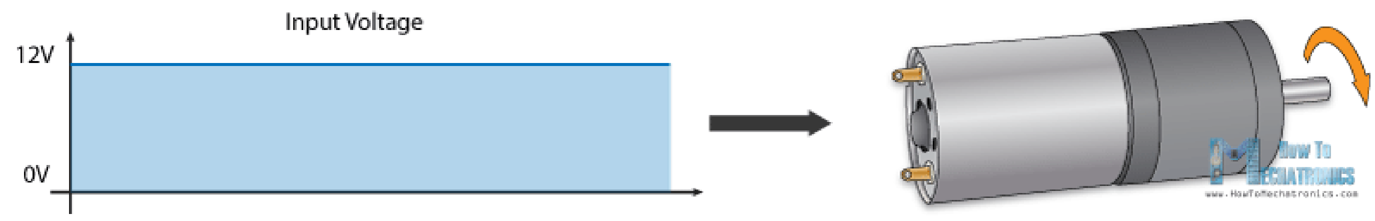
Arduino DC Motor Control Tutorial – L298N | PWM | H-Bridge

In this Arduino Tutorial we will learn how to control DC motors using Arduino. We well take a look at some basic techniques for controlling DC motors and make two example through which we will learn how to control DC motors using the L298N driver and the Arduino board.

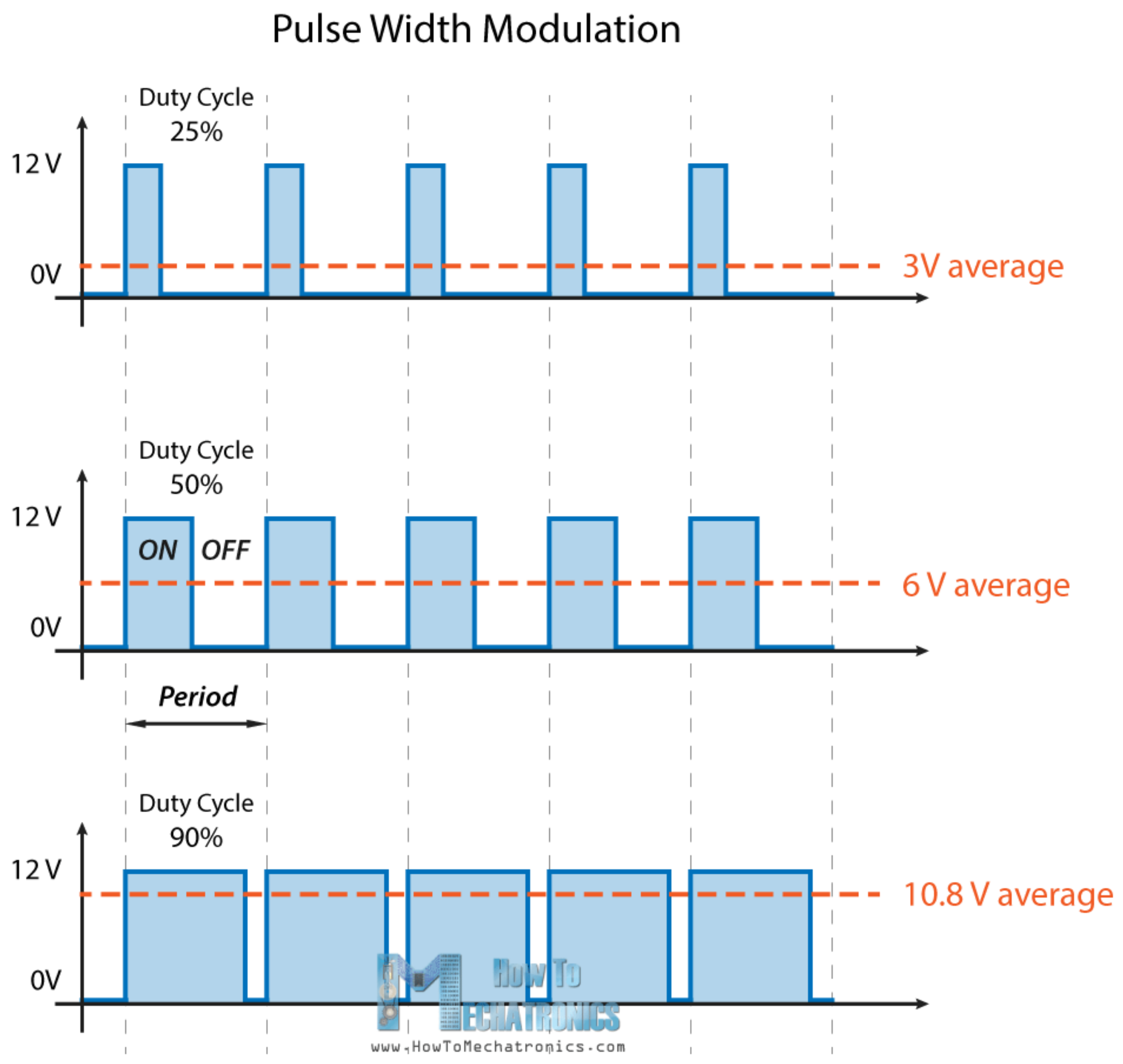
Overview

We can control the speed of the DC motor by simply controlling the input voltage to the motor and the most common method of doing that is by using PWM signal.

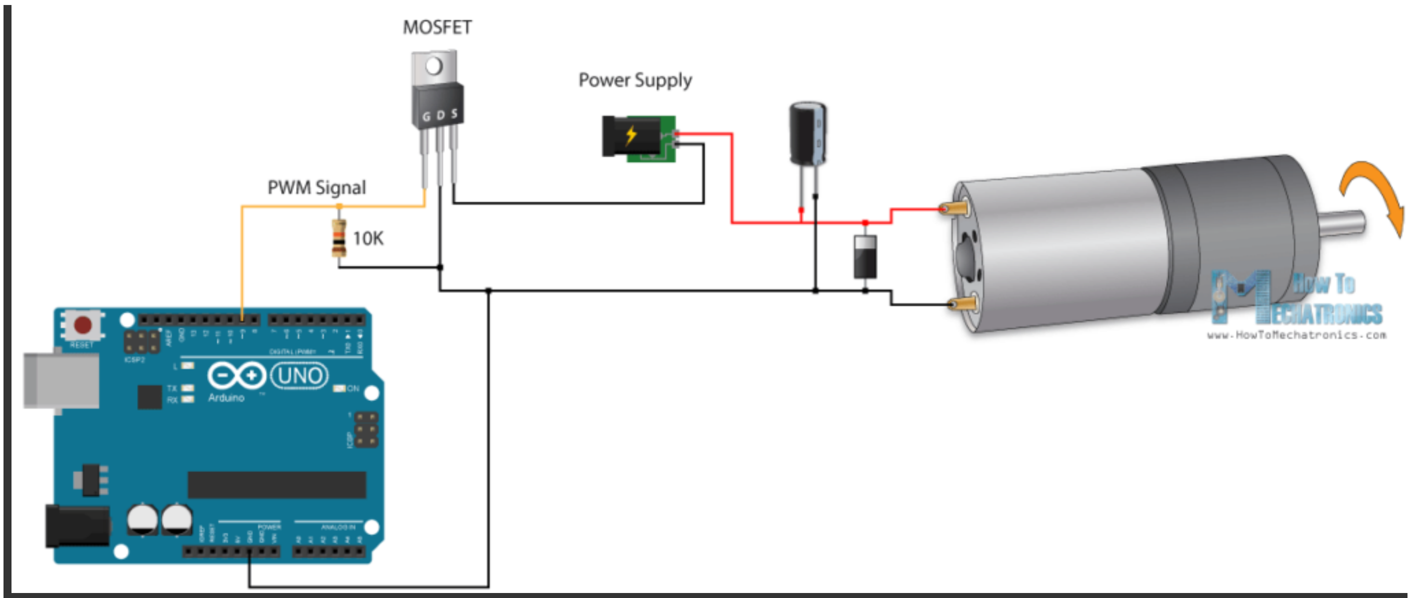


PWM DC Motor Control

PWM, or pulse width modulation is a technique which allows us to adjust the average value of the voltage that's going to the electronic device by turning on and off the power at a fast rate. The average voltage depends on the duty cycle, or the amount of time the signal is ON versus the amount of time the signal is OFF in a single period of time.



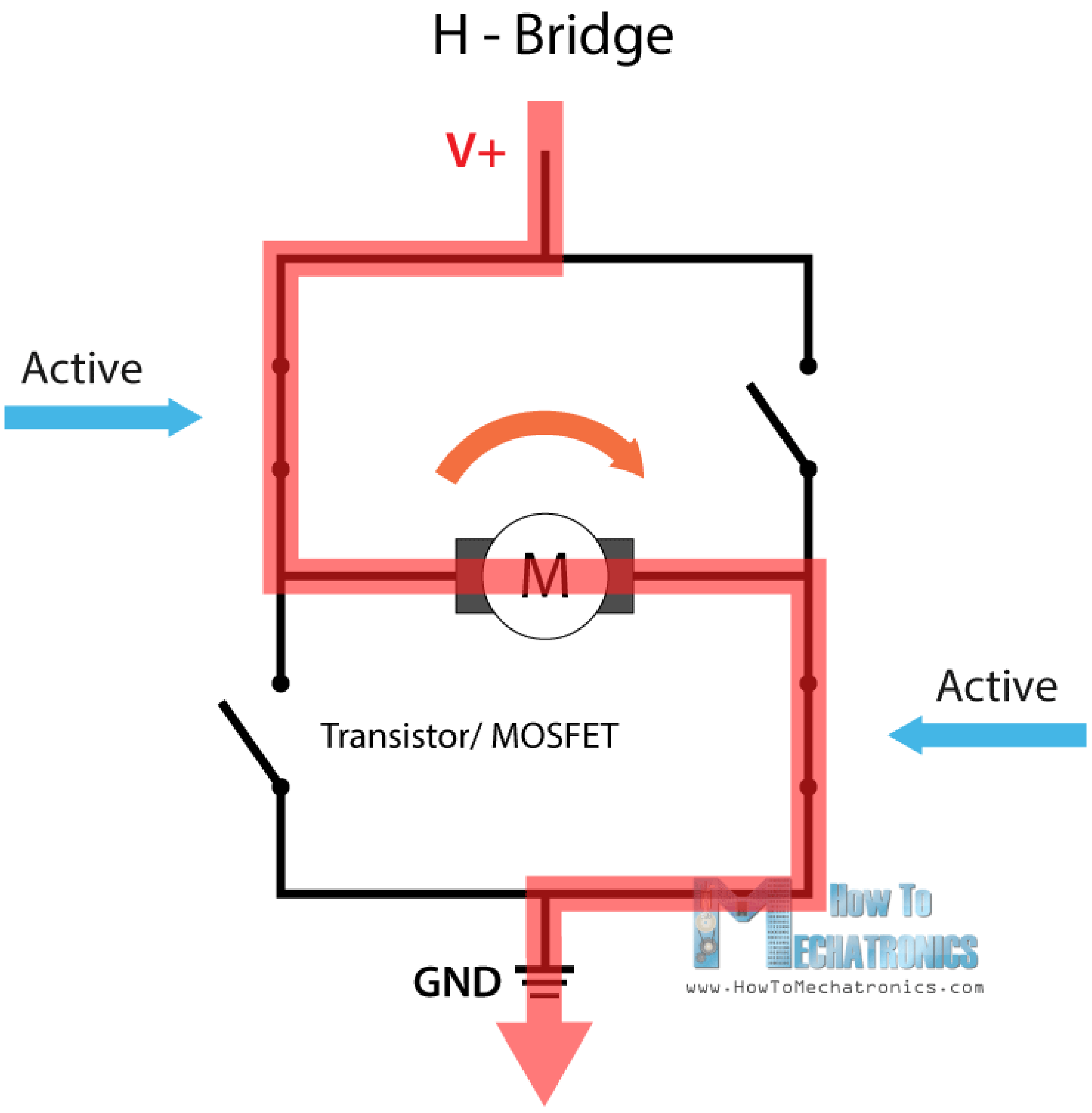
So depending on the size of the motor, we can simply connect an Arduino PWM output to the base of transistor or the gate of a MOSFET and control the speed of the motor by controlling the PWM output. The low power Arduino PWM signal switches on and off the gate at the MOSFET through which the high power motor is driven.



Note: Arduino GND and the motor power supply GND should be connected together.

H-Bridge DC Motor Control

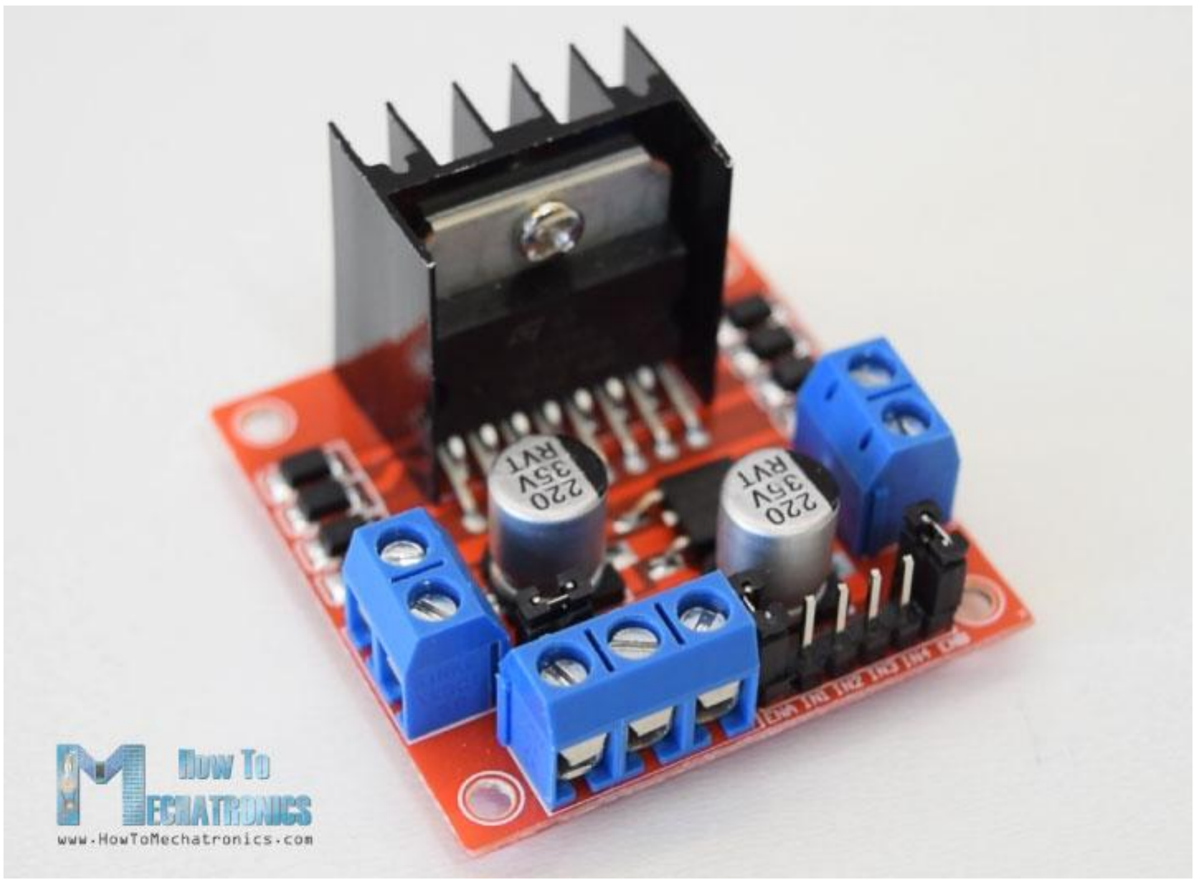
On the other hand, for controlling the rotation direction, we just need to inverse the direction of the current flow through the motor, and the most common method of doing that is by using an H-Bridge. An H-Bridge circuit contains four switching elements, transistors or MOSFETs, with the motor at the center forming an H-like configuration. By activating two particular switches at the same time we can change the direction of the current flow, thus change the rotation direction of the motor.



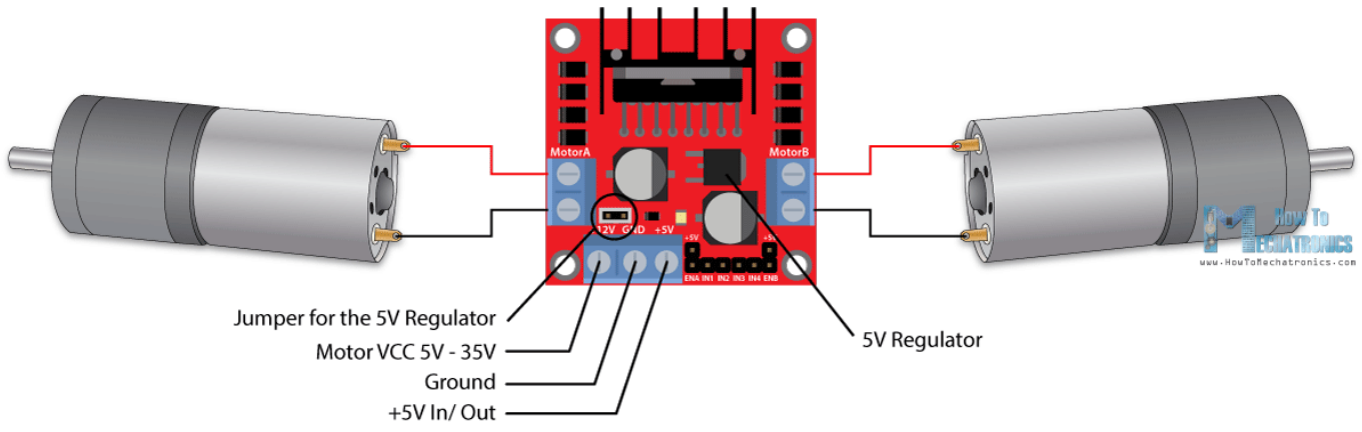
So if we combine these two methods, the PWM and the H-Bridge, we can have a complete control over the DC motor. There are many DC motor drivers that have these features and the L298N is one of them.

L298N Driver

The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.

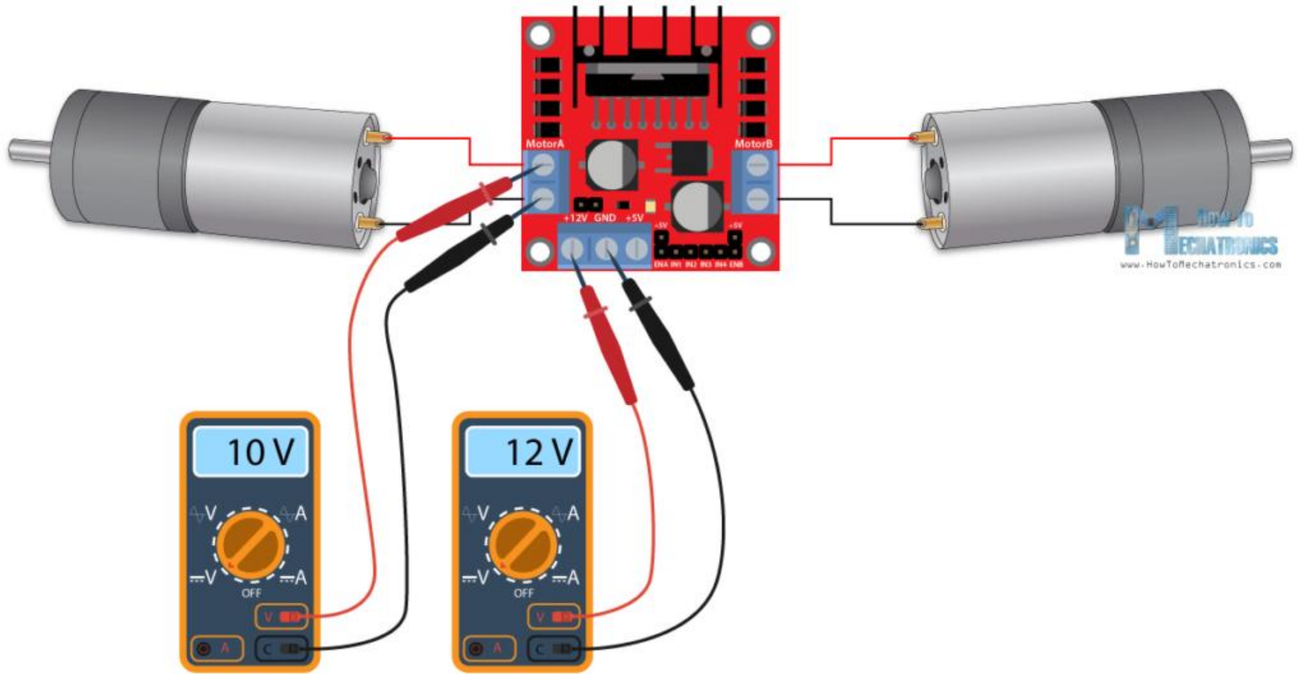


Let’s take a closer look at the pinout of L298N module and explain how it works. The module has two screw terminal blocks for the motor A and B, and another screw terminal block for the Ground pin, the VCC for motor and a 5V pin which can either be an input or output.

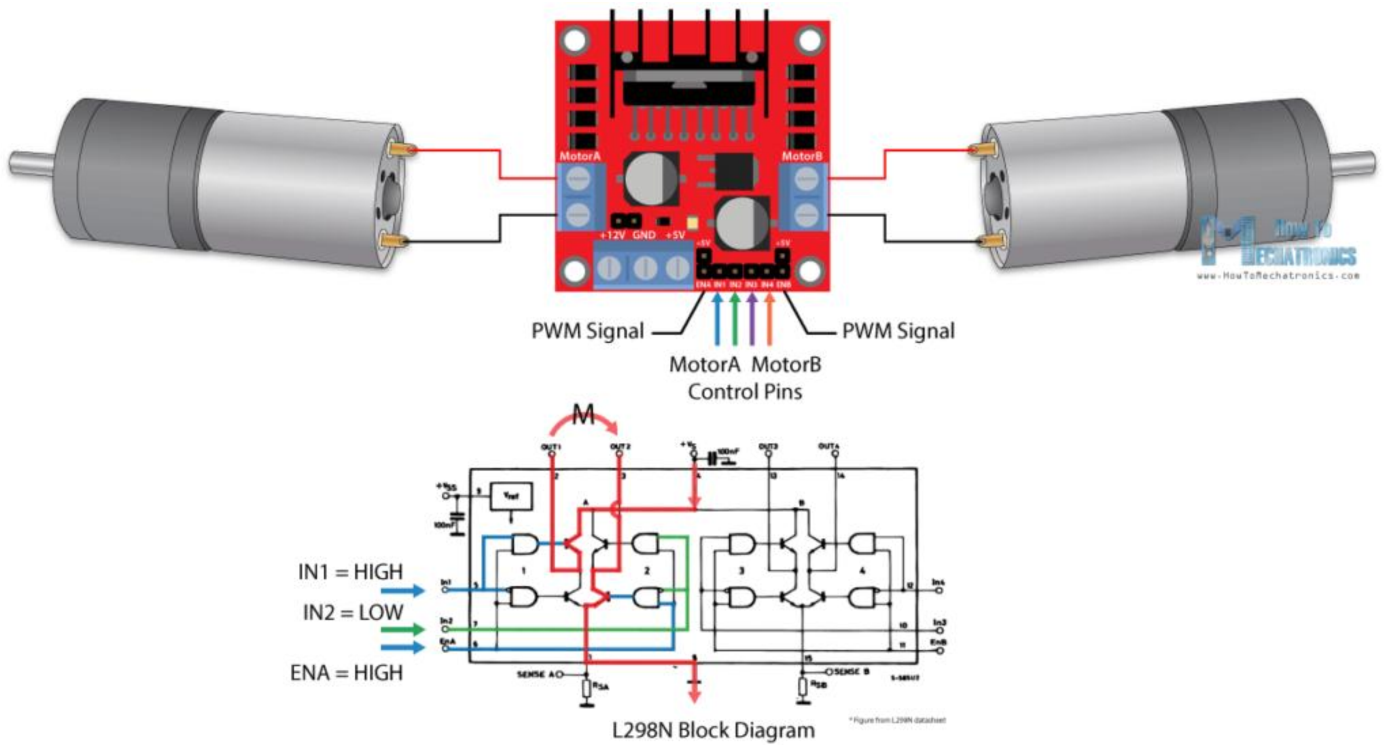


This depends on the voltage used at the motors VCC. The module have an onboard 5V regulator which is either enabled or disabled using a jumper. If the motor supply voltage is up to 12V we can enable the 5V regulator and the 5V pin can be used as output, for example for powering our Arduino board. But if the motor voltage is greater than 12V we must disconnect the jumper because those voltages will cause damage to the onboard 5V regulator. In this case the 5V pin will be used as input as we need connect it to a 5V power supply in order the IC to work properly.

We can note here that this IC makes a voltage drop of about 2V. So for example, if we use a 12V power supply, the voltage at motors terminals will be about 10V, which means that we won't be able to get the maximum speed out of our 12V DC motor.



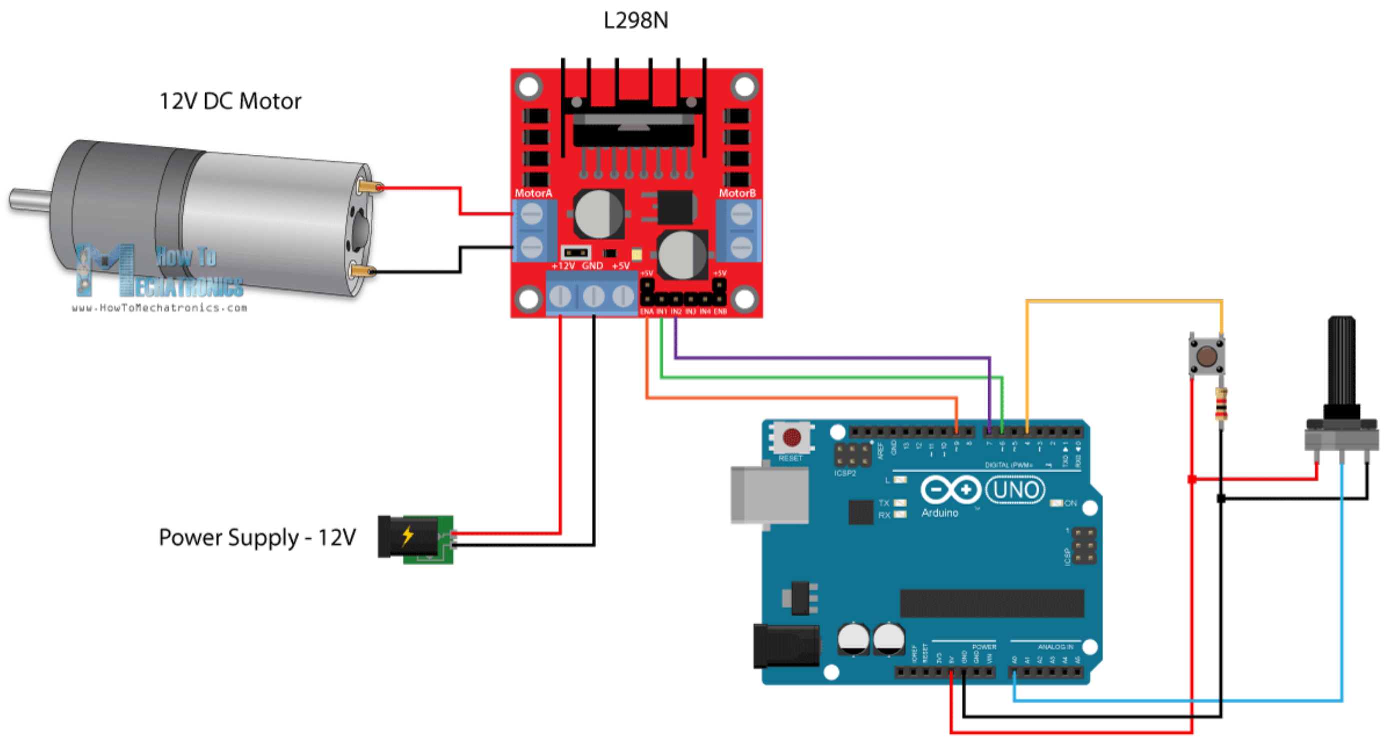
Next are the logic control inputs. The Enable A and Enable B pins are used for enabling and controlling the speed of the motor. If a jumper is present on this pin, the motor will be enabled and work at maximum speed, and if we remove the jumper we can connect a PWM input to this pin and in that way control the speed of the motor. If we connect this pin to a Ground the motor will be disabled.



Next, the Input 1 and Input 2 pins are used for controlling the rotation direction of the motor A, and the inputs 3 and 4 for the motor B. Using these pins we actually control the switches of the H-Bridge inside the L298N IC. If input 1 is LOW and input 2 is HIGH the motor will move forward, and vice versa, if input 1 is HIGH and input 2 is LOW the motor will move backward. In case both inputs are same, either LOW or HIGH the motor will stop. The same applies for the inputs 3 and 4 and the motor B.

Arduino and L298N

Now let's make some practical applications. In the first example we will control the speed of the motor using a potentiometer and change the rotation direction using a push button. Here's the circuit schematics.



So we need an L298N driver, a DC motor, a potentiometer, a push button and an Arduino board.

Arduino Code

Here's the Arduino code:

```
/* Arduino DC Motor Control - PWM | H-Bridge | L298N - Example 01

by Dejan Nedelkovski, www.HowToMechatronics.com
*/

#define enA 9
#define in1 6
#define in2 7
#define button 4

```

```
int rotDirection = 0;
int pressed = false;

void setup() {
pinMode(enA, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(button, INPUT);
// Set initial rotation direction
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
}

void loop() {
int potValue = analogRead(A0); // Read potentiometer value
int pwmOutput = map(potValue, 0, 1023, 0 , 255); // Map the potentiometer value from 0 to 255
analogWrite(enA, pwmOutput); // Send PWM signal to L298N Enable pin

// Read button - Debounce
if (digitalRead(button) == true) {
pressed = !pressed;
}
while (digitalRead(button) == true);
delay(20);

// If button is pressed - change rotation direction
if (pressed == true & rotDirection == 0) {
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
rotDirection = 1;
delay(20);
}
// If button is pressed - change rotation direction
if (pressed == false & rotDirection == 1) {
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
rotDirection = 0;
delay(20);
}
}
```

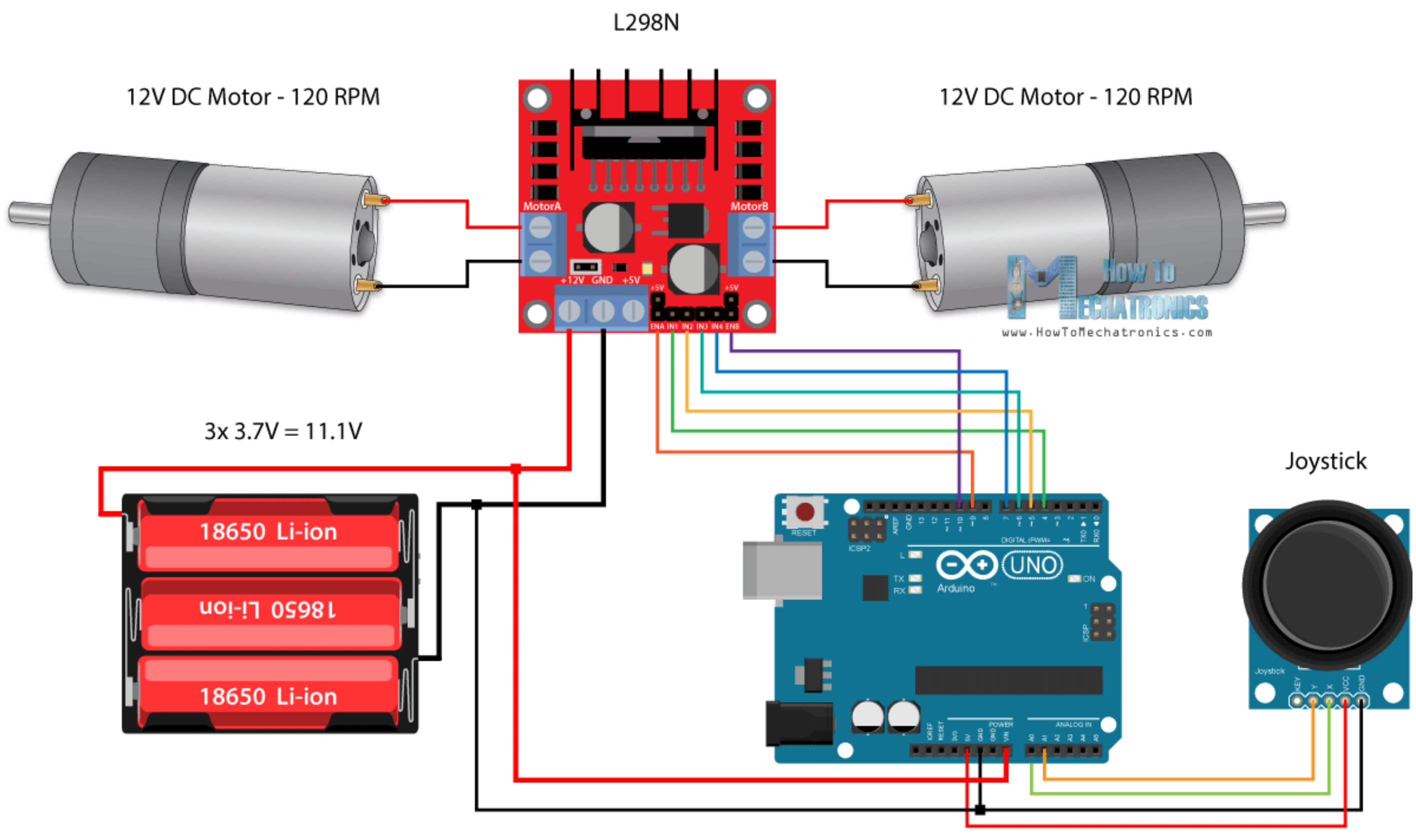
Description: So first we need to define the pins and some variables needed for the program. In the setup section we need to set the pin modes and the initial rotation direction of the motor. In the loop section we start by reading the potentiometer value and then map the value that we get from it which is from 0 to 1023, to a value from 0 to 255 for the PWM signal, or that's 0 to 100%

duty cycle of the PWM signal. Then using the analogWrite() function we send the PWM signal to the Enable pin of the L298N board, which actually drives the motor.

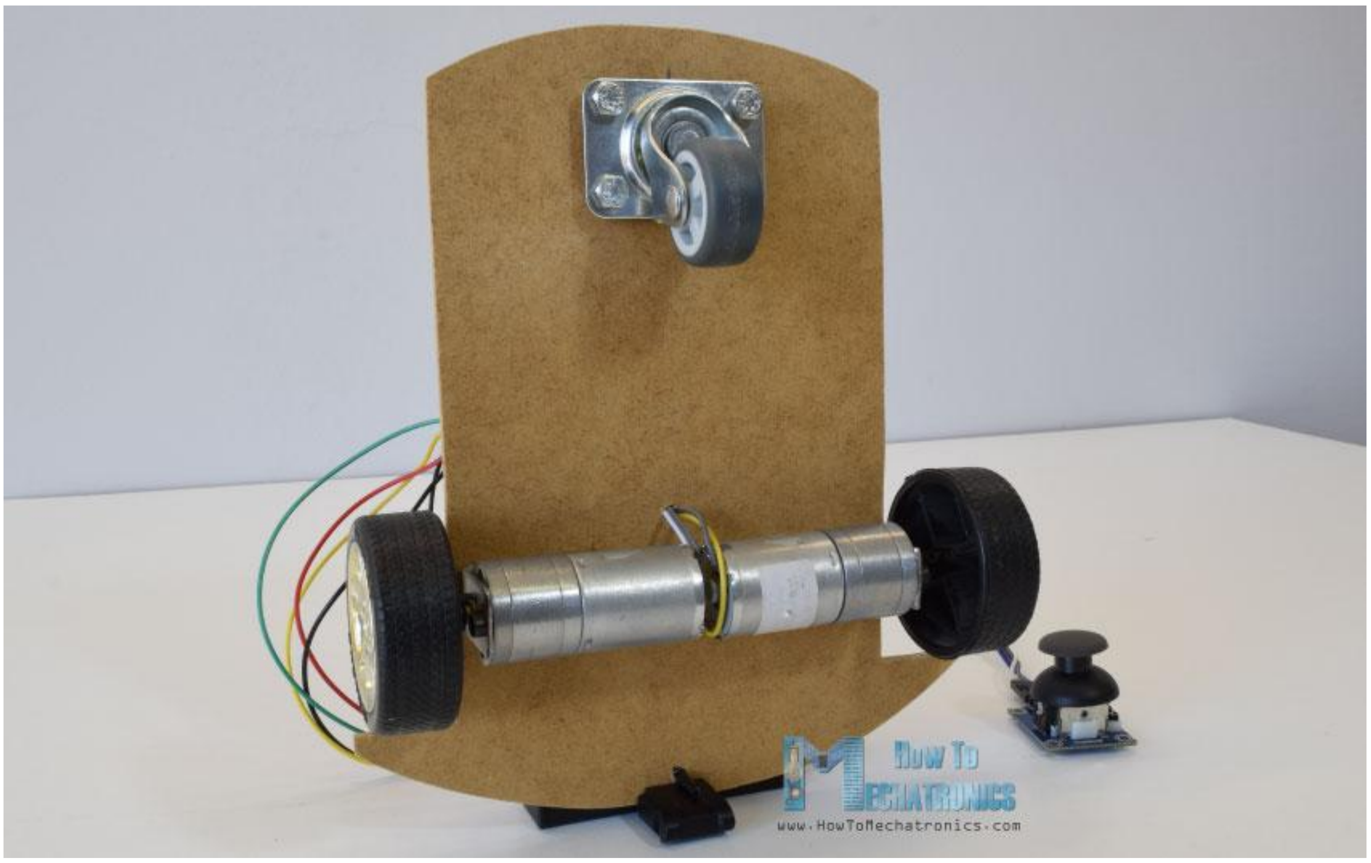
Next, we check whether we have pressed the button, and if that's true, we will change the rotation direction of the motor by setting the Input 1 and Input 2 states inversely. The push button will work as toggle button and each time we press it, it will change the rotation direction of the motor.

Arduino Robot Car Control using L298N Driver

So once we have learned this, now we can build our own Arduino robot car. Here's the circuit schematic:



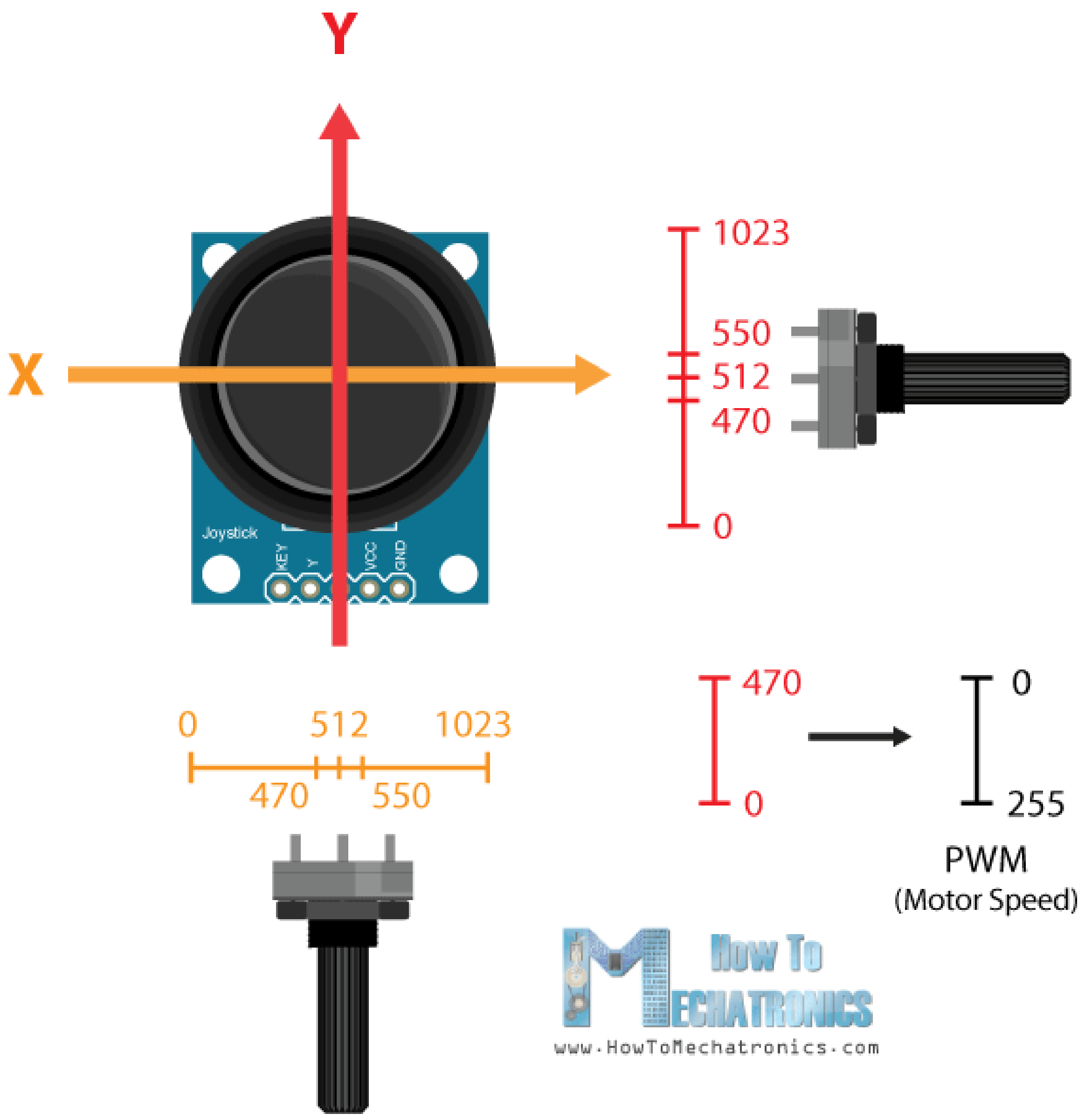
All we need is 2 DC Motors, the L298N driver, an Arduino board and a joystick for the control. As for the power supply, I chose to use three 3.7V Li-ion batteries, providing total of 11V. I made the chassis out of 3 mm tick plywood, attached the motors to it using metal brackets, attached wheels to the motors and in front attached a swivel wheel.



Now let’s take a look at the Arduino code and see how it works. (Down below you can find the complete code)

```
int xAxis = analogRead(A0); // Read Joysticks X-axis
int yAxis = analogRead(A1); // Read Joysticks Y-axis
```

After defining the pins, in the loop section, we start with reading the joystick X and Y axis values. The joystick is actually made of two potentiometers which are connected to the analog inputs of the Arduino and they have values from 0 to 1023. When the joystick stays in its center position the value of both potentiometers, or axes is around 512.



We will add a little tolerance and consider the values from 470 to 550 as center. So if we move the Y axis of joystick backward and the value goes below 470 we will set the two motors rotation direction to backward using the four input pins. Then, we will convert the declining values from 470 to 0 into increasing PWM values from 0 to 255 which is actually the speed of the motor.

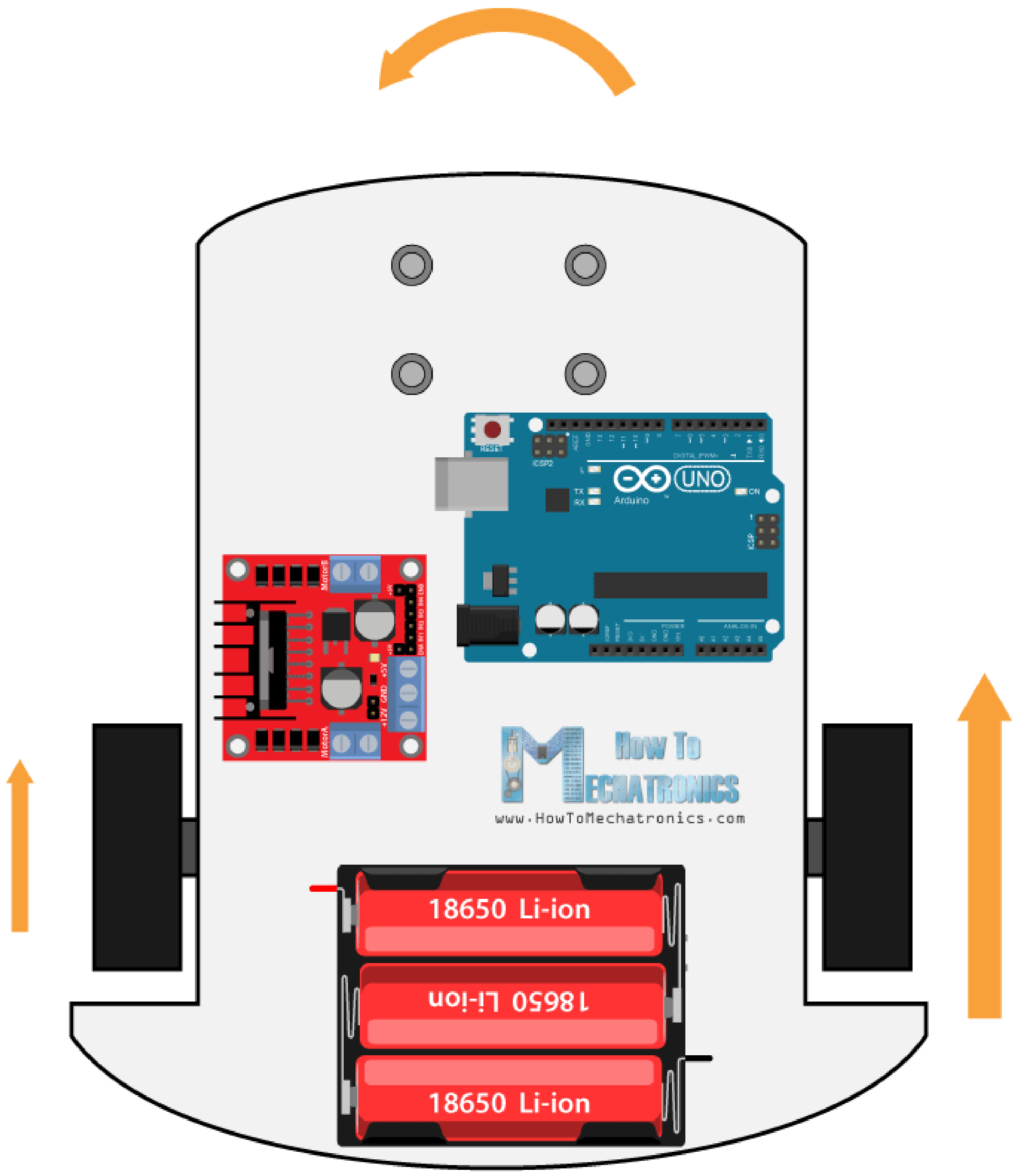
```
// Y-axis used for forward and backward control
if (yAxis < 470) {
// Set Motor A backward
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
// Set Motor B backward
digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);
// Convert the declining Y-axis readings for going backward from 470 to 0 into 0 to 255
   value for the PWM signal for increasing the motor speed
motorSpeedA = map(yAxis, 470, 0, 0, 255);
motorSpeedB = map(yAxis, 470, 0, 0, 255);
}
```

Similar, if we move the Y axis of the joystick forward and the value goes above 550 we will set the motors to move forward and convert the readings from 550 to 1023 into PWM values from 0 to 255. If the joystick stays in its center the motors speed will be zero.

Next, let's see how we use the X axis for the left and right control of the car.

```
// X-axis used for left and right control
if (xAxis < 470) {
// Convert the declining X-axis readings from 470 to 0 into increasing 0 to 255 value
int xMapped = map(xAxis, 470, 0, 0, 255);
// Move to left - decrease left motor speed, increase right motor speed
motorSpeedA = motorSpeedA - xMapped;
motorSpeedB = motorSpeedB + xMapped;
// Confine the range from 0 to 255
if (motorSpeedA < 0) {
motorSpeedA = 0;
}
if (motorSpeedB > 255) {
motorSpeedB = 255;
}
}
```

So again, first we need to convert the X axis readings into speed values from 0 to 255. For moving left, we use this value to decrease the left motor speed and increase the right motor speed. Here, because of the arithmetic functions we use two additional "if" statements to confine the range of the motor speed from 0 to 255.



The same method is used for moving the car to the right.

Depending on the applied voltage and the motor itself, at lower speeds the motor is not able to start moving and it produces a buzzing sound. In my case, the motors were not able to move if the value of the PWM signal was below 70. Therefore using this two if statements I actually confined to speed range from 70 to 255. At the end we just send the final motor speeds or PWM signal to the enable pins of the L298N driver.

```
// Prevent buzzing at low speeds (Adjust according to your motors. My motors couldn't start moving if PWM value was below value of 70)
if (motorSpeedA < 70) {
motorSpeedA = 0;
}
if (motorSpeedB < 70) {
motorSpeedB = 0;
}
analogWrite(enA, motorSpeedA); // Send PWM signal to motor A
analogWrite(enB, motorSpeedB); // Send PWM signal to motor B
```

Here’s the complete code of the Arduino robot car example:

```
/* Arduino DC Motor Control - PWM | H-Bridge | L298N
Example 02 - Arduino Robot Car Control
by Dejan Nedelkovski, www.HowToMechatronics.com
*/

#define enA 9
#define in1 4
#define in2 5
#define enB 10
#define in3 6
#define in4 7

int motorSpeedA = 0;
int motorSpeedB = 0;

void setup() {
pinMode(enA, OUTPUT);
pinMode(enB, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
}

void loop() {
int xAxis = analogRead(A0); // Read Joysticks X-axis
int yAxis = analogRead(A1); // Read Joysticks Y-axis

// Y-axis used for forward and backward control
if (yAxis < 470) {
// Set Motor A backward
digitalWrite(in1, HIGH);
digitalWrite(in2, LOW);
// Set Motor B backward
digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);
// Convert the declining Y-axis readings for going backward from 470 to 0 into 0 to 255 value for the PWM signal for increasing the motor speed
motorSpeedA = map(yAxis, 470, 0, 0, 255);
motorSpeedB = map(yAxis, 470, 0, 0, 255);
}
else if (yAxis > 550) {
// Set Motor A forward
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
// Set Motor B forward
digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
// Convert the increasing Y-axis readings for going forward from 550 to 1023 into 0 to
255 value for the PWM signal for increasing the motor speed
motorSpeedA = map(yAxis, 550, 1023, 0, 255);
motorSpeedB = map(yAxis, 550, 1023, 0, 255);
}
// If joystick stays in middle the motors are not moving
else {
motorSpeedA = 0;
motorSpeedB = 0;
}

// X-axis used for left and right control
if (xAxis < 470) {
// Convert the declining X-axis readings from 470 to 0 into increasing 0 to 255 value
int xMapped = map(xAxis, 470, 0, 0, 255);
// Move to left - decrease left motor speed, increase right motor speed
motorSpeedA = motorSpeedA - xMapped;
motorSpeedB = motorSpeedB + xMapped;
// Confine the range from 0 to 255
if (motorSpeedA < 0) {
motorSpeedA = 0;
}
if (motorSpeedB > 255) {
motorSpeedB = 255;
}
}
if (xAxis > 550) {
// Convert the increasing X-axis readings from 550 to 1023 into 0 to 255 value
int xMapped = map(xAxis, 550, 1023, 0, 255);
// Move right - decrease right motor speed, increase left motor speed
motorSpeedA = motorSpeedA + xMapped;
motorSpeedB = motorSpeedB - xMapped;
// Confine the range from 0 to 255
if (motorSpeedA > 255) {
motorSpeedA = 255;
}
if (motorSpeedB < 0) {
motorSpeedB = 0;
}
}
```

```
// Prevent buzzing at low speeds (Adjust according to your motors. My motors couldn't start moving if PWM value was below value of 70)
if (motorSpeedA < 70) {
motorSpeedA = 0;
}
if (motorSpeedB < 70) {
motorSpeedB = 0;
}
analogWrite(enA, motorSpeedA); // Send PWM signal to motor A
analogWrite(enB, motorSpeedB); // Send PWM signal to motor B
}
```

So that would be all for this tutorial, and in my next video we will upgrade this Arduino robot car, by adding a Bluetooth and Radio devices for enabling smartphone and wireless control.