EL MÓDULO CONTROLADOR DE MOTORES L298N *(versión 17-4-19)*

Controlar la velocidad y el sentido de giro de dos motores de corriente continua.

OBJETIVOS

- Presentar el **módulo controlador de motores L298N H-bridge.**
- Explicar cómo se conecta y se utiliza.
- Usarlo para controlar **dos motores de corriente continua**.

MATERIAL REQUERIDO.

	**Arduino Uno** o equivalente.
	Un módulo controlador de motores **L298N.**
	Un par de **ruedas y motores CC.**
	Algunos **cables** de Protoboard macho-macho y macho-hembra.

INTRODUCCIÓN

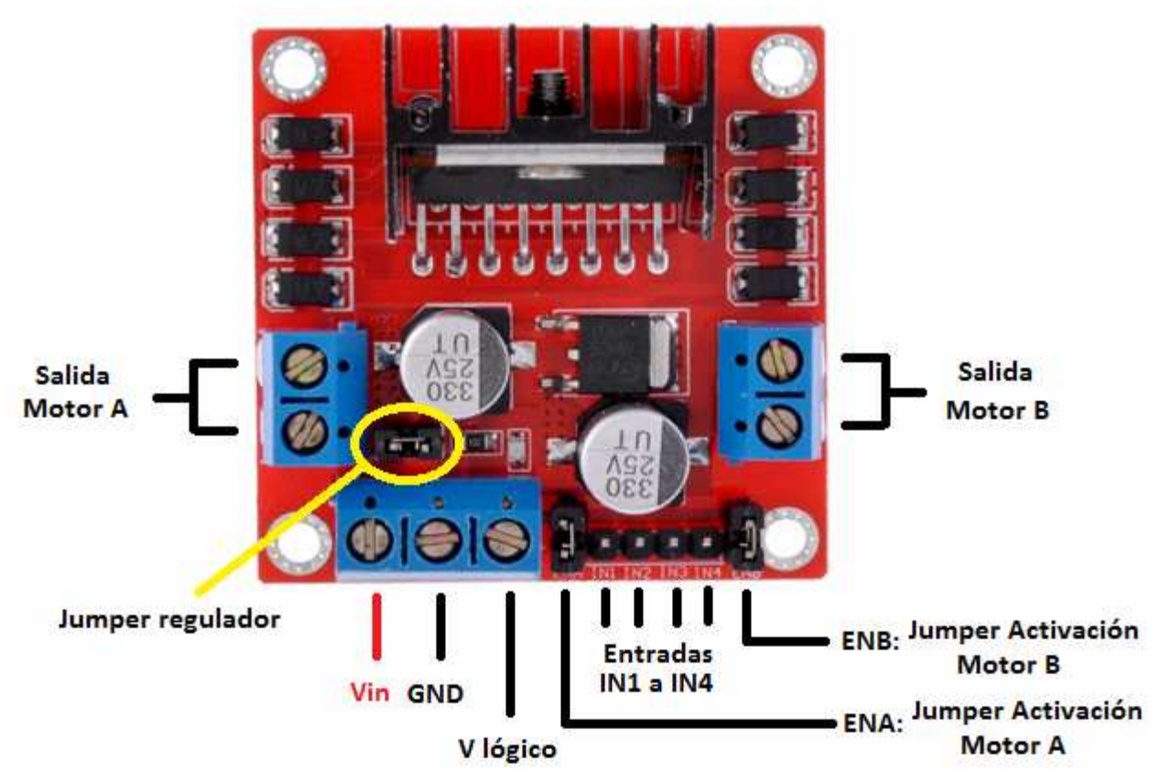
El módulo controlador de motores **L298N** H-bridge nos permite controlar la velocidad y la dirección de dos motores de corriente continua o un motor paso a paso de una forma muy sencilla, gracias a los 2 los dos **H-bridge** que monta.

Ya hemos hablado de ellos antes, pero básicamente un **puente-H** o **H-bridge** es un componente formado por 4 transistores que nos permite invertir el sentido de la corriente, y de esta forma podemos invertir el sentido de giro del motor.

El rango de tensiones en el que trabaja este módulo va desde 3V hasta 35V, y una intensidad de hasta 2A. A la hora de alimentarlo hay que tener en cuenta que la electrónica del módulo consume unos 3V, así que los motores reciben 3V menos que la tensión con la que alimentemos el módulo.

Además el **L298N** incluye un **regulador de tensión** que nos permite obtener del módulo una tensión de 5V, perfecta para alimentar nuestro**Arduino**. Eso sí, este regulador sólo funciona si alimentamos el módulo con una tensión máxima de 12V.

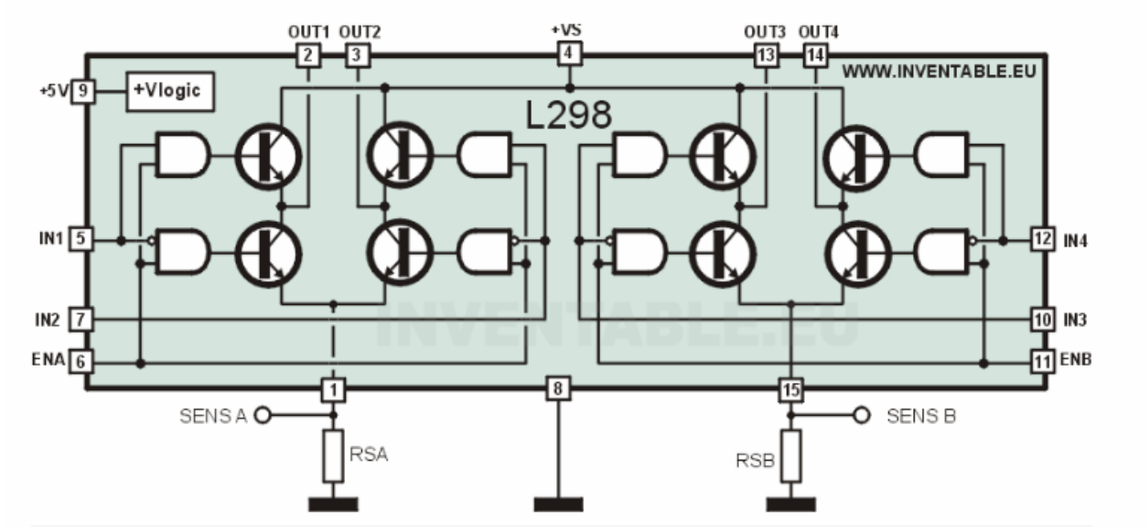
Es un módulo que se utiliza mucho en proyectos de robótica, por su facilidad de uso y su reducido precio.



Características

- **Voltaje de alimentación, mínimo de 5 V. Posee dos entradas, una de 5V para controlar la parte lógica y otra para alimentar las salidas al motor, que pueden ser de 5V o más.**
- **La tarjeta tiene la opción de habilitar un regulador LM7805 integrado en ella para alimentar la parte lógica con lo que se puede alimentar la tarjeta con 12V por ejemplo.**
- **Corriente máxima 2 Amperios.**
- **Posee 6 entradas de control (ver tabla de control)**
- **Admite entradas de señal PWM para el control de velocidad.**
- **Dimensiones: 43 mm x 23,9 mm x 43 mm.**
- **Salidas: para 2 motores de DC o para un motor bipolar paso a paso.**

Detalles sobre el INTEGRADO



CONEXIÓN Y FUNCIONAMIENTO DEL MODULO

La entrada de tensión **Vin** admite tensiones entre 3V y 35V, y justo a su derecha en la imagen tenemos el pin que debemos conectar a **GND**.

La tercera conexión de ese grupo **V lógico** puede funcionar de dos maneras:

- Si el **jumper del regulador** está **cerrado** activaremos el regulador de tensión del **L298N**, y en **V lógico** tendremos una salida de 5V, que podremos usar para lo que queramos, por ejemplo para alimentar una placa Arduino.
- Si el **quitamos** el **jumper** desactivaremos el regulador, necesitaremos alimentar la parte lógica del módulo, así que tendremos que colocar una tensión de 5V por la conexión **V lógico** para que el módulo funcione.
- **¡Cuidado!** Si introducimos corriente por V lógico con el jumper de regulación puesto podríamos dañar el módulo.
- Además el **regulador** sólo funciona con tensiones hasta **12V** en **Vin**, por encima de este valor tendremos que quitar el jumper y alimentar la parte lógica del módulo desde otra fuente.

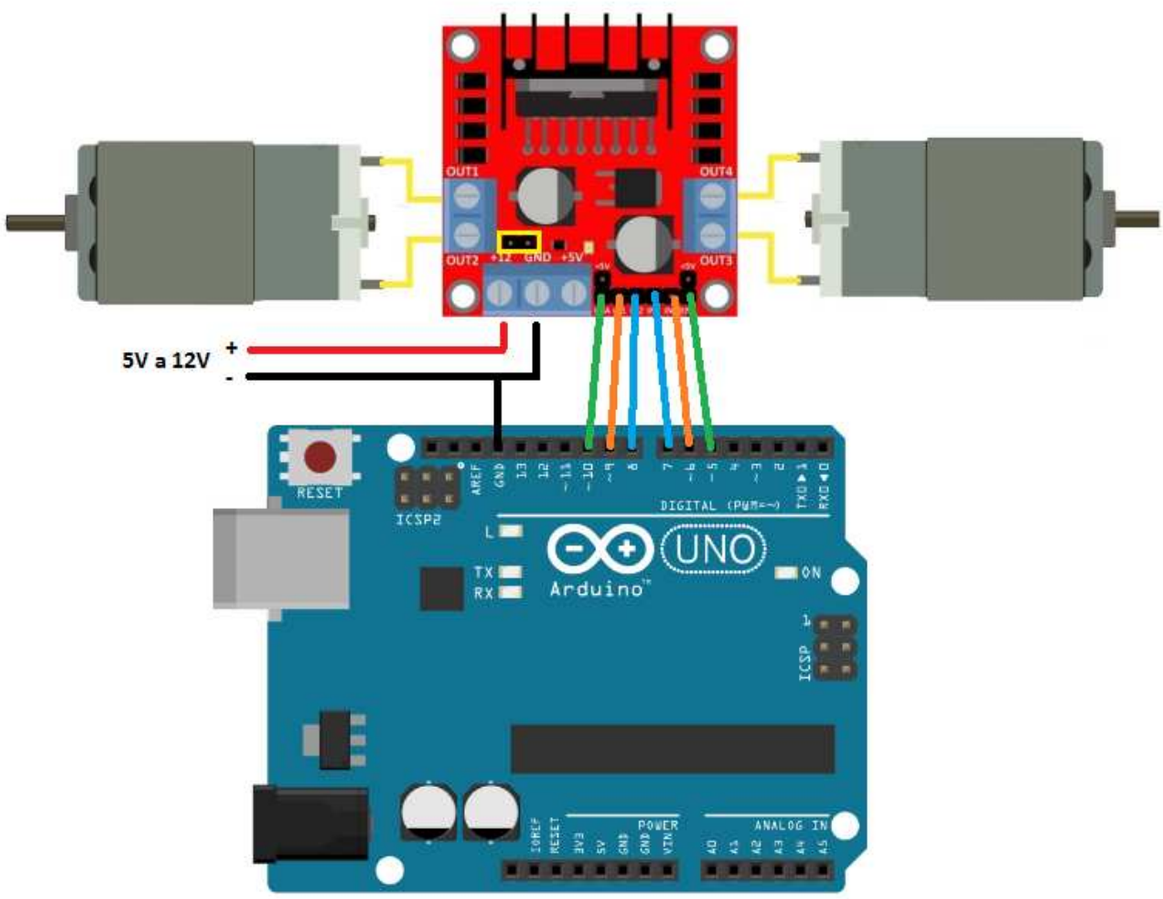
El resto de conexiones se usan de una u otra forma dependiendo si vamos a manejar dos motores de continua o un motor paso a paso. En esta sesión nos vamos a centrar en el control de **motores DC.**

Las salidas para los motores A y B nos darán la energía para mover los motores. Tened en cuenta la polaridad al conectarlos, para que cuando más tarde hagamos que se muevan adelante, funcionen como deberían. Si no fuera así, no tendríamos más que invertir las conexiones.

Los pines **IN1** e **IN2** nos sirven para controlar el sentido de giro del **motor A**, y los pines **IN3** e **IN4** el del **motor B**. Funcionan de forma que si IN1 está a HIGH e IN2 a LOW, el motor A gira en un sentido, y si está IN1 a LOW e IN2 a HIGH lo hace en el otro. Y lo mismo con los pines IN3 e IN4 y el motor B.

Para controlar la velocidad de giro de los motores tenemos que quitar los jumpers y usar los pines **ENA** y **ENB**. Los conectaremos a dos salidas**PWM** de Arduino de forma que le enviemos un valor entre 0 y 255 que controle la velocidad de giro. Si tenemos los jumpers colocados, los motores girarán a la siempre a la misma velocidad.

El esquema de montaje que vamos a utilizar va a ser el siguiente, aunque podéis usar los pines que queráis siempre que respetemos que los que conectemos a **ENA** y **ENB** sean **PWM**.



PROGRAMACIÓN

Sabiendo todo esto, ya estamos en posición de empezar a programar. Vamos a hacer un programilla que mueva los motores en ambos sentidos, adelante y atrás, y en sentidos contrarios el uno del otro, variando también la velocidad de movimiento.

Comenzamos asignando los pines que vamos a usar y declarándolos como salida:

(Ejemplo de programa sugerido por esta publicación)

```
//Programa pruebaModulo1

// Motor A

int ENA = 10;

int IN1 = 9;

int IN2 = 8;

// Motor B

int ENB = 5;

int IN3 = 7;
```

```
int IN4 = 6;

void setup ()

{

 // Declaramos todos los pines como salidas

 pinMode (ENA, OUTPUT);

 pinMode (ENB, OUTPUT);

 pinMode (IN1, OUTPUT);

 pinMode (IN2, OUTPUT);

 pinMode (IN3, OUTPUT);

 pinMode (IN4, OUTPUT);

}
```

Y ahora vamos a crear las **funciones** para mover los motores. Primero para moverlos hacia delante a plena potencia:

```
void Adelante ()

{

 //Direccion motor A

 digitalWrite (IN1, HIGH);

 digitalWrite (IN2, LOW);

 analogWrite (ENA, 255); //Velocidad motor A

 //Direccion motor B

 digitalWrite (IN3, HIGH);

 digitalWrite (IN4, LOW);

 analogWrite (ENB, 255); //Velocidad motor B

}
```

Y ahora para moverse en el sentido contrario a la mitad de potencia:

```
void Atras ()

{

 //Direccion motor A

 digitalWrite (IN1, LOW);
```

```
 digitalWrite (IN2, HIGH);
 analogWrite (ENA, 128); //Velocidad motor A
 //Direccion motor B
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, HIGH);
 analogWrite (ENB, 128); //Velocidad motor B
}
```

Y ahora vamos a girar cada motor en un sentido, cada uno con una velocidad además:

```
void Derecha ()
{
 //Direccion motor A
 digitalWrite (IN1, HIGH);
 digitalWrite (IN2, LOW);
 analogWrite (ENA, 200); //Velocidad motor A
 //Direccion motor B
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, HIGH);
 analogWrite (ENB, 100); //Velocidad motor A
}
void Izquierda ()
{
 //Direccion motor A
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, HIGH);
 analogWrite (ENA, 50); //Velocidad motor A
 //Direccion motor B
 digitalWrite (IN3, HIGH);
 digitalWrite (IN4, LOW);
 analogWrite (ENB, 150); //Velocidad motor A
```

```
}
```

Y una función más para pararlos:

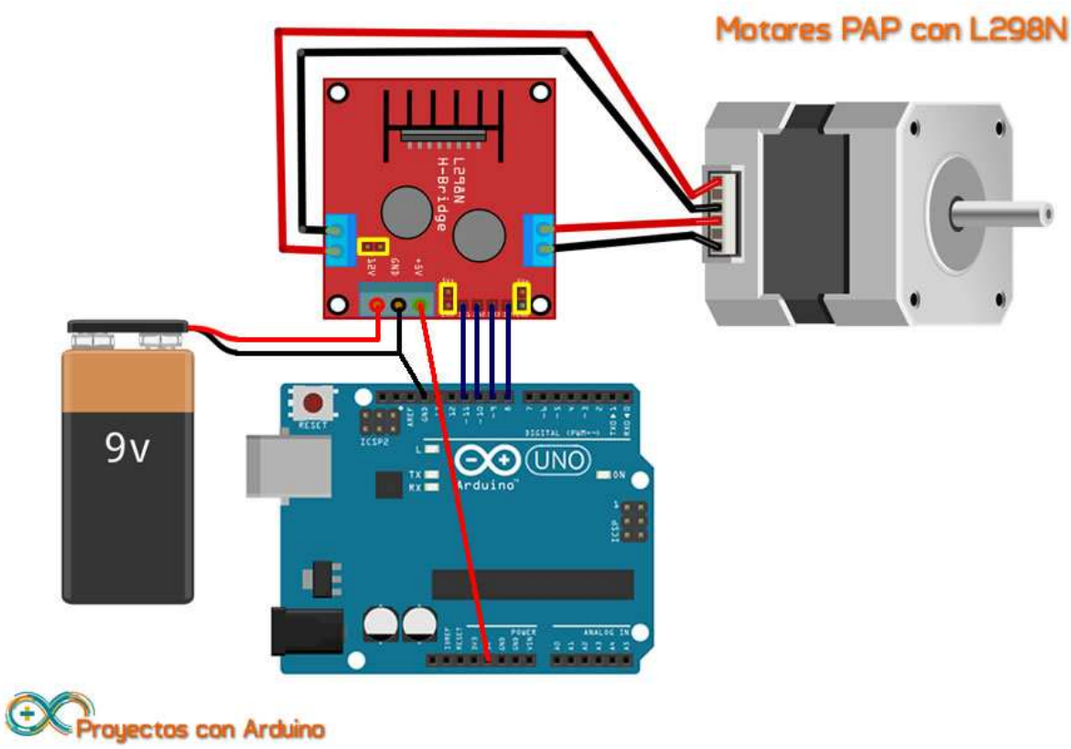
```
void Parar ()

{

 //Direccion motor A

 digitalWrite (IN1, LOW);

 digitalWrite (IN2, LOW);

 analogWrite (ENA, 0); //Velocidad motor A

 //Direccion motor B

 digitalWrite (IN3, LOW);

 digitalWrite (IN4, LOW);

 analogWrite (ENB, 0); //Velocidad motor A

}
```

Vamos a combinar estas funciones en el loop, haciendo que cada una se ejecute durante un periodo de tiempo:

```
void loop ()

{

 Adelante ();

 delay (5000);

 Atras ();

 delay (3000);

 Derecha ();

 delay (2000);

 Izquierda ();

 delay (2000);

 Parar ();

 delay (4000);

}
```

Si al ejecutarlo resulta que los motores se mueven en direcciones que no son las esperadas, podéis o bien intercambiar las conexiones del motor en los bornes del **L298N** o invertir el estado de los pines IN1 a IN4.

USANDO EL MODULO PARA CONTROLAR UN MOTOR PASO A PASO



CONTINUARA