

# Aprendiendo Arduino

Aprendiendo a manejar Arduino en profundidad

ARCHIVO DE LA ETIQUETA: SERVO

## Motores Arduino

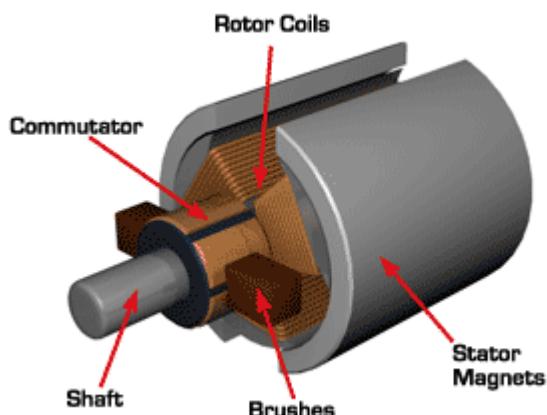
Como ya se ha visto anteriormente, un pin de Arduino solo puede tener valores de 0 y 5 voltios y dar hasta 40 mA de corriente. Esto es insuficiente para mover casi cualquier motor del tipo que sea, por lo tanto si queremos que Arduino maneje un motor, deberemos usar un driver.

Un motor driver es un amplificador de corriente cuya función es tomar una pequeña señal de control de baja corriente y convertirla en una señal de alta corriente que pueda alimentar el motor. Hay muchos tipos de motor drivers en función del motor a manejar, máximo voltaje, máxima corriente de salida, etc...

Más información: <http://www.futureelectronics.com/en/drivers/motor-driver.aspx>

### Motor DC

Un motor de corriente continua convierte la energía eléctrica en mecánica. Se compone de dos partes: el estator y el rotor. El estator es la parte mecánica del motor donde están los polos del imán. El rotor es la parte móvil del motor con devanado y un núcleo, al que llega la corriente a través de las escobillas. Si queremos cambiar el sentido de giro del rotor, tenemos que cambiar el sentido de la corriente que le proporcionamos al rotor, es decir, basta con invertir la polaridad de la pila o batería.



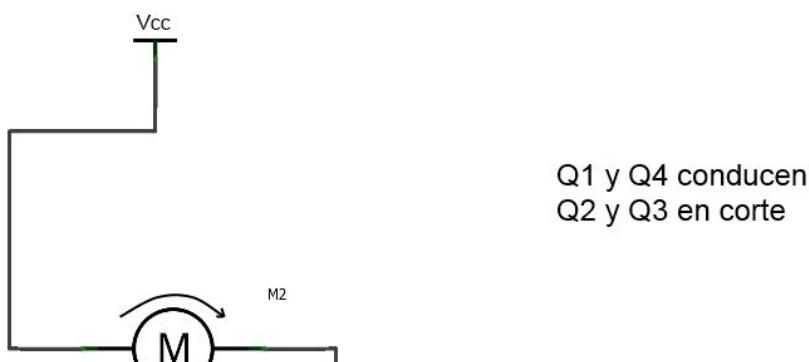
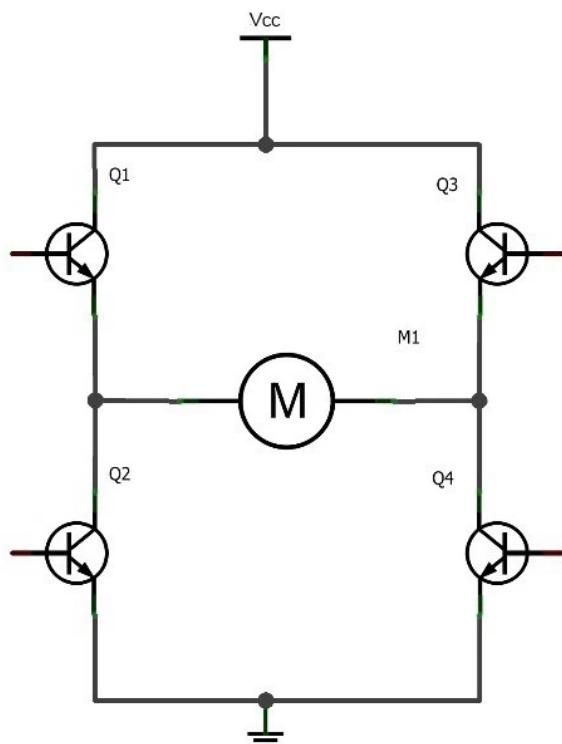
Para controlar un motor DC desde Arduino, tendremos que usar un driver de motores para proporcionar más corriente al motor ya que las salidas del Arduino sólo dan hasta 40mA y más voltaje porque este tipo de motores suelen ser de 6V o más. El driver debemos alimentarlo con una fuente de alimentación externa con el voltaje y corriente eléctrica suficiente para mover el motor..

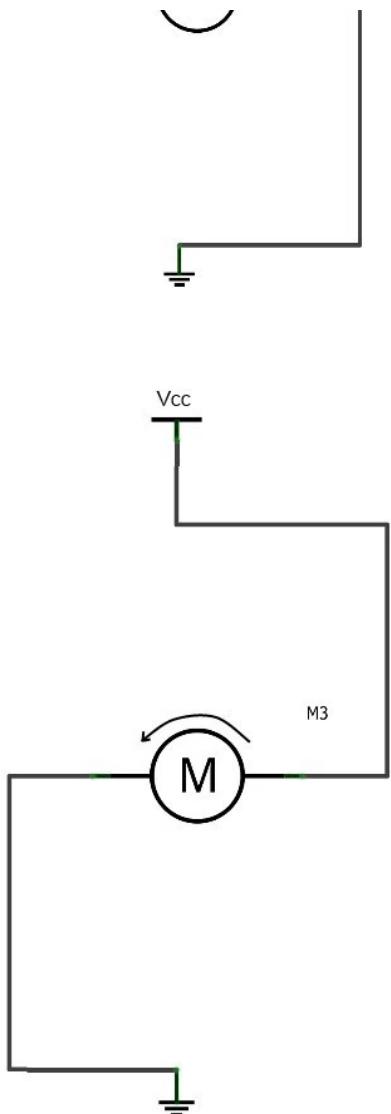
Más información:

- Muy buen tutorial: <http://www.prometec.net/motorcc/>
- Cómo funciona un motor DC: <https://www.baldengineer.com/videos/brushed-dc-motor-video-tutorial>

## Driver L293D

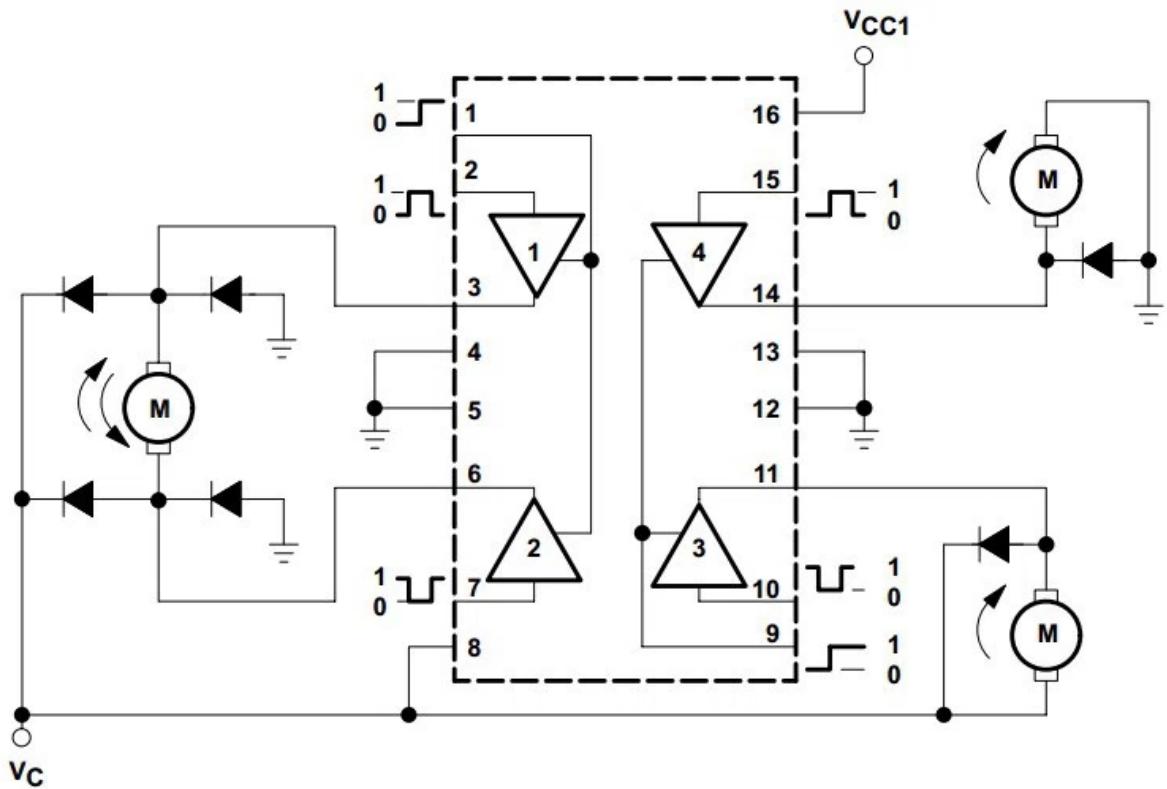
El L293D es un integrado para controlar motores DC que usa doble puente en H. Es un sistema para controlar el sentido de giro de un motor DC usando cuatro transistores y también para variar la velocidad del motor. En la imagen vemos que los transistores se comportan como interruptores y dependiendo que transistores conducen y cuáles no cambia la polarización del motor y con esto el sentido de giro.





Q2 y Q3 conducen  
Q1 y Q4 en corte

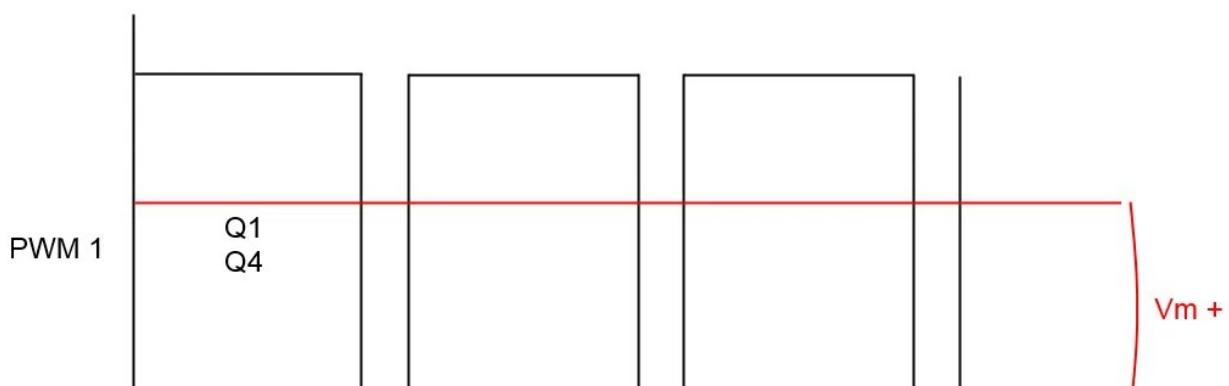
El L293D tiene dos puentes H y proporciona 600mA al motor y soporta un voltaje entre 4,5V y 36V tal y como pone en el datasheet: <http://www.ti.com/lit/ds/symlink/l293d.pdf>. Estas especificaciones nos indican el tipo de motores que podremos usar con este driver.

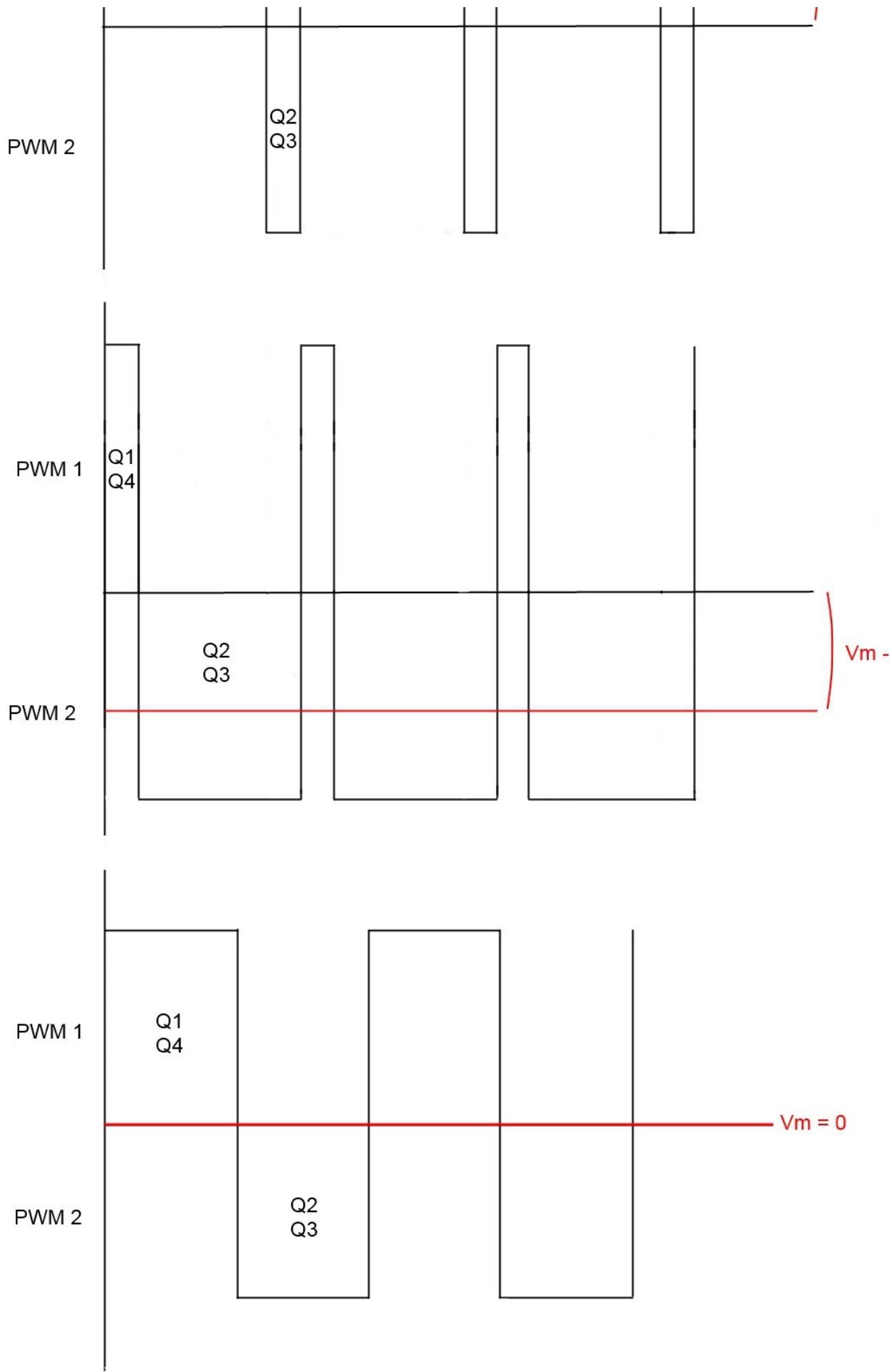


Nosotros usaremos la parte de la izquierda (los diodos externos hay que ponerlos para evitar las corrientes inducidas del motor si usamos el modelo L293 y están incluidos en el integrado si usamos el L293D). Como se aprecia en la imagen, los pins 3 y 6 son las salidas y se conectan a los bornes del motor. Y los pins 2 y 7 son las entradas donde conectaremos las salidas del Arduino. Dependiendo del valor aplicado entre los pins 2 y 7 el motor girará en un sentido o en otro.

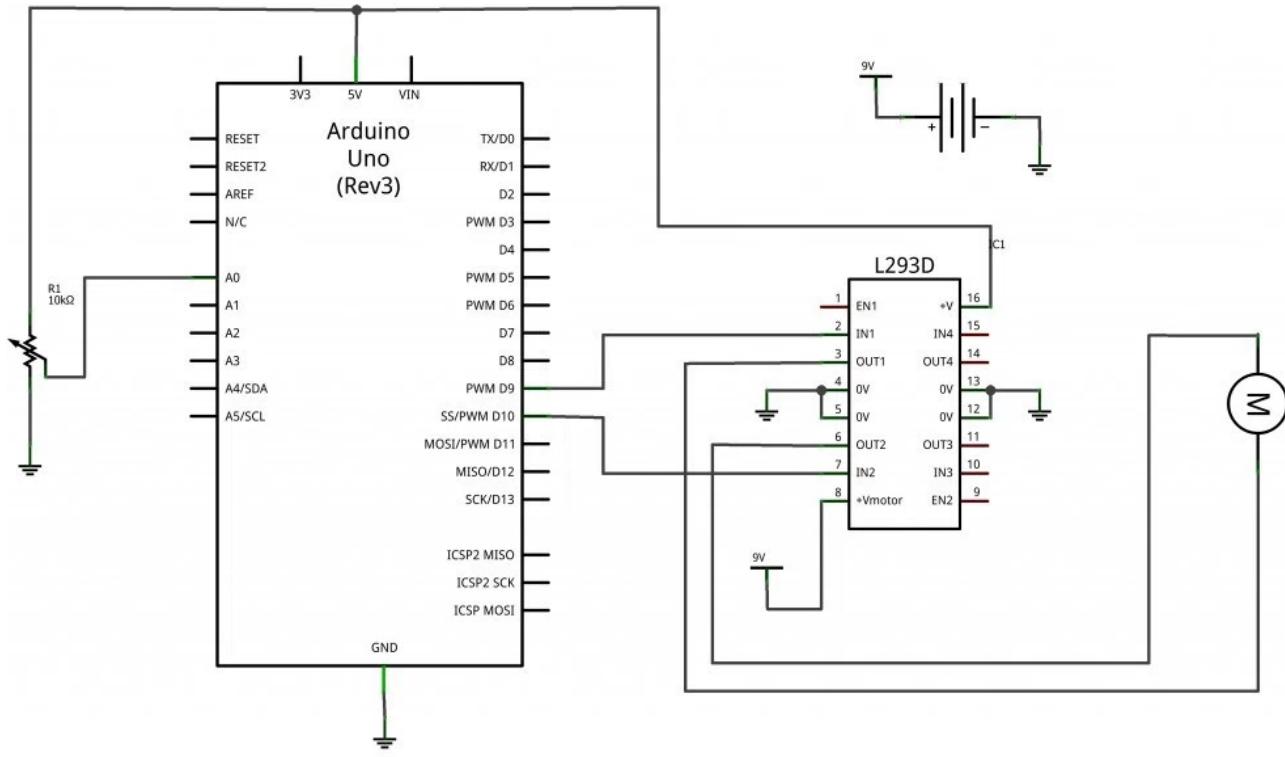
**IMPORTANTE:** si se utiliza el driver L293, hay que poner diodos para evitar dañar el integrado con las corrientes parásitas generadas por los propios solenoides de las cargas. No obstante el modelo L293D no los necesita, ya que, los lleva incorporados el propio integrado, lo que se hace "más sencillo" y "económico" su uso. También es cierto que L293 al no llevar los diodos integrados nos permite escoger los que mejor se adapten a nuestras cargas o necesidades.

Para controlar la velocidad del motor se usa la técnica de PWM. Sabemos que hay que atacar los pins 2 y 7 del L293D desde dos salidas del Arduino. En estas dos salidas habrá un PWM a cada una. Pero tenemos que invertir un PWM. ¿Qué quiere decir invertir? Pues que cuando en un PWM tengamos un pulso a un valor alto, en el otro PWM el mismo pulso sea valor bajo. En la imagen lo entenderemos de una manera más gráfica.





## Montaje:



Más información y código en: <http://diymakers.es/control-velocidad-y-sentido-de-motor-dc/>

Ejemplo del playground de Arduino:

<http://playground.arduino.cc/Main/DirectionalMotorControlWithL293D>

## Driver L298

Otro driver de motor muy utilizado es el L298 (datasheet) que es el utilizado por el Arduino motor shield. Este driver es similar en funcionamiento al anterior pero posee un sensor de corriente muy útil y puede manejar corrientes más grandes que el L293D.

Arduino motor shield: <https://www.arduino.cc/en/Main/ArduinoMotorShieldR3>

## Montaje:

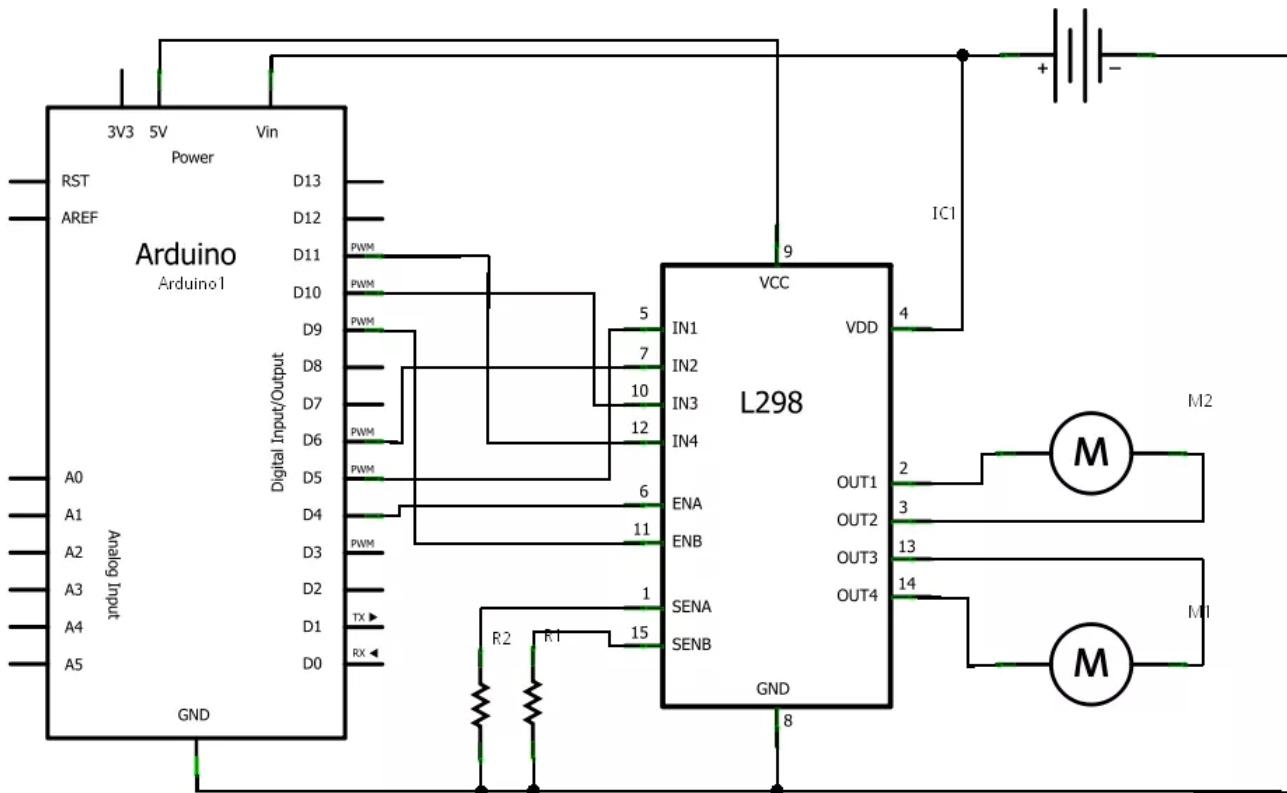
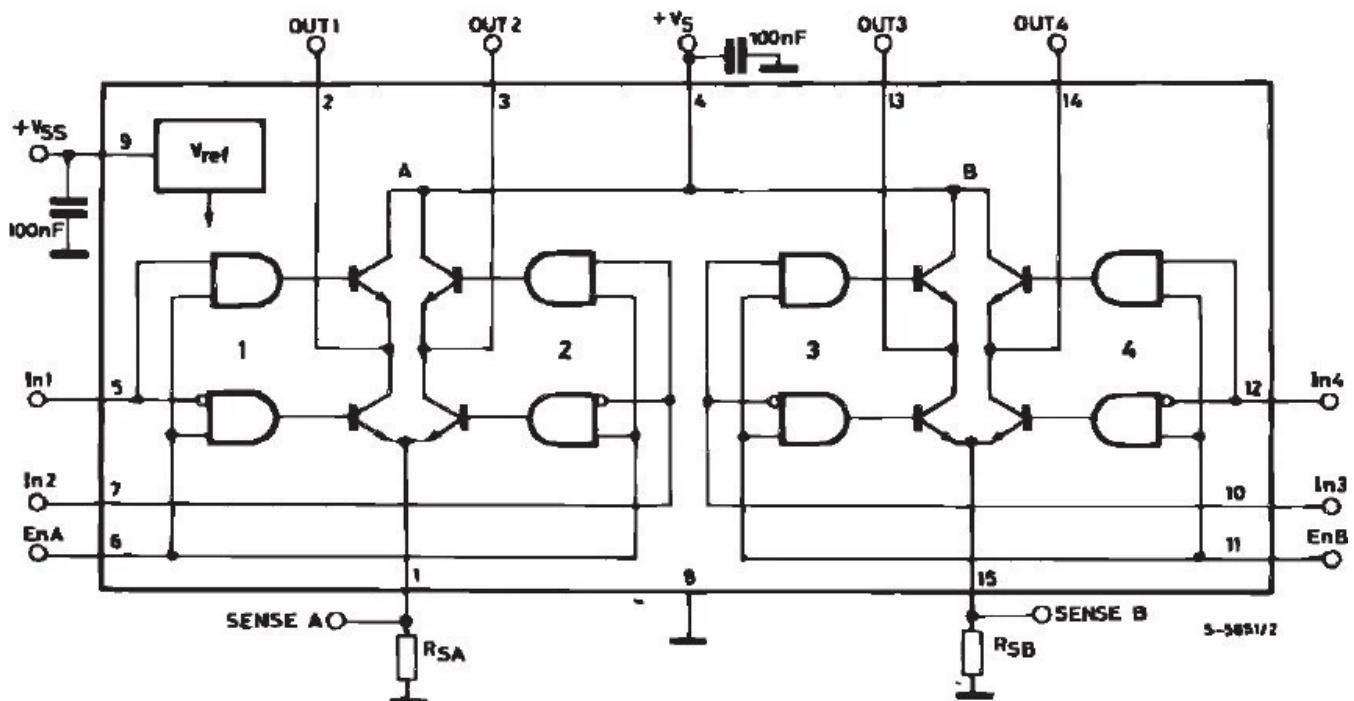
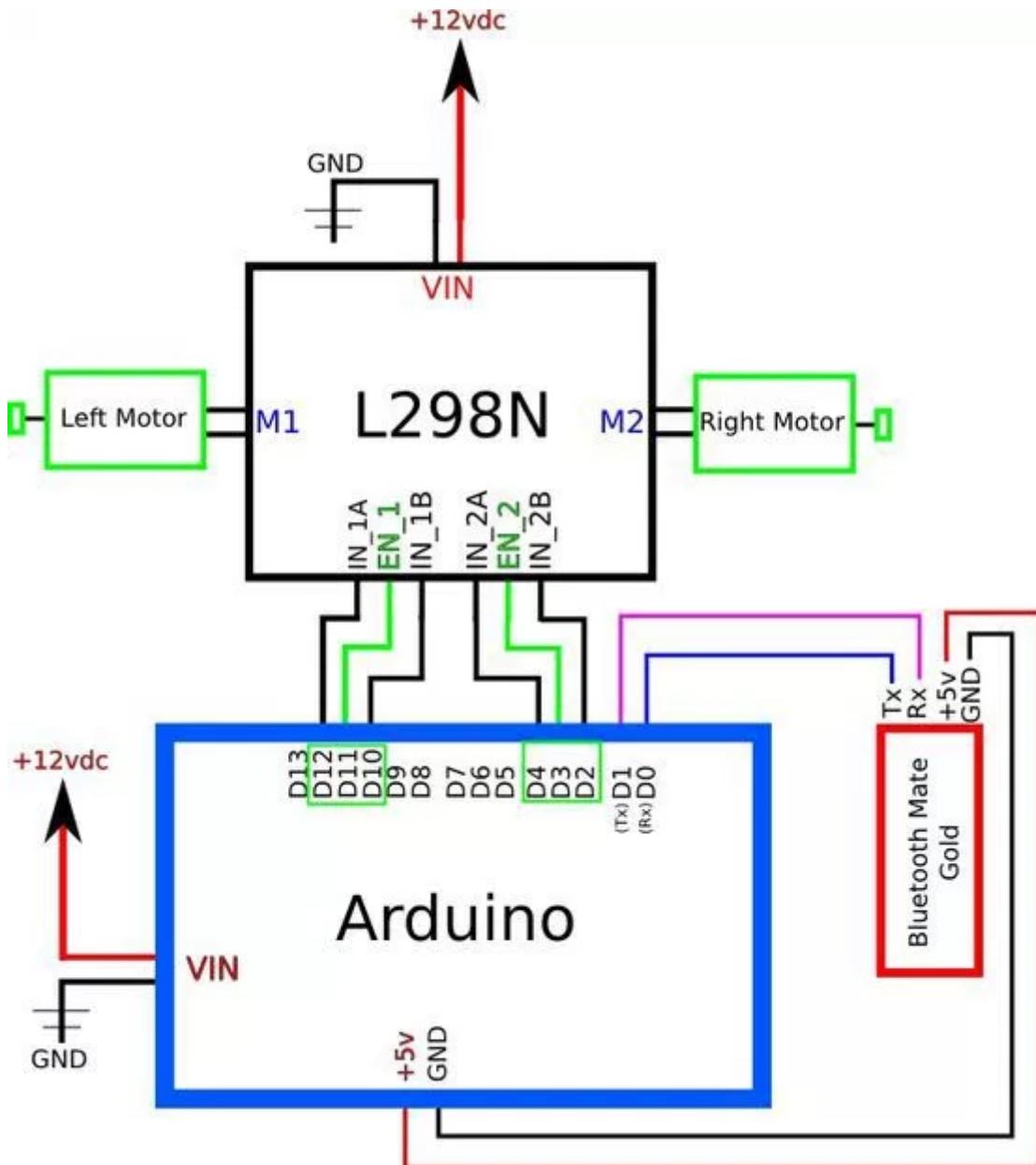


Diagrama:

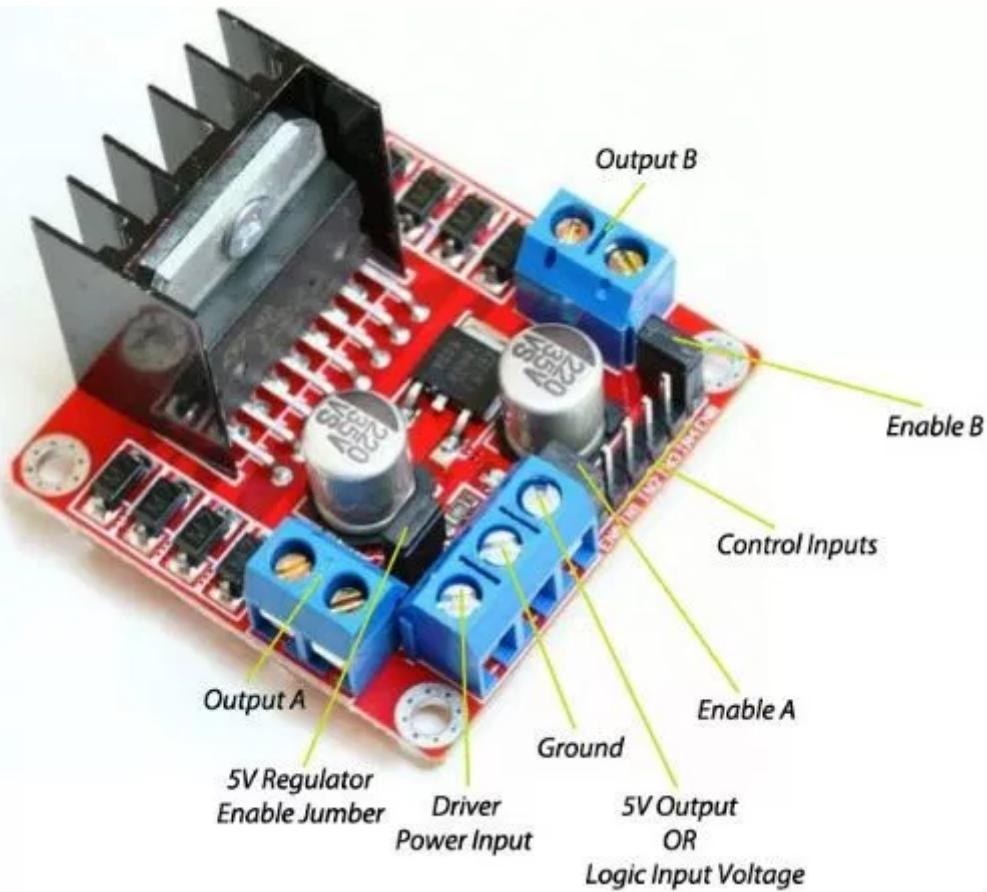


Esquema de conexión:



**L298N Breakout Board** (Esta breakout board es muy sencilla y bien documentada):

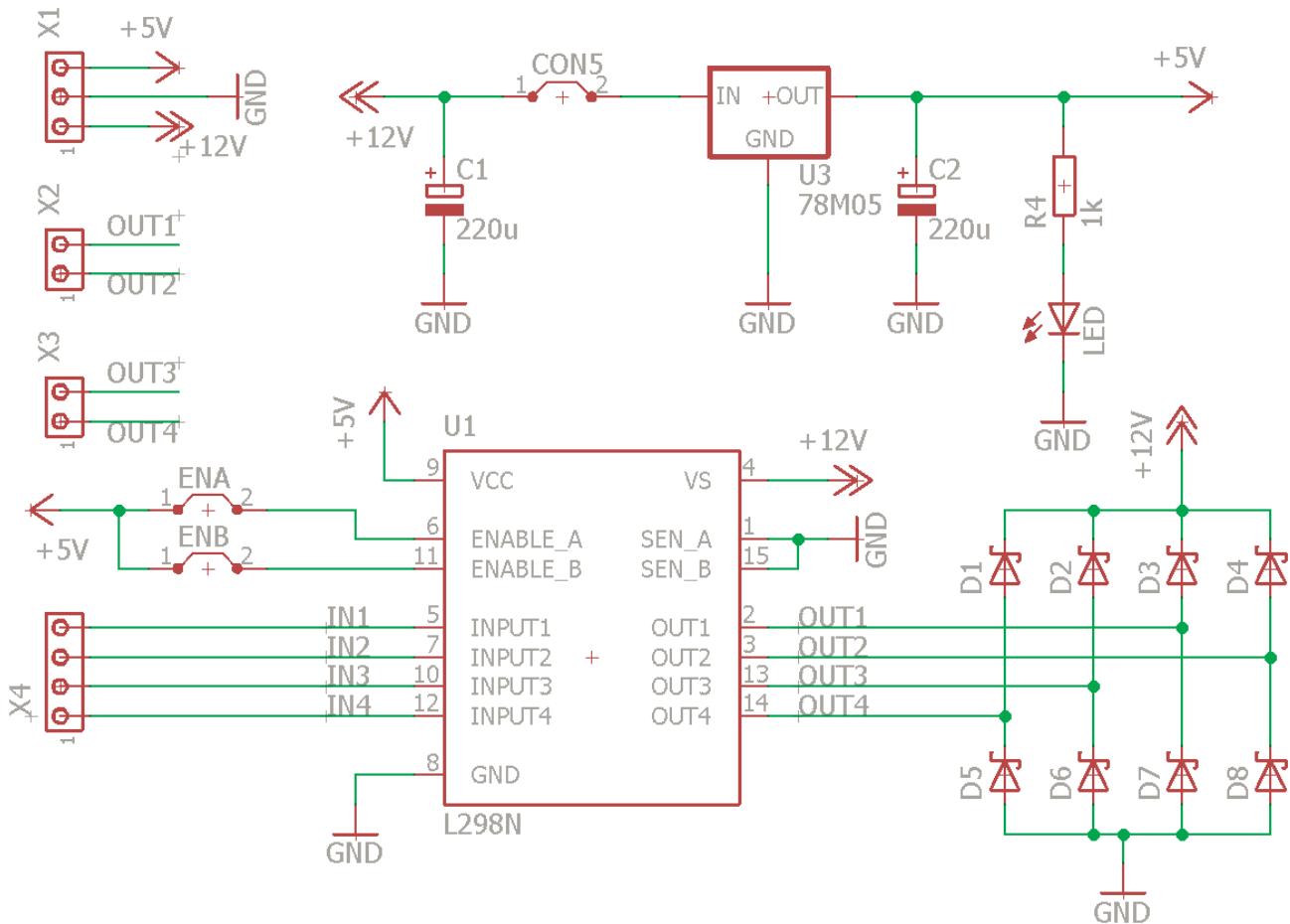
- Producto: <http://www.geeetech.com/l298n-stepper-motor-driver-board-p-567.html>
- Wiki: [http://www.geeetech.com/wiki/index.php/L298N\\_Motor\\_Driver\\_Board](http://www.geeetech.com/wiki/index.php/L298N_Motor_Driver_Board) (con esquemático)
- El que usa el coche: [http://www.vetco.net/catalog/product\\_info.php?products\\_id=14337](http://www.vetco.net/catalog/product_info.php?products_id=14337)
- Test: <https://developer.mbed.org/teams/TVZ-Mechatronics-Team/code/L298N-Breakout-Test/>



#### Specification:

- chipset: L298N
- Driving power supply voltage Vs: +5V to +46V
- Peak current of driving power supply Io: 2A
- Vss: +5V to +7V
- Current of logic power supply: 0 – 36mA
- PWM control signal range:
  - Low level: -0.3V < Vin < 1.5V
  - High level: 2.3V < Vin < Vss
- Enable signal range:
  - Low level: -0.3V
  - High level: 2.3V < Vin < Vss
- Maximum power consumption: 25W
- Working temperature: -25C to 130C
- Regulador de tensión para los 5V.

#### Esquemático:



Diferencias entre el L293 y el L298:

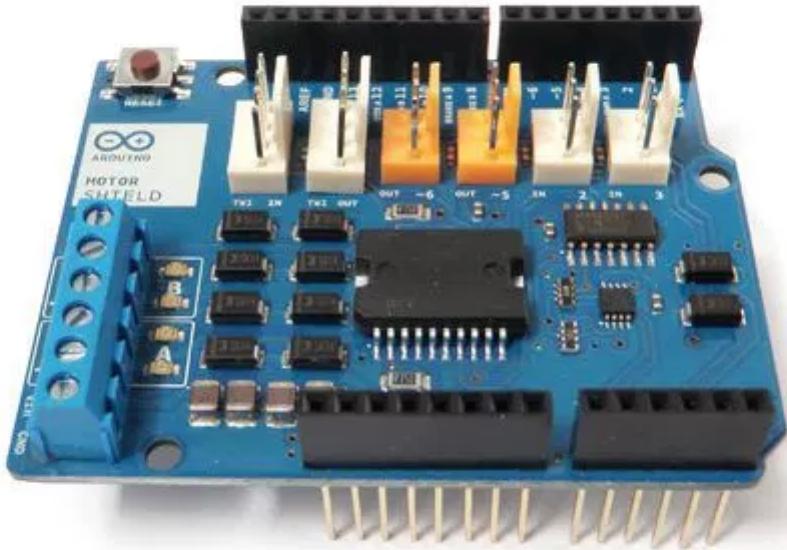
- L293 es un quadruple half-H driver y el L298 es un dual full-H driver
- El L293 al tener 4 canales permite manejar motores paso a paso de 4 hilos y el L298 solo tiene dos canales perfecto para manejar dos motores DC. En el L293 las cuatro líneas de entrada/salida son independientes.
- La corriente de salida por canal en el L293 es de 1A, mientras que en el L298 es de 2A, por ese motivo el L298 puede llevar un disipador.
- Los diodos protectores del L293D deben ponerse externamente en el L293 y L298

Video brushed DC motors <https://www.baldengineer.com/videos/brushed-dc-motor-video-tutorial>

Buen tutorial de uso de Arduino con L298: <http://www.electrontools.com/Home/WP/2016/05/06/puente-h-con-driver-l298/>

## Arduino Motor Shield

El Arduino motor shield está basado en el L298 que es un dual full-bridge diseñado para cargas inductivas como relés, solenoides y motores DC o paso a paso. Me permite manejar dos motores DC controlando las velocidad y dirección de cada motor de forma independiente. También permite medir la corriente absorbida por cada motor entre otras características. La corriente máxima que puede manejar con 2A por canal y el voltaje máximo de suministro es de 50V.



Esquemático: [https://www.arduino.cc/en/uploads/Main/arduino\\_MotorShield\\_Rev3-schematic.pdf](https://www.arduino.cc/en/uploads/Main/arduino_MotorShield_Rev3-schematic.pdf)

El L298 necesita dos alimentaciones, una para la parte de control (5V) y otra para los motores (12V). La de los motores en tu caso la coge del Vin al que conectas la fuente conmutada de 12V.

Los pines de alimentación en el Arduino motor shield son:

- Vin en la borna de conexiones, es la entrada de alimentación a los motores conectados al shield. Una fuente de alimentación externa conectada a esta borna también alimenta a la placa Arduino sobre la que va montada el shield. Cortando el jumper “Vin Connect” se elimina esta alimentación a Arduino.
  - GND es la masa de los motores que va unida a la masa del Arduino.

Este shield dispone de dos canales llamados A y B y cada uno usa 4 de los pines de Arduino para manejar o monitorizar el motor. En total se usan 8 pines. Se pueden usar los canales separados para manejar dos motores DC o combinarlos para manejar un motor paso a paso bipolar.

Los pinos son:

Function	pins per Ch. A	pins per Ch. B
<i>Direction</i>	D12	D13
<i>PWM</i>	D3	D11
<i>Brake</i>	D9	D8

---

*Current Sensing*

A0

A1

Si no se necesita el freno y el sensor de corriente, es posible deshabilitar estas características cortando los correspondientes jumpers en la parte trasera del shield. Los pines de freno al ponerlos a HIGH frenan el motor en lugar de dejarlos correr libremente al cortar la alimentación.

Mediante la función analogRead() es posible leer la corriente en los pines de current sensing como una entrada analógica normal. Está calibrado para medir 3.3V para la corriente máxima de 2A.

Esquema: [https://www.arduino.cc/en/uploads/Main/arduino\\_MotorShield\\_Rev3-schematic.pdf](https://www.arduino.cc/en/uploads/Main/arduino_MotorShield_Rev3-schematic.pdf)

Más información: <https://www.arduino.cc/en/Main/ArduinoMotorShieldR3>

Ejemplos de uso:

- <https://www.arduino.cc/en/Tutorial/DueMotorShieldDC>
- <http://www.arduino.org/learning/tutorials/boards-tutorials/arduino-motor-shield-two-dc-motors-example>
- <http://www.instructables.com/id/Arduino-Motor-Shield-Tutorial/>

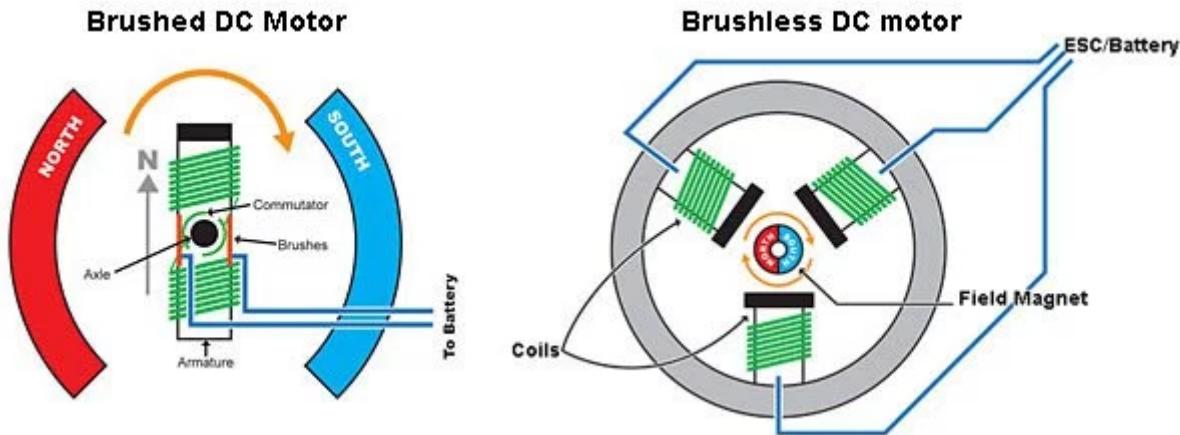
## **Motor DC Brushless**

Un motor DC sin escobillas o motor brushless es un motor eléctrico que no emplea escobillas para realizar el cambio de polaridad en el rotor. Los motores eléctricos solían tener un colector de delgas o un par de anillos rozantes. Estos sistemas, que producen rozamiento, disminuyen el rendimiento, desprenden calor y ruido, requieren mucho mantenimiento y pueden producir partículas de carbón que manchan el motor de un polvo que, además, puede ser conductor.

Los primeros motores sin escobillas fueron los motores de corriente alterna asincrónos. Hoy en día, gracias a la electrónica, se muestran muy ventajosos, ya que son más baratos de fabricar, pesan menos y requieren menos mantenimiento, pero su control era mucho más complejo. Esta complejidad prácticamente se ha eliminado con los controles electrónicos.

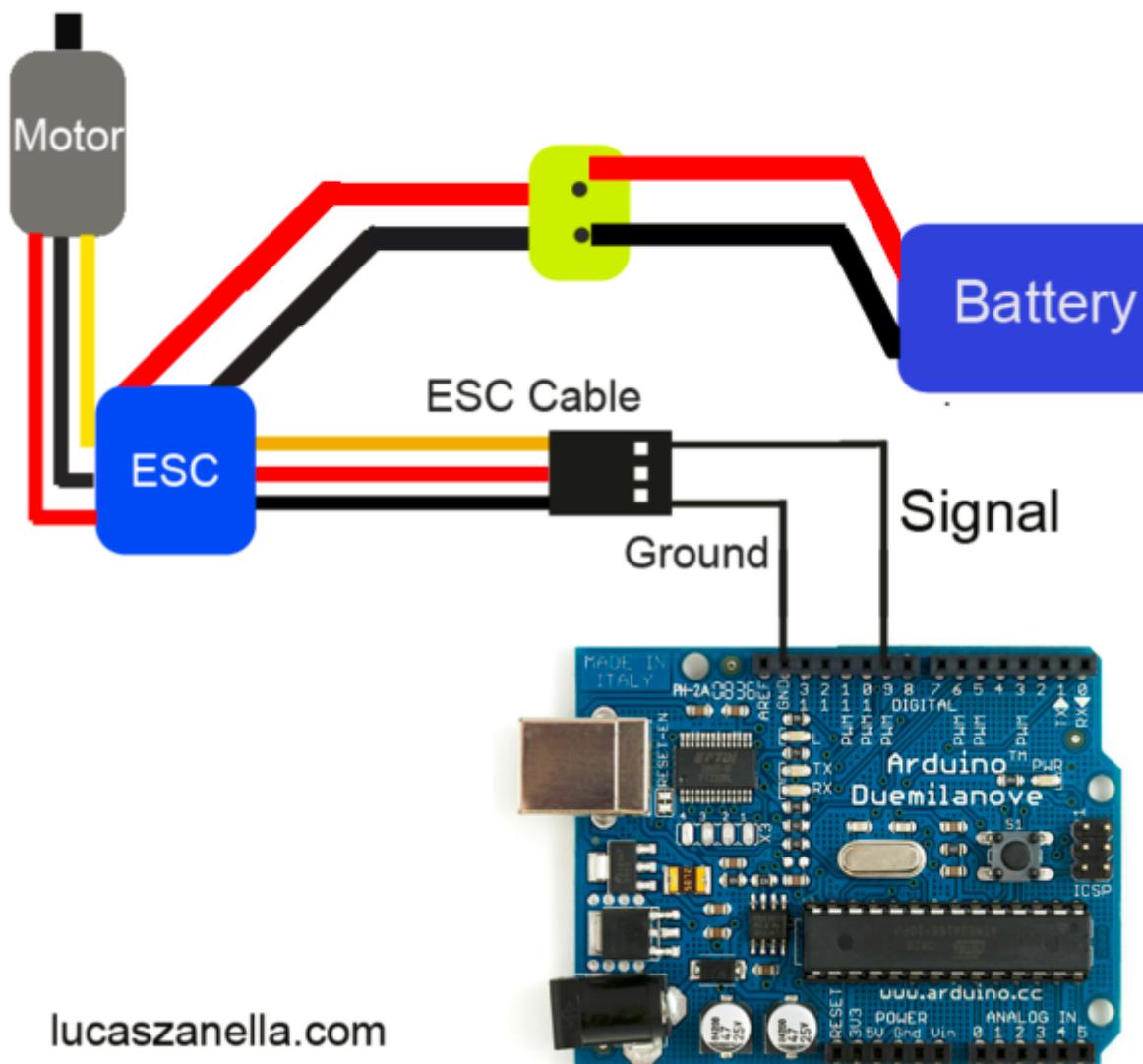
Más información:

- [https://en.wikipedia.org/wiki/Brushless\\_DC\\_electric\\_motor](https://en.wikipedia.org/wiki/Brushless_DC_electric_motor)
- [https://es.wikipedia.org/wiki/Motor\\_el%C3%A9ctrico\\_sin\\_escobillas](https://es.wikipedia.org/wiki/Motor_el%C3%A9ctrico_sin_escobillas)



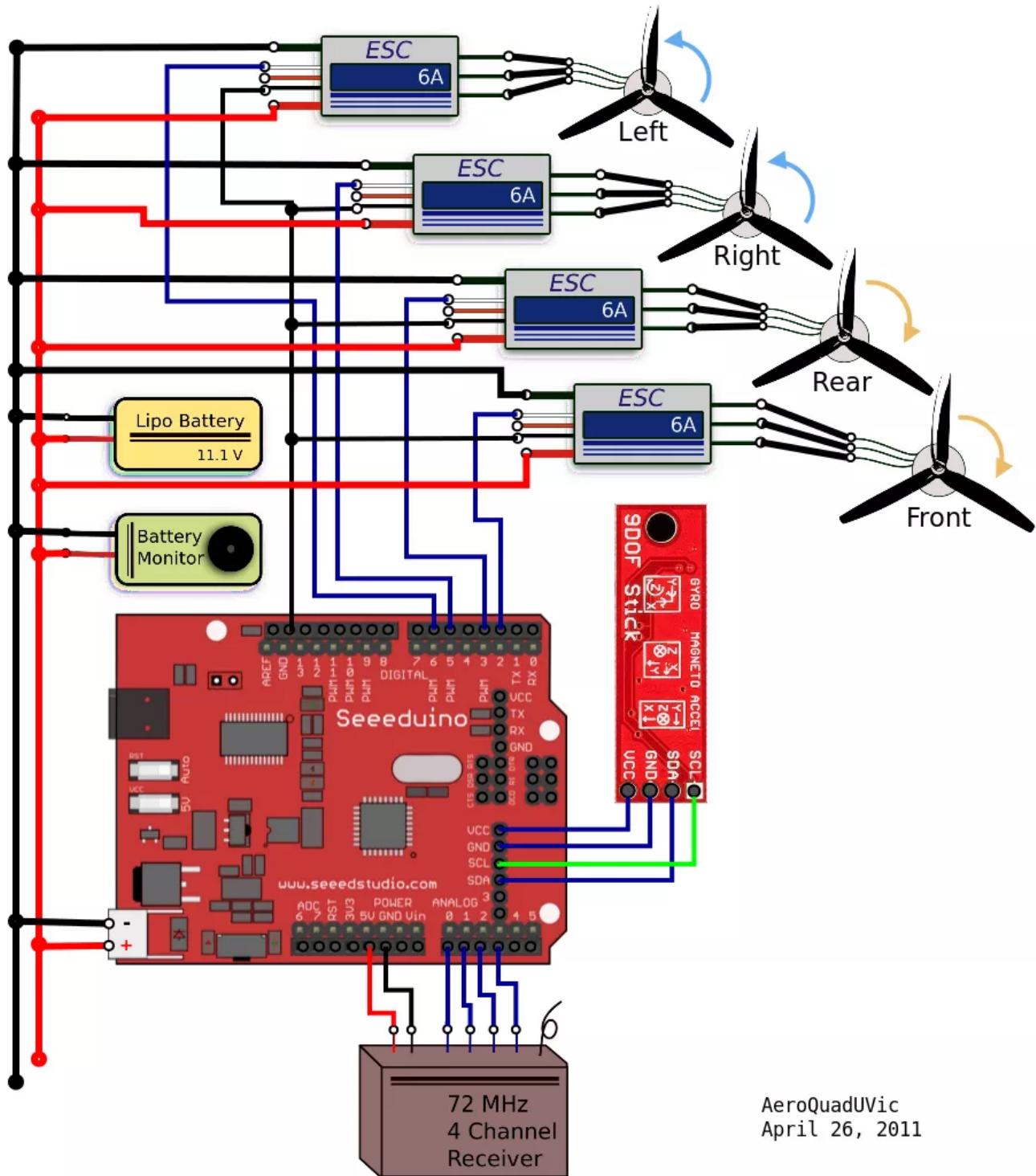
Para controlar los motores brushless necesitaremos un ESC (Electronic Speed Control)

[https://en.wikipedia.org/wiki/Electronic\\_speed\\_control](https://en.wikipedia.org/wiki/Electronic_speed_control)



El inversor debe convertir la corriente continua de la batería o fuente de alimentación en alterna de una frecuencia determinada. Algunas aplicaciones serían los coches y aviones con radiocontrol, que funciona con pilas.

Los motores brushless se usan habitualmente en los drones y son trifásicos con un variador para controlar de forma muy exacta la velocidad del motor.



AeroQuadUVic  
April 26, 2011

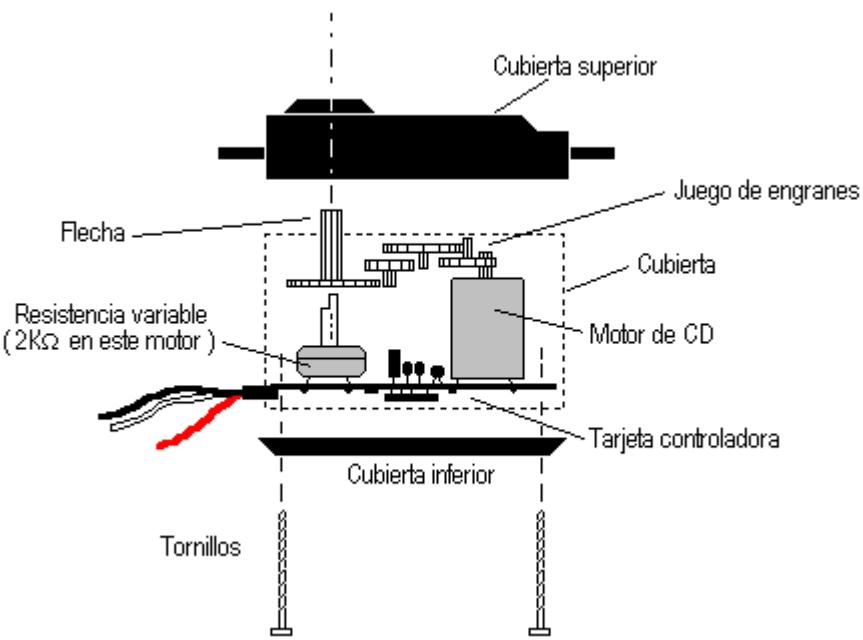
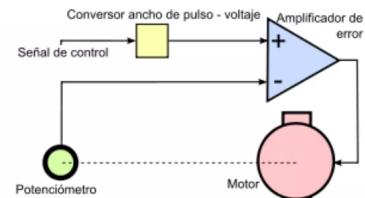
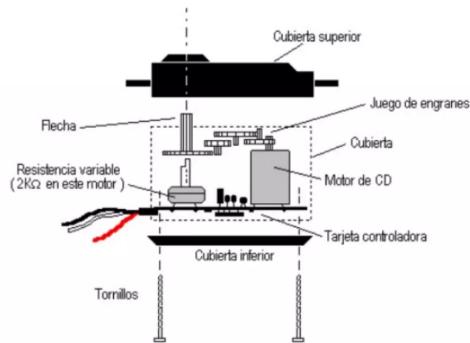
Más información: <https://learn.adafruit.com/adafruit-motor-selection-guide/brushless-dc-motor-control>

## Servomotor

Servomotor (o también llamado servo) es similar a un motor de corriente continua pero con la capacidad de posicionarse en una posición determinada y permanecer fija en esta. Normalmente el ángulo es de 0 a 180 grados y se alimentan a 5 voltios mínimo.

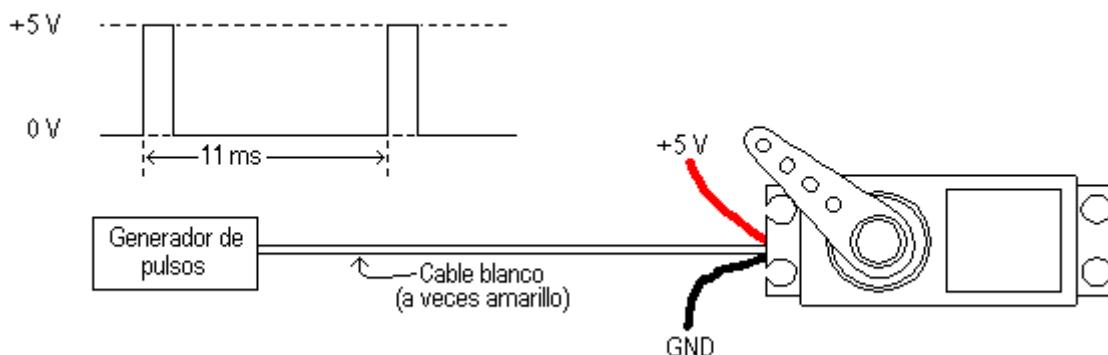
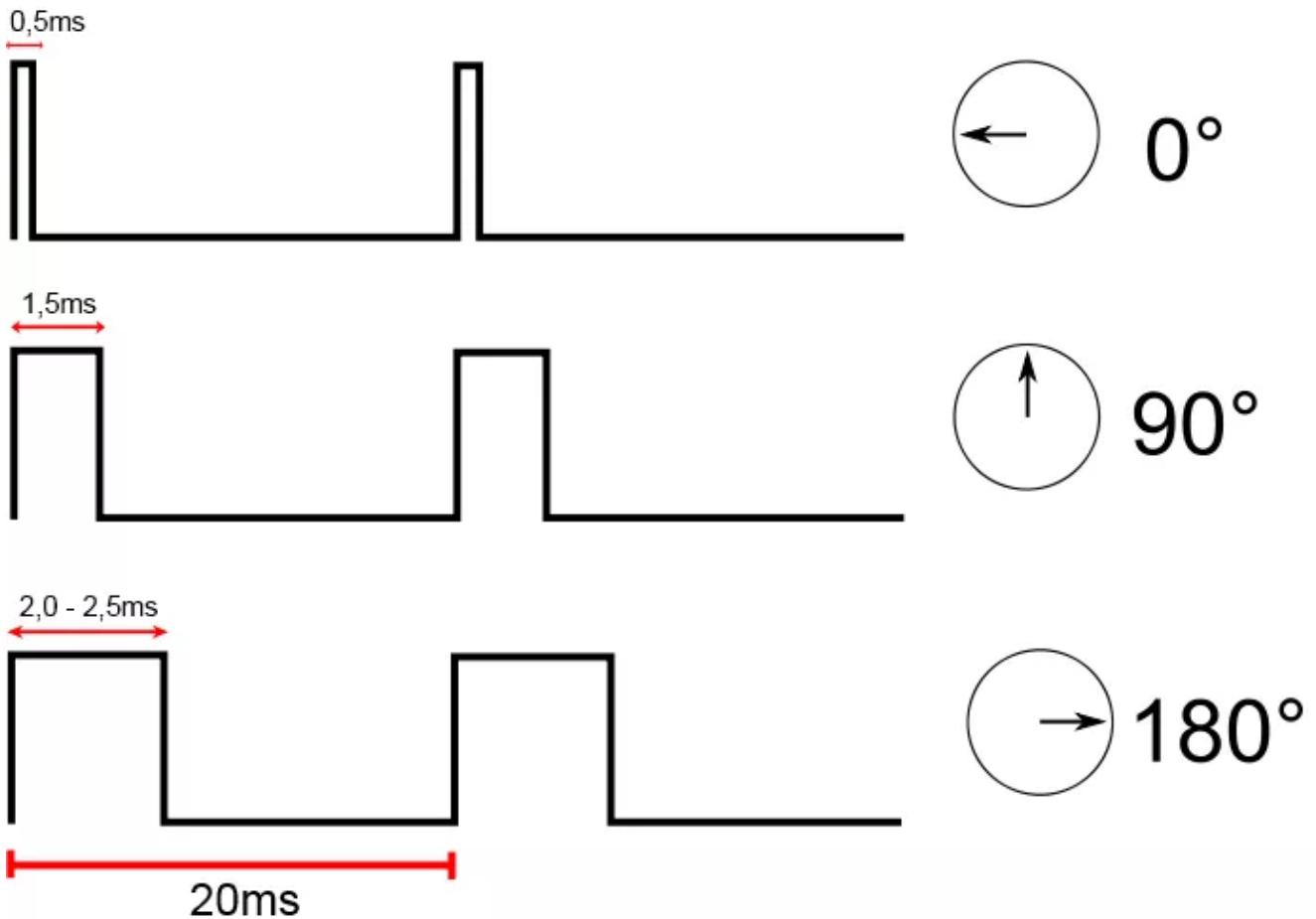
Algunos servos servos pueden alimentarse directamente desde Arduino sin necesidad de un driver, lo que supone una ventaja. En caso de usar varios servos con un mismo Arduino, este no podrá alimentarlos y habrá que usar una fuente de alimentación externa.

Un servomotor está formado por un motor de corriente continua, una caja reductora, un juego de engranajes, un potenciómetro y un circuito de control. Puede aguantar cierto peso a través del par o torque del servo indicado en sus características. Normalmente se indica con Kg/cm, que quiere decir los kilos que aguanta a 1 cm de distancia.



Para controlar un servo, se usa el PWM. La mayoría de los servos trabajan en una frecuencia de 50 Hz (20ms). Cuando se manda un pulso, la anchura de este determina la posición angular del servo. La anchura varía según el servomotor pero normalmente es entre 0,5ms a 2,5ms.

En Arduino la frecuencia por defecto de PWM es de 500 Hz, pero con la librería servo conseguimos una señal PWM con la frecuencia y duty cycle adecuada.



El Arduino utiliza la librería `<Servo.h>` para controlar los servos y dispone de las siguientes funciones:

- <http://arduino.cc/en/Reference/Servo>
- <http://arduino.cc/en/Reference/ServoAttach>
- <http://arduino.cc/en/Reference/ServoWrite>
- <http://arduino.cc/en/Reference/ServoRead>

Fuente, más información y código en: <http://diymakers.es/controlar-servomotor-con-mando-ir/>

Más información de servos:

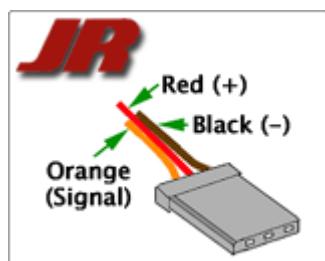
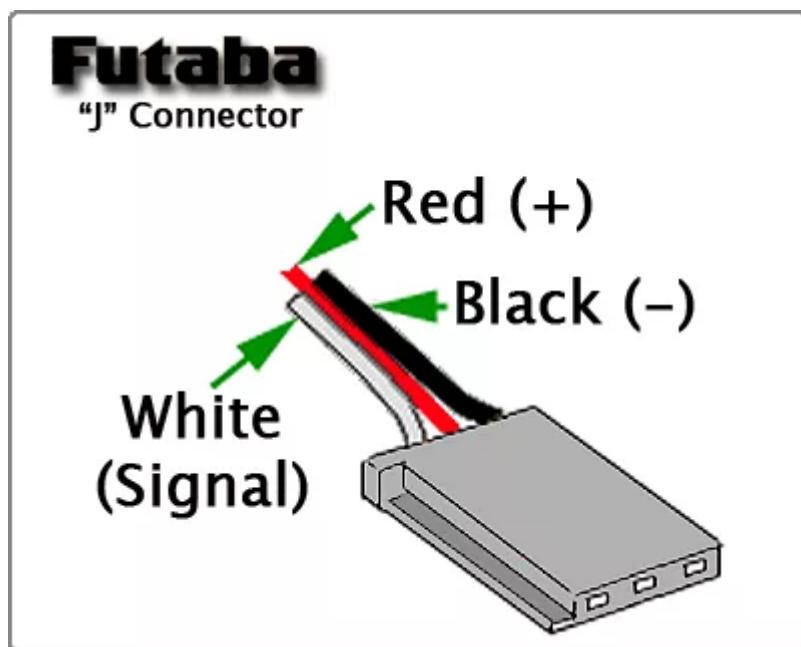
[http://platea.pntic.mec.es/vgonzale/cyr\\_0204/ctrl\\_rob/robotica/sistema/motores\\_servo.htm](http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/sistema/motores_servo.htm)

**IMPORTANTE:** La librería servo soporta hasta 12 servos en la mayoría de los Arduinos, pero deshabilita el uso de PWM en los pines 9 y 10 puesto que hace uso del timer que controla esas señales. En el caso del mega hay otras restricciones.

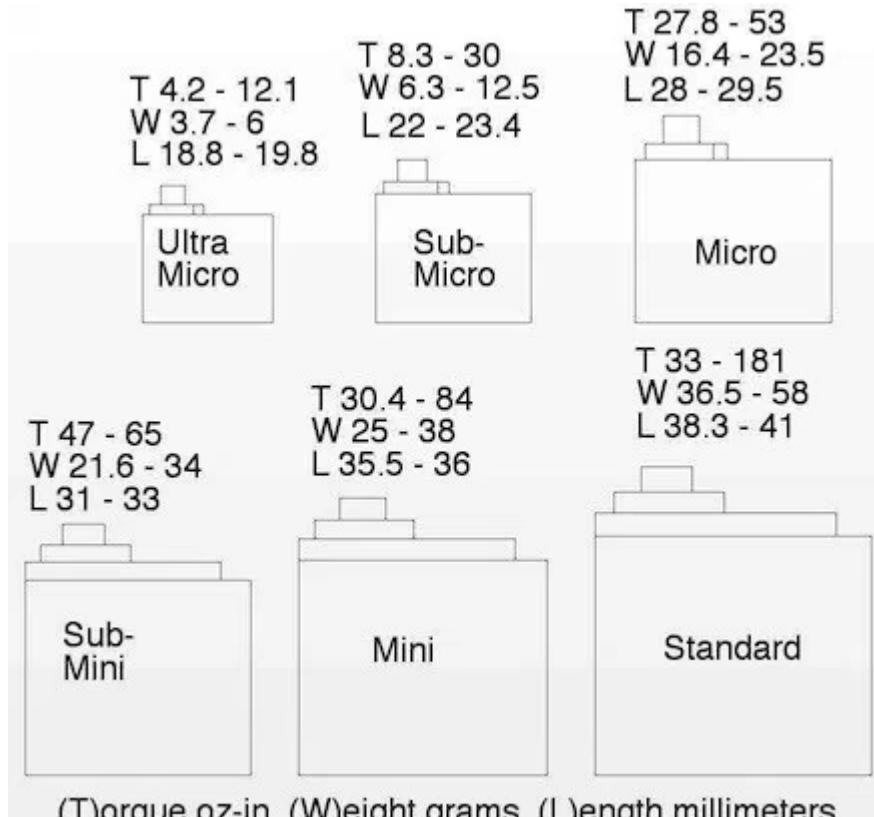
Más información:

- <http://en.wikipedia.org/wiki/Servomotor>
- <http://www.info-ab.uclm.es/labelec/solar/electronica/elementos/servomotor.htm>
- Video explicativo de como funciona un servo: <https://www.youtube.com/watch?v=tsrAP8EgcbQ>

Todos los servos tienen 3 cables de conexión uno para alimentación DC (normalmente 5-6V), otro para masa y el tercero para la señal.



Los tamaños de los servos son:



Características de los servos:

- **Torque:** determina la máxima fuerza rotacional que puede aplicar el servo, se mide en kg.cm. Cambia a mínimo y máximo voltaje
- **Voltaje:** mínimo y máximo
- **Corriente:** con carga y sin carga.
- **Velocidad:** la medida del tiempo que tarda en rotar un cierto número de grados, se mide en sec/60°. Es diferente a mínimo y máximo voltaje.

Tipos de servos, en función de cómo es procesada la señal:

- **Analógico:** aplica señales de voltaje on off para controlar la velocidad. El voltaje es constante. La frecuencia está estandarizada a 50 ciclos por segundo.
- **Digital:** tiene un microprocesador dentro que analiza la señal y la procesa en pulsos de voltaje de alta frecuencia hasta 300 pulsos por segundo. Los pulsos serán más cortos en tiempo. Con esto se consigue una mayor velocidad y un torque constante. Estos servos tienen un tiempo de respuesta mayor y una mejor capacidad de retención. Estos servos tienen la desventaja de un mayor consumo eléctrico.

Más información sobre servos analógicos y digitales: <http://www.futabarc.com/servos/digitalservos.pdf>

La mayoría de los servos estándar y low cost, usan un motor DC estándar de 3 polos. Hay servos con motores de 5 polos que hacen que el servo sea más rápido en aceleración y mayor torque en el arranque. Pero también hay servos:

- **Coreless:** el núcleo de acero es sustituido por un bobinado ligero.

- **Brushless.** Son más eficiente y dan más fuerza, además de una vida útil mayor.

El consumo eléctrico de un servo depende de la carga y el modelo del servo. Un servo analógico estándar consume entre 250-700mA. Por lo tanto si alimentamos varios servos desde el pin de 5V de Arduino es posible que nos sea posible moverlos por la limitación en amperios de Arduino, esa limitación viene del regulador de tensión que alimenta el bus de 5V de Arduino. Por ejemplo en el caso de Arduino UNO el regulador es el NCP1117ST50T3G que como puede verse en su datasheet

[http://www.onsemi.com/pub\\_link/Collateral/NCP1117-D.PDF](http://www.onsemi.com/pub_link/Collateral/NCP1117-D.PDF) puede dar hasta 1 amperio, aunque soporta picos mayores.

Algunos servos comerciales y características:

- <https://www.servocity.com/>
- <https://www.servocity.com/servos/hitec-servos>
- <https://www.servocity.com/servos/futaba-servos>
- Servos lineales:<https://www.servocity.com/servos/heavy-duty-linear-servos>

Ranuras de los servos: <https://www.servocity.com/servo-spline-info>

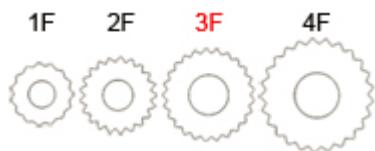


Tabla de selección de servos: [https://www.ia.omron.com/data\\_pdf/guide/14/servo\\_selection\\_tg\\_e\\_1\\_1\\_3-13\(further\\_info\).pdf](https://www.ia.omron.com/data_pdf/guide/14/servo_selection_tg_e_1_1_3-13(further_info).pdf)

## Controladores para servos

Un servo puede controlarse directamente con un Arduino para generar la señal de control que hemos visto anteriormente.

También es posible adquirir controladores de servo y especialmente con capacidad de radiocontrol o para manejar muchos canales.

Controladores servo de pololu: <https://www.pololu.com/category/12/rc-servo-controllers>

- Controlador de 6 canales: <https://www.pololu.com/product/1350>
- Controlador servo de 18 canales: <https://www.servocity.com/mini-maestro-18-channel-usb> e información: <https://www.servocity.com/files/index/download/id/1456932868/>

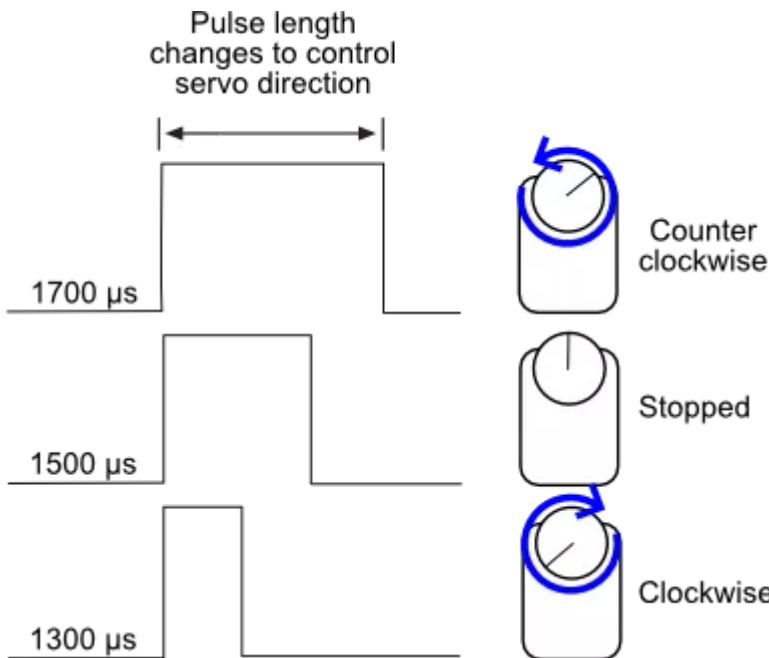
Torobot controlador de 32 canales para servos:

- <http://www.bajdi.com/torobot-32-channel-servo-controller/>
- [http://letsmakerobots.com/files/32\\_Servo\\_Controller\\_Manual.pdf](http://letsmakerobots.com/files/32_Servo_Controller_Manual.pdf)
- <http://www.elabpeers.com/32-channel-servo-controller.html>

## Servo de rotación continua

Un servo de rotación continua es un motor cuyo circuito electrónico nos permite controlar la dirección de giro. A diferencia de los servos anteriormente mencionados, no se detiene en una posición, sino que gira continuamente. Son muy utilizados en robótica y en muchas aplicaciones electrónicas, como en lectores de DVD.

En un servo de rotación continua, la función `write()` configura la velocidad del servo en lugar del ángulo de posición. En este caso 0 es máxima velocidad en giro contrario al sentido horario, 180 es máxima velocidad en sentido horario y 90 motor parado.



Trucar un servo a rotación continua: <http://www.ardumania.es/trucar-servo-a-rotacion-continua/>

Servos de rotación continua comerciales:

- <http://tienda.bricogEEK.com/motores/118-servomotor-de-rotacion-continua-sm-s4303r.html>
- [https://www.servocity.com/html/hsr-2645cr\\_servo\\_continuous\\_r.html#.VqiH0\\_nhDct](https://www.servocity.com/html/hsr-2645cr_servo_continuous_r.html#.VqiH0_nhDct)
- <https://www.pololu.com/product/1248>

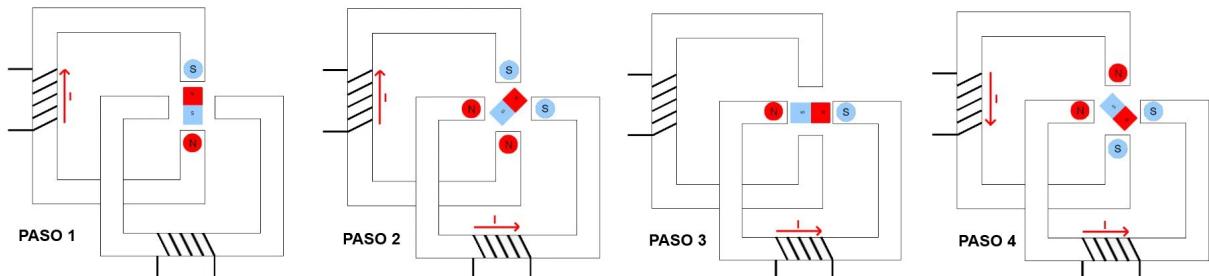
Tutorial sencillo de BQ: <http://diwo.bq.com/muevete-el-servo-de-rotacion-continua/>

## Motor paso a paso

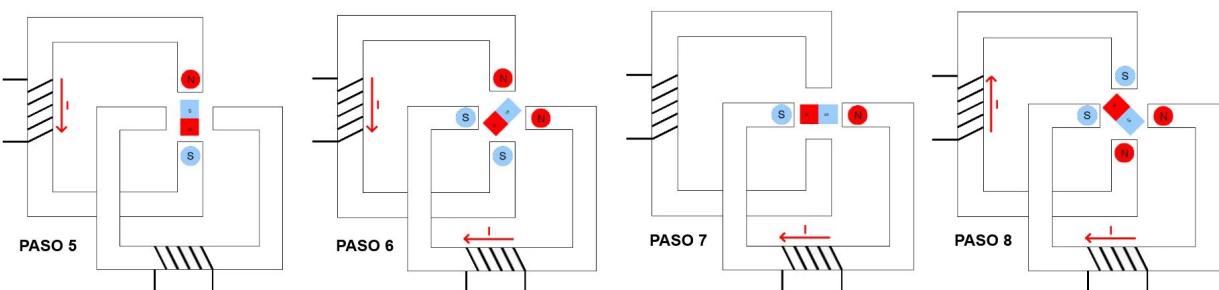
Un motor paso a paso (también llamado stepper) es un dispositivo electromagnético que convierte impulsos eléctricos en movimientos mecánicos de rotación. La principal característica de estos motores es que se mueven un paso por cada impulso que reciben. Normalmente los pasos pueden ser de 1,8º a 90º por paso, dependiendo del motor. Son motores con mucha precisión, que permiten quedar fijos en una posición (como un servomotor) y también son capaces de girar libremente en un sentido u otro (como un motor DC).



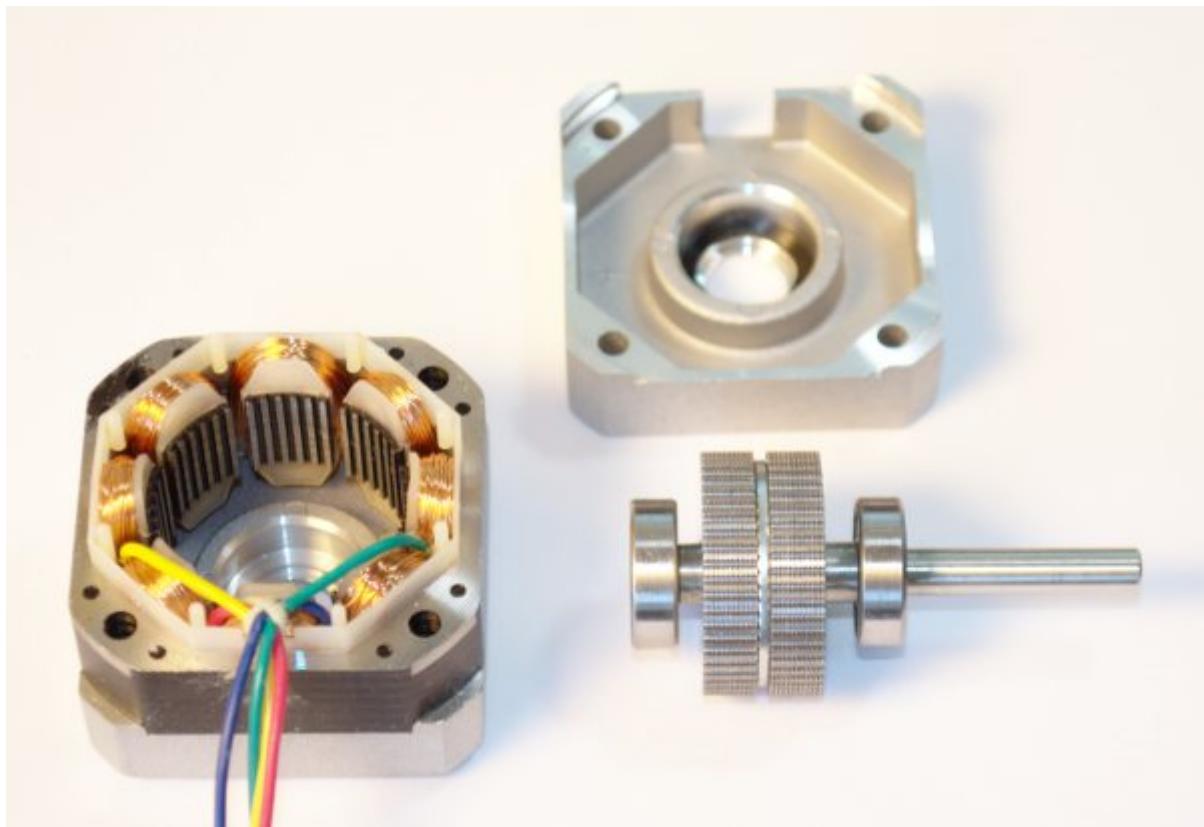
Cuando circula corriente por una o más bobinas del estator se crea un campo magnético creando los polos Norte-Sur. Luego el rotor se equilibrará magnéticamente orientando sus polos Norte-Sur hacia los polos Sur-Norte del estator. Cuando el estator vuelve a cambiar la orientación de sus polos a través de un nuevo impulso recibido hacia sus bobinas, el rotor volverá a moverse para equilibrarse magnéticamente. Si se mantiene esta situación, obtendremos un movimiento giratorio permanente del eje. El ángulo de paso depende de la relación entre el nombre de polos magnéticos del estator y el nombre de polos magnéticos del rotor.



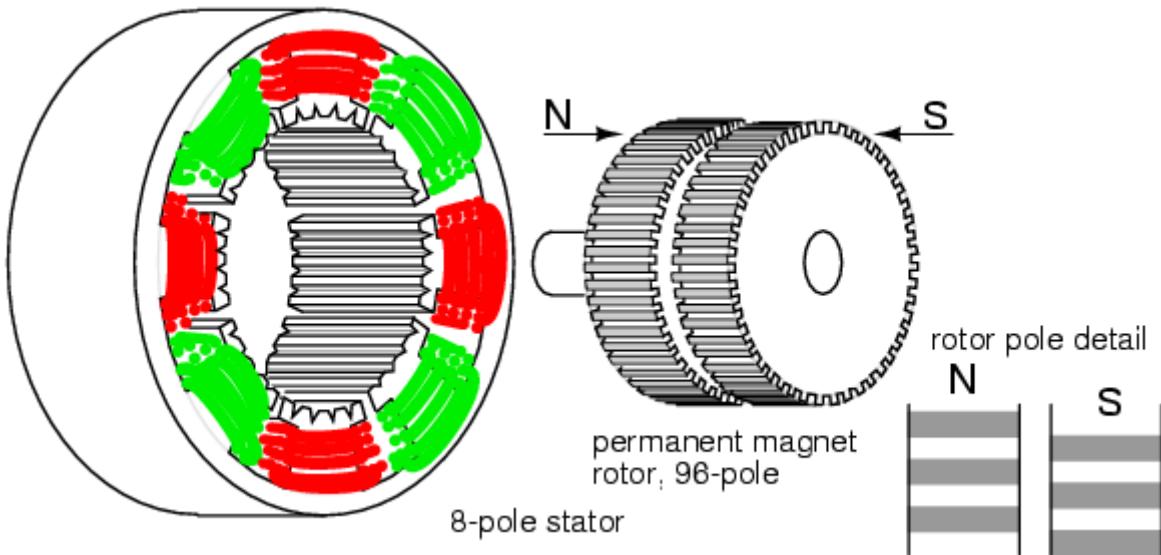
MOTOR PASO A PASO DE 45° DE PASO ANGULAR



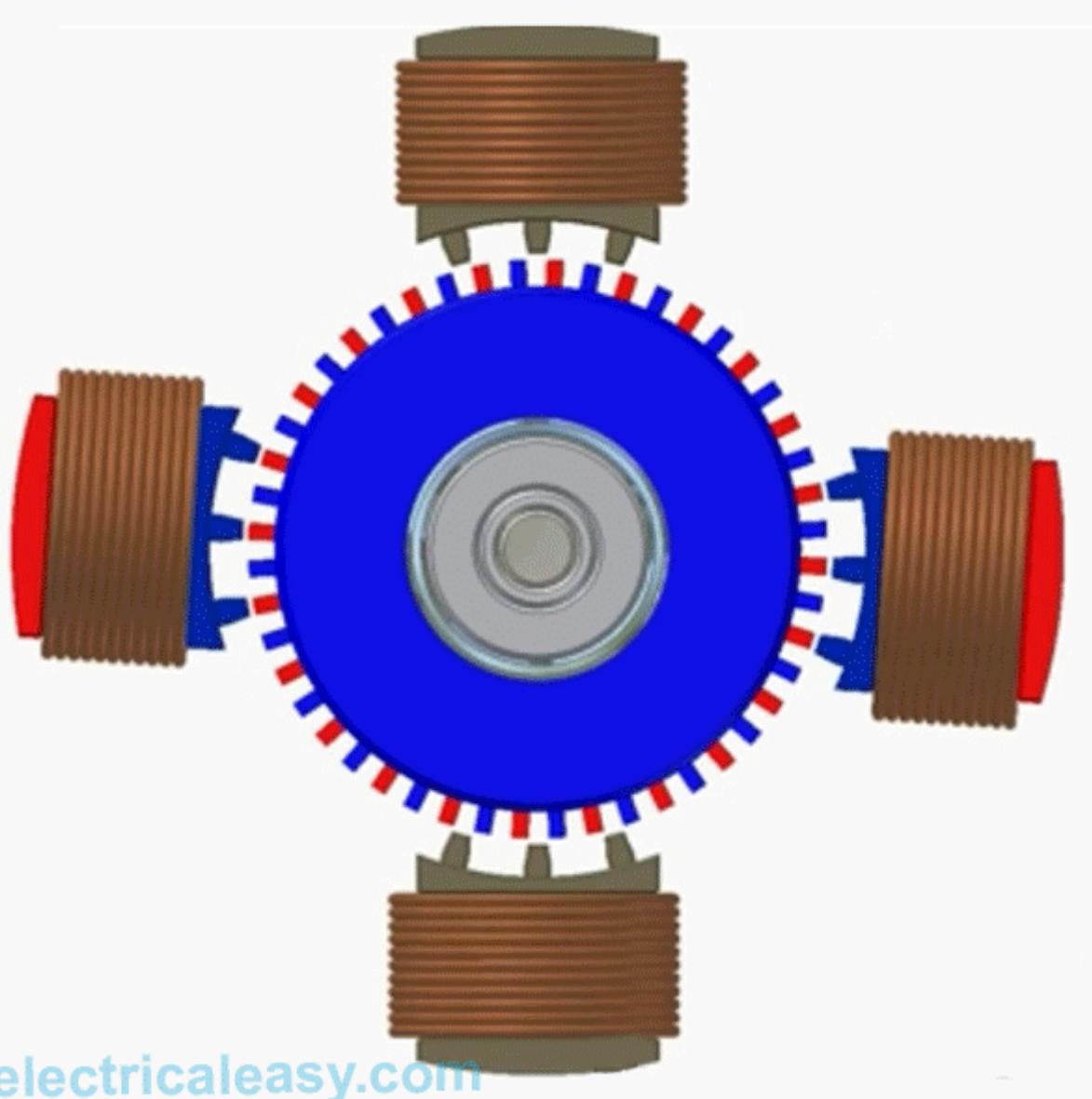
Elementos de un motor paso a paso:



En la construcción de un motor paso a paso el estator puede tener 4, 8, etc... polos y los imanes permanentes del rotor pueden tener múltiples polos. En función del número de ellos serán los pasos que conforman una vuelta del motor paso a paso.



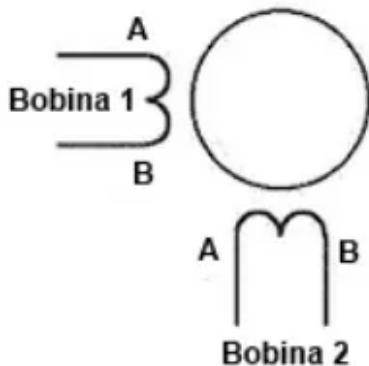
Funcionamiento:



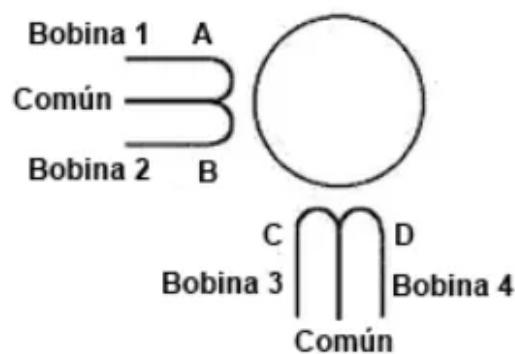
Los motores paso a paso se dividen en dos grandes grupos: bipolar y unipolar. Muy buena comparativa de motores paso a paso unipolares o bipolares: <https://www.330ohms.com/blogs/blog/85507012-motores-a-pasos-unipolares-o-bipolares>

- Conexión motores unipolares: <https://www.arduino.cc/en/Reference/StepperUnipolarCircuit>
- Conexión motores bipolares: <https://www.arduino.cc/en/Reference/StepperBipolarCircuit>

### Motor paso a paso bipolar



### Motor paso a paso unipolar



La mayoría de los motores que encontraremos son bipolares puesto que son más baratos, con mayor torque y más pequeños. La desventaja es que el control es más complicado.

Motores paso a paso de Pololu: <https://www.pololu.com/category/87/stepper-motors>

Más información sobre los motores paso a paso:

- [http://es.wikipedia.org/wiki/Motor\\_paso\\_a\\_paso](http://es.wikipedia.org/wiki/Motor_paso_a_paso)
- <http://www.productoscnc.es/hyperpages/unintroamotorespap.htm>
- <https://elchals.wordpress.com/2015/04/12/motores-paso-a-paso-todo-lo-que-deseabas-saber-y-no-te-cotaron-en-barrio-sesamo/>
- <http://www.prometec.net/motores-paso-a-paso/>
- <https://www.allaboutcircuits.com/textbook/alternating-current/chpt-13/stepper-motors/>
- <https://www.staticboards.es/blog/motores-paso-paso/>

### Motor Paso a Paso vs Servo

Los motores paso a paso son más lentos, su rotación es más precisa, es de fácil configuración y control, además mientras que los servos requieren un mecanismo de retroalimentación y circuitos de soporte para accionamiento de posicionamiento, un motor paso a paso tiene control de posición a través de su naturaleza de rotación por incrementos fraccionales.

Los motores paso a paso son adecuados para las impresoras 3D y dispositivos similares en los que la posición es fundamental.

La velocidad es uno de los criterios más importantes a tener en cuenta al elegir entre servo y motor paso a paso. Hay una relación inversa entre velocidad y par en los motores por pasos. Cuando la velocidad se incrementa, el par decrece. Los motores servo tienen un par constante hasta la velocidad nominal. Como criterio general para aplicaciones con velocidades bajas los motores paso a paso son una buena elección porque producen un par más alto que el servomotor de tamaño equivalente. Los servomotores tienen la capacidad de producir un par pico en cortos períodos de tiempo que es hasta 8 veces su par nominal.

continuo. Esto es particularmente útil cuando la resistencia del movimiento no es constante. Los motores por pasos no tienen esta capacidad.

En algunos casos que requieren alta velocidad, alta aceleración o aplicaciones críticas, el bucle cerrado de los servos es importante. Esto ocurre por ejemplo cuando se trabaja con accesorios caros en los que los fallos no son aceptables.

Guía para comprar motores paso a paso: <https://www.staticboards.es/blog/motores-paso-paso/>

Guía de selección de motores: <https://learn.adafruit.com/adafruit-motor-selection-guide/types-of-motors>

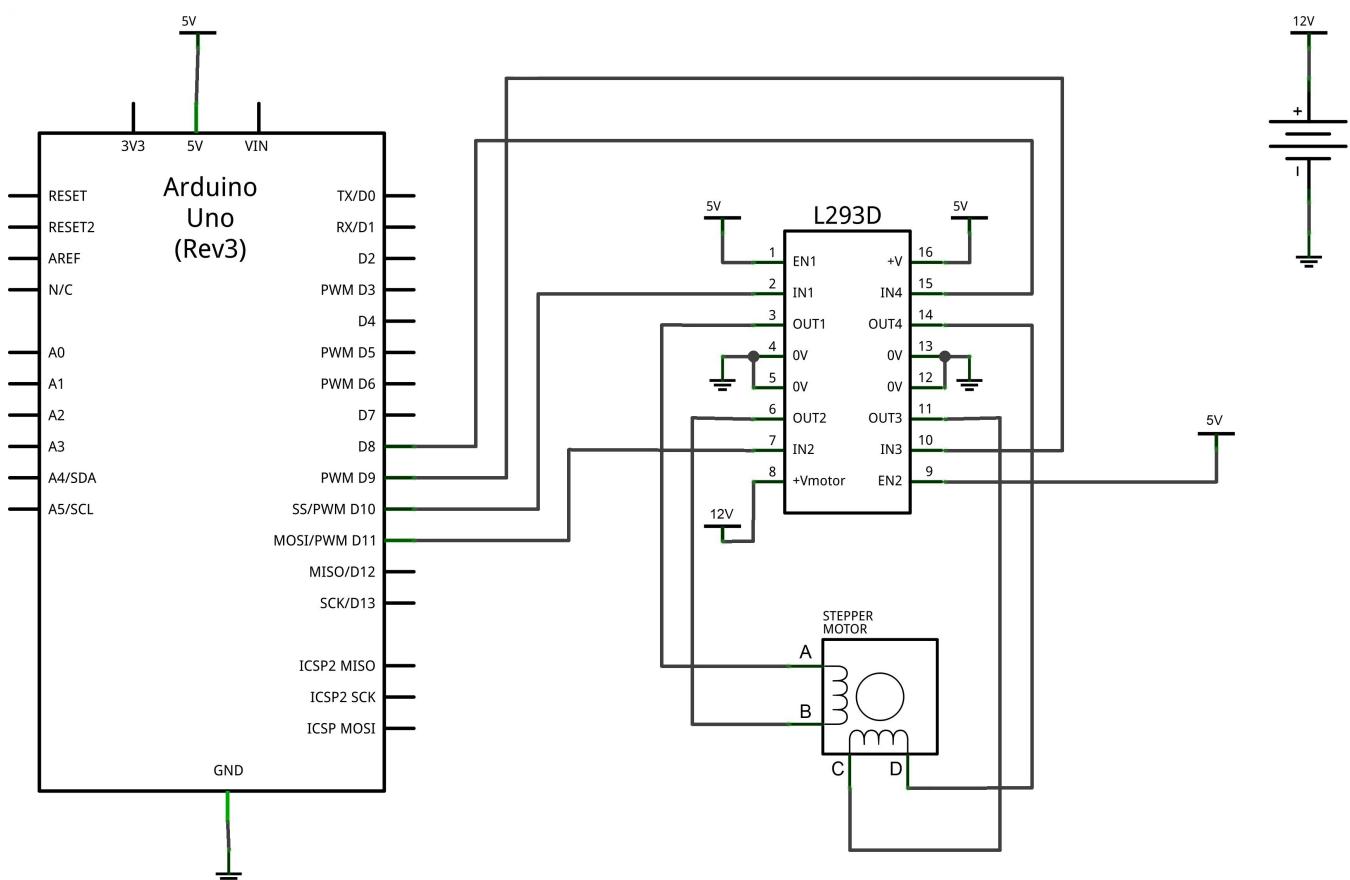
Web con muchos motores: <https://www.servocity.com/>

**NOTA:** En caso de que tengamos un motor DC y queramos hacer un control de posición, podemos añadirle un encoder y si queremos más par es posible añadir una reductora. Ejemplo de un motor CC con reductora y encoder como alternativa a steppers y servos:

[https://www.servocity.com/html/26\\_rpm\\_planetary\\_gearmotor\\_w\\_.html#.VpfFM\\_nhDct](https://www.servocity.com/html/26_rpm_planetary_gearmotor_w_.html#.VpfFM_nhDct)

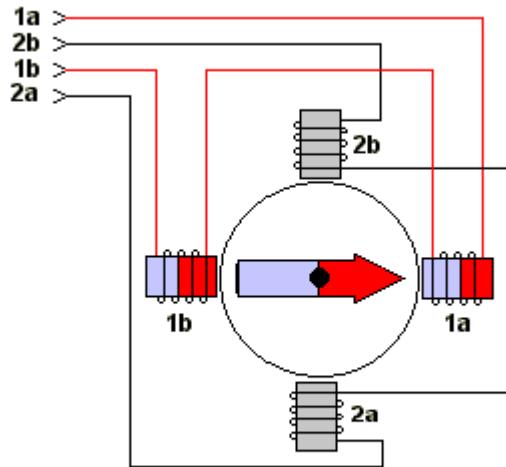
## Drivers Motores Paso a Paso

Los motores bipolares son más baratos pero más complejos de controlar ya que el flujo de corriente tiene que cambiar de dirección a través de las bobinas con una secuencia determinada. Para esto debemos conectar cada una de las dos bobinas en un puente en H (H-Bridge). Para esto, utilizaremos el integrado L293 que contiene dos H-Bridge ([datasheet](#)).



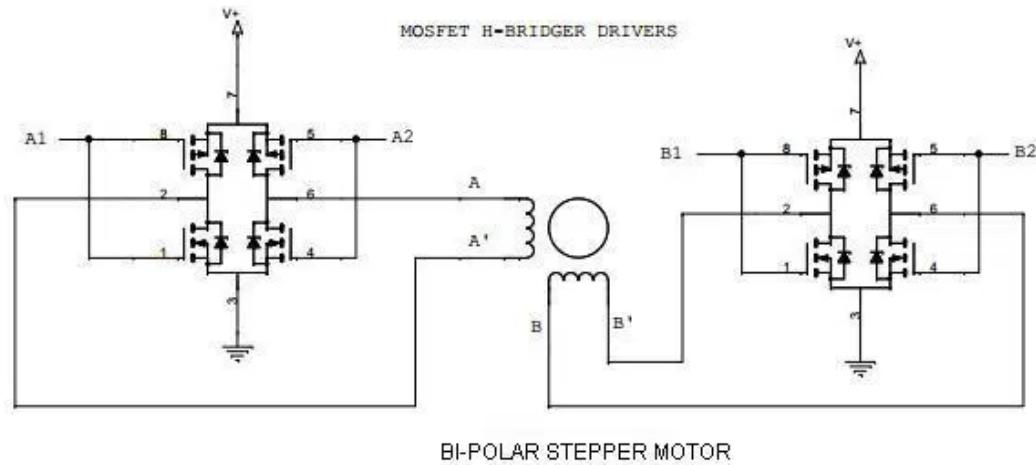
También es posible usar el L298 para controlar un motor paso a paso.

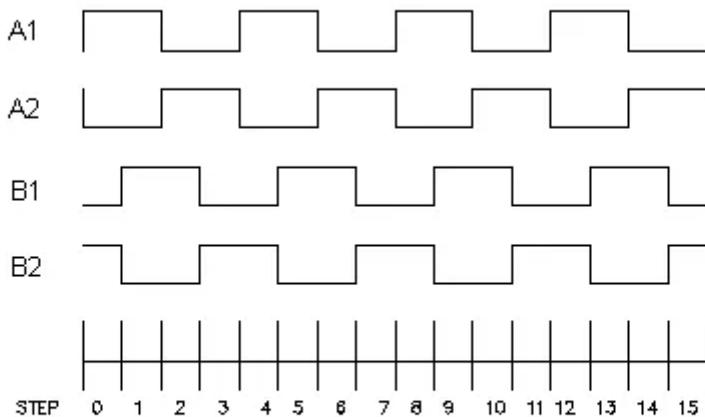
Motor bipolar:



Conceptual Model of Bipolar Stepper Motor

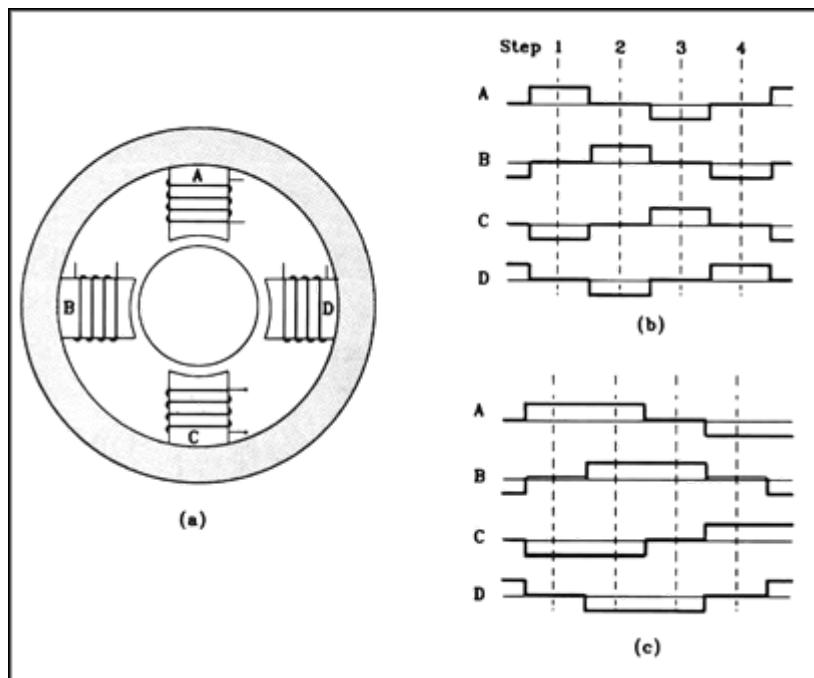
Señal que mandamos para controlar un paso a paso (4 pasos):





BI-POLAR STEPPER MOTOR DRIVER WAVEFORM

En pasos de 90 grados, como se magnetiza cada núcleo:



Más información: <http://www.hteck.ca/electronics/stepper-motor-drv/sm-driver.html>

Para controlar motores paso a paso con Arduino, utilizaremos la librería **Stepper.h**:

- <http://arduino.cc/en/Reference/Stepper/>
- <http://www.tigoe.net/pcomp/code/circuits/motors/stepper-motors/>

Con esta librería mandamos los pasos mediante programación en lugar de hacerlo con una señal del microcontrolador.

Step	wire 1	wire 2	wire 3	wire 4

1	High	low	high	low
2	low	high	high	low
3	low	high	low	high
4	high	low	low	high

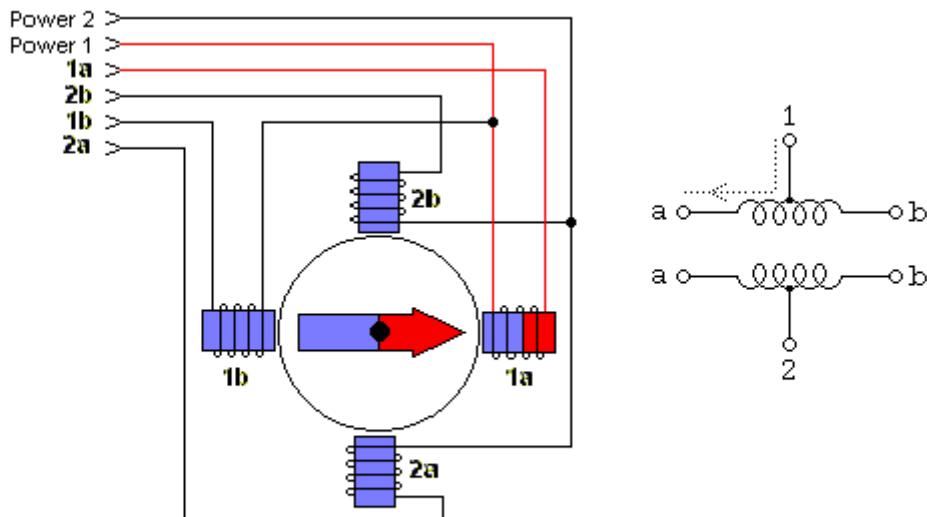
Más información y código en:

- <http://diymakers.es/mover-motores-paso-paso-con-arduino/>

Para los motores paso a paso podemos usar las mismas shields que para los motores DC:

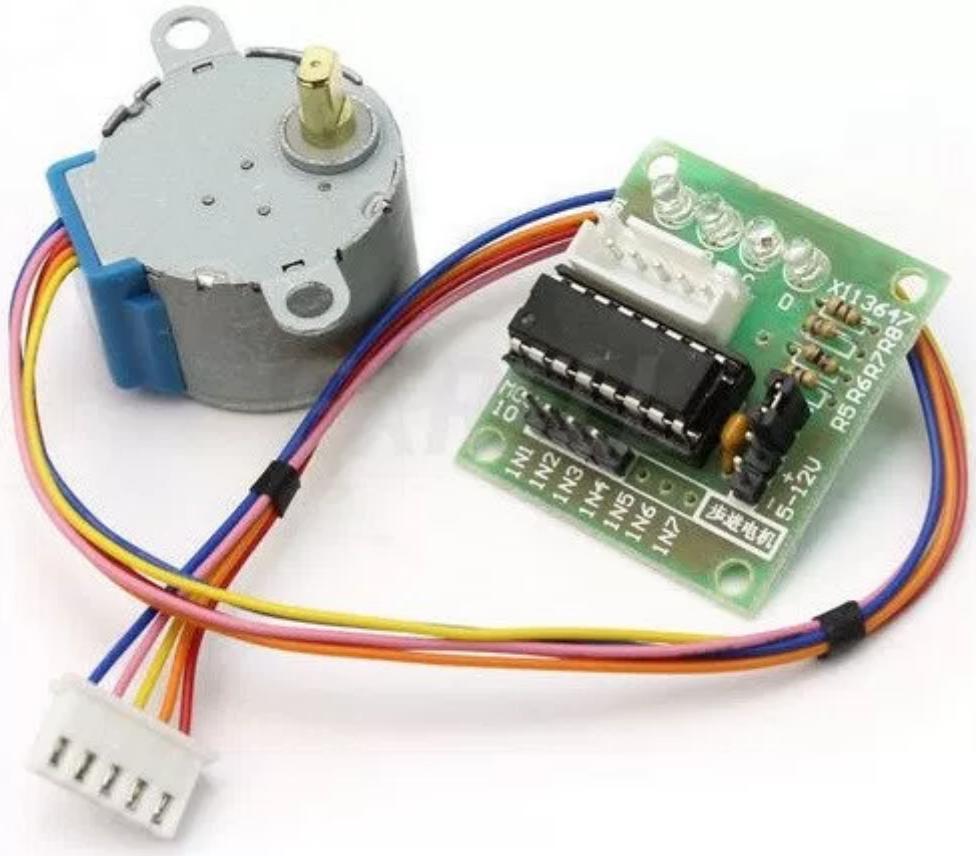
- <https://www.arduino.cc/en/Main/ArduinoMotorShieldR3>
- <https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino>
- <https://learn.adafruit.com/adafruit-motor-shield>

Para el caso de un **motor unipolar** es más sencillo el control y no es necesario un L293D sino que es suficiente con unos transistores o unos pares darlington.

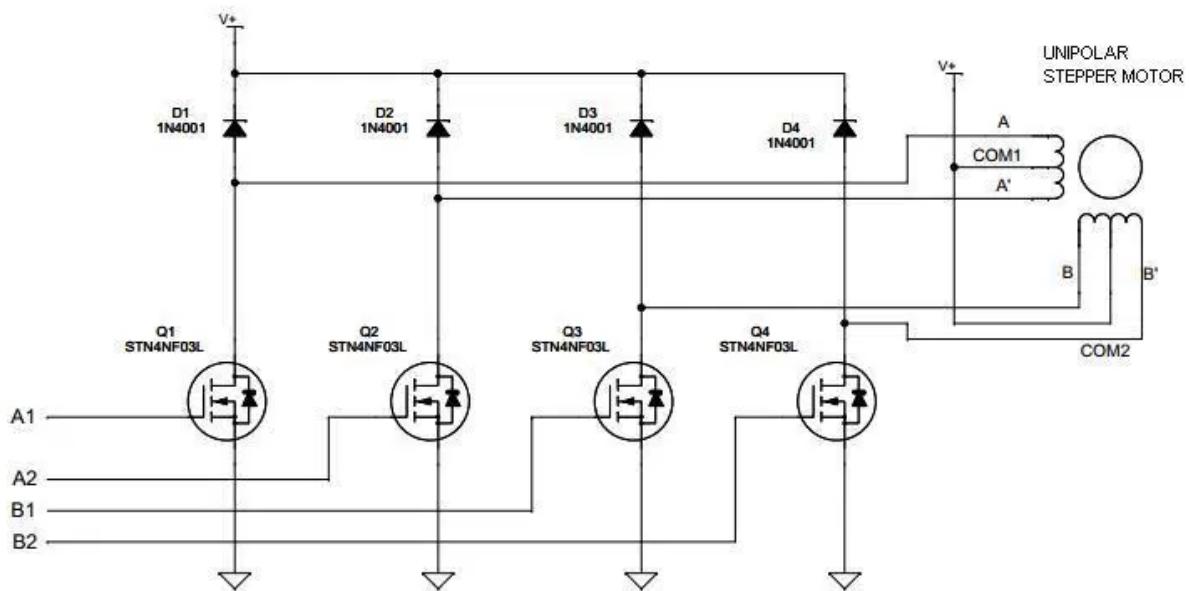


Conceptual Model of Unipolar Stepper Motor

Motor unipolar 28BYJ-48 controlado con un chip [ULN2003A](#) que es un array de transistores Darlington, que soporta hasta 500 mA.

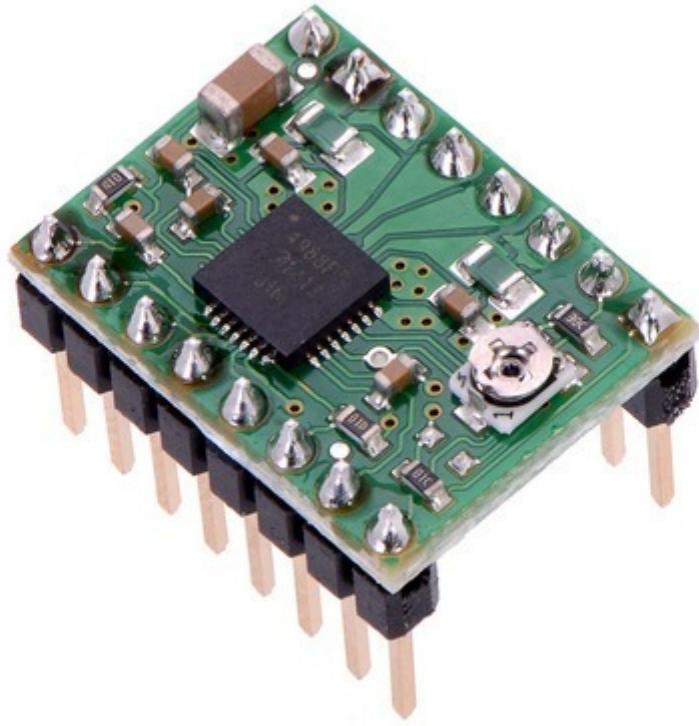


Esquema de conexión usando 4 pines del microcontrolador:



## Drivers Pololu

Los drivers de motores paso a paso más famosos son los Pololu, en concreto los A4988 y DRV8825.



Drivers de Pololu: <https://www.pololu.com/category/120/stepper-motor-drivers>, estos drivers son los más usado en impresoras 3D y máquinas CNC.

Drivers de un motor paso a paso:

- A4998: <https://www.pololu.com/product/1182>
  - Datasheet: <https://www.pololu.com/file/0J450/A4998.pdf>
- DRV8825 High Current: <https://www.pololu.com/product/2133>
  - Datasheet del driver: [https://www.pololu.com/file/download/drv8825.pdf?file\\_id=0J590](https://www.pololu.com/file/download/drv8825.pdf?file_id=0J590)
- DRV8834 <https://www.pololu.com/product/2134>

En el caso del A4998 cuenta con dos filas de 8 pines a cada lado. Por un lado tiene los pines para controlar la parte digital, y por el otro tiene la parte dedicada a la corriente de los motores. Incluye un potenciómetro para regular la corriente máxima que circula por las bobinas del motor.

En la parte digital tenemos los siguientes pines:

---

ENABLE	Permite que el driver pueda enviar corriente al motor
--------	---

---

MS1	Configuración de los micropasos
-----	---------------------------------

---

MS2	Configuración de los micropasos
-----	---------------------------------

---

MS3	Configuración de los micropasos
-----	---------------------------------

---

RESET	Normalmente conectado a Sleep para que el driver funcione
-------	---

---

SLEEP	Normalmente conectado a Reset para que el driver funcione
-------	---

---

STEP El arduino manda un pulso para que el motor avance un paso

DIR Indicamos la dirección de giro del motor que queremos

En la parte de potencia tenemos estos pines:

VMOT Alimentación de los motores (entre 8V y 35V)

GND Ground de la parte de motores

2B Bobina 2

2A Bobina 2

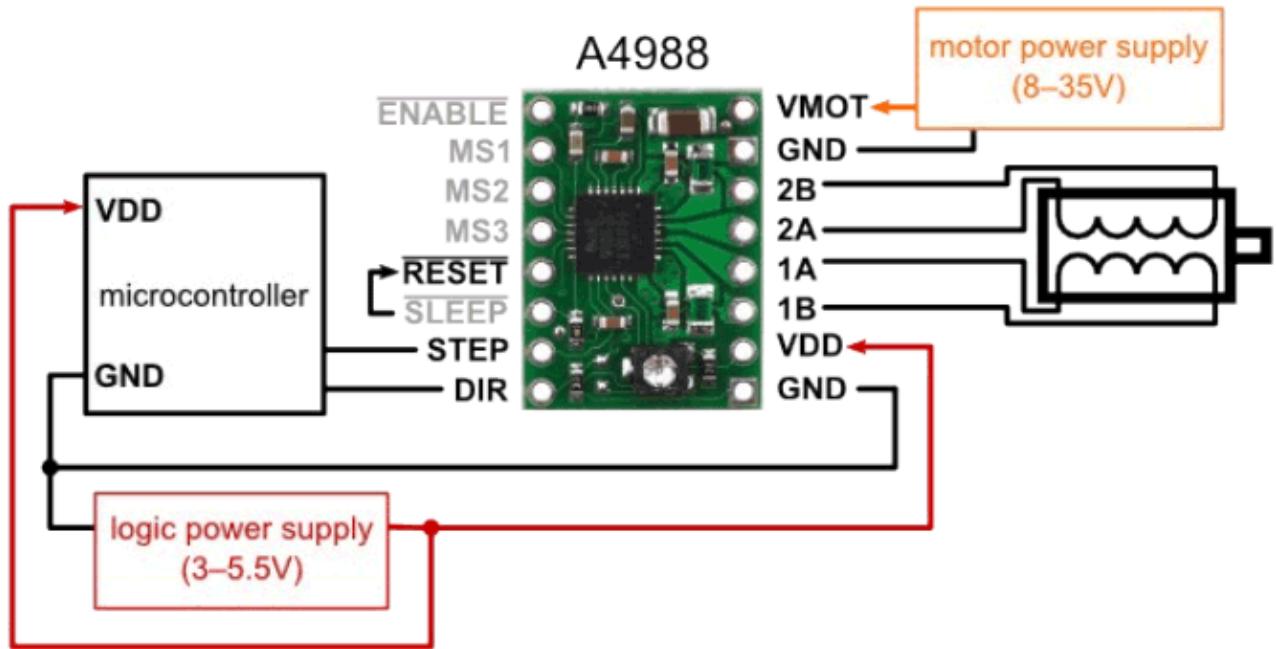
1A Bobina 1

1B Bobina 1

VDD Alimentación de la parte digital (5V)

GND Ground de la parte digital

Esquema de conexión A4998:



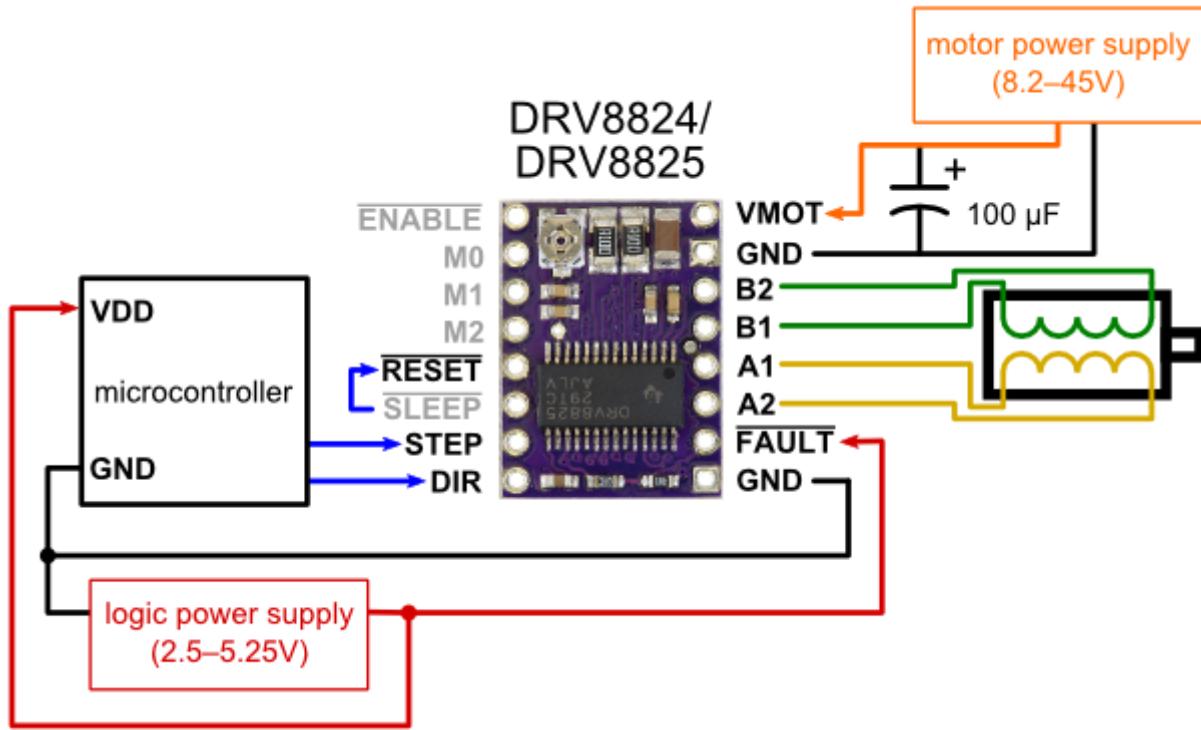
Los drivers basados en el Texas Instruments DRV8825 se usa en aplicaciones para más fuerza en los motores. Este es el chip más popular entre los usuarios más avanzados de CNC y de Impresoras 3D.

Fantastica guía de drivers <https://www.staticboards.es/blog/drv8825-vs-a4988/>

Tiene dos ventajas fundamentales sobre el chip de Allegro A4988.

- Es más potente. Permite una corriente de pico de 2.5A.
- Con la misma configuración de jumpers, permite 32 micropasos, que da un movimiento más fluido.

Esquema de conexión DRV8825:



Cuando configuramos una impresora 3D o una máquina CNC, tenemos que recordar los DRV8825 tiene 32 micropasos, y el A4988 tiene 16 micropasos. El DRV8825 funciona con 45V y el A4988 sólo 35V. El DRV8825 puede manejar 1.5A sin necesidad de un disipador adicional. El A4988 sólo llega a los 1.2A

Los tiempos de las señales son ligeramente distintos. Los DRV8825 necesitan un tiempo mínimo de pulso de 1.9µs y el A4988 de 1µs. Esto puede afectar a la velocidad máxima a la que puedes mover el driver, ya que tienes que esperar a que el driver detecte el pulso el doble de tiempo.

Estos drivers permiten pasar más o menos corriente a tu motor. No todos los motores están preparados para soportar la misma cantidad de corriente circulando por las bobinas. Si circula demasiada corriente los motores se van a sobrecalentar.

Unos drivers calibrados mantienen los motores a una temperatura adecuada. Para calibrarlos hay que comprobar que motor estamos usando y la corriente que admite, luego y con un polímetro ajustar el potenciómetro a la corriente adecuada.

**Con estos drivers el Arduino manda los pulsos al driver pero con el L293D o el L298N el pulso lo genera en la programación de Arduino con la librería Stepper.h.**

Disponemos de la librería <https://github.com/laurb9/StepperDriver> de Arduino para manejar motores paso a paso con estos drivers de Pololu. Tiene un código más depurado y puedes configurar la velocidad de motor en RPM y los microsteps.

Más información de estos drivers: <https://www.luisllamas.es/motores-paso-paso-arduino-driver-a4988-drv8825/>

Ejemplo de uso de DRV8825 con Arduino sin librería:

[http://www.geeetech.com/wiki/index.php/DRV8825\\_Motor\\_Driver\\_Board](http://www.geeetech.com/wiki/index.php/DRV8825_Motor_Driver_Board)

Tutoriales:

- <http://www.instructables.com/id/Drive-a-Stepper-Motor-with-an-Arduino-and-a-A4988/>
- <http://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/>

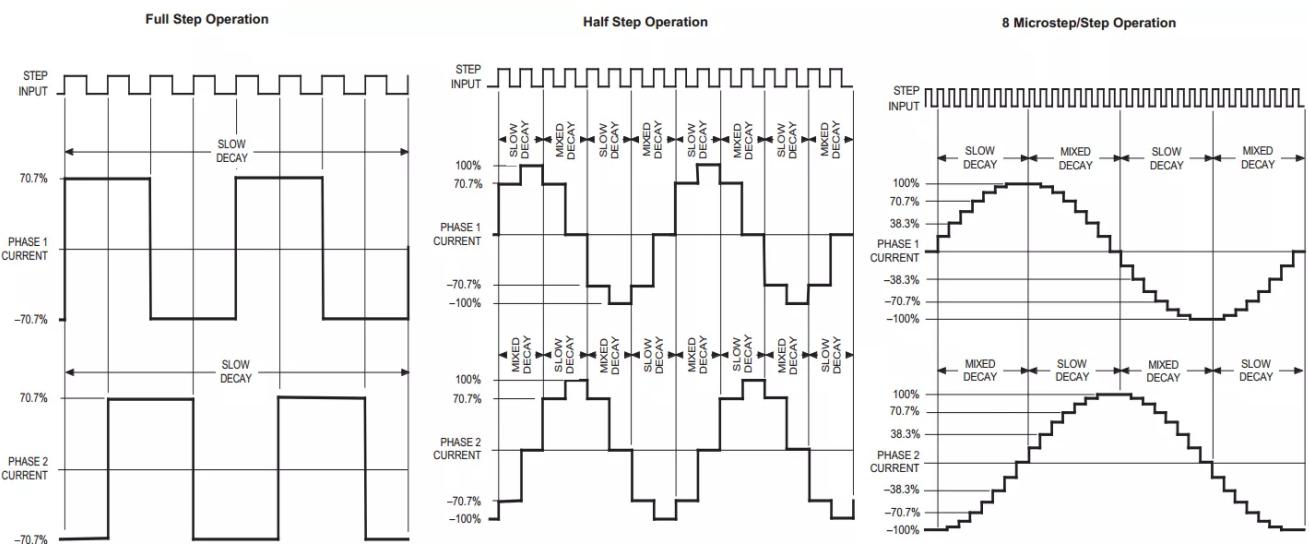
## Microstepping

La técnica de microstep permite multiplicar hasta por 32 el número de pasos de un motor paso a paso, haciendo que tenga un movimiento más suave. Consiste en no usar la corriente completa en cada bobina cada pulso, sino aplicar un porcentaje de la corriente simulando una onda senoidal que se aplica a las bobinas del motor.

Los motores paso a paso más habituales tienen 200 pasos por vuelta, por lo tanto cada paso es de 1,8°.

Cuando usamos los full steps la salida del driver al motor es una onda cuadrada y produce movimientos bruscos. Cuando mayor es el microstepping, la salida del driver más se parece a una onda senoidal y el motor se mueve de una forma más suave. Al aumentar los micro steps, el torque disminuye, de forma que si aumentamos mucho los pasos intermedios llega un punto que el motor no puede mover la carga. Normalmente microstepping de 1/4, 1/8 o 1/16 son adecuados para obtener un movimiento suave.

El microstepping nos dice cuantos micro steps necesitamos para hacer un paso completo, por ejemplo un microstepping de 1/4 significa que se dan 4 micropasos para hacer un paso completo de 1,8° y una vuelta son 800 micropasos.



Ver video <https://www.youtube.com/watch?v=bkqoKWP4Oy4> que explica como funciona un motor paso a paso y como controlarlo. A partir del minuto 5 habla de microstepping.

En el caso de los drivers de pololu los microsteps se configuran mediante 3 pines. En el caso del A4988 la tabla de configuración es:

**Table 1: Microstepping Resolution Truth Table**

<b>MS1</b>	<b>MS2</b>	<b>MS3</b>	<b>Microstep Resolution</b>	<b>Excitation Mode</b>
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

Más información sobre microstepping:

- <http://www.nmbtc.com/step-motors/engineering/full-half-and-microstepping/>
- <http://blog.poscope.com/stepper-motor-driver/>
- <http://www.zaber.com/microstepping-tutorial>

Esta entrada se publicó en Arduino, Curso Iniciación 2017, Motores y está etiquetada con Arduino, Curso Iniciación 2017, Motor DC, Motor Paso a Paso, Motores, Servo, Servomotor en 24 junio, 2017 [<https://aprendiendoarduino.wordpress.com/2017/06/24/motores-arduino/>].

---

## Proyectos Sencillos con Arduino

Hagamos unos proyectos sencillos con Arduino usando sensores, actuadores y comunicaciones.

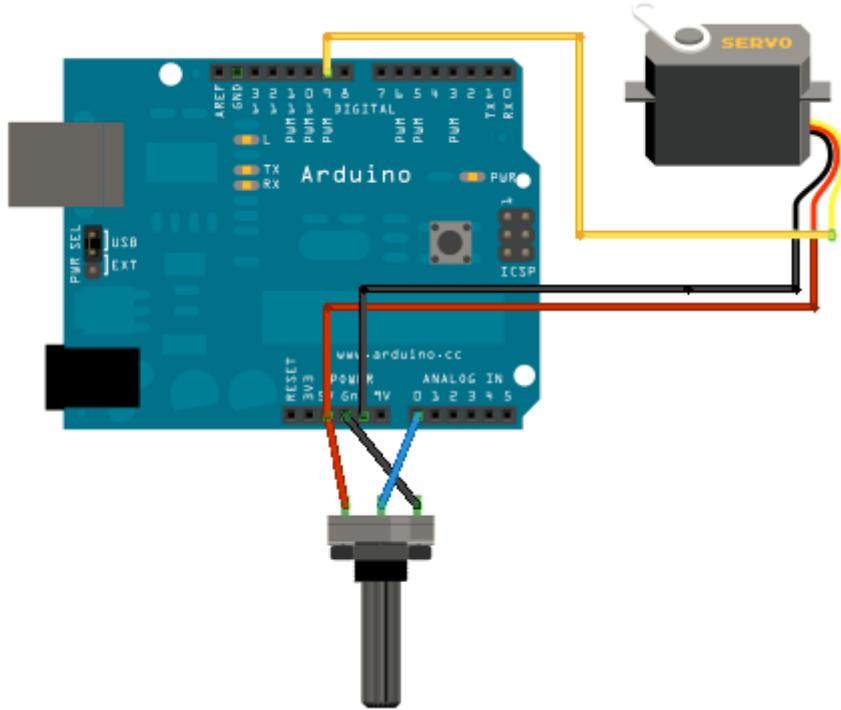
### Menú interactivo con Arduino

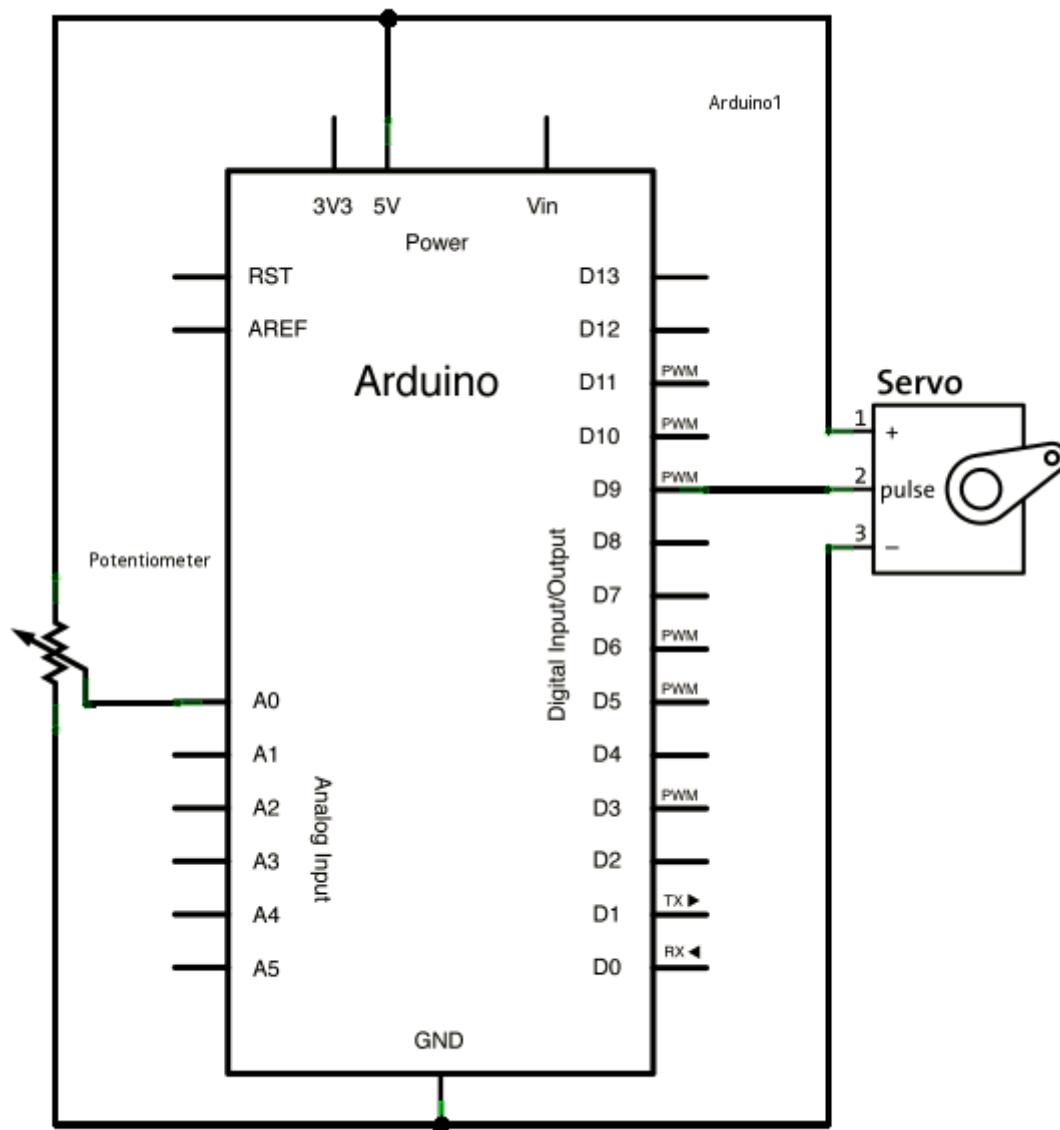
Con todo lo visto anteriormente de comunicación serie, operadores, estructuras de control y funciones, hacer un ejemplo de un menú interactivo donde se dan varias opciones y pulsando cada una de ellas se ejecuta una acción concreta. Si el valor pulsado no es ninguna de las opciones avisar y volver a mostrar el menú hasta que se pulse una opción correcta.

**Solución:** [https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio46-Estructuras\\_de\\_Control](https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio46-Estructuras_de_Control)

## Mover un Servo

Controlar la posición de un servo con un potenciómetro.





**Tutorial:** <http://arduino.cc/en/Tutorial/Knob>

**Solución:** <https://github.com/jcrespo/Aprendiendo-Arduino/tree/master/Ejercicio55-Servo/Knob>

## Mover un Servo con un Acelerómetro

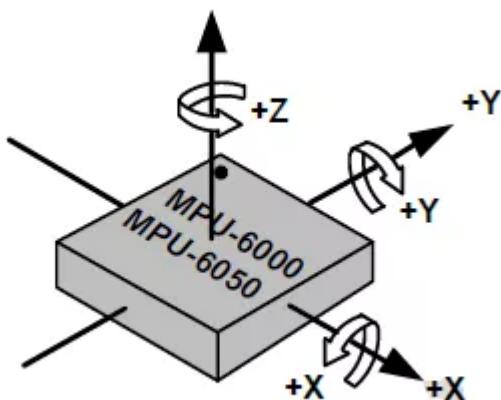
Una IMU (Inertial Measurement Unit) es un dispositivo capaz de medir la fuerza (aceleración) y la velocidad. Generalmente consta de un Acelerómetro y un Giroscopio. Por lo tanto una IMU no mide ángulos, por lo menos no directamente, requiere algunos cálculos.

Un dispositivo I2C muy interesante es el MPU-6050 que nos sirve para probar e introducirnos en el mundo de los giroscopios y acelerómetros.

- <https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/>
- Datasheet: [https://www.cdiweb.com/datasheets/invensense/MPU-6050\\_DataSheet\\_V3%204.pdf](https://www.cdiweb.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf)

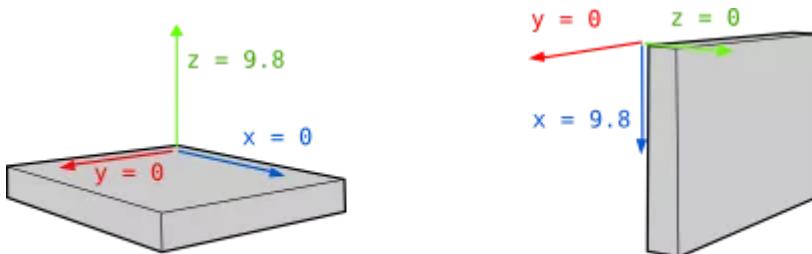
Para esta práctica vamos a utilizar una Breakout board bastante típica llamada GY-521, que incluye la IMU MPU-6050 y un regulador de tensión, con lo que podemos alimentar a tanto 3.3V como a 5V.

El siguiente diagrama muestra la orientación de los ejes de sensibilidad y la polaridad de rotación.



**El acelerómetro** mide la aceleración. La aceleración puede expresarse en 3 ejes: X, Y y Z, las tres dimensiones del espacio. Por ejemplo, si mueves la IMU hacia arriba, el eje Z marcará un cierto valor. Si es hacia delante, marcará el eje X, etc. La gravedad de la Tierra tiene una aceleración de aprox. 9.8 m/s<sup>2</sup>, perpendicular al suelo como es lógico. Así pues, la IMU también detecta la aceleración de la gravedad terrestre. Gracias a la gravedad terrestre se pueden usar las lecturas del acelerómetro para saber cuál es el ángulo de inclinación respecto al eje X o eje Y.

Supongamos que la IMU esté perfectamente alineada con el suelo. Entonces, como puedes ver en la imagen, el eje Z marcará 9.8, y los otros dos ejes marcarán 0. Ahora supongamos que giramos la IMU 90 grados. Ahora es el eje X el que está perpendicular al suelo, por lo tanto marcará la aceleración de la gravedad.

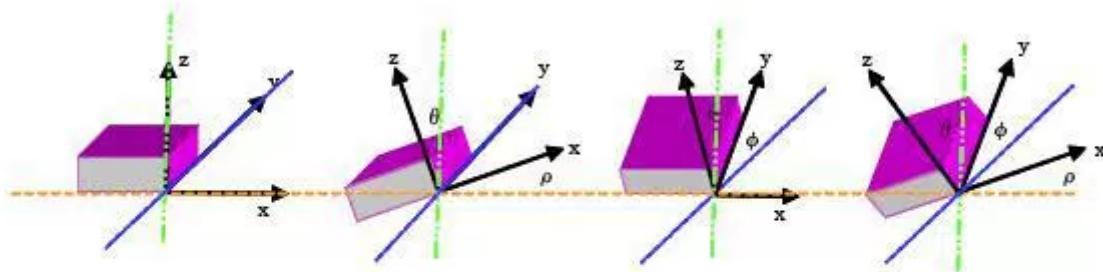


Paralelo al suelo

Girado 90º

Si sabemos que la gravedad es 9.8 m/s<sup>2</sup>, y sabemos qué medida dan los tres ejes del acelerómetro, por trigonometría es posible calcular el ángulo de inclinación de la IMU. Una buena fórmula para calcular el ángulo es:

$$\text{AnguloY} = \text{atan} \left( \frac{x}{\sqrt{y^2 + z^2}} \right)$$



**Figure 8. Three Axis for Measuring Tilt**

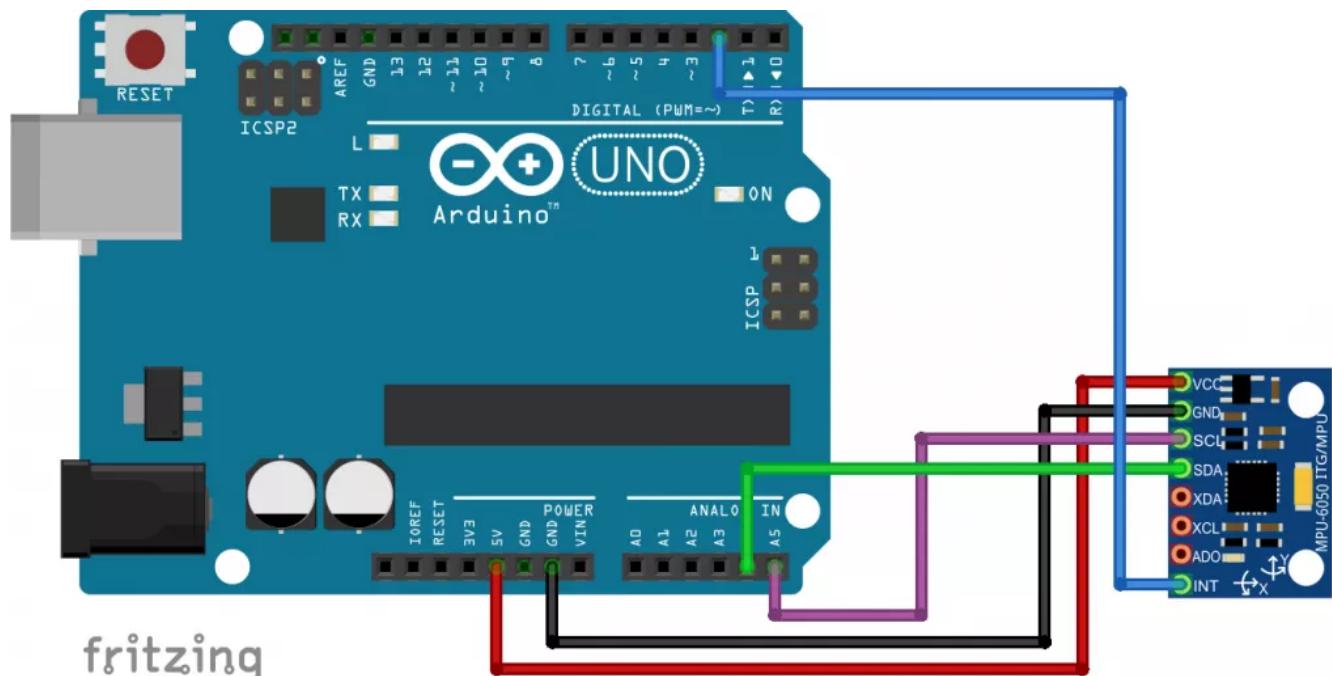
$$\rho = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right)$$

$$\phi = \arctan\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right)$$

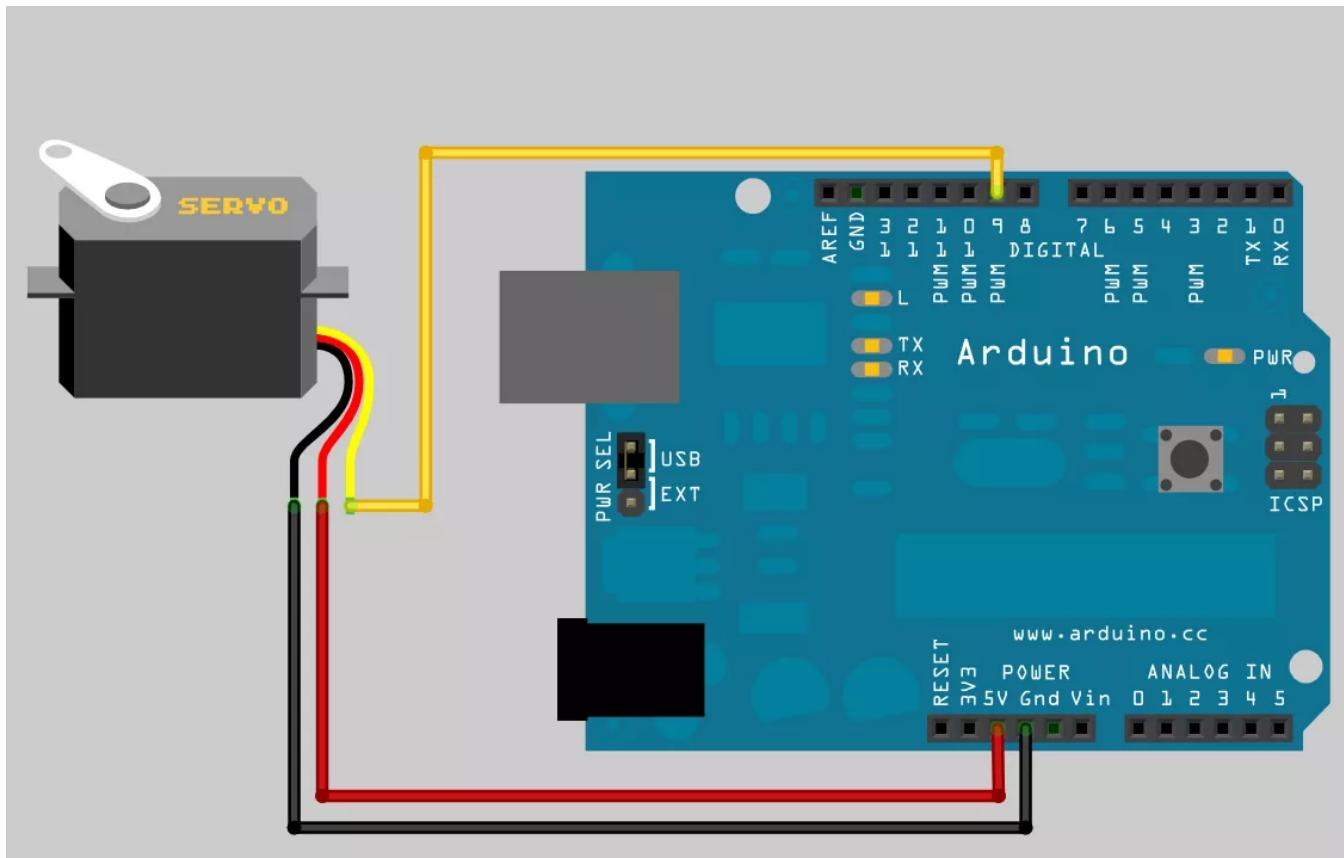
$$\theta = \arctan\left(\frac{\sqrt{A_x^2 + A_y^2}}{A_z}\right)$$

Dado que el ángulo se calcula a partir de la gravedad, no es posible calcular el ángulo Z (giro sobre si mismo) con esta fórmula ni con ninguna otra. Para hacerlo se necesita otro componente: el magnetómetro, que es un tipo de brújula digital. El MPU-6050 no lleva, y por tanto nunca podrá calcular con precisión el ángulo Z. Sin embargo, para la gran mayoría de aplicaciones sólo se necesitan los ejes X e Y.

Esquema de conexión IMU:



## Esquema conexión servo:



Mover el servo en función del ángulo en el eje x obtenido de la IMU.

**Solución:** [https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio66-Servo\\_IMU/IMU\\_1servo](https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio66-Servo_IMU/IMU_1servo)

Hacer el mismo ejemplo pero con un sistema de dos grados de libertad con dos servos y moviéndose en función de los grados obtenidos del IMU en los ejes x e y.



**Solución:** [https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio66-Servo\\_IMU/IMU\\_2servos](https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio66-Servo_IMU/IMU_2servos)

## Proyectos con Piezas Impresas 3D

Ahora disponemos de las herramientas para hacer un proyecto completo, imprimiendo las piezas en 3D, montando un arduino y los sensores y actuadores en las piezas.

Robot impreso en 3D: <http://otto.strikingly.com/>



Tutoriales de construcción:

- <http://www.instructables.com/id/Otto-Build-You-Own-Robot-in-Two-Hours/?ALLSTEPS>
- <https://www.hackster.io/otto/otto-build-your-own-robot-in-two-hours-5f2a1c>



Código: <https://github.com/OttoDIY>

Brazo robot para montar con unos servos y tornillos:

- <https://www.thingiverse.com/thing:34829>
- <https://www.thingiverse.com/thing:1454048>

Más proyectos impresos 3D con Arduino:

- <http://www.makeuseof.com/tag/5-coolest-3d-printed-arduino-projects/>
- <http://www.instructables.com/tag/type-id/category-technology/channel-3D-Printing/keyword-arduino/>
- <http://diy3dprinting.blogspot.com.es/2016/01/esp8266-controlled-kame-3d-printed.html>
- <http://www.lamja.com/?p=451>

Esta entrada se publicó en Arduino, Motores, Proyecto y está etiquetada con Arduino, Impresion 3D, IMU,

Proyectos, Servo en 21 diciembre, 2016

[<https://aprendiendoarduino.wordpress.com/2016/12/21/proyectos-sencillos-con-arduino/>] .

---

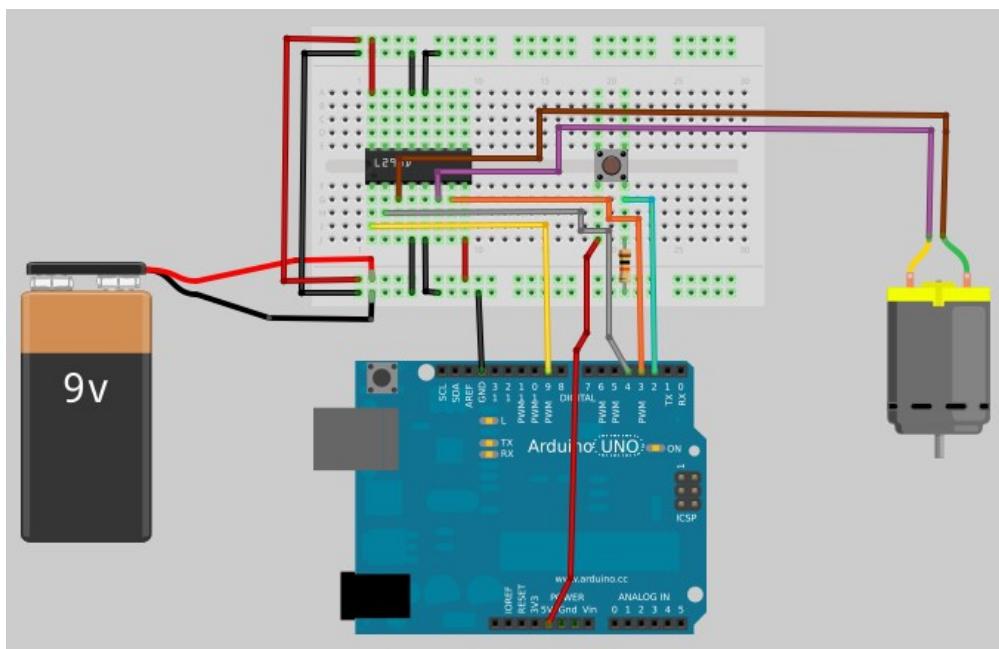
## Uso de Motores

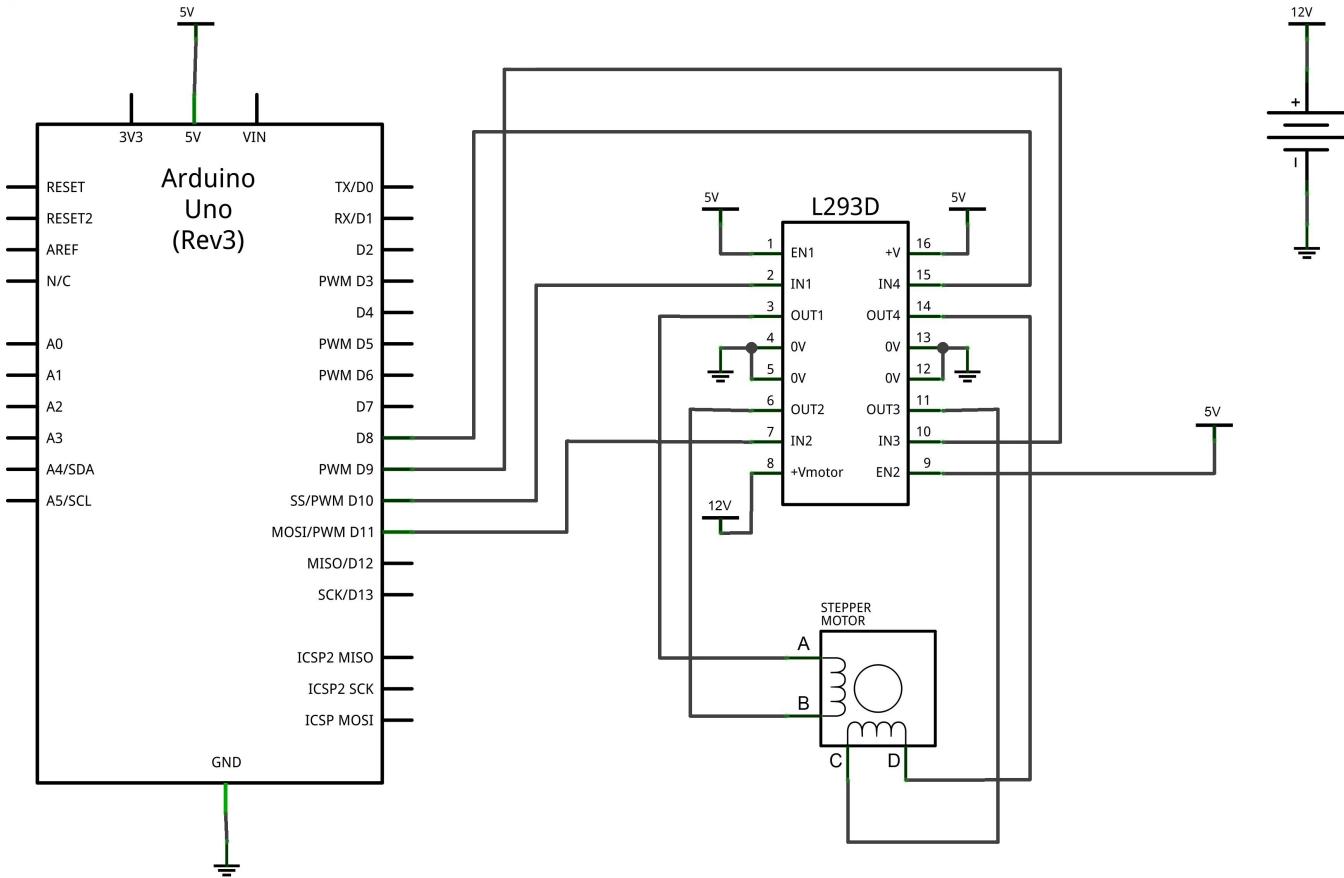
## Motores DC

**Ejercicio.** Mover un motor DC de 9V usando un integrado L293D (Quadruple Half-H driver). Para controlar la velocidad del motor se usará un potenciómetro conectado al pin A0. Además se usarán dos botones, uno conectado al pin digital 4 para controlar el sentido de giro del motor y otro conectado al pin digital 5 que controlará el encendido y apagado del motor. Con cada pulsación encendemos y apagamos el motor o usamos una dirección de giro u otra con el otro botón.

Datasheet: [https://www.arduino.cc/documents/datasheets/H-bridge\\_motor\\_driver.PDF](https://www.arduino.cc/documents/datasheets/H-bridge_motor_driver.PDF)

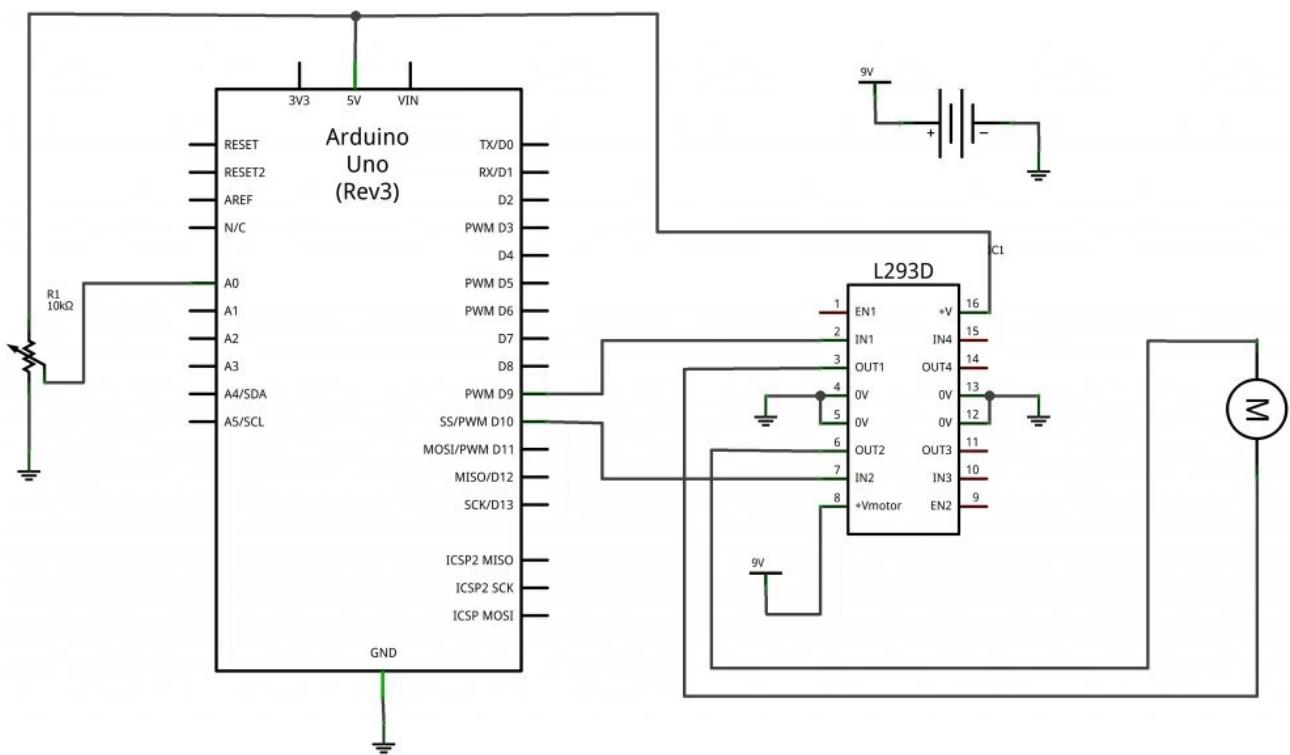
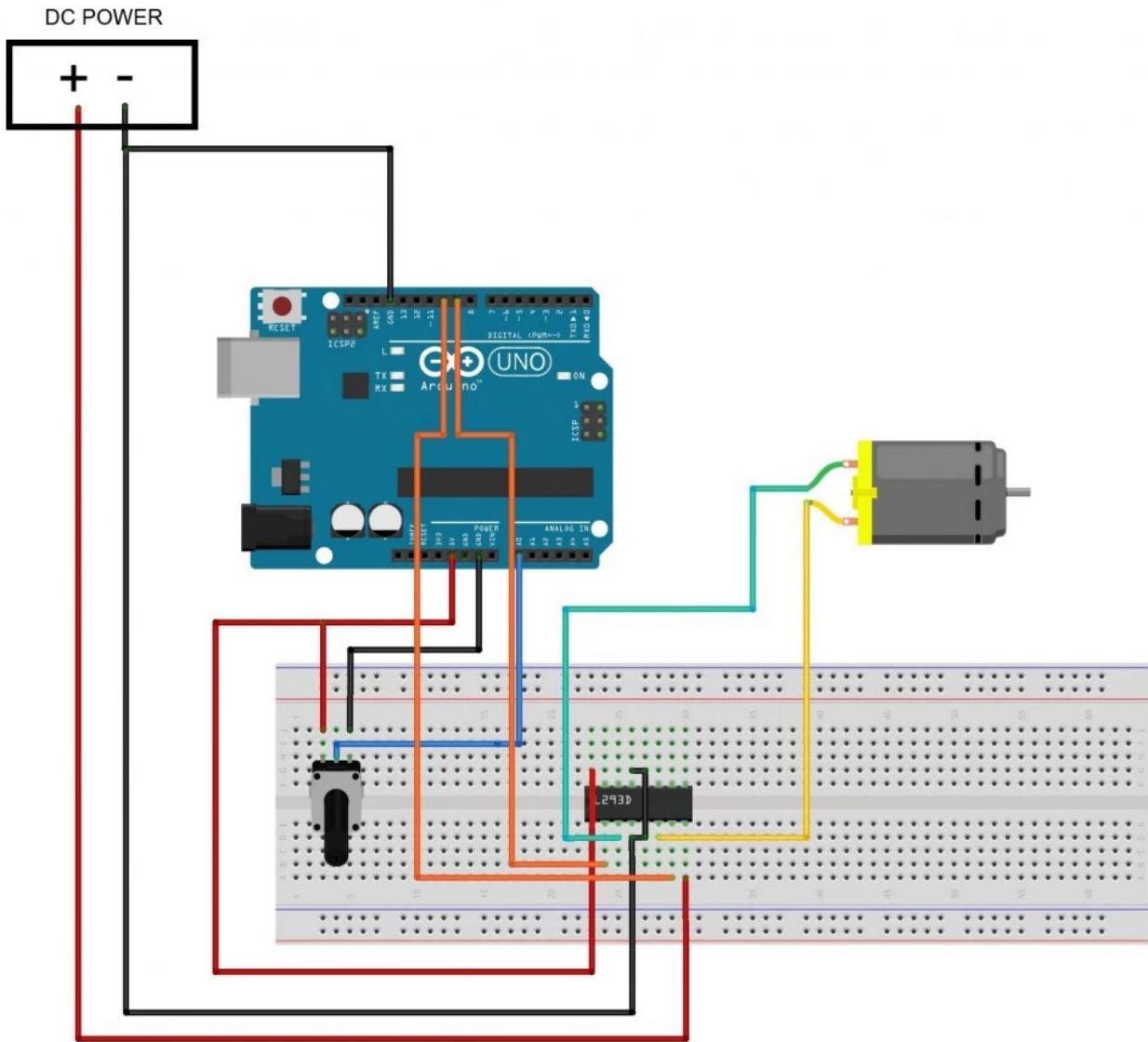
Montaje:





**Solución:** [https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio21-MotorDC\\_1](https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio21-MotorDC_1)

**Ejercicio Avanzado.** Añadir rampa de aceleración/deceleración para arranque, parada y cambio de sentido para que pase por cero con una compilación de “seguridad” y sin ella. Añadir una parada de emergencia que tenga preferencia sobre todo y pare de golpe al pulsar un botón de emergencia (seta de emergencia), ver manejo de prioridades.



**Solución:** [https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio21-MotorDC\\_2](https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio21-MotorDC_2)

**Ejercicio Motor Shield:** Uso del Arduino Motor Shield para controlar la dirección de giro un motor DC. En este caso es necesario alimentar Arduino con una fuente de alimentación o mediante una batería, puesto que con la energía de USB no es posible mover el motor. También es posible alimentar Arduino por USB y alimentar independientemente el shield a través de las bornas marcadas con + y -. El voltaje debe ser el correspondiente al que use el motor DC.

**Tutorial:** <https://www.arduino.cc/en/Tutorial/DueMotorShieldDC>

**Solución:** [https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio47-Motor\\_Shield](https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio47-Motor_Shield)

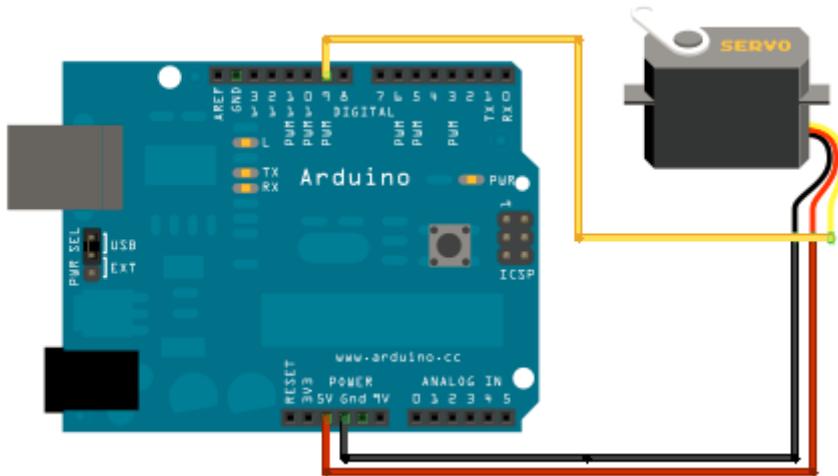
## Servo

**Ejercicio:** Controlar la posición de un servo con un potenciómetro.

**Solución:** <http://arduino.cc/en/Tutorial/Knob>

**Ejercicio:** Programar un barrido continuo del 0 a 180º en un servo. Activar y desactivar el barrido con una pulsación de un botón. p.e. activación de un limpiaparabrisas.

**Solución:** <http://arduino.cc/en/Tutorial/Sweep>



Esta entrada se publicó en Arduino, Motores, Práctica y está etiquetada con Arduino, Motor DC, Motores, Práctica, Servo, Servomotor en 4 julio, 2016 [https://aprendiendoarduino.wordpress.com/2016/07/04/uso-de-motores/].

# Motores

Como ya se ha visto anteriormente, un pin de Arduino solo puede tener los valores de 0 y 5 voltios y dar hasta 40 mA. Esto es insuficiente para mover un motor del tipo que sea, por lo tanto si queremos que Arduino maneje un motor, deberemos usar un driver.

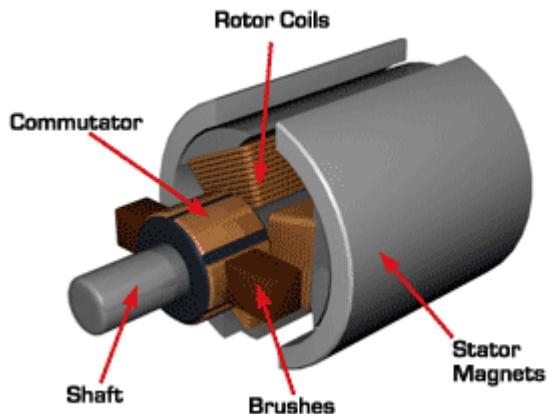
Un motor driver es un amplificador de corriente cuya función es tomar una pequeña señal de control de baja corriente y convertirla en una señal de alta corriente que pueda alimentar el motor.

Hay muchos tipos de motor drivers en función del motor a manejar, máximo voltaje, máxima corriente de salida, etc...

Más información: <http://www.futureelectronics.com/en/drivers/motor-driver.aspx>

## **Motor DC**

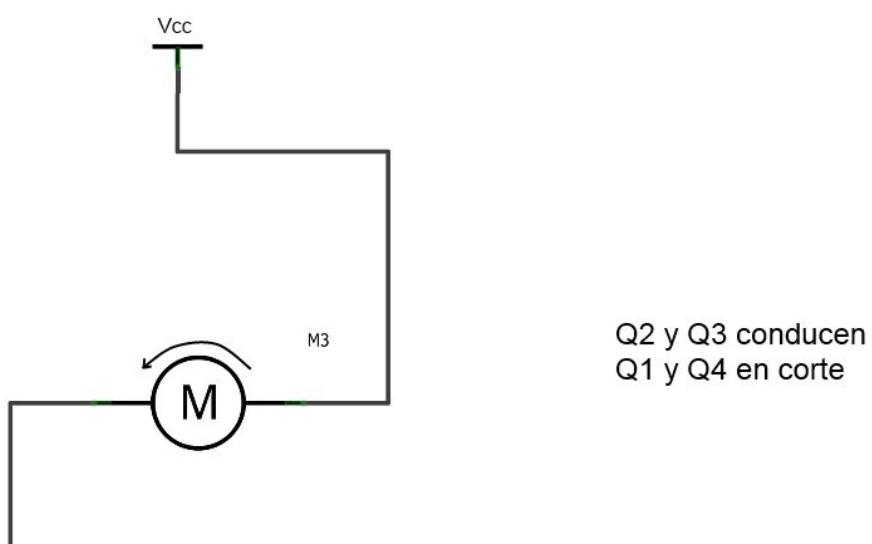
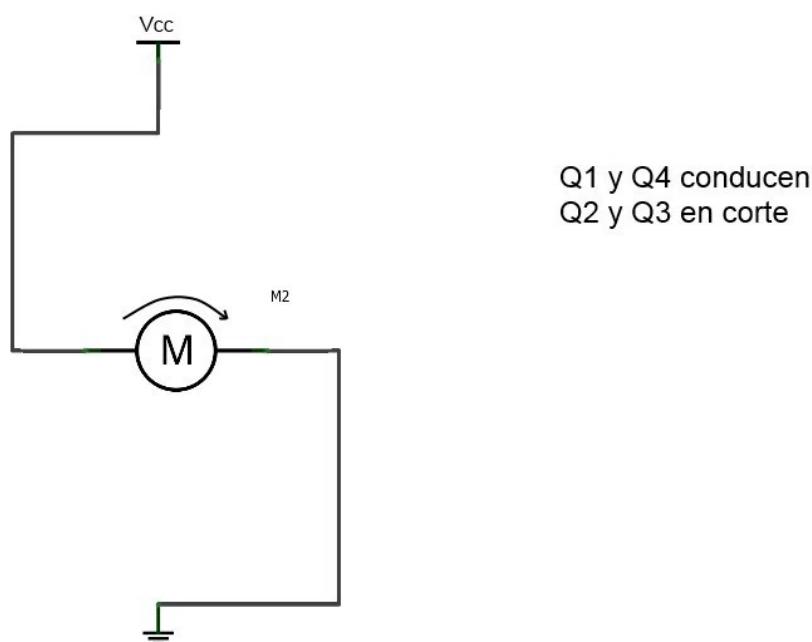
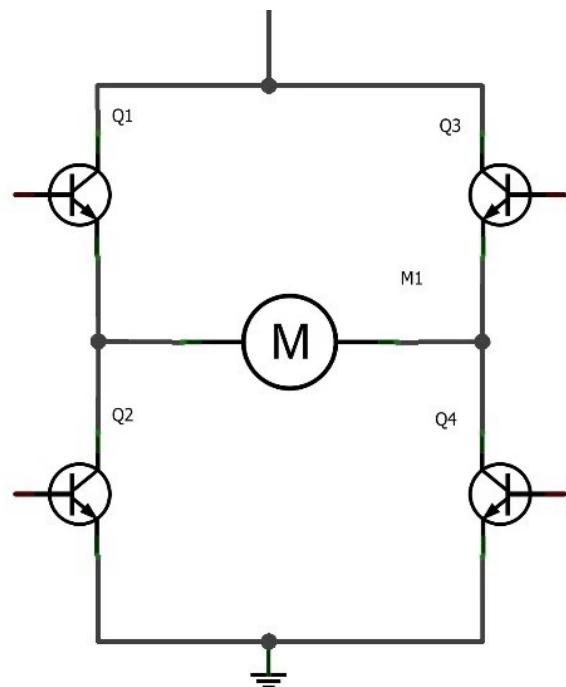
Un motor de corriente continua convierte la energía eléctrica en mecánica. Se compone de dos partes: el estator y el rotor. El estator es la parte mecánica del motor donde están los polos del imán. El rotor es la parte móvil del motor con devanado y un núcleo, al que llega la corriente a través de las escobillas. Si queremos cambiar el sentido de giro del rotor, tenemos que cambiar el sentido de la corriente que le proporcionamos al rotor, basta con invertir la polaridad de la pila o batería.

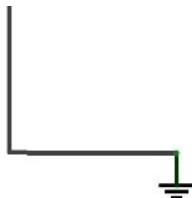


Para controlar un motor DC desde Arduino, tendremos que usar un driver para motores para proporcionar más corriente al motor ya que las salidas del Arduino sólo dan hasta 40mA. Con el driver podemos alimentar el motor con una fuente de alimentación externa.

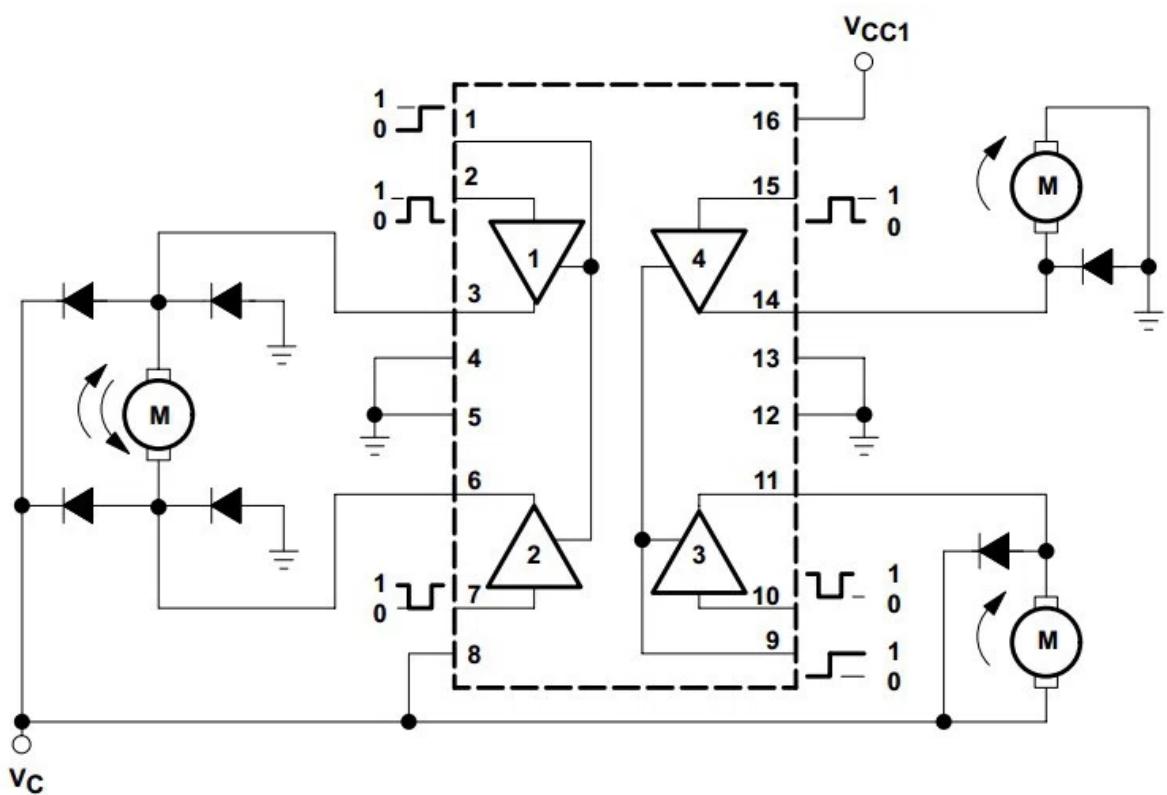
Más información: <http://www.prometec.net/motorcc/>

El L293D es un integrado para controlar motores DC que usa el sistema puente en H. Es un sistema para controlar el sentido de giro de un motor DC usando cuatro transistores y también la velocidad del motor. En la imagen vemos que los transistores se comportan como interruptores y dependiendo que transistores conducen y cuáles no cambia la polarización del motor, y con esto el sentido de giro.





El L293D tiene dos puentes H y proporciona 600mA al motor y soporta un voltaje entre 4,5V y 36V tal y como pone en el datasheet: <http://www.ti.com/lit/ds/symlink/l293d.pdf>

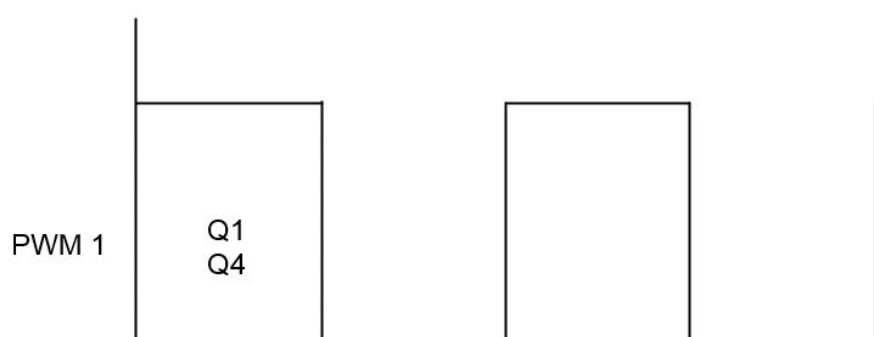
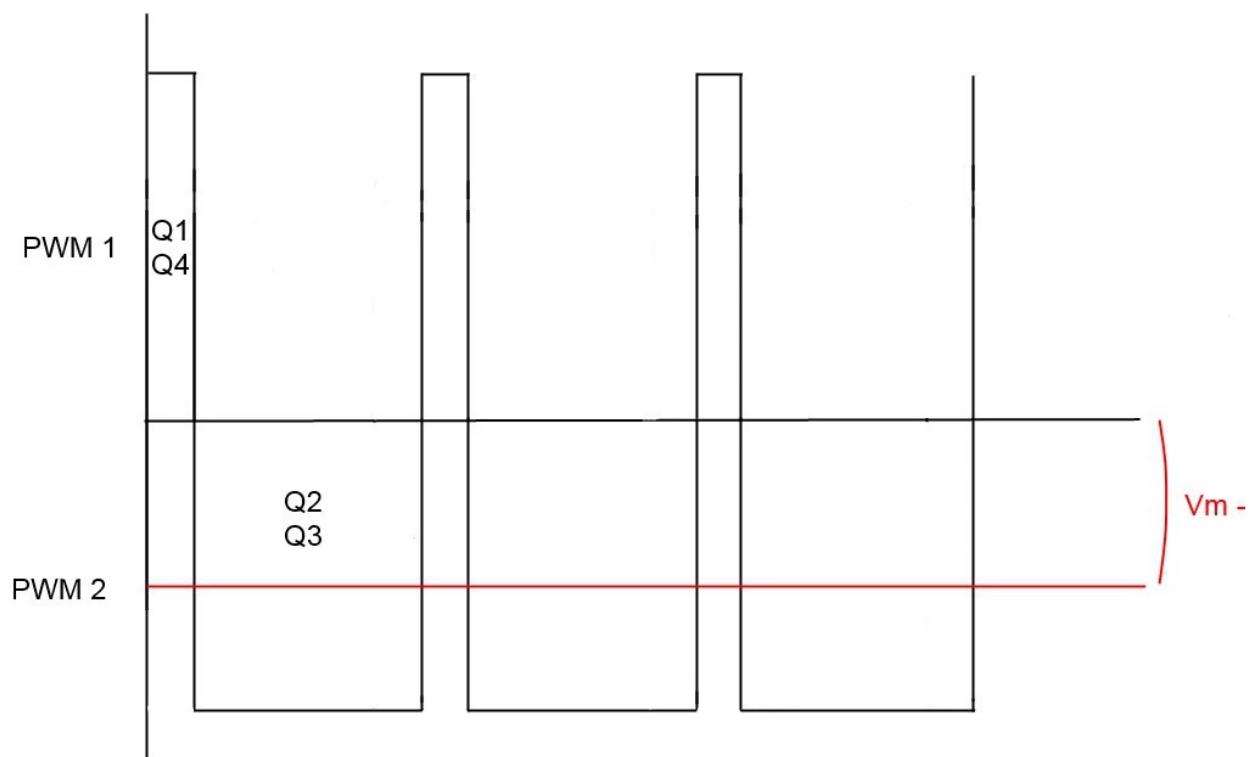
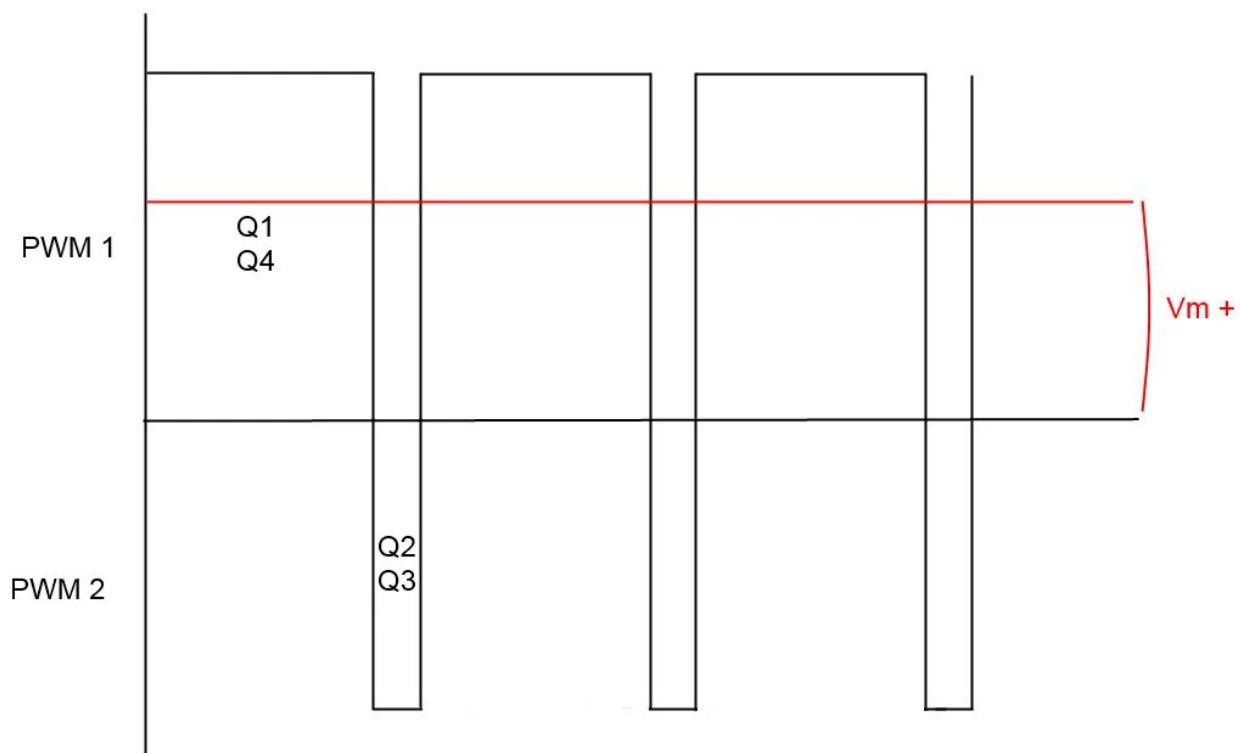


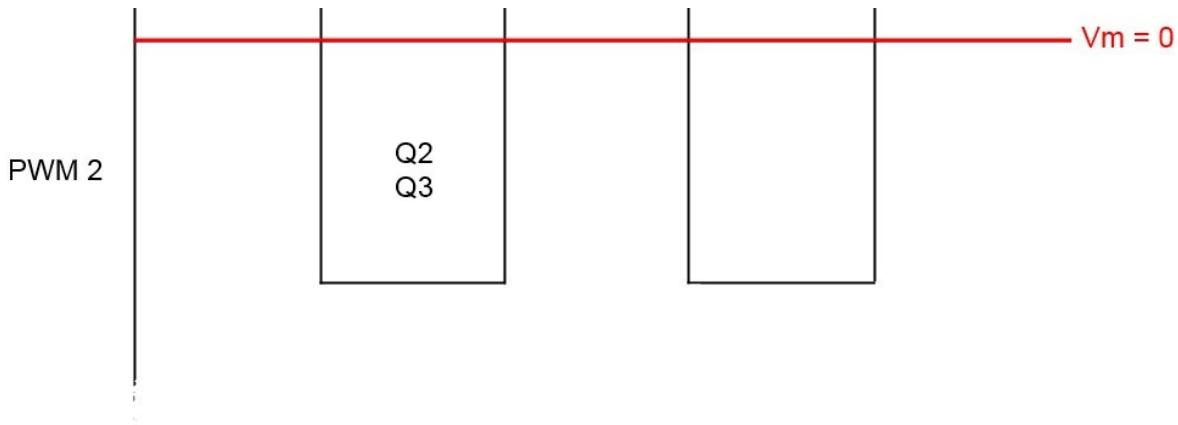
Nosotros usaremos la parte de la izquierda (los diodos externos hay que ponerlos para evitar las corrientes inducidas del motor si usamos el modelo L293 y están incluidos en el integrado si usamos el L293D). Como se aprecia en la imagen, los pins 3 y 6 son las salidas y se conectan a los bornes del motor. Y los pins 2 y 7 son las entradas donde conectaremos las salidas del Arduino. Dependiendo qué valor ponemos entre los pines 2 y 7 el motor girará en un sentido o en otro.

Es muy IMPORTANTE que se utiliza el driver L293, hay que poner diodos para evitar dañar el integrado con las corrientes parásitas generadas por los propios solenoides de las cargas. No obstante el modelo L293D no los necesita, ya que, los lleva incorporados el propio integrado, lo que se hace "más sencillo" y "económico" su uso. También es cierto que L293 al no llevarlos integrados nos permite escoger los que mejor se adapten a nuestras cargas o necesidades.

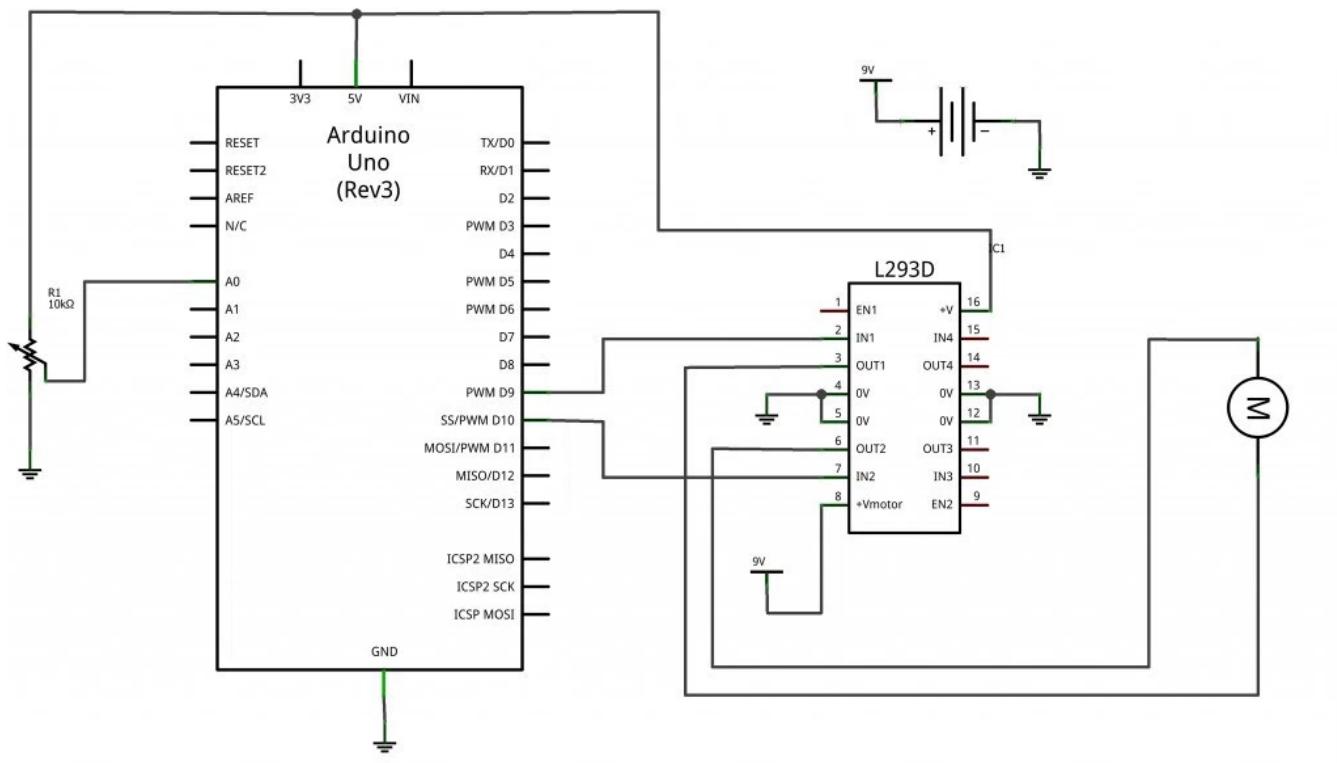
Para controlar la velocidad del motor se usa la técnica de PWM. Sabemos que hay que atacar los pins 2 y 7 del L293D desde dos salidas del Arduino. En estas dos salidas habrá un PWM a cada una. Pero tenemos que invertir un PWM. ¿Qué quiere decir invertir? Pues que cuando en un PWM tengamos un pulso a un

valor alto, en el otro PWM el mismo pulso sea valor bajo. En la imagen lo entenderemos de una manera más gráfica.





Montaje:



Más información y código en: <http://diymakers.es/control-velocidad-y-sentido-de-motor-dc/>

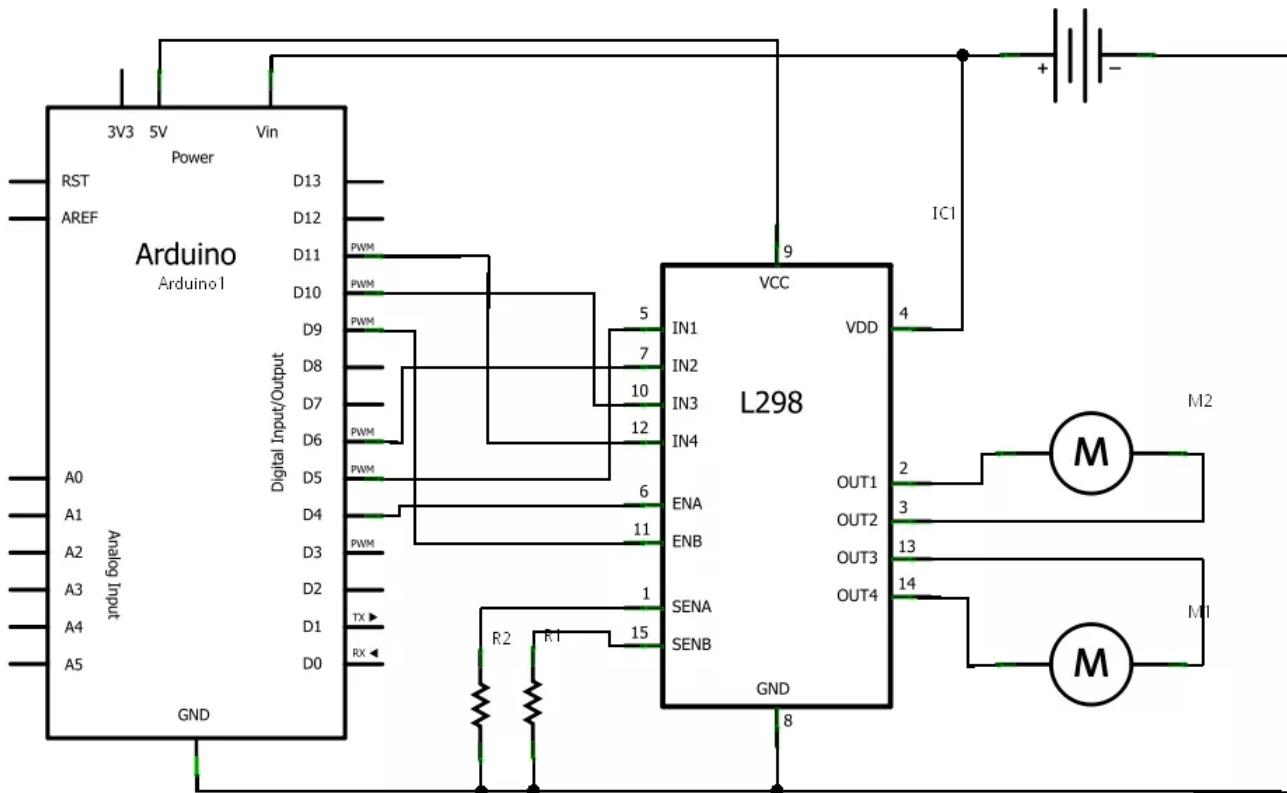
Tutorial para controlar un motor con Arduino: <http://www.allaboutcircuits.com/projects/use-an-arduino-to-control-a-motor/>

Ejemplo del playground de Arduino:

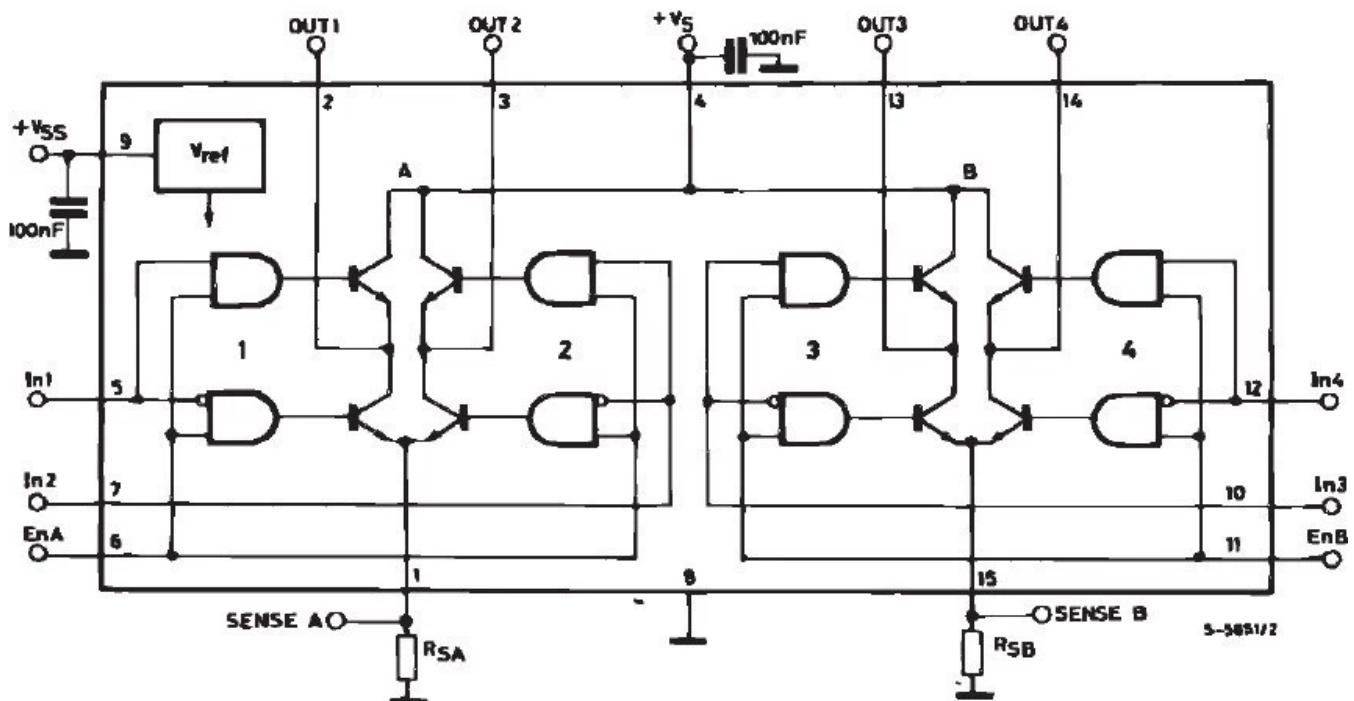
<http://playground.arduino.cc/Main/DirectionalMotorControlWithL293D>

Otro driver de motor muy utilizado es el L298 (datasheet) que es el utilizado por el motor shield. Este driver es similar en funcionamiento al anterior pero posee un sensor de corriente muy útil.

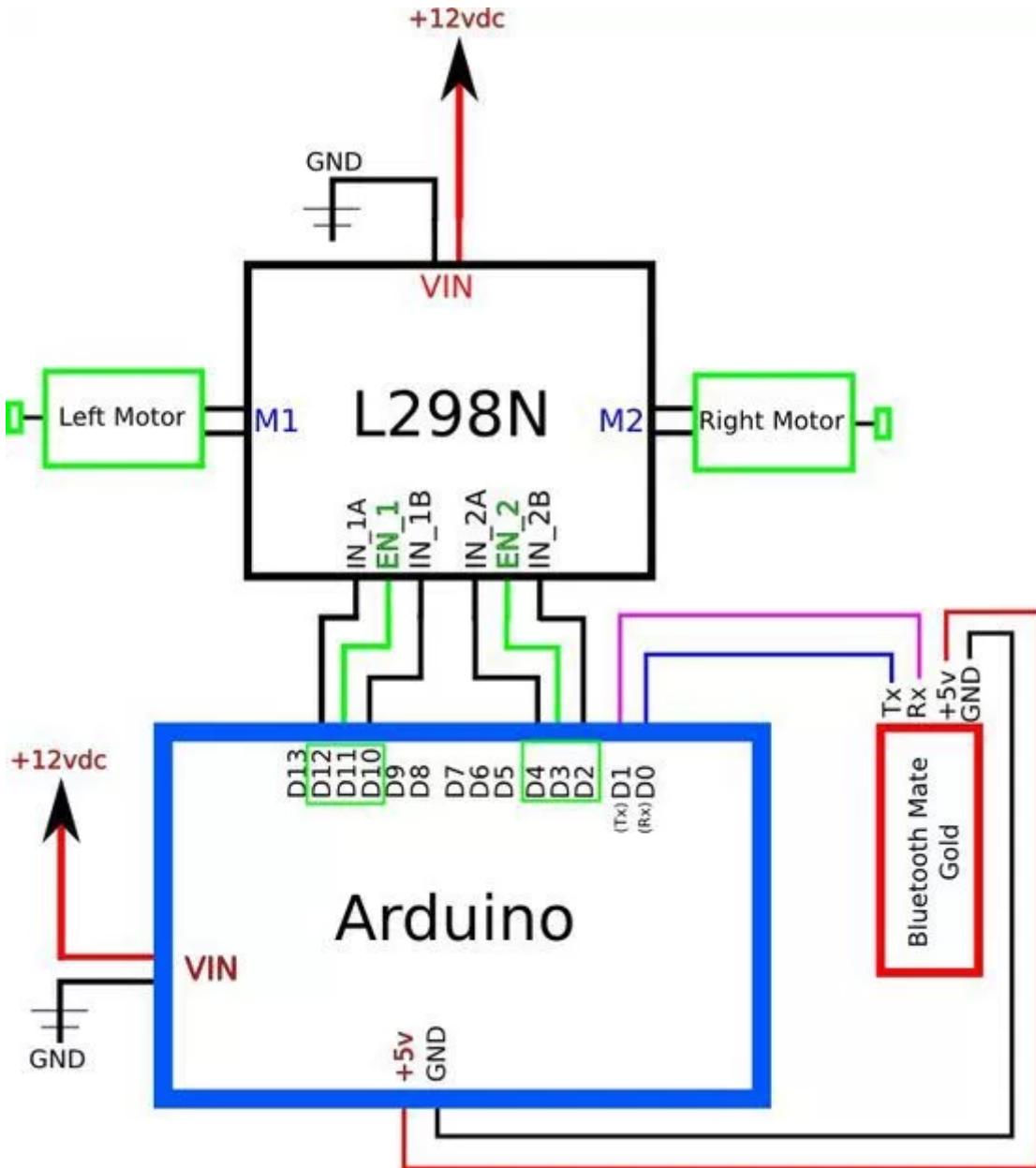
Montaje



Diagrama



Esquema de conexión

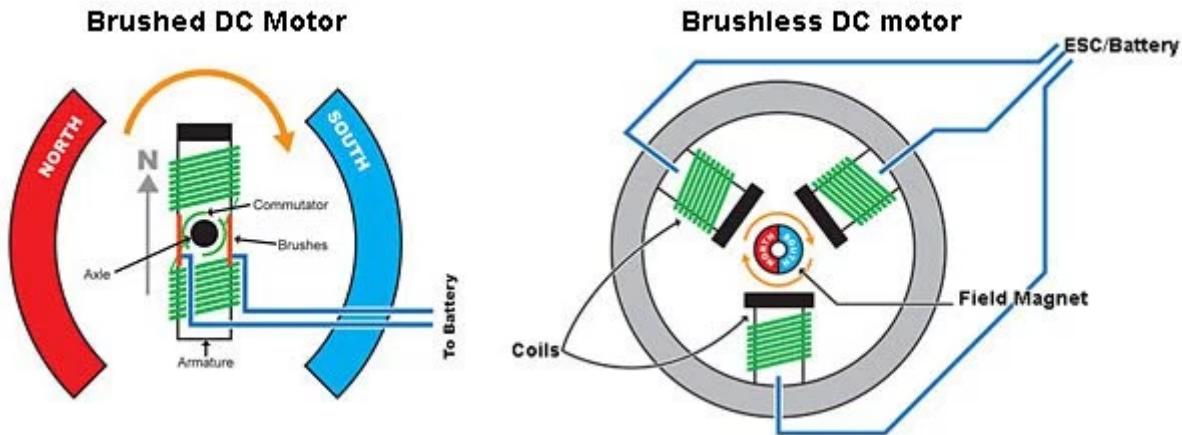


Diferencias entre el L293 y el L298:

- L293 es un quadruple half-H driver y el L298 es un dual full-H driver
- El L293 al tener 4 canales permite manejar motores paso a paso de 4 hilos y el L298 solo tiene dos canales perfecto para manejar dos motores DC. En el L293 las cuatro líneas de entrada/salida son independientes.
- La corriente de salida por canal en el L293 es de 1A, mientras que en el L298 es de 2A, por ese motivo el L298 puede llevar un disipador.
- Los diodos protectores del L293D deben ponerse externamente en el L293 y L298

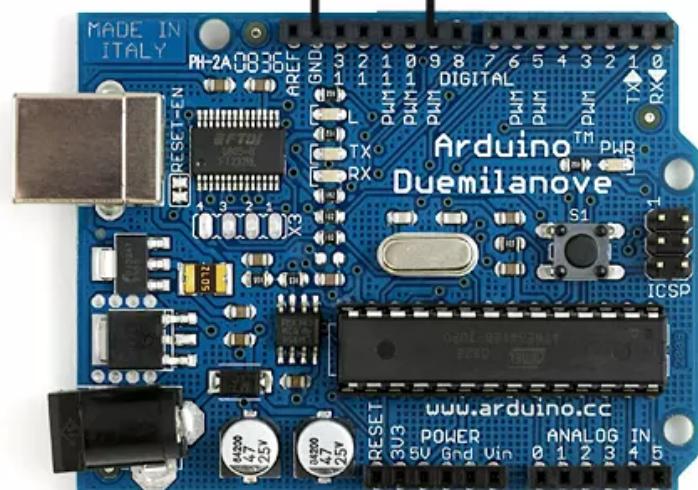
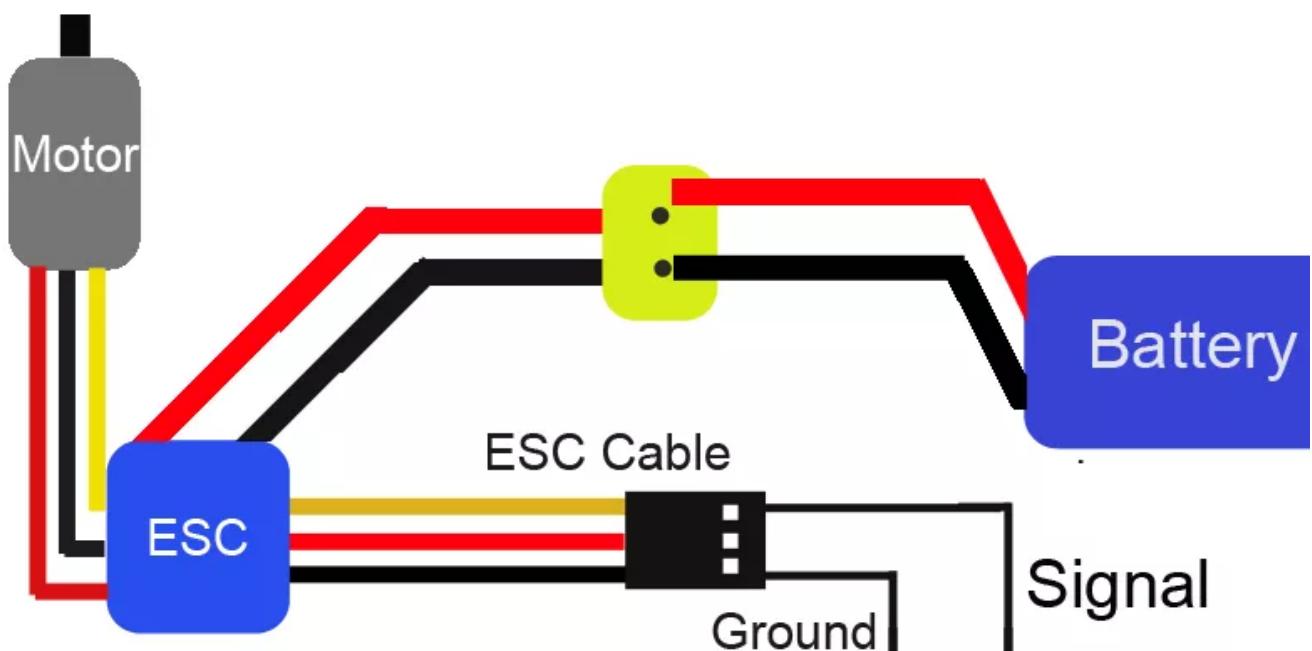
## Motores DC Brushless

Hay motores DC brushless: [https://en.wikipedia.org/wiki/Brushless\\_DC\\_electric\\_motor](https://en.wikipedia.org/wiki/Brushless_DC_electric_motor)



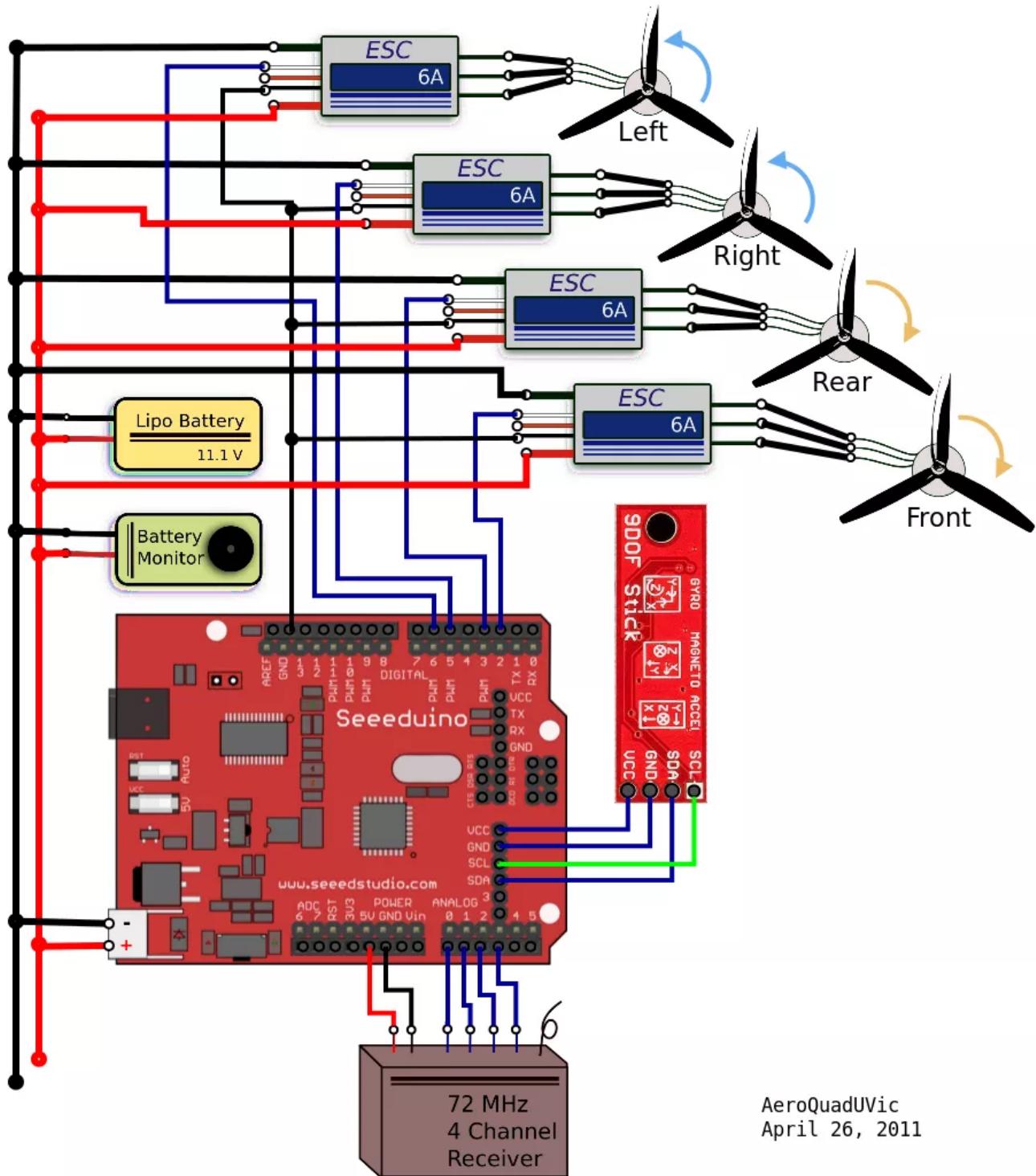
Para controlar los motores brushless necesitaremos un ESC (Electronic Speed Control)

[https://en.wikipedia.org/wiki/Electronic\\_speed\\_control](https://en.wikipedia.org/wiki/Electronic_speed_control)



lucaszanella.com

Los motores brushless se usan habitualmente en los drones son trifásicos con un variador para controlar de forma muy exacta la velocidad del motor.

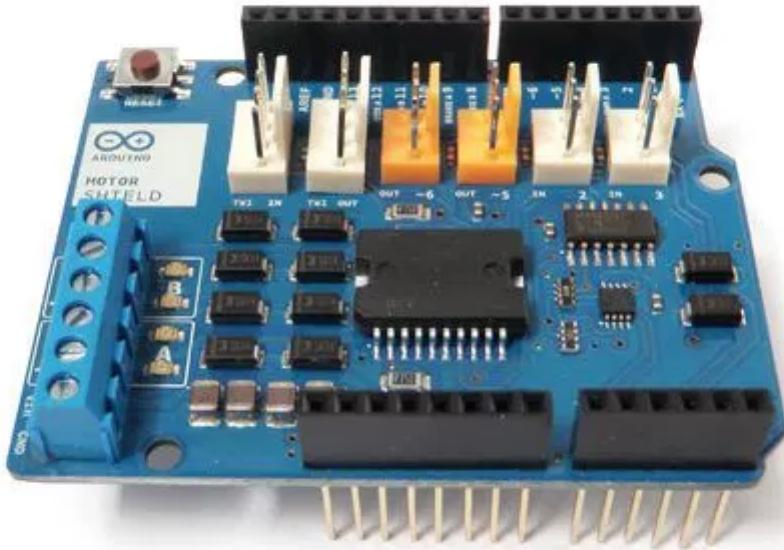


AeroQuadUVic  
April 26, 2011

Más información: <https://learn.adafruit.com/adafruit-motor-selection-guide/brushless-dc-motor-control>

## Arduino Motor Shield

El Arduino motor shield está basado en el L298 que es un dual full-bridge diseñado para cargas inductivas como relés, solenoides y motores DC o paso a paso. Me permite manejar dos motores DC controlando las velocidad y dirección de cada motor de forma independiente. También permite medir la corriente absorbida por cada motor entre otras características. La corriente máxima que puede manejar con 2A por canal y el voltaje máximo de suministro es de 50V.



Esquemático: [https://www.arduino.cc/en/uploads/Main/arduino\\_MotorShield\\_Rev3-schematic.pdf](https://www.arduino.cc/en/uploads/Main/arduino_MotorShield_Rev3-schematic.pdf)

Arduino motor shield puede ser alimentado por una fuente externa, por lo tanto debemos alimentar al Arduino conectado al motor shield mediante el alimentador y no es posible alimentarlo por USB porque la corriente del motor puede exceder la corriente máxima del estándar USB. El L298 también tiene un alimentación lógica que toma de los 5V de Arduino.

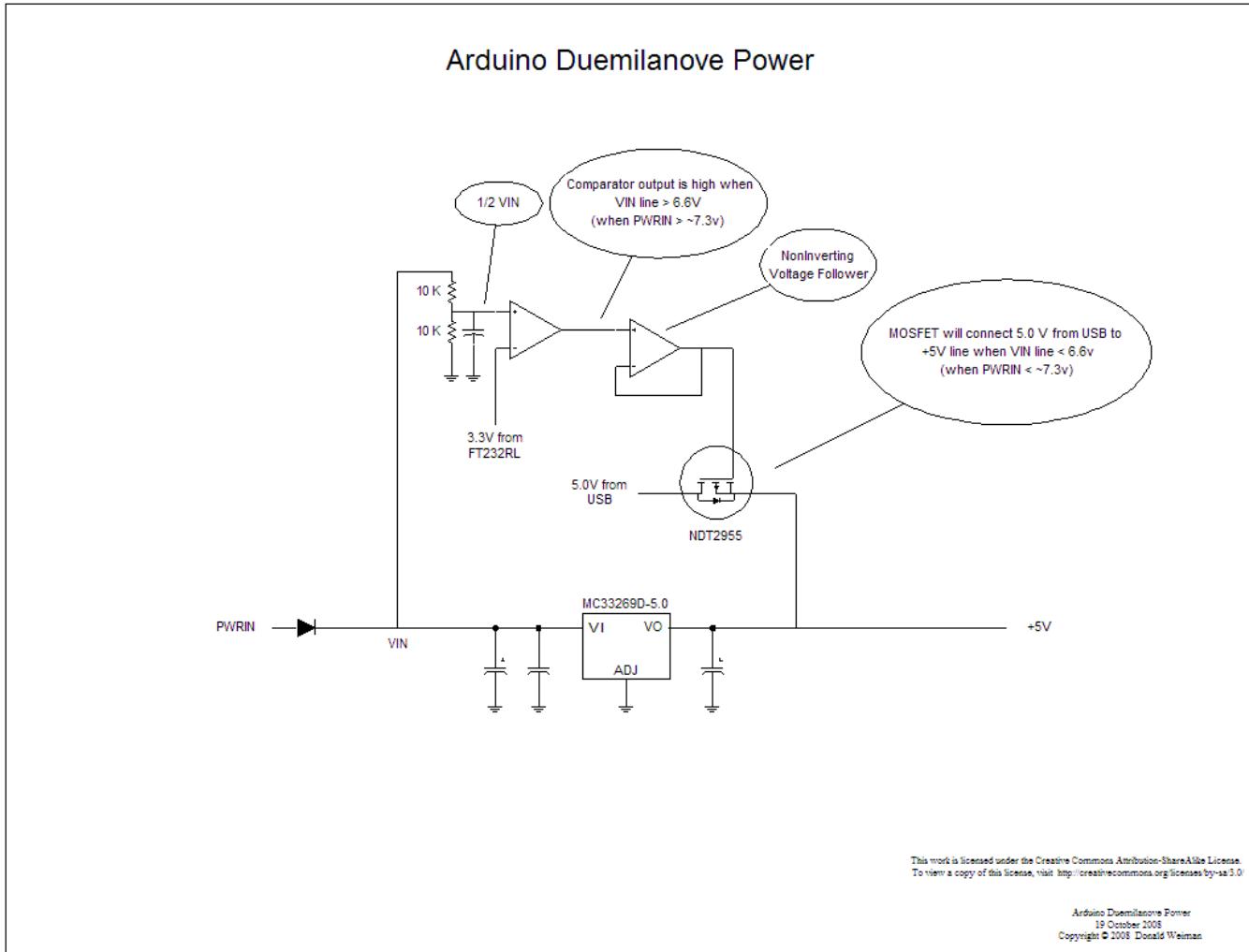
El L298 necesita dos alimentaciones, una para la parte de control (5V) y otra para los motores (12V). La de los motores en tu caso la coge del Vin al que conectas la fuente conmutada de 12V.

Al alimentar desde LSP5 (Borna marcada con Vin MAX 12V), es el VMOT que alimenta Vin según el esquemático de motor shield y al L298P. La alimentación lógica del L298P se hace a través del regulador de tensión del Arduino.

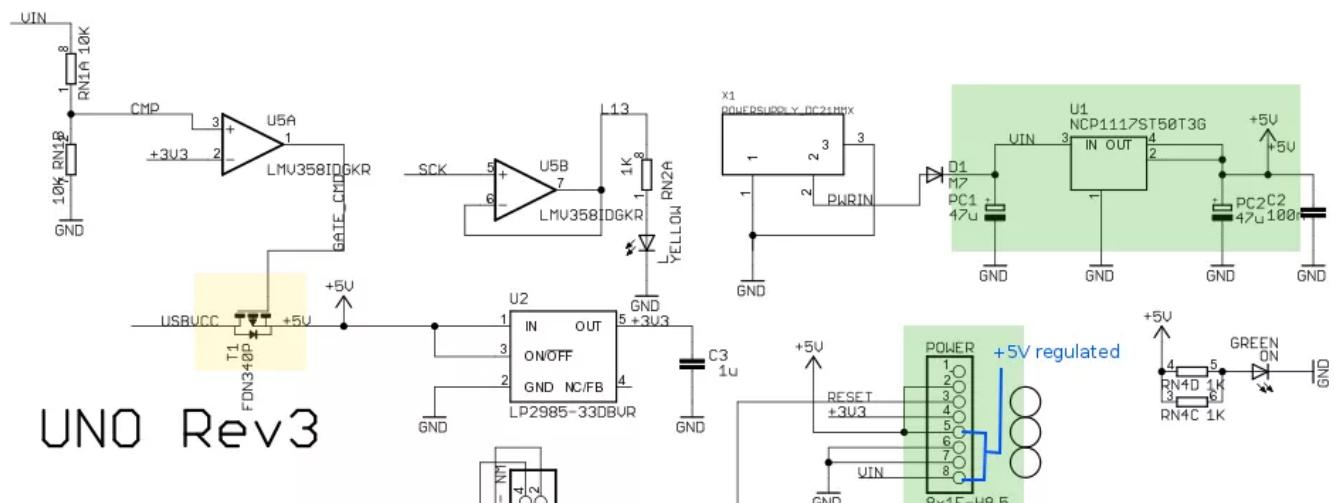
Al alimentar el Arduino mediante el Vin el Motor Shield, puede ser problemático, puesto que al alimentar Arduino a través del Vin no tenemos la protección del diodo M7 ni el condensador para estabilizar el voltaje, por lo que en caso de intercambiar la polaridad se puede dañar Arduino y además si al arrancar un motor el voltaje de Vin cae por debajo de un valor puede ocurrir que el regulador de tensión [NCP1117](#) no sea capaz de dar 5V a su salida y en ese caso Arduino se reiniciaría. Además el NCP1117 tiene una caída de tensión, por lo que para alimentar el Arduino a través de Vin como indica la documentación de Arduino debe ser con un valor entre 7 y 12 V.

Tener en cuenta que si alimento a través de Vin, pero tengo conectado el USB a Arduino, entonces, el MOSFET [FDN340P](#) permitirá alimentar el microcontrolador cuando  $Vin/2 < 3.3V$  y no se produce el reinicio

de la MCU, actuando como una fuente de alimentación de backup.



Esquema del Arduino UNO: <https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf>



Para evitar posibles daños al Arduino sobre el que va montado el motor shield, si el motor requiere más de 9V, es recomendable separar las líneas de alimentación del shield y del Arduino, esto es posible cortando el jumper "Vin Connect" que está en la parte trasera del shield.

En caso de problemas, la solución es alimentar por separado Arduino y el motor shield y cortar el Vin-Connect.

Los pines de alimentación en el Arduino motor shield son:

- Vin en la borna de conexiones, es la entrada de alimentación a los motores conectados al shield. Una fuente de alimentación externa conectada a esta borna también alimenta a la placa Arduino sobre la que va montada el shield. Cortando el jumper “Vin Connect” se elimina esta alimentación a Arduino.
- GND es la masa de los motores que va unida a la masa del Arduino.

Este shield dispone de dos canales llamados A y B y cada uno usa 4 de los pines de Arduino para manejar o monitorizar el motor. En total se usan 8 pines. Se pueden usar los canales separados para manejar dos motores DC o combinarlos para manejar un motor paso a paso bipolar.

Los pines son:

Function	pins per Ch. A	pins per Ch. B
<i>Direction</i>	D12	D13
<i>PWM</i>	D3	D11
<i>Brake</i>	D9	D8
<i>Current Sensing</i>	A0	A1

Si no se necesita el freno y el sensor de corriente, es posible deshabilitar estas características cortando los correspondientes jumpers en la parte trasera del shield.

Los pines de freno al ponerlos a HIGH frenan el motor en lugar de dejarlos correr libremente al cortar la alimentación.

Mediante la función analogRead() es posible leer la corriente en los pines de current sensing como una entrada analógica normal. Está calibrado para medir 3.3V para la corriente máxima de 2A.

Esquema: [https://www.arduino.cc/en/uploads/Main/arduino\\_MotorShield\\_Rev3-schematic.pdf](https://www.arduino.cc/en/uploads/Main/arduino_MotorShield_Rev3-schematic.pdf)

Más información: <https://www.arduino.cc/en/Main/ArduinoMotorShieldR3>

Ejemplos de uso:

- <https://www.arduino.cc/en/Tutorial/DueMotorShieldDC>
- <http://www.arduino.org/learning/tutorials/boards-tutorials/arduino-motor-shield-two-dc-motors-example>
- <http://www.instructables.com/id/Arduino-Motor-Shield-Tutorial/>

**NOTA:** El **Motor Shield** de arduino.cc ha sido retirado, pero arduino.org comercializa el Arduino Motor Shield que es el mismo: <http://www.arduino.org/products/shields/arduino-motor-shield>

## Otros shields y breakout board para motores DC

### Adafruit Motor Shield:

- Producto: <https://www.adafruit.com/products/1438>
- Tutorial: <https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino>
- Librería: <https://github.com/adafruit/Adafruit-Motor-Shield-library/>

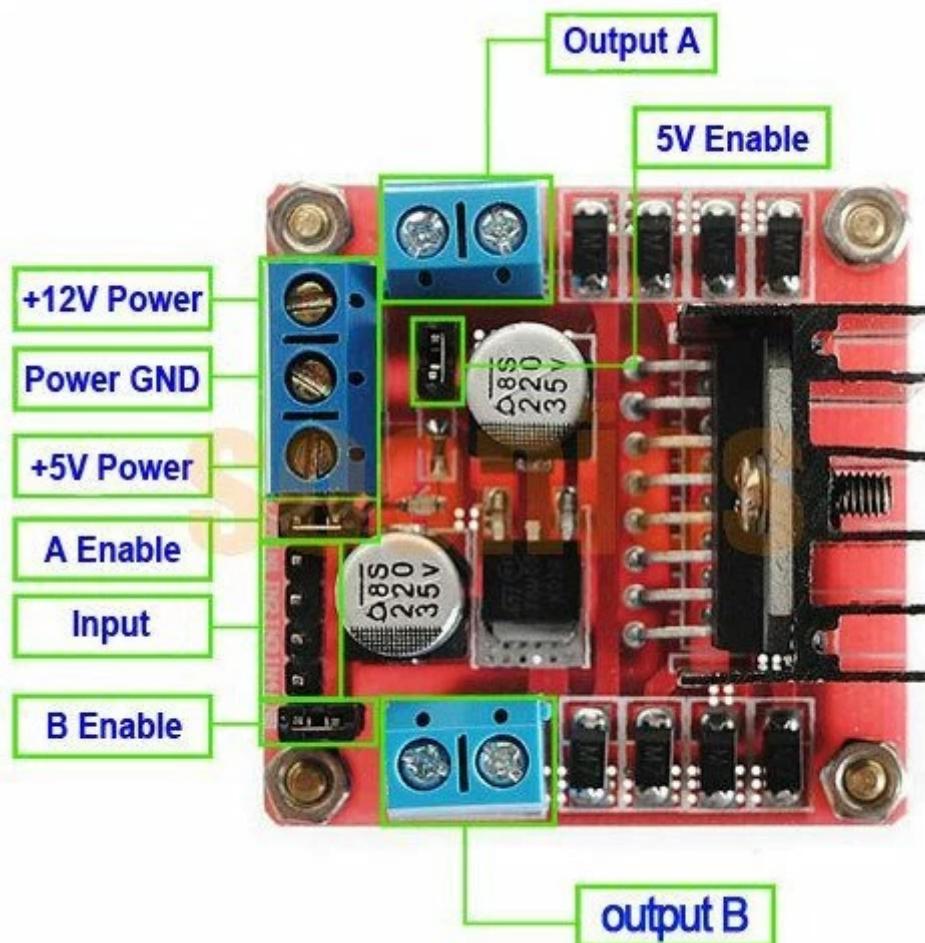
**L298N Breakout Board** (Esta breakout board es muy sencilla y bien documentada):

- Producto: <http://www.geeetech.com/l298n-stepper-motor-driver-board-p-567.html>
- Wiki: [http://www.geeetech.com/wiki/index.php/L298N\\_Motor\\_Driver\\_Board](http://www.geeetech.com/wiki/index.php/L298N_Motor_Driver_Board) (con esquemático)
- El que usa el coche: [http://www.vetco.net/catalog/product\\_info.php?products\\_id=14337](http://www.vetco.net/catalog/product_info.php?products_id=14337)
- Test: <https://developer.mbed.org/teams/TVZ-Mechatronics-Team/code/L298N-Breakout-Test/>

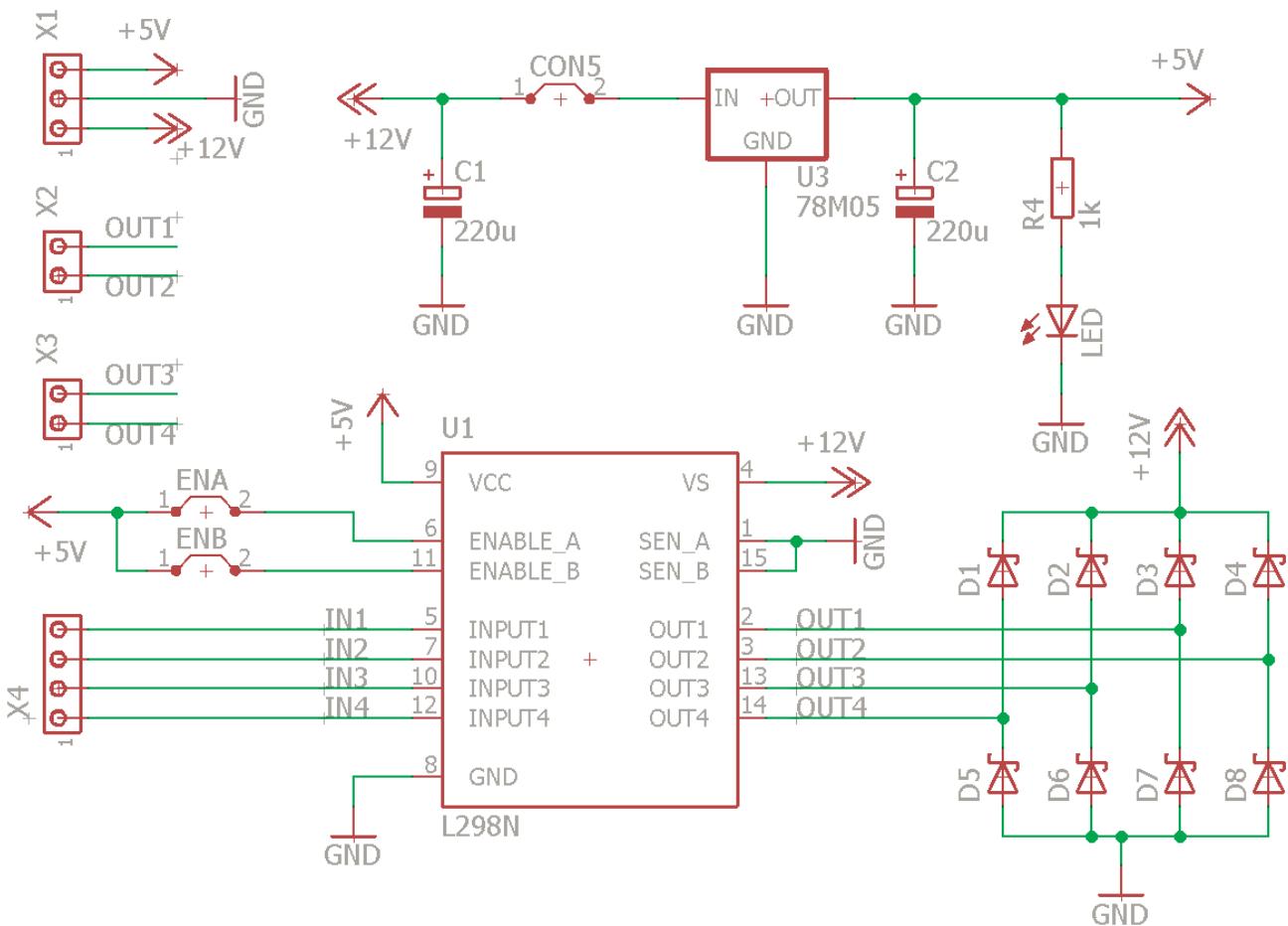
Specification:

- chipset: L298N
- Driving power supply voltage Vs: +5V to +46V
- Peak current of driving power supply Io: 2A
- Vss: +5V to +7V
- Current of logic power supply: 0 – 36mA
- PWM control signal range:
  - Low level: -0.3V < Vin < 1.5V
  - High level: 2.3V < Vin < Vss
- Enable signal range:
  - Low level: -0.3V
  - High level: 2.3V < Vin < Vss
- Maximum power consumption: 25W
- Working temperature: -25C to 130C
- Regulador de tensión para los 5V.





Esquemático:

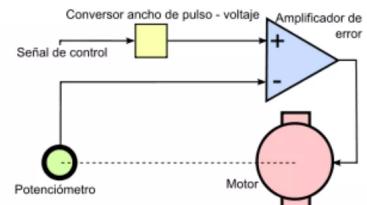
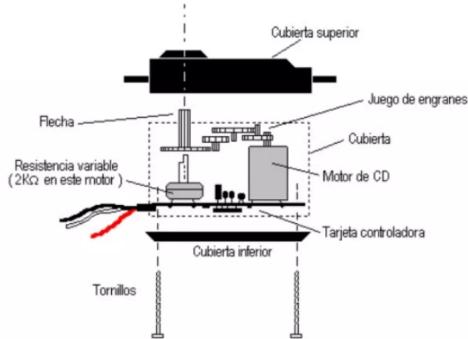


## Servomotor

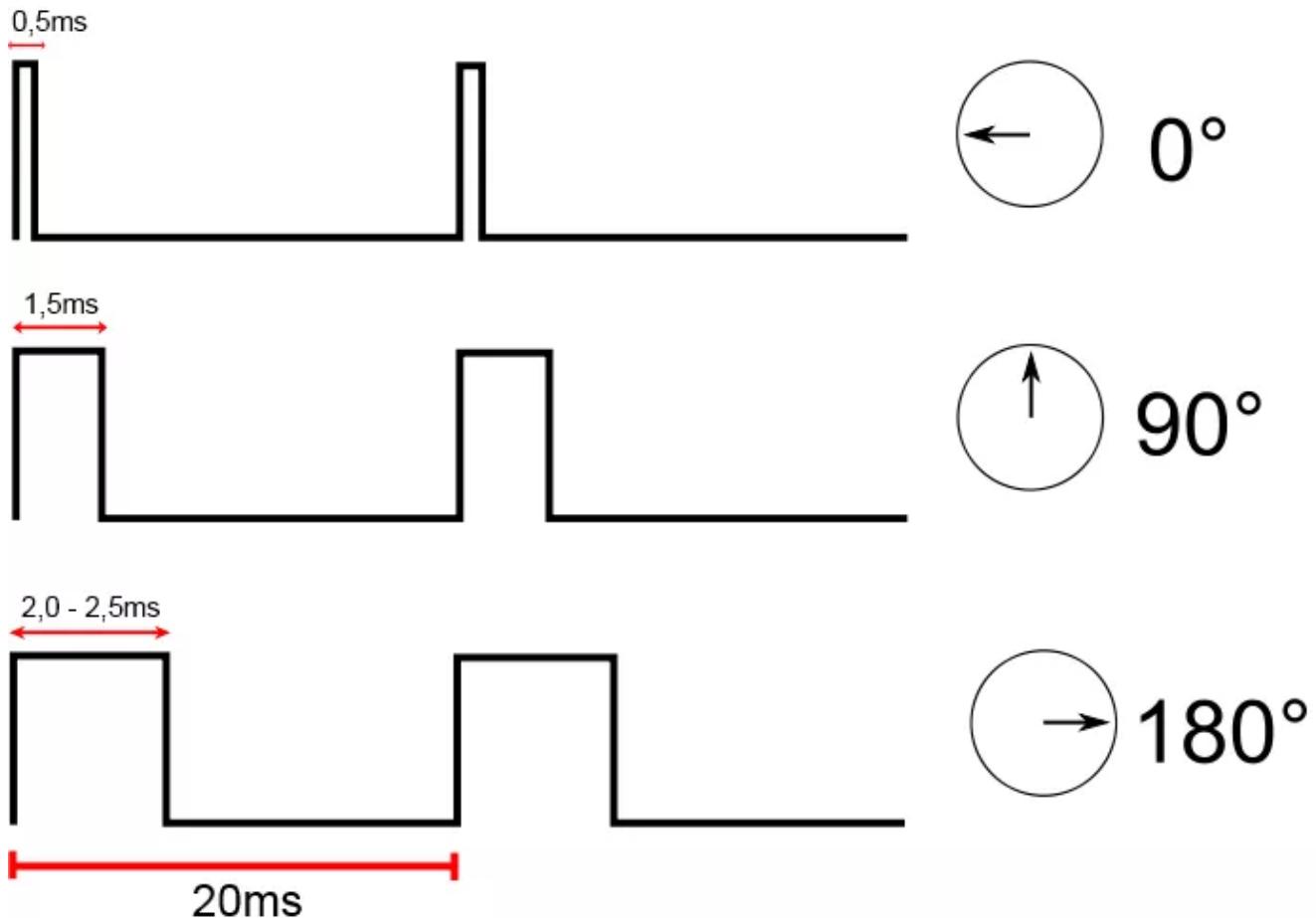
Servomotor (o también llamado servo) es similar a un motor de corriente continua pero con la capacidad de posicionarse en una posición determinada y permanecer fija en esta. Normalmente el ángulo es de 0 a 180 grados, y se alimentan a 5 voltios mínimo.

Algunos servos pueden alimentarse directamente desde Arduino sin necesidad de un driver, lo que supone una ventaja en algunos casos.

Un servomotor está formado por un motor de corriente continua, una caja reductora, un juego de engranajes, un potenciómetro y un circuito de control. Puede aguantar cierto peso a través del par o torque del servo indicado en sus características. Normalmente se indica con Kg/cm, que quiere decir los kilos que aguanta a 1 cm de distancia.



Para controlar un servo, se usa el PWM. La mayoría trabaja en una frecuencia de 50 Hz (20ms). Cuando se manda un pulso, la anchura de este determina la posición angular del servo. La anchura varía según el servomotor pero normalmente es entre 0,5ms a 2,5ms.



El Arduino utiliza la librería `<Servo.h>` para controlar los servos y usa las siguientes funciones:

- <http://arduino.cc/en/Reference/Servo>
- <http://arduino.cc/en/Reference/ServoAttach>
- <http://arduino.cc/en/Reference/ServoWrite>
- <http://arduino.cc/en/Reference/ServoRead>

Fuente, más información y código en: <http://diymakers.es/controlar-servomotor-con-mando-ir/>

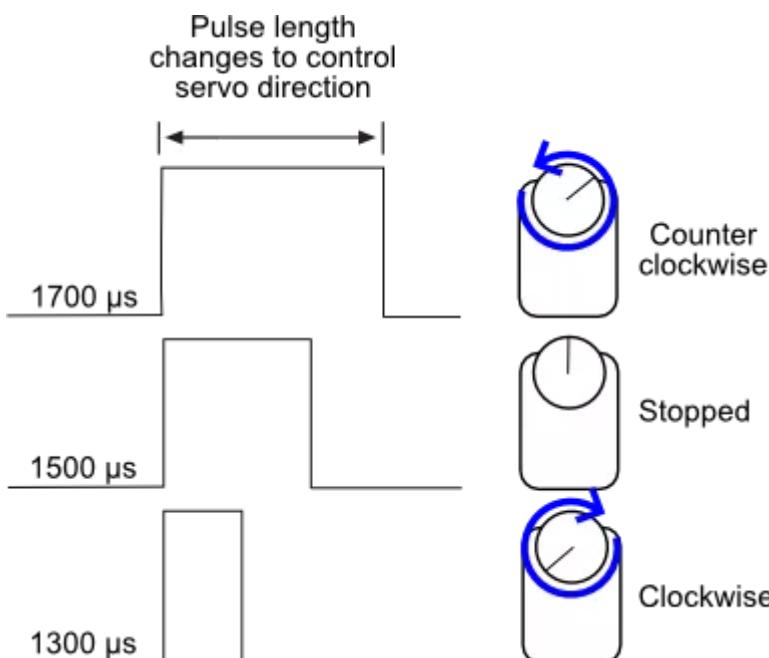
Más información: <http://en.wikipedia.org/wiki/Servomotor>

Si queremos tener control de bucle cerrado con un motor DC como tenemos con los servos, podemos hacerlo incorporando un encoder al motor DC.

## Servo de rotación continua

Un servo de rotación continua es un motor cuyo circuito electrónico nos permite controlar la dirección de giro. A diferencia de los servos anteriormente mencionados, no se detiene en una posición, sino que gira continuamente. Son muy utilizados en robótica y en muchas aplicaciones electrónicas, como en lectores de DVD.

En un servo de rotación continua, la función `write()` configura la velocidad del servo en lugar del ángulo de posición. En este caso 0 es máxima velocidad en giro contrario al sentido horario, 180 es máxima velocidad en sentido horario y 90 motor parado.



Trucar un servo a rotación continua: <http://www.ardumania.es/trucar-servo-a-rotacion-continua/>

Servo de rotación continua comerciales:

- <http://tienda.bricogeek.com/motores/118-servomotor-de-rotacion-continua-sm-s4303r.html>
- [https://www.servocity.com/html/hsr-2645cr\\_servo\\_continuous\\_r.html#.VqiH0\\_nhDct](https://www.servocity.com/html/hsr-2645cr_servo_continuous_r.html#.VqiH0_nhDct)
- <https://www.pololu.com/product/1248>

Tutorial sencillo de BQ: <http://diwo.bq.com/muevete-el-servo-de-rotacion-continua/>

Tutorial más completo de servos de rotación continua:

- <http://www.element14.com/community/community/design-challenges/enchanted-objects/blog/2015/04/03/review-3-digital-continuous-rotation-360-servo>
- <http://www.element14.com/community/community/design-challenges/enchanted-objects/blog/2015/04/04/review-4-digital-continuous-rotation-360-servo-part-2>

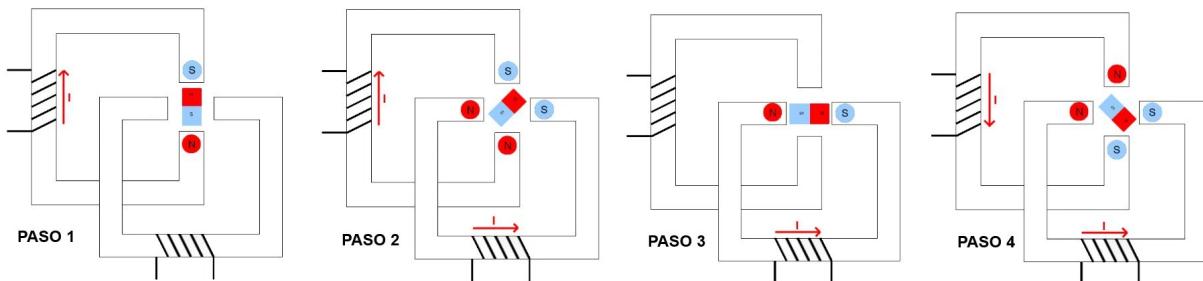
Calibrador de servo: <http://www.instructables.com/id/Servo-calibrator-GUI/>

Servo driver para Raspberry Pi: <https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/overview>

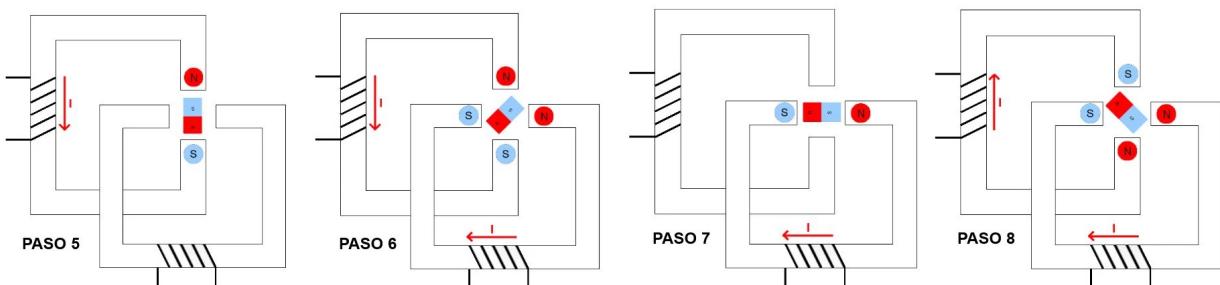
## Motor paso a paso

Un motor paso a paso (también llamado stepper) es un dispositivo electromagnético que convierte impulsos eléctricos en movimientos mecánicos de rotación. La principal característica de estos motores es que se mueven un paso por cada impulso que reciben. Normalmente los pasos pueden ser de  $1,8^\circ$  a  $90^\circ$  por paso, dependiendo del motor. Son motores con mucha precisión, que permiten quedar fijos en una posición (como un servomotor) y también son capaces de girar libremente en un sentido u otro (como un motor DC).

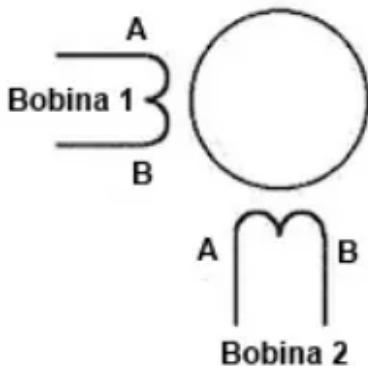
Cuando circula corriente por una o más bobinas del estator se crea un campo magnético creando los polos Norte-Sur. Luego el rotor se equilibrará magnéticamente orientando sus polos Norte-Sur hacia los polos Sur-Norte del estator. Cuando el estator vuelve a cambiar la orientación de sus polos a través de un nuevo impulso recibido hacia sus bobinas, el rotor volverá a moverse para equilibrarse magnéticamente. Si se mantiene esta situación, obtendremos un movimiento giratorio permanente del eje. El ángulo de paso depende de la relación entre el nombre de polos magnéticos del estator y el nombre de polos magnéticos del rotor.



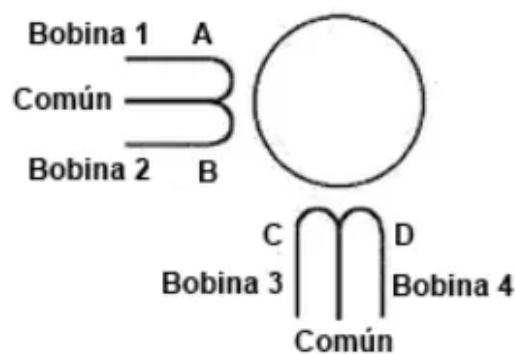
MOTOR PASO A PASO DE  $45^\circ$  DE PASO ANGULAR



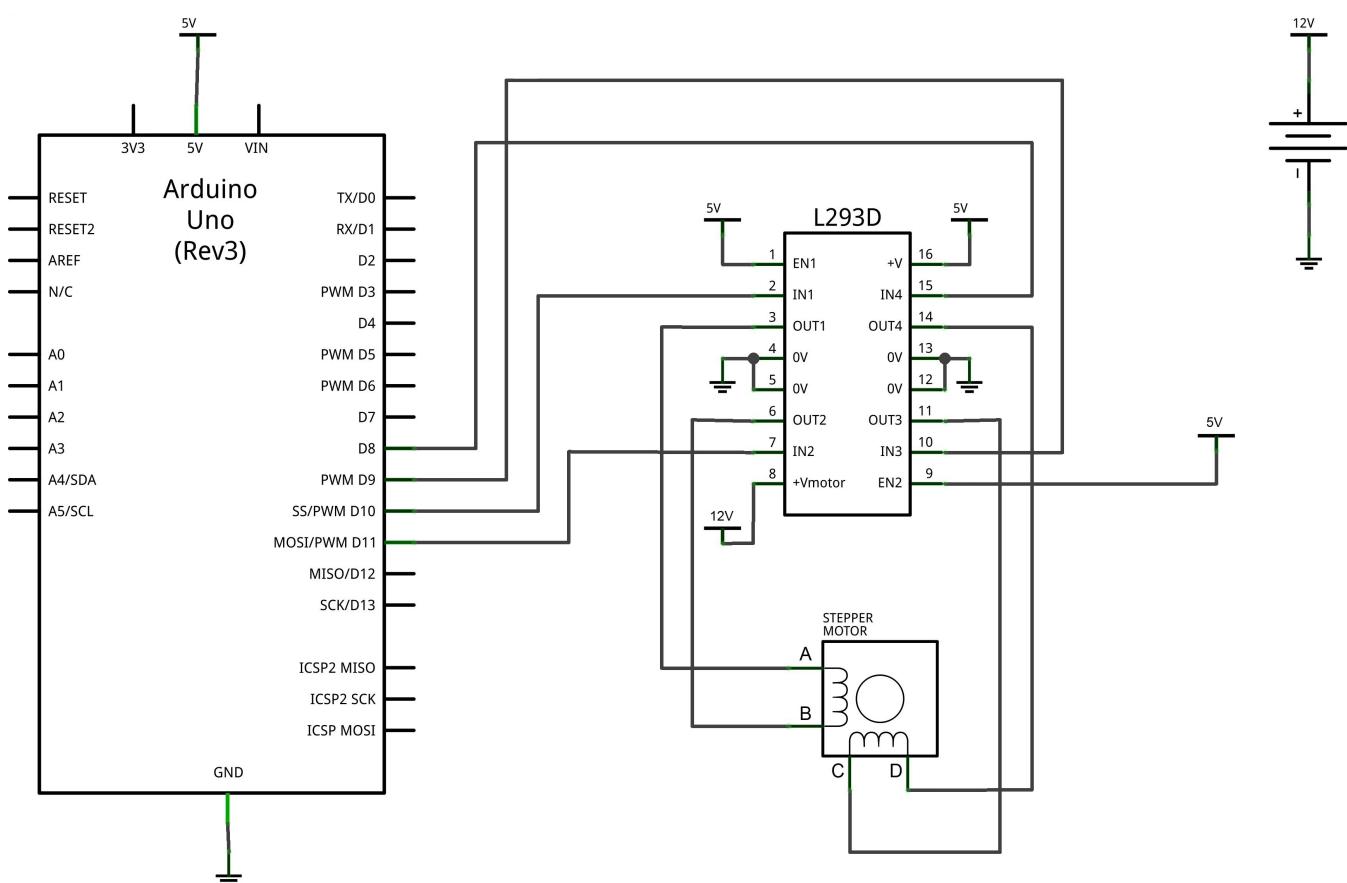
## **Motor paso a paso bipolar**



### **Motor paso a paso unipolar**



Los motores bipolares son más complejos de controlar ya que el flujo de corriente tiene que cambiar de dirección a través de las bobinas con una secuencia determinada. Para esto debemos conectar cada una de las dos bobinas en un puente en H (H-Bridge). Para esto, utilizaremos el integrado L293 que contiene dos H-Bridge ([datasheet](#)).



Para controlar motores paso a paso con Arduino, utilizaremos la librería <Stepper.h>:

- <http://arduino.cc/en/Reference/Stepper/>
  - <http://www.tigoe.net/pcomp/code/circuits/motors/stepper-motors/>

Fuente, más información y código en:

- <http://diymakers.es/mover-motores-paso-paso-con-arduino/>
- [http://es.wikipedia.org/wiki/Motor\\_paso\\_a\\_paso](http://es.wikipedia.org/wiki/Motor_paso_a_paso)

Un ejemplo de driver de un motor paso a paso <https://www.pololu.com/product/2133>

Datasheet del driver: [https://www.pololu.com/file/download/drv8825.pdf?file\\_id=0J590](https://www.pololu.com/file/download/drv8825.pdf?file_id=0J590)

Este tipo de motor es más lento, su rotación es más precisa, es de fácil configuración y control, además mientras que los servos requieren un mecanismo de retroalimentación y circuitos de soporte para accionamiento de posicionamiento, un motor paso a paso tiene control de posición a través de su naturaleza de rotación por incrementos fraccionales.

Los motores paso a paso son adecuados para las impresoras 3D y dispositivos similares en los que la posición es fundamental.

Además de los drivers de motores paso a paso como los puente H y los darlington array, es posible usar otros drivers como Easydriver que funciona dando los pasos y la dirección. Más información:

- <http://www.schmalzhaus.com/EasyDriver/>
- <https://www.sparkfun.com/products/12779>

Para los motores paso a paso podemos usar las mismas shields que para los motores DC:

- <https://www.arduino.cc/en/Main/ArduinoMotorShieldR3>
- <https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino>
- <https://learn.adafruit.com/adafruit-motor-shield>

## Más Motores

Los de alterna monofásicos podemos controlar su encendido, apagado y sentido de giro con un relé o un contactor y estos a su vez controlados con un Arduino como se ha visto anteriormente.

Los motores trifásicos habrá que usar un arrancador para controlar su encendido y apagado y un variador para controlar la velocidad y podríamos controlarlos estos mediante señales adecuadas desde Arduino como modbus.

Disponemos de shields para controlar múltiples motores, ya sean DC, servos o paso a paso.

A la hora de seleccionar un motor, hay varios factores que deben evaluarse antes de decidir qué tipo utilizar. Estos factores incluyen velocidad, par, bucle abierto o cerrado, resolución, precio y mantenimiento.

La velocidad es uno de los criterios más importantes a tener en cuenta al elegir entre servo y motor paso a paso. Hay una relación inversa entre velocidad y par en los motores por pasos. Cuando la velocidad se

incrementa, el par decrece. Los motores servo tienen un par constante hasta la velocidad nominal. Como criterio general, por encima de 1000 rpm, debe seleccionarse servo. Si la velocidad está por debajo de 500 rpm, los motores por pasos son una buena elección porque producen un par más alto que el servomotor de tamaño equivalente. Los servomotores tienen la capacidad de producir un par pico en cortos períodos de tiempo que es hasta 8 veces su par nominal continuo. Esto es particularmente útil cuando la resistencia del movimiento no es constante. Los motores por pasos no tienen esta capacidad.

En algunos casos que requieren la alta velocidad, alta aceleración o aplicaciones críticas, el bucle cerrado es importante. Esto ocurre por ejemplo cuando se trabaja con accesorios caros en los que los fallos no son aceptables.

Guía de selección de motores: <https://learn.adafruit.com/adafruit-motor-selection-guide/types-of-motors>

Web como muchos motores: <https://www.servocity.com/>

Por ejemplo de un motor CC con reductora y encoder:

[https://www.servocity.com/html/26\\_rpm\\_planetary\\_gearmotor\\_w\\_.html#.VpfFM\\_nhDct](https://www.servocity.com/html/26_rpm_planetary_gearmotor_w_.html#.VpfFM_nhDct)

Esta entrada se publicó en Arduino, Motores y está etiquetada con Arduino, Arduino Motor Shield, L298N, Motor DC, Motor Paso a Paso, Motores, Servo, Servo Rotación Continua, Servomotor, Stepper en 4 julio, 2016 [<https://aprendiendoarduino.wordpress.com/2016/07/04/motores/>] .

---

## Práctica: Motores

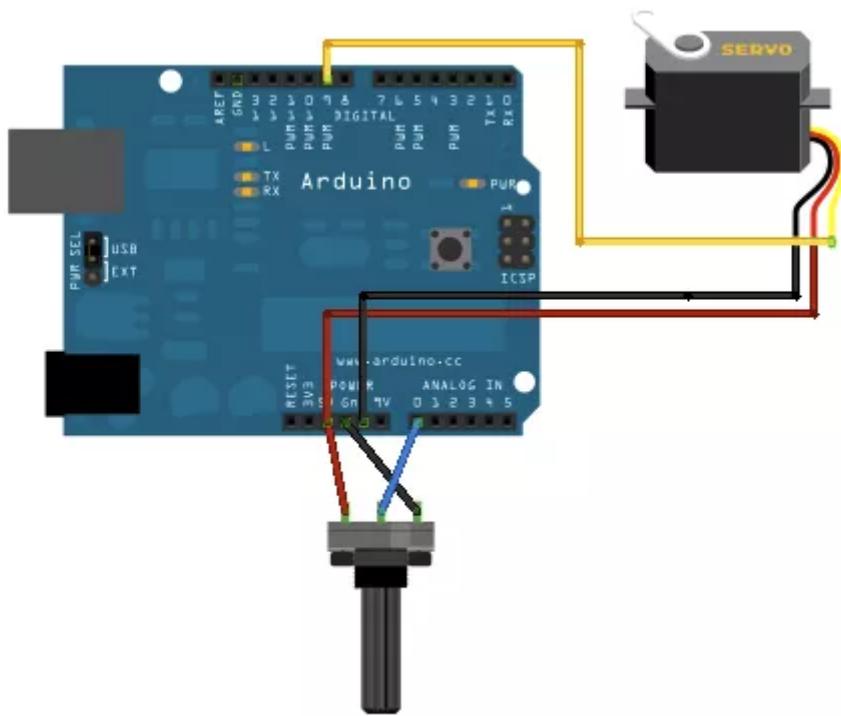
### Servo

Ejercicio: Controlar la posición de un servo con un potenciómetro.

Solución: <http://arduino.cc/en/Tutorial/Knob>

Ejercicio: Programar un barrido continuo del 0 a 180° en un servo. Activar y desactivar el barrido con una pulsación de un botón. p.e. activación de un limpiaparabrisas.

Solución: <http://arduino.cc/en/Tutorial/Sweep>



Esta entrada se publicó en Actuadores, Arduino, Motores, Práctica y está etiquetada con Arduino,

Avanzado, Motores, Práctica, Servo en 31 marzo, 2015

[<https://aprendiendoarduino.wordpress.com/2015/03/31/practica-motores/>] .

---

Entradas antiguas

c