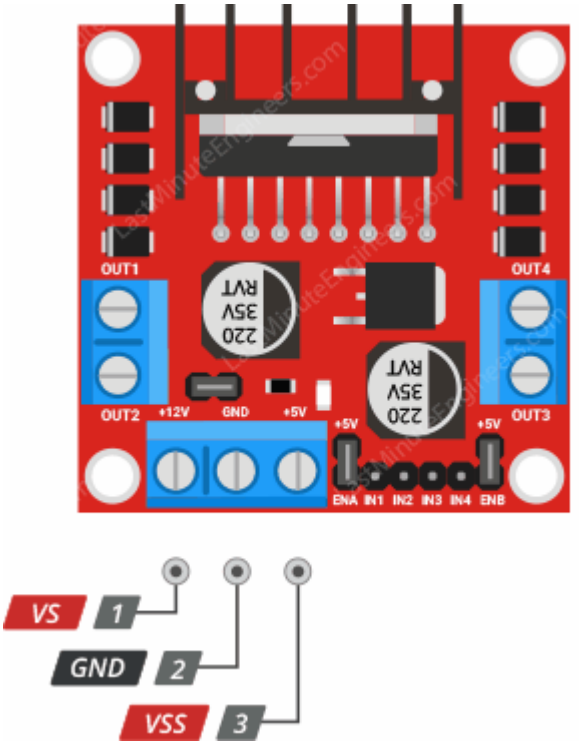
The L298N module has 11 pins that allow it to communicate with the outside world. The pinout is as follows:



Let's get acquainted with each pin one by one.

Power Pins

The L298N motor driver module receives power from a 3-pin, 3.5mm-pitch screw terminal.



The L298N motor driver has two input power pins: VS and VSS.

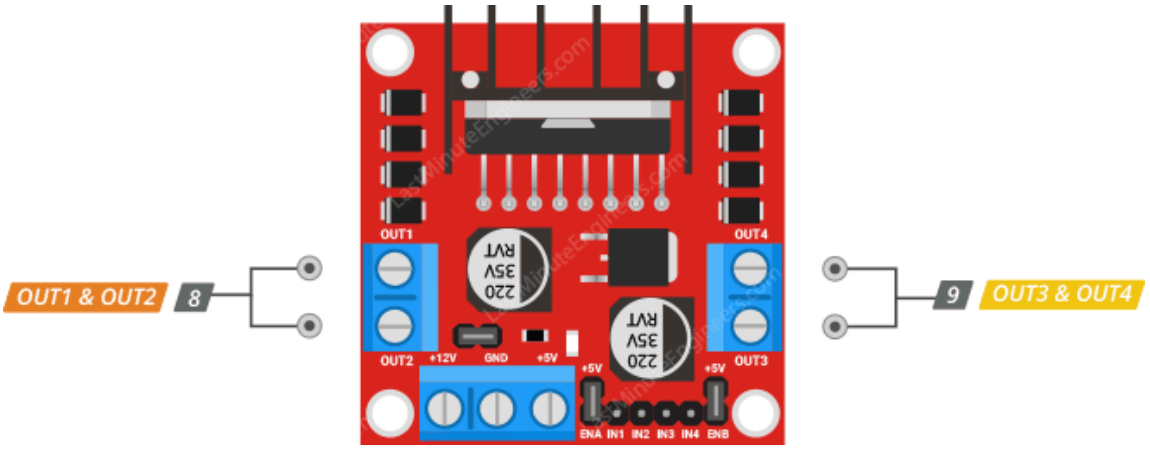
VS pin powers the IC's internal H-Bridge, which drives the motors. This pin accepts input voltages ranging from 5 to 12V.

VSS is used to power the logic circuitry within the L298N IC, and can range between 5V and 7V.

GND is the common ground pin.

Output Pins

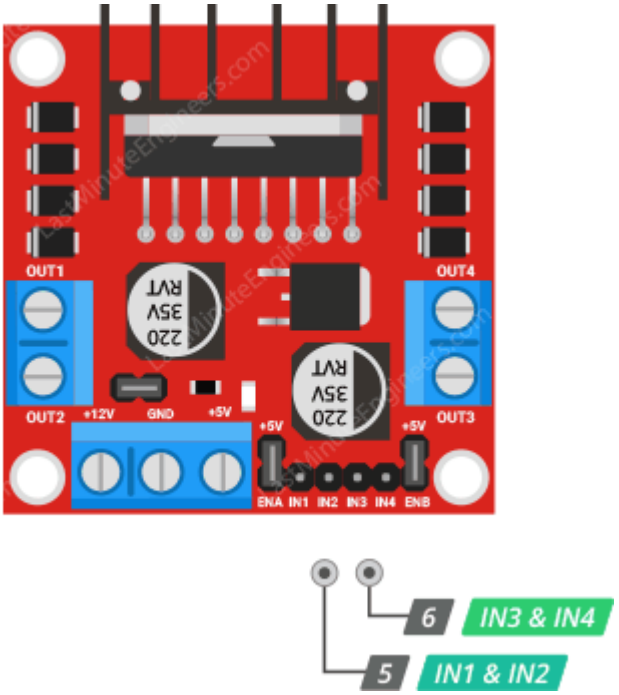
The output channels of the L298N motor driver, OUT1 and OUT2 for motor A and OUT3 and OUT4 for motor B, are broken out to the edge of the module with two 3.5mm-pitch screw terminals. You can connect two 5-12V DC motors to these terminals.



Each channel on the module can supply up to 2A to the DC motor. The amount of current supplied to the motor, however, depends on the capacity of the motor power supply.

Direction Control Pins

The direction control pins allow you to control whether the motor rotates forward or backward. These pins actually control the switches of the H-Bridge circuit within the L298N chip.

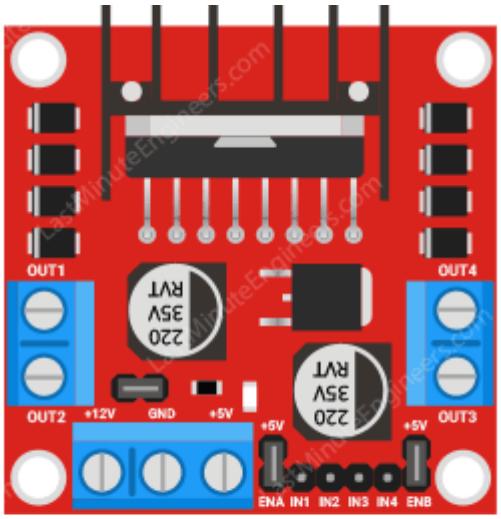


The module has two direction control pins. The IN1 and IN2 pins control the spinning direction of motor A; While IN3 and IN4 control the spinning direction of motor B.

The spinning direction of the motor can be controlled by applying logic HIGH (5V) or logic LOW (Ground) to these inputs. The chart below shows various combinations and their outcomes.

Speed Control Pins

The speed control pins ENA and ENB are used to turn on/off the motors and control their speed.

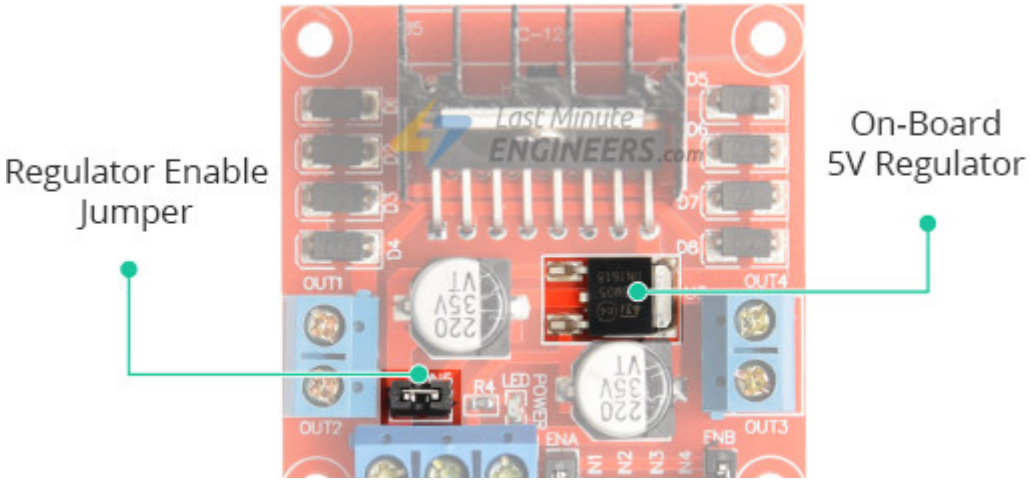


Pulling these pins HIGH will cause the motors to spin, while pulling them LOW will stop them. However, with Pulse Width Modulation (PWM), the speed of the motors can be controlled.

The module usually comes with a jumper on these pins. When this jumper is in place, the motor spins at full speed. If you want to control the speed of the motors programmatically, remove the jumpers and connect them to the Arduino's PWM-enabled pins.

On-board 5V Regulator and Jumper

The module includes a 78M05 5V regulator that can be enabled or disabled via a jumper.



When this jumper is in place, the 5V regulator is enabled, and the logic power supply (VSS) is derived from the motor power supply (VS). In this case, the 5V input terminal acts as the output pin, delivering 5V 0.5A. You can use it to power an Arduino or other circuitry that needs 5V power.

When the jumper is removed, the 5V regulator is disabled, and we have to supply 5V separately through the VSS pin.

Warning:

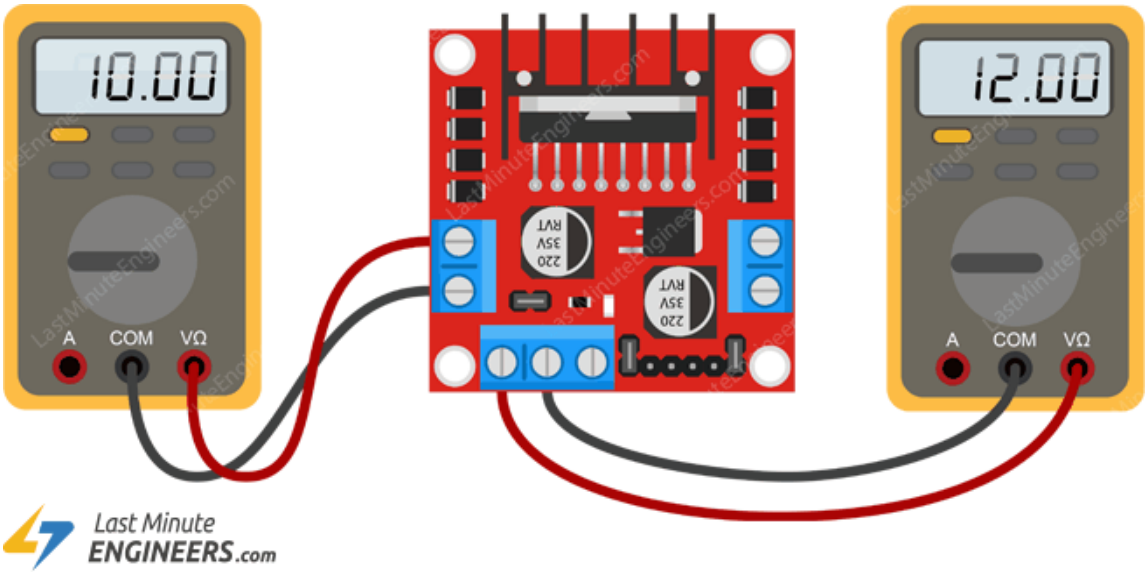
If the motor power supply is less than 12V, you can keep the jumper in place. If it is greater than 12V, the jumper must be removed to prevent damage to the onboard 5V regulator.

Also, do not supply power to both the VSS and VS pins while the jumper is in place.

Voltage Drop of L298N

The L298N has a voltage drop of approximately 2V. This is due to the fact that internal switching transistors have a voltage drop of approximately 1V when forward biased, and because an H-Bridge requires the current to pass through two transistors, the total voltage drop is 2V.

So, if you connect 12V to the motor power supply terminal, the motors will receive approximately 10V. This means that a 12V DC motor will never spin at full speed.



In order to get the motor to run at its maximum speed, the motor power supply should have a voltage that is slightly higher (+2V) than the actual voltage requirement of the motor.

Taking into account a voltage drop of 2V, if you are using 5V motors, you will need to provide 7V at the motor power supply terminal. If you have 12V motors then your motor supply voltage should be 14V.

This excess voltage drop results in significant power dissipation in the form of heat. This is why the L298N based motor drivers require a big

heatsink.

Wiring an L298N Motor Driver Module to an Arduino

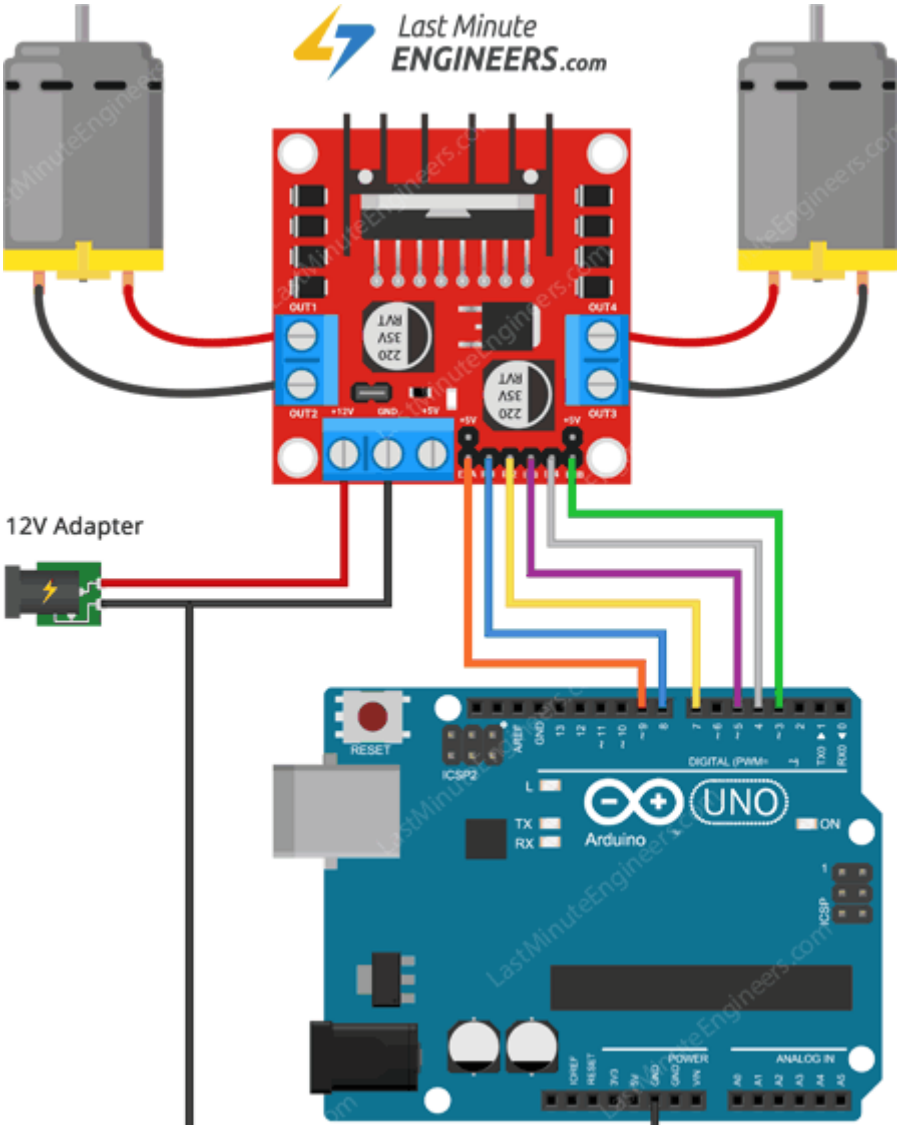
Now that we know everything about the module, we can start hooking it up to our Arduino!

Let's begin by connecting the motor power supply. In our experiment, we are using DC gearbox motors, also called "TT" motors, which are often found in two-wheel-drive robots. They are rated for 3 to 12V. We will therefore connect an external 12V power source to the VS terminal. Because L298N has a voltage drop of about 2V, the motors will receive 10V and spin at a slightly lower RPM. But that's okay.

Next, we need to supply 5V to the logic circuitry of the L298N. We'll use the on-board 5V regulator to draw 5V from the motor power supply, so keep the 5V-EN jumper in place.

Now connect the L298N module's Input and Enable pins (ENA, IN1, IN2, IN3, IN4 and ENB) to the six Arduino digital output pins (9, 8, 7, 5, 4 and 3). Note that both Arduino output pins 9 and 3 are PWM-enabled.

Finally, wire one motor to terminal A (OUT1 and OUT2) and the other to terminal B (OUT3 and OUT4). You can swap out your motor's connections. There is technically no right or wrong way.



Arduino Example Code

The sketch below will show you how to control the speed and spinning direction of a DC motor using the L298N Motor Driver and can serve as the basis for more practical experiments and projects.

The sketch moves the motor in one direction for one revolution, then in the opposite direction. There is also some acceleration and deceleration involved.

When accelerating or decelerating the motor, you may hear it humming, especially at lower PWM values. This is normal; there is nothing to be concerned about. This happens because the DC motor requires a minimum amount of voltage to operate.

```
// Motor A connections
int enA = 9;
int in1 = 8;
int in2 = 7;
// Motor B connections
int enB = 3;
int in3 = 5;
int in4 = 4;

void setup() {
        // Set all the motor control pins to outputs
        pinMode(enA, OUTPUT);
        pinMode(enB, OUTPUT);
        pinMode(in1, OUTPUT);
        pinMode(in2, OUTPUT);
        pinMode(in3, OUTPUT);
        pinMode(in4, OUTPUT);

        // Turn off motors - Initial state
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
}

void loop() {
        directionControl();
        delay(1000);
        speedControl();
        delay(1000);
}

// This function lets you control spinning direction
of motors
void directionControl() {
        // Set motors to maximum speed
        // For PWM maximum possible values are 0 to
255
```

```
        analogWrite(enA, 255);
        analogWrite(enB, 255);

        // Turn on motor A & B
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
        digitalWrite(in3, HIGH);
        digitalWrite(in4, LOW);
        delay(2000);

        // Now change motor directions
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);
        delay(2000);

        // Turn off motors
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
}

// This function lets you control speed of the
motors
void speedControl() {
        // Turn on motors
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);

        // Accelerate from zero to maximum speed
        for (int i = 0; i < 256; i++) {
                analogWrite(enA, i);
                analogWrite(enB, i);
                delay(20);
        }

        // Decelerate from maximum speed to zero
        for (int i = 255; i >= 0; --i) {
                analogWrite(enA, i);
                analogWrite(enB, i);
                delay(20);
        }

        // Now turn off motors
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
}
```

Code Explanation:

The Arduino code is fairly simple. It does not require any libraries to work. The sketch starts by declaring the Arduino pins that are connected to the L298N's control pins.

```
// Motor A connections
int enA = 9;
int in1 = 8;
int in2 = 7;
// Motor B connections
int enB = 3;
int in3 = 5;
int in4 = 4;
```

In the setup section of the code, all of the motor control pins, including the direction and speed control pins, are configured as digital OUTPUT. And the direction control pins are pulled LOW to initially disable both motors.

```
void setup() {
        // Set all the motor control pins to outputs
        pinMode(enA, OUTPUT);
        pinMode(enB, OUTPUT);
        pinMode(in1, OUTPUT);
        pinMode(in2, OUTPUT);
        pinMode(in3, OUTPUT);
        pinMode(in4, OUTPUT);

        // Turn off motors - Initial state
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
}
```

In the loop section of the code, we call two user-defined functions with a one-second delay.

```
void loop() {
        directionControl();
        delay(1000);
        speedControl();
        delay(1000);
}
```

These functions are:

- `directionControl()` – This function causes both motors to spin at full speed for two seconds. It then reverses the spinning direction of the motors and spins for two seconds. Finally, it stops the motors.

```
void directionControl() {
        // Set motors to maximum speed
        // For PWM maximum possible values are 0 to
255
        analogWrite(enA, 255);
        analogWrite(enB, 255);

        // Turn on motor A & B
```

```
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
        digitalWrite(in3, HIGH);
        digitalWrite(in4, LOW);
        delay(2000);

        // Now change motor directions
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);
        delay(2000);

        // Turn off motors
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
}
```

- `speedControl()` – This function uses the analogWrite() function to generate a PWM signal that accelerates both motors from zero to maximum speed before decelerating them back to zero. Finally, it stops the motors.

```
void speedControl() {
        // Turn on motors
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, LOW);
        digitalWrite(in4, HIGH);

        // Accelerate from zero to maximum speed
        for (int i = 0; i < 256; i++) {
                analogWrite(enA, i);
                analogWrite(enB, i);
                delay(20);
        }

        // Decelerate from maximum speed to zero
        for (int i = 255; i >= 0; --i) {
                analogWrite(enA, i);
                analogWrite(enB, i);
                delay(20);
        }

        // Now turn off motors
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
}
```