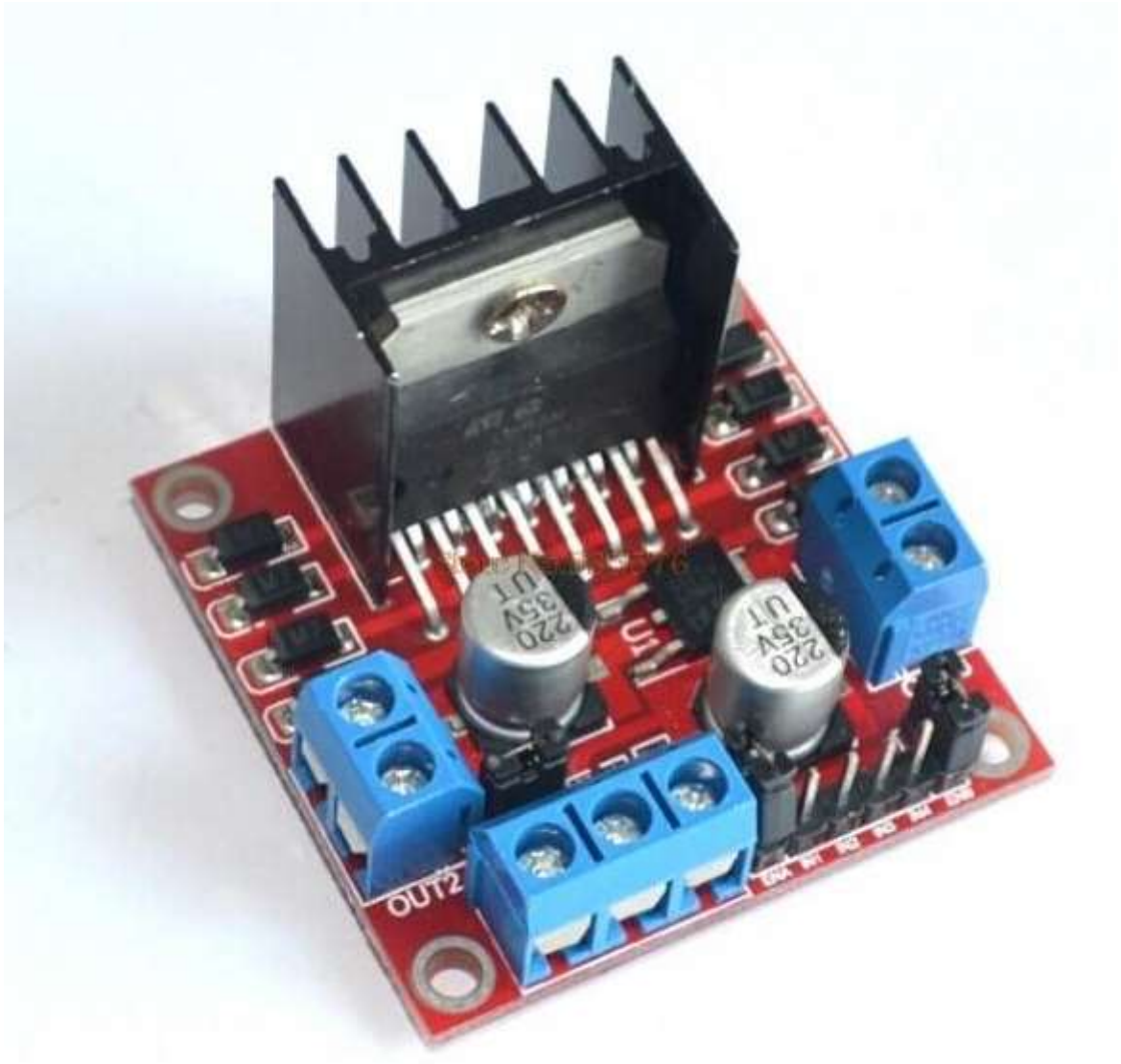
Tutorial - L298N Dual Motor Controller Module 2A and Arduino

In this tutorial we'll explain how to use our L298N H-bridge Dual Motor Controller Module 2A with Arduino. This allows you to control the speed and direction of two DC motors, or control one bipolar stepper motor with ease. The L298N H-bridge module can be used with motors that have a voltage of between 5 and 35V DC.

There is also an onboard 5V regulator, so if your supply voltage is up to 12V you can also source 5V from the board.

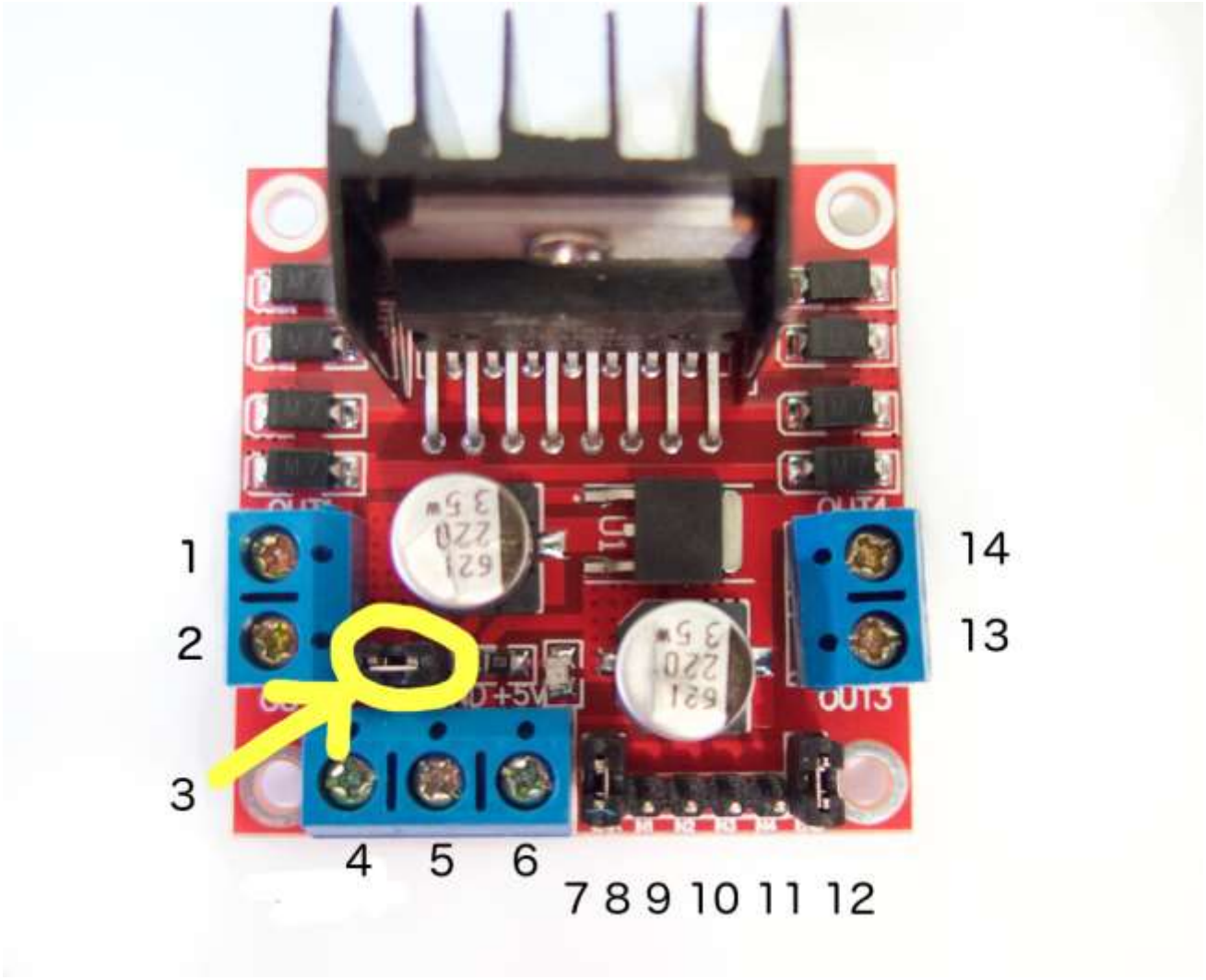
So let's get started!



First we'll run through the connections, then explain how to control DC motors then a stepper motor.

Module pinouts

Consider the following image - match the numbers against the list below the image:



1. DC motor 1 "+" or stepper motor A+
2. DC motor 1 "-" or stepper motor A-
3. 12V jumper - remove this if using a supply voltage greater than 12V DC. This enables power to the onboard 5V regulator
4. Connect your motor supply voltage here, maximum of 35V DC. Remove 12V jumper if >12V DC
5. GND
6. 5V output if 12V jumper in place, ideal for powering your Arduino (etc)
7. DC motor 1 enable jumper. Leave this in place when using a stepper motor. Connect to PWM output for DC motor speed control.
8. IN1
9. IN2
10. IN3
11. IN4
12. DC motor 2 enable jumper. Leave this in place when using a stepper motor. Connect to PWM output for DC motor speed control.

13. DC motor 2 "+" or stepper motor B+
14. DC motor 2 "-" or stepper motor B-

Controlling DC Motors

To control one or two DC motors is quite easy. First connect each motor to the A and B connections on the L298N module. If you're using two motors for a robot (etc) ensure that the polarity of the motors is the same on both inputs. Otherwise you may need to swap them over when you set both motors to forward and one goes backwards!

Next, connect your power supply - the positive to pin 4 on the module and negative/GND to pin 5. If you supply is up to 12V you can leave in the 12V jumper (point 3 in the image above) and 5V will be available from pin 6 on the module. This can be fed to your Arduino's 5V pin to power it from the motors' power supply. Don't forget to connect Arduino GND to pin 5 on the module as well to complete the circuit.

Now you will need six digital output pins on your Arduino, two of which need to be PWM (pulse-width modulation) pins. PWM pins are denoted by the tilde ("~") next to the pin number, for example:

Finally, connect the Arduino digital output pins to the driver module. In our example we have two DC motors, so digital pins D9, D8, D7 and D6 will be connected to pins IN1, IN2, IN3 and IN4 respectively. Then connect D10 to module pin 7 (remove the jumper first) and D5 to module pin 12 (again, remove the jumper).

The motor direction is controlled by sending a HIGH or LOW signal to the drive for each motor (or channel). For example for motor one, a HIGH to IN1 and a LOW to IN2 will cause it to turn in one direction, and a LOW and HIGH will cause it to turn in the other direction.

However the motors will not turn until a HIGH is set to the enable pin (7 for motor one, 12 for motor two). And they can be turned off with a LOW to the same pin(s). However if you need to control the speed of the motors, the PWM signal from the digital pin connected to the enable pin can take care of it.

This is what we've done with the DC motor demonstration sketch. Two DC motors and an Arduino Uno are connected as described above, along with an external power supply. Then enter and upload the following sketch:

```
// connect motor controller pins to Arduino digital pins
// motor one
int enA = 10;
int in1 = 9;
int in2 = 8;
// motor two
int enB = 5;
int in3 = 7;
int in4 = 6;
```

```
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);
}
void demoOne()
{
  // this function will run the motors in both directions at a fixed speed
  // turn on motor A
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // set speed to 200 out of possible range 0~255
  analogWrite(enA, 200);
  // turn on motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  // set speed to 200 out of possible range 0~255
  analogWrite(enB, 200);
  delay(2000);
  // now change motor directions
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  delay(2000);
  // now turn off motors
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void demoTwo()
{
  // this function will run the motors across the range of possible speeds
  // note that maximum speed is determined by the motor itself and the
operating voltage
  // the PWM values sent by analogWrite() are fractions of the maximum
speed possible
  // by your hardware
  // 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);
}
void loop()
{
  demoOne();
  delay(1000);
  demoTwo();
  delay(1000);
}
```

So what's happening in that sketch? In the function *demoOne()* we turn the motors on and run them at a PWM value of 200. This is not a speed value, instead power is applied for 200/255 of an amount of time at once.

Then after a moment the motors operate in the reverse direction (see how we changed the HIGHs and LOWs in the *digitalWrite()* functions?).

To get an idea of the range of speed possible of your hardware, we run through the entire PWM range in the function *demoTwo()* which turns the motors on and them runs through PWM values zero to 255 and back to zero with the two *for* loops.

Finally this is demonstrated in the following video - using a well-worn tank chassis with two DC motors: