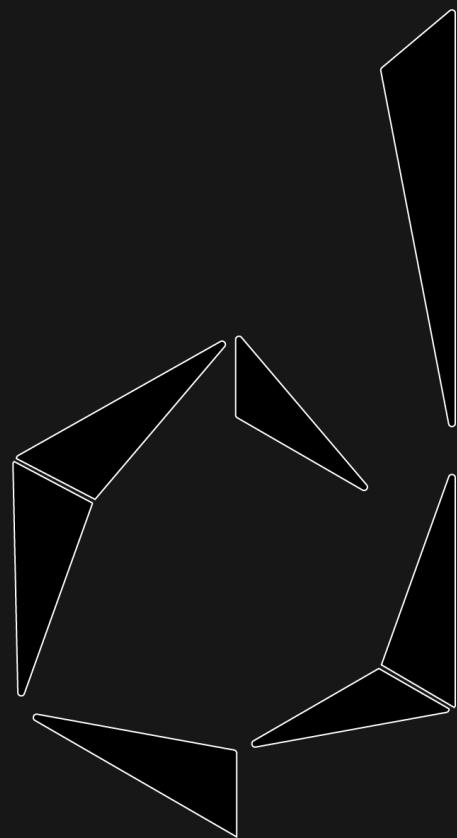


# MECHATRONICS ENGINEERING



## DI\_CERO

DIEGO CERVANTES RODRÍGUEZ

BEVISIONEERS, THE MERCEDES BENZ FELLOWSHIP

ENERDRAIS: BYCICLE FRONT WHEEL FOR CLEAN ENERGY GENERATION

## Milestone 1

# Contenido

<b>Partes y Tipos de Motores Eléctricos AC/DC .....</b>	3
<b>Partes y Tipos de Motores Eléctricos AC/DC .....</b>	4
Tipos de Motores Eléctricos AC/DC .....	4
Bobinas y Electroimanes .....	5
Partes de un Motor DC con Escobillas: Comutador, Estator, Armadura y Rotor .....	7
Conexión en Serie o Paralelo del Estator y Rotor de un Motor DC .....	8
<b>Manejo de la Corriente en las Bobinas de los Motores .....</b>	9
Diodo de Marcha Libre.....	9
Etapa de Control y Etapa de Potencia en un Circuito con Motores .....	10
<b>Etapa de Potencia:</b> Corriente Demandada por el Motor al Agregar una Carga .....	10
<b>Etapa de Control:</b> Aislamiento de los Circuitos Lógicos .....	11
<b>Baterías LiPo (Litio Polímero).....</b>	14
Datos Importantes de Uso: Voltaje Nominal, Capacidad y Tasa de Descarga .....	14
Carga y Descarga de las Baterías LiPo: Curvas de Descarga y Tasa de Carga .....	17
Cargador de Baterías LiPo IMAX B6AC.....	19
<b>Proceso de Carga de una Batería LiPo con el Cargador IMAXB6AC.....</b>	21
Medidor de Tensión para Baterías LiPo .....	21
Cuidados de una Batería LiPo .....	22
<b>Motor Brushless, BLDC o Sin Escobillas .....</b>	27
<b>Motor Brushless MT2204 2300 Kv .....</b>	36
<b>Motor Brushless A2212 1400 Kv .....</b>	36
<b>Electronic Speed Controller (ESC) 30A.....</b>	37
<b>Electronic Speed Controller (ESC) 30A BLHeli .....</b>	38
<b>Código Arduino - Control de Velocidad con Código:.....</b>	39
<b>Código Arduino - Control de Velocidad con Código y Duty Cycle Personalizado:.....</b>	39
<b>Código Arduino - Control de Movimiento con Potenciómetro:.....</b>	40
<b>Código Arduino - Control de Movimiento con JoyStick:.....</b>	41
<b>Módulo KY-023: Joystick .....</b>	41
Control de Motores Brushless en <b>Verilog y VHDL.....</b>	43
<b>Código Verilog – Control de Movimiento con Switches:.....</b>	44
Divisor de Reloj y Creación de Señal PWM: .....	44

Código UCF:.....	45
<b>Código VHDL – Control de Movimiento con Switches:.....</b>	<b>45</b>
Divisor de Reloj y Creación de Señal PWM: .....	45
Código UCF:.....	47
Simulación PWM:.....	47
Referencias: .....	48



# Partes y Tipos de Motores Eléctricos AC/DC



# Partes y Tipos de Motores Eléctricos AC/DC

## Tipos de Motores Eléctricos AC/DC

A continuación, se presenta la clasificación de los motores eléctricos:

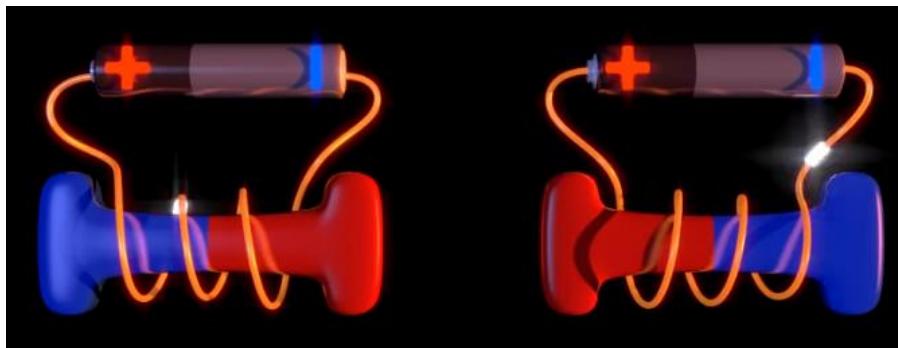
- **Motores DC:** Son impulsados por medio de corriente directa (pilas, rectificadores, etc).
  - **Motores con Escobillas:** Son motores que **realizan su conmutación con unas terminales llamadas escobillas**, que debido a su fricción deterioran el motor a través del tiempo.
    - ⊕ **Motor DC Sencillo:** **Motor que gira continuamente sin control** a altas velocidades.
    - ⊕ **Motorreductor:** Es un motor DC con escobillas sencillo, pero que cuenta con una serie de engranajes en su punta que reducen su velocidad, pero aumentan su torque (fuerza de rotación). **Este motor gira de forma continua sin control.**
    - ⊕ **Servomotor:** El servomotor es un motor DC con escobillas sencillo, pero que cuenta con el control **más preciso** de posición y velocidad de todos los motores eléctricos, debido a que tiene una entrada de retroalimentación. **Este motor puede detenerse en un punto deseado y mantener esa posición.**
  - **Motores sin Escobillas (Brushless) o BLDC:** Son motores que **realizan su conmutación sin las terminales llamadas escobillas, lo hacen a través de controladores externos** y duran más porque como no tienen escobillas, no se deterioran al generar fricción cuando giran, por lo que son más rápidos, fuertes y mejores al usarse como generadores eléctricos.
    - ⊕ **Motor BLDC Outrunner:** Es un motor trifásico, por lo que cuenta con 3 terminales para controlar el motor (A, B y C) y en su base puede o no tener 3 sensores de efector hall, que ayudan a determinar la posición del rotor. **El estator (parte fija) de este se encuentra en su centro y su rotor (parte que gira) en su perímetro.** **Este motor gira de forma continua sin control.**
    - ⊕ **Motor BLDC Inrunner:** Es también un motor trifásico que cuenta con 3 terminales para controlar el motor y a veces incluye 3 sensores de efector hall para determinar la posición de su rotor. Su principal diferencia con el motor **Outrunner** es que **el estator (parte fija) de este se encuentra en su perímetro y su rotor (parte que gira) en su centro**, girando de forma similar a los **motores DC con escobillas.** **Este motor gira de forma continua sin control.**
    - ⊕ **Motor a pasos (o Motor paso a paso):** Es un motor unipolar o bipolar que cuenta con 4 o 2 bobinas para controlar su movimiento, dependiendo de si es unipolar o bipolar, sus bobinas tienen un punto común de alimentación o no. Además, en la punta de cada bobina cuenta con conductores ferromagnéticos en forma de dientes que crearán los pasos del motor, haciendo que gire algo lento, pero proporcionándole **gran precisión** en su control de posición y velocidad. **Este motor puede detenerse en un punto deseado y mantener esa posición.**
- **Motores AC:** Funcionan mediante el suministro de corriente alterna (toma de luz o generador AC).
  - **Motor Síncrono:** La velocidad del rotor es igual a la velocidad del campo magnético en el estator. **Este motor gira de forma continua sin control.**
  - **Motor Asíncrono:** La velocidad del rotor NO es igual a la velocidad del campo magnético en el estator. **Este motor gira de forma continua sin control.**

## Bobinas y Electroimanes

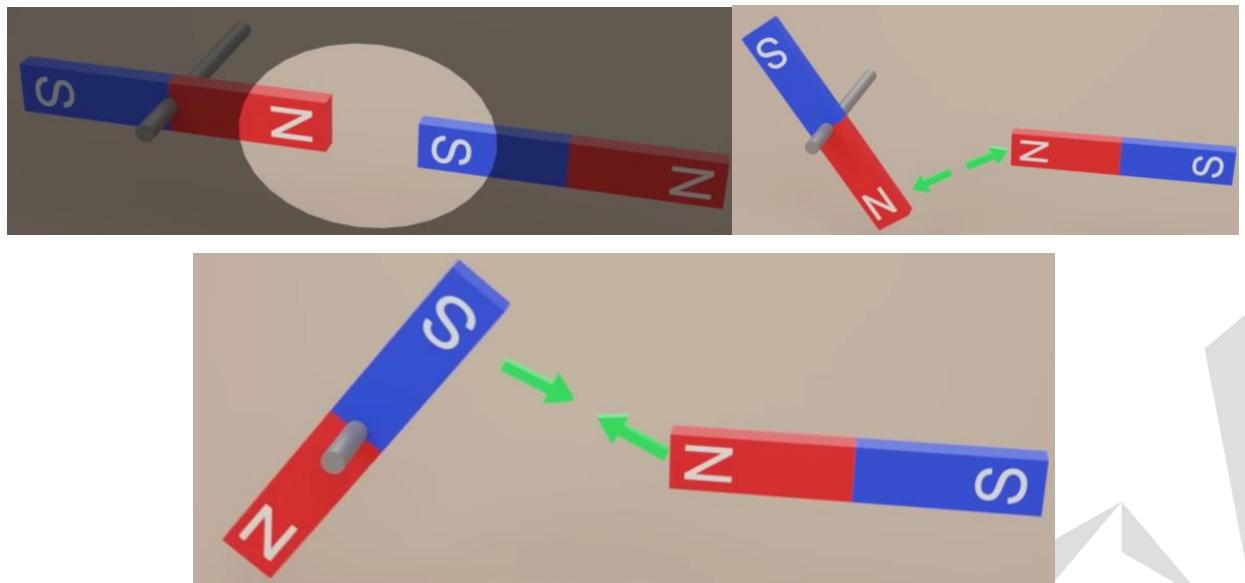
Para comprender el funcionamiento de cualquier motor eléctrico se debe conocer el funcionamiento básico de un electroimán, un electroimán suele estar compuesto de:

- **Metal ferromagnético:** Es un material que es conductor de campo magnético.
- **Bobina o inductor:** Es un alambre compuesto de un metal conductor eléctrico (usualmente cobre) enrollado alrededor del material ferromagnético mencionado previamente.

Al activar el electroimán, el metal del núcleo se comportará como un imán mientras exista un flujo de corriente eléctrica en el cable, pudiendo cambiar su polaridad según la forma en la que se orienten las vueltas del embobinado o invirtiendo el sentido de la fuente de alimentación (la pila). **Al hacerlo por mucho tiempo el metal se calienta por un efecto llamado corriente de Foucault (o Eddy).**

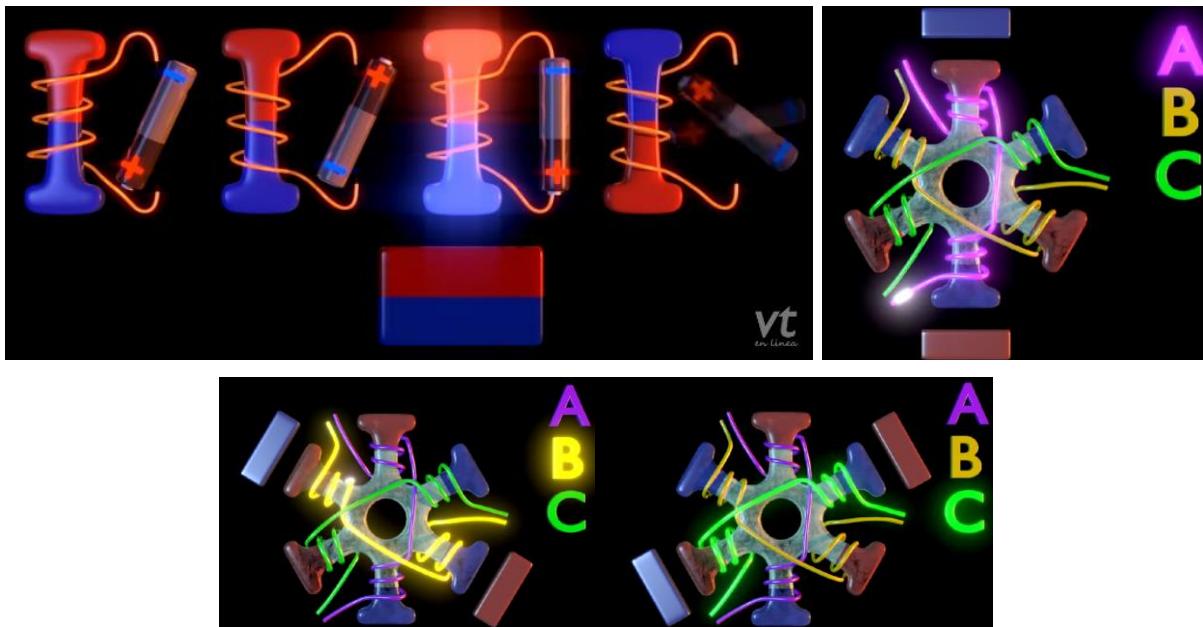


Esta acción convierte un metal ferromagnético cualquiera en un imán, y por medio de un magneto real (imán permanente) u otro electroimán se puede atraer o repeler sus polos (siempre y cuando el electroimán se encuentre encendido), generando así el **movimiento mecánico del motor**, siguiendo el principio magnético donde se dicta que **polos iguales se repelen y polos opuestos se atraen**.

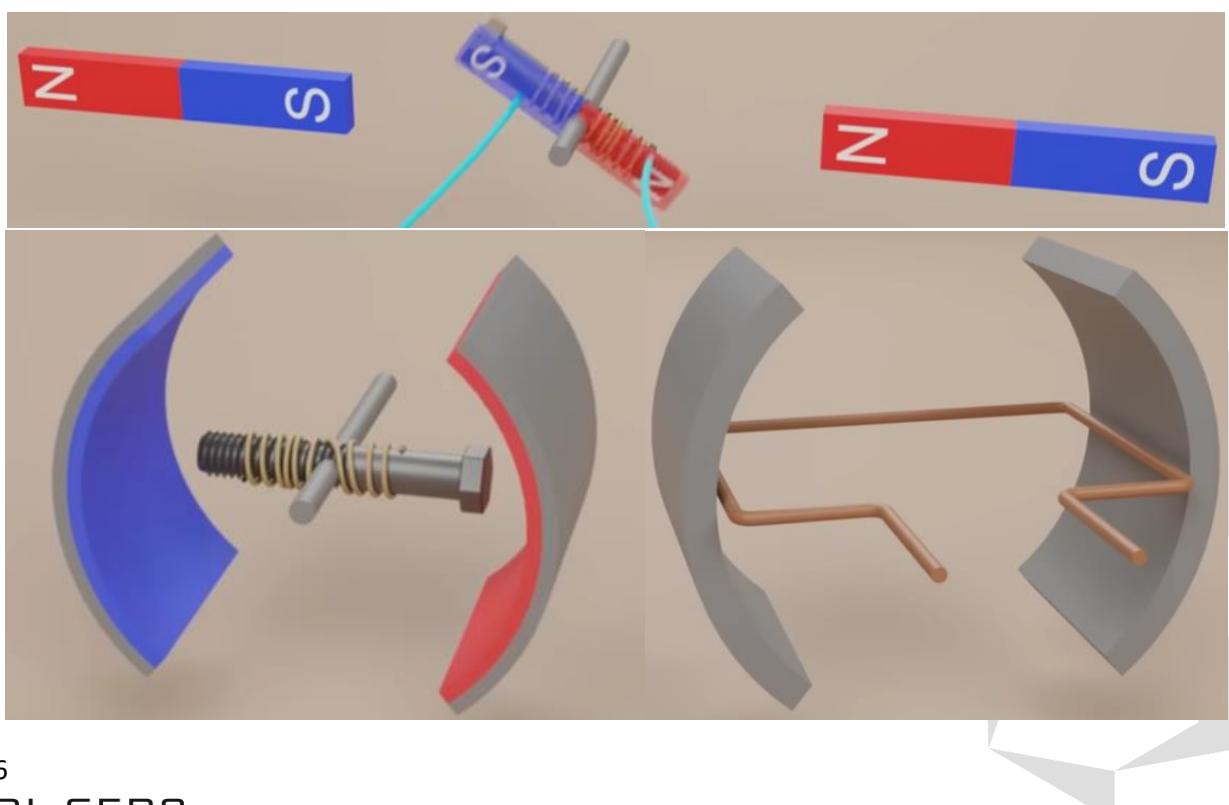


Basandonos en los conceptos explicados previamente, existen dos maneras globales de efectuar el movimiento de un motor eléctrico, que serán explicadas a continuación:

- Giro del imán permanente:** Se puede contar con varios **electroimanes estáticos** para **move el imán permanente de un motor** a la posición donde se requiera, en este ejemplo sencillo, solo uno debe estar encendido a la vez. El concepto es más aplicado en **motores DC sin escobillas**.



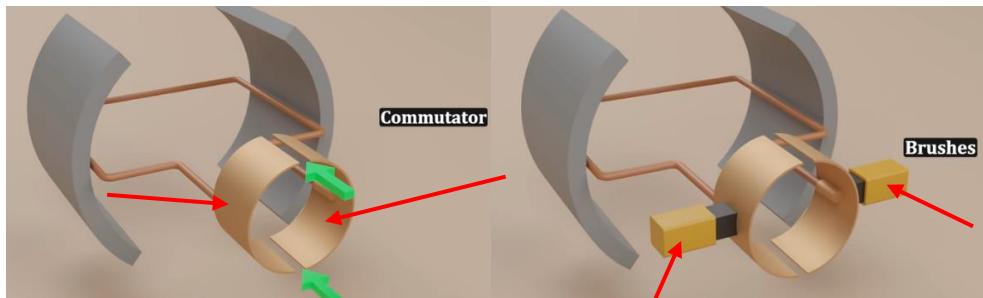
- Giro del electroimán:** Se puede contar con dos o más **iman permanentes estáticos** para **move el electroimán de un motor**, donde en vez de tener una bobina con un material ferromagnético dentro, se usarán varias bobinas que cambien el sentido de su corriente para invertir el sentido de su polaridad y al hacerlo serán movidas por medio dos o más imanes permanentes a su lado. Este concepto es más aplicado en **motores DC con escobillas**.



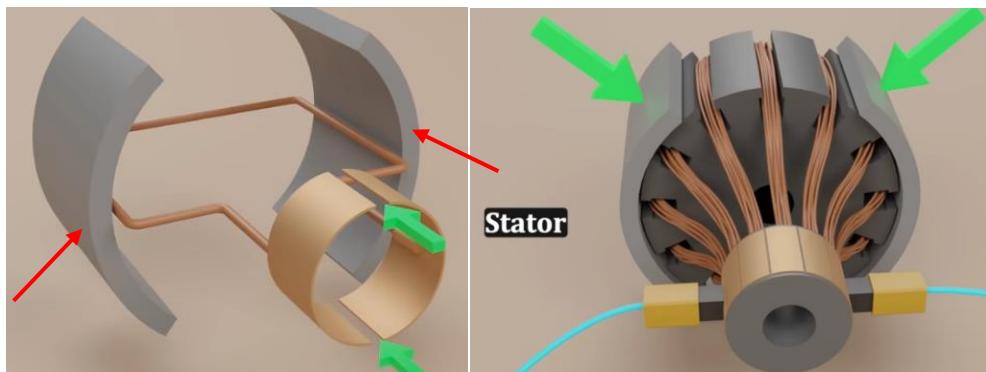
## Partes de un Motor DC con Escobillas: Comutador, Estator, Armadura y Rotor

Ahora que ya conocemos el efecto básico que mueve a los motores, debemos conocer el nombre de cada una de sus partes.

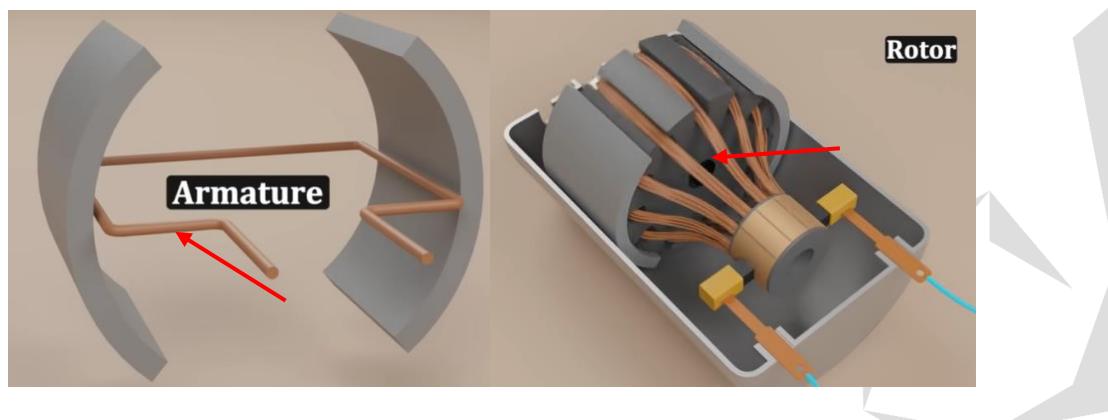
Los motores con escobillas se alimentan con corriente directa y la parte que realiza su conmutación, osea el cambio de dirección en la corriente de las bobinas pertenecientes a los electroimanes para generar su movimiento mecánico, son las mismas **escobillas** (o **brushes**) en conjunto con el **comutador** (o **colector**), ya que estas invierten el sentido de la polaridad del **electroimán interno que gira**.



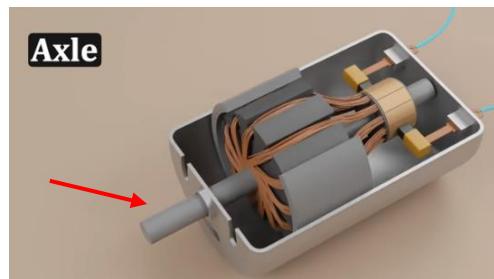
Los imanes permanentes usados para mover los electroimanes que se encienden y apagan en el motor son denominados como **estator** (stator), siendo la parte que se mantiene estática en el motor.



El metal ferromagnético usado como núcleo del electroimán se llama **rotor** y está conformado por varias láminas aisladas en vez de ser completamente sólido, para así evitar que se creen corrientes parásitas llamadas corrientes de Foucault (o Eddy) que lo calienten en vez de moverlo, además, las bobinas que lo rodean se llaman **armadura** (armature) y la combinación de ambas es la **parte que gira en el motor**.



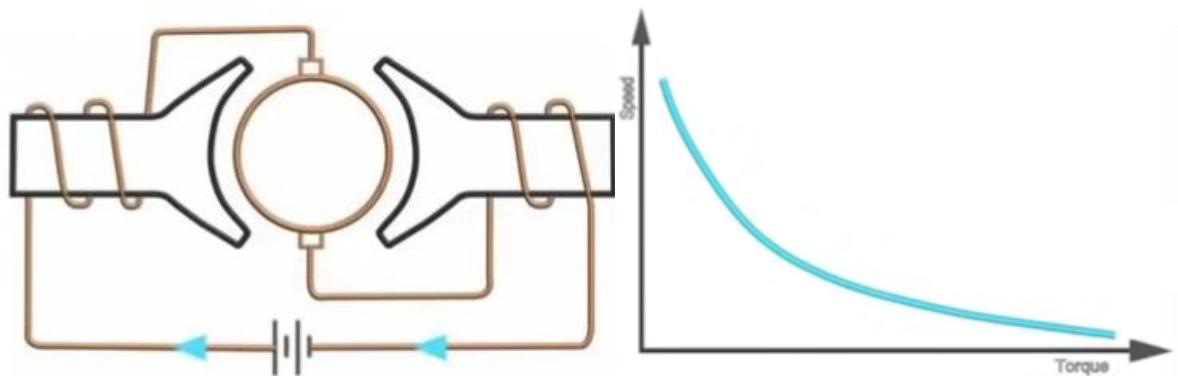
En la parte central del rotor se coloca el **eje del motor** (axle) y con esto finalizamos de nombrar todas las partes importantes de un motor DC con escobillas.



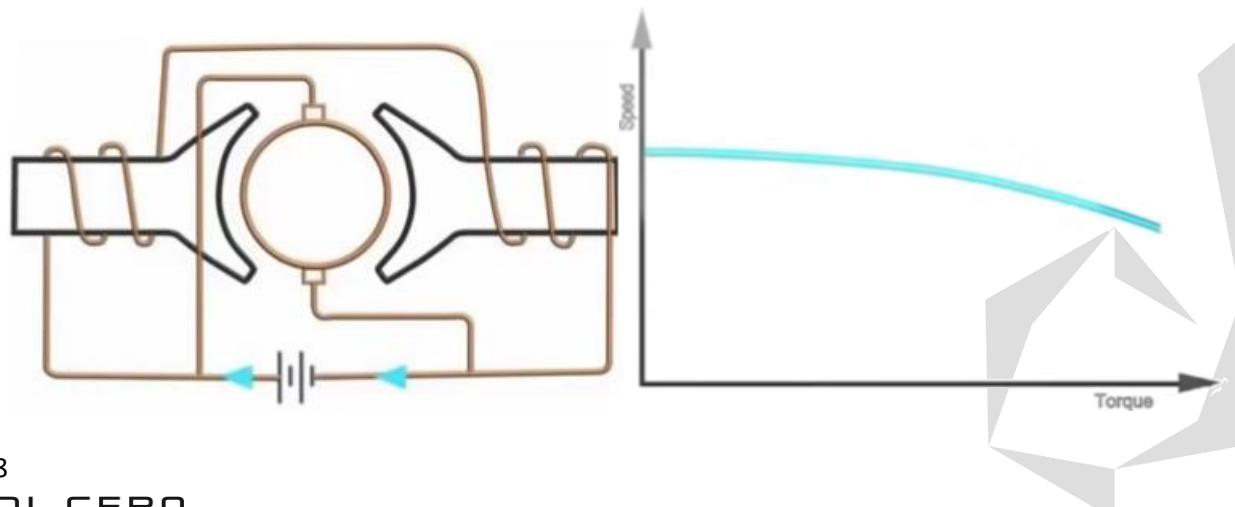
### Conexión en Serie o Paralelo del Estator y Rotor de un Motor DC

La estructura descrita anteriormente es la más común en motores DC con escobillas, pero a veces el **estator no es un imán permanente, sino otro electroimán que puede estar conectado en serie o en paralelo con el electroimán del rotor**, logrando así las diferentes características descritas a continuación:

- **Motor DC conectado en Serie:** Este motor tiene un buen torque o par (fuerza de rotación) de arranque (al iniciar el movimiento del motor), pero su velocidad disminuye drásticamente al ponerle una carga, osea un peso que el motor mueva.



- **Motor DC conectado en Paralelo (Motor en Derivación o Shunt):** Este motor tiene un bajo torque de arranque (al iniciar el movimiento del motor), pero es capaz de mantener una velocidad casi constante al ponerle una carga, osea un peso que el motor mueva.



# Manejo de la Corriente en las Bobinas de los Motores

Como el corazón de todos los tipos de motores es el electroimán, que está conformado por inductores (bobinas), una de sus propiedades fundamentales es que se opone a cambios bruscos en la corriente del circuito debido a su inductancia.

**De acuerdo con la ley de Faraday, la autoinductancia es la capacidad de una bobina para generar una fuerza electromotriz (fem o emf) en respuesta a un cambio brusco en la corriente que lo atraviesa.** La **fem** en realidad es una tensión eléctrica con polaridad inversa a la corriente cambiante, lo que causa una oposición a su flujo y ralentiza su cambio, su ecuación es la siguiente, donde:

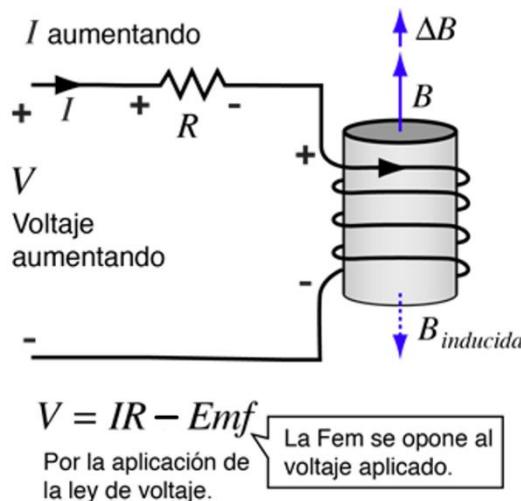
$B$  = Flujo magnético inducido en un material [Tesla]

$V_{fem}$  = Tensión de fuerza electromotriz inducida [V]

$L$  = Inductancia [Henrys]

$I$  = Corriente cambiante del circuito [Amperes]

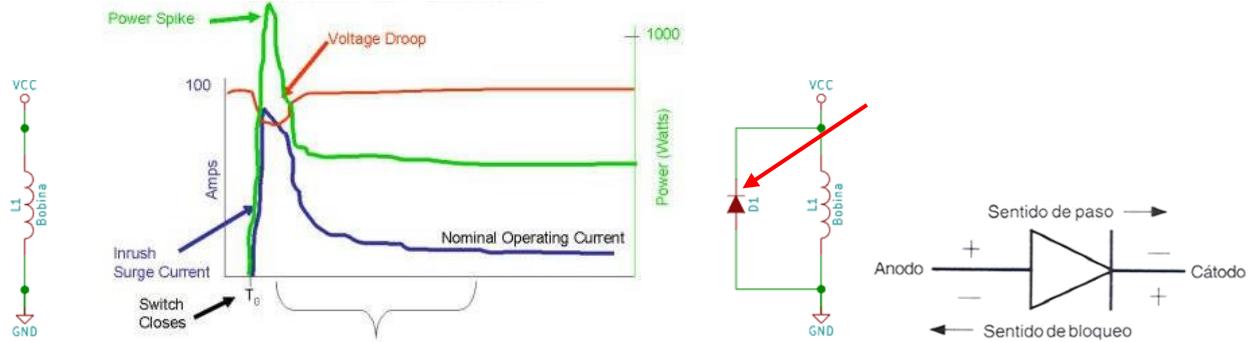
$$V_{fem} = V_{Emf} = L \left( \frac{dI}{dt} \right) [V]$$



## Diodo de Marcha Libre

Una situación muy sencilla que se puede analizar para exemplificar el efecto de la inductancia en los motores eléctricos es cuando se alimenta uno directamente:

- 1) En un inicio su bobina no está energizada, pero al conectarse por primera vez, se creará un campo magnético en ella.
- 2) **Al apagar su alimentación, la corriente cambiará bruscamente a ser cero**, situación que tratará de ser impedida por el inductor del motor.
  - a. Esto ocasionará que se cree una enorme **fem** (**tensión en sentido contrario**) en las bobinas del motor, que tratará de impedir que la corriente en la bobina cambie a ser cero bruscamente, aunque obviamente no lo podrá lograr, ya que la alimentación ha sido removida. **Esta tensión generada puede ser hasta 10 veces mayor que la inicial en el circuito**, pero en sentido contrario, lo cual dañará a la fuente de alimentación.
- 3) Para evitar que la tensión inducida en el circuito dañe la fuente o los demás dispositivos electrónicos, se agrega el **diodo de marcha libre** o **diodo en antiparalelo**, el cual es un simple diodo, conectado en paralelo al motor y puesto en sentido contrario a la alimentación.
  - a. **Cuando el circuito esté encendido**: El diodo de marcha libre no hará nada.
  - b. **Cuando el circuito se apague**: Se creará la **fem** (**tensión en sentido contrario**) y es ahí cuando el **diodo de marcha libre** permitirá que esa tensión fluya a través de él, dejando que descargue su corriente sin dañar a los demás dispositivos electrónicos.



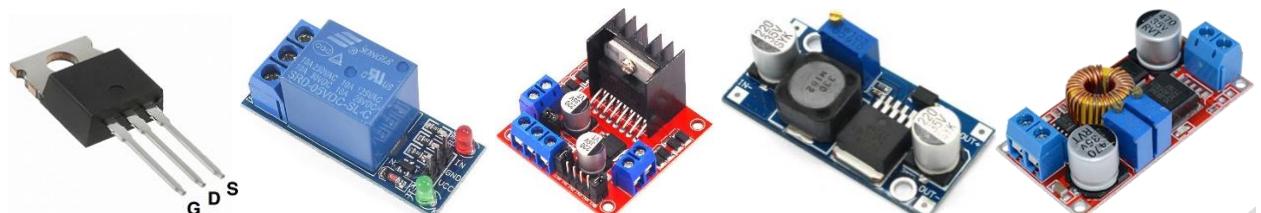
## Etapa de Control y Etapa de Potencia en un Circuito con Motores

### Etapa de Potencia: Corriente Demandada por el Motor al Agregar una Carga

Cuando un motor se enfrenta a una **carga** (fuerza o peso que se opone a su giro normal), debe generar suficiente torque para intentar superarla. El **torque** que ejerce el motor en su eje **está relacionado con la corriente que fluye a través de sus bobinas**. A medida que aumenta la carga, el motor necesita generar más **torque**, por lo que requiere un aumento en su **corriente** para superar la fuerza de oposición ocasionada por la **carga** y mantener su velocidad.

Por esta cuestión es que el circuito se divide en dos partes, una llamada **etapa de potencia** y otra llamada **etapa de lógica o control**:

- **Etapa de Potencia:** En esta etapa se manejan **tensiones o corrientes grandes** que pueden ser cambiantes o no, como la ocasionada en un motor cuando se le agrega una **carga** que se opone a su rotación.
  - **La etapa de potencia generalmente involucra componentes como:**
    - Transistores MOSFET, relevadores (relés), controladores de motor (drivers), puentes H, convertidores de CD a CD boost y/o buck, etc.



- Motores CD con escobillas sencillas, motorreductores, servomotores, motores sin escobillas (brushless o BLDC), motores a pasos, motores AC, etc.



- **Etapa de Control:** Esta etapa maneja **tensiones y corrientes pequeñas**, es la parte del circuito que controla la **etapa de potencia**.
  - La etapa de control generalmente involucra componentes como:
    - Microcontroladores, FPGA, CPU (Raspberry Pi), sensores, circuitos lógicos, transistores BJT, amplificadores operacionales y otros dispositivos que operan a niveles de potencia más bajos o se centran en el procesamiento y la toma de decisiones.



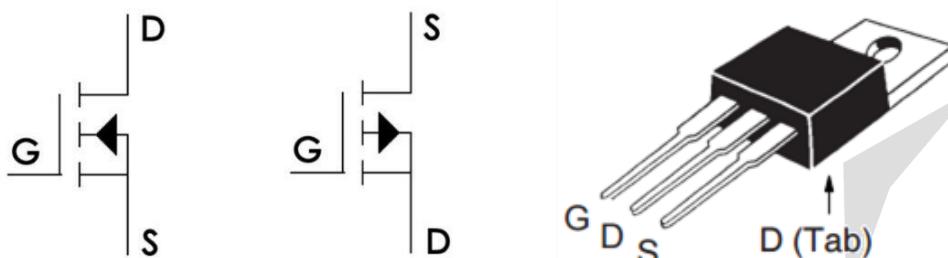
En pocas palabras, la etapa de control es el cerebro del circuito y la etapa de potencia es su músculo.

### Etapa de Control: Aislamiento de los Circuitos Lógicos

El objetivo principal de separar el circuito en una **etapa de control** y otra de **potencia** es que no se dañen los componentes electrónicos que no manejan altos niveles de tensión y corriente, para ello se utilizan ciertos dispositivos electrónicos que aislan la etapa de control, separándola de la etapa de potencia, **incluso se pueden utilizar fuentes de alimentación distintas en cada etapa**. Se suelen usar los 3 siguientes componentes electrónicos para realizar el aislamiento del circuito:

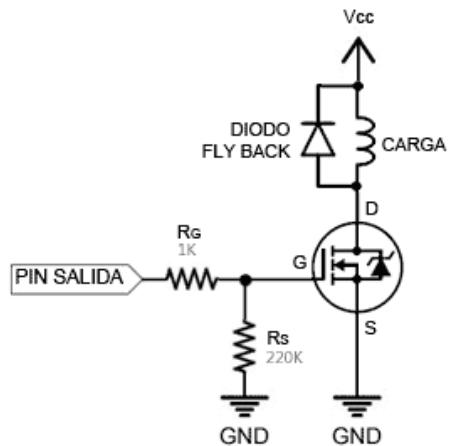
- **Transistor MOSFET:** Es un dispositivo electrónico que puede utilizarse en una configuración de **amplificador o interruptor** para separar la etapa de potencia y la etapa lógica.
  - **Amplificador:** Los transistores **pueden amplificar señales de control de bajo voltaje y corriente provenientes de la etapa lógica para controlar dispositivos electrónicos que consumen mayor tensión y corriente en la etapa de potencia**.
  - **Interruptor:** Los transistores **pueden usarse como interruptores para encender o apagar la fuente u otros dispositivos electrónicos de la etapa de potencia por medio de una señal proveniente de la etapa lógica**.
    - Se utilizan las diferentes configuraciones del transistor MOSFET, llamadas canal N o P, según la polaridad (dirección) de la tensión en el circuito.

MOSFET Canal N    MOSFET Canal P

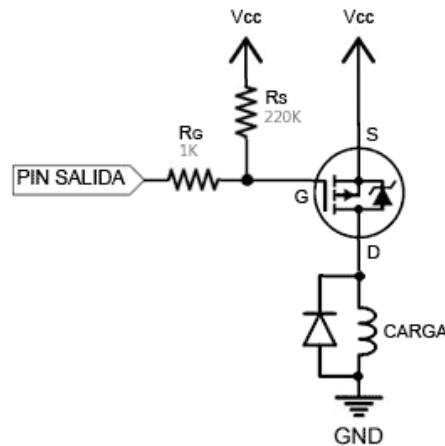


- A continuación, se muestra un ejemplo de conexión que aísla ambas etapas a través de un **transistor MOSFET**:

CANAL - N



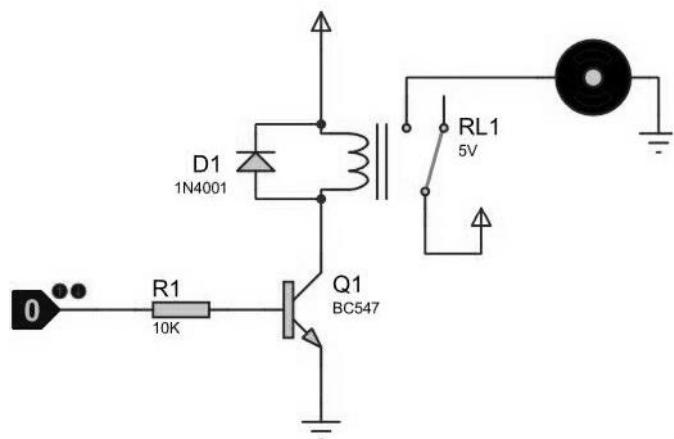
CANAL - P



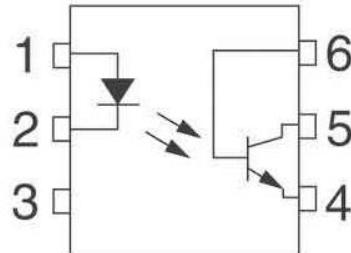
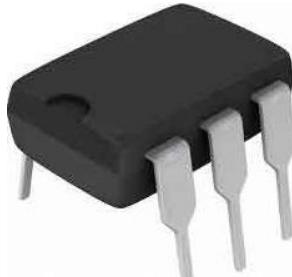
- **Relevador:** Un relevador o relé es un interruptor electromagnético que utiliza un electroimán interno para conectar o desconectar dos contactos mecánicos.
  - Los relés proporcionan un aislamiento eléctrico entre la **etapa de potencia** y la **etapa de control**, ya que no hay conexión eléctrica directa entre ellas.
  - Se pueden usar relevadores de forma individual o adquirir módulos que tengan varios.



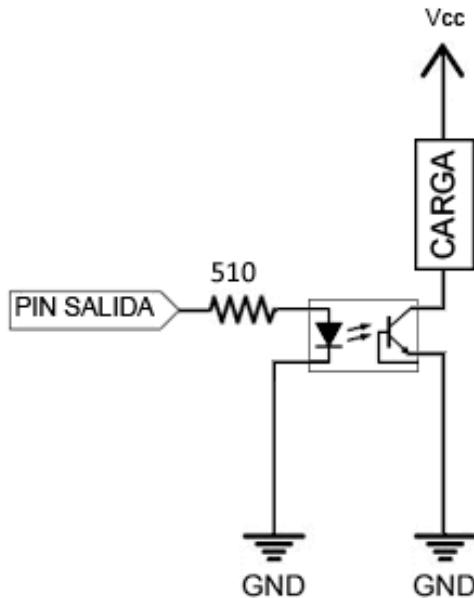
- A continuación, se muestra un ejemplo de conexión que aísla ambas etapas a través de un **relevador o relé**:



- **Optoacoplador (optoisolador):** Es un dispositivo electrónico que utiliza una combinación de un **emisor de luz** (generalmente un diodo emisor de luz o LED) y un **receptor de luz** (generalmente un fototransistor) para proporcionar un aislamiento eléctrico entre la **etapa de potencia** y la **etapa lógica**.
  - El optoacoplador proporciona un acoplamiento óptico; y **es el mejor aislamiento que se le puede dar a la etapa de control**, ya que la protege de interferencias electromagnéticas, sobretensiones, ruidos y otros problemas que podrían generarse en la **etapa de potencia**.

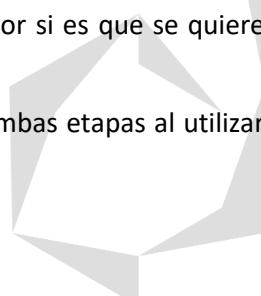


- A continuación, se muestra un ejemplo de conexión que aísla ambas etapas a través de un **optoacoplador**:



En los ejemplos posteriores donde se explicará el funcionamiento y control de los motores no se separa el circuito en una **etapa de potencia** y **etapa lógica**, esto se realiza por simplicidad en la demostración, pero ya en aplicaciones reales si se debe realizar el aislamiento de ambas partes del circuito, **si no se realiza la separación, cuando el motor demande más potencia, dañará al microcontrolador, FPGA o Raspberry Pi** e incluso podría dañar el puerto serial que conecta la tarjeta de desarrollo con la computadora, por ello es importante también no colocarle ninguna carga al motor si es que se quiere probar su funcionamiento conectando el circuito directamente.

En este caso en particular del motor BLDC si se está haciendo la separación de ambas etapas al utilizar una **batería LiPo** para su alimentación.



# Baterías LiPo (Litio Polímero)

## Datos Importantes de Uso: Voltaje Nominal, Capacidad y Tasa de Descarga

Las baterías LiPo (Litio Polímero) son muy utilizadas en proyectos de robótica debido a que son livianas y poseen gran densidad energética, por lo que se les puede exigir una cantidad alta de corriente. Sin embargo, son muy peligrosas, por lo que es importante conocer su funcionamiento y cuidados antes de utilizarse, sus aspectos más importantes son:

- **Voltaje Nominal:** El término nominal se refiere a un valor de referencia, lo que implica que en realidad el voltaje no será de un valor fijo, sino que oscilará entre un valor mínimo y máximo alrededor de ese voltaje nominal, dependiendo de si la carga de la batería está en su nivel máximo o mínimo.
  - Una sola batería LiPo cuenta con un **voltaje nominal** de aproximadamente **3.7V**.
    - Su **tensión máxima** es de **4.2V**.
    - Su **tensión mínima por seguridad** es de **3.3V**.
  - Las baterías LiPo pueden estar compuestas de varias **celdas, denotadas por la letra S**, indicando que varias de ellas están conectadas en serie para aumentar su tensión.
    - Una batería de **2S** entrega:  $V = 3.7 * \#Celdas = 3.7 * 2 = 7.2V$

Las fuentes de alimentación son dispositivos que proveen a un circuito electrónico con tensión y corriente (como las baterías LiPo), estas se pueden conectar entre sí de 2 formas, en serie o paralelo, proporcionando las siguientes características cada una:

- **Conexión en serie:** Se suma la tensión entregada y la corriente se iguala a la más baja de todas las fuentes conectadas en serie.

$$V_{Total} = V1 + V2 + \dots + V_n$$

$I = Constante = La menor corriente entradada por las fuentes$

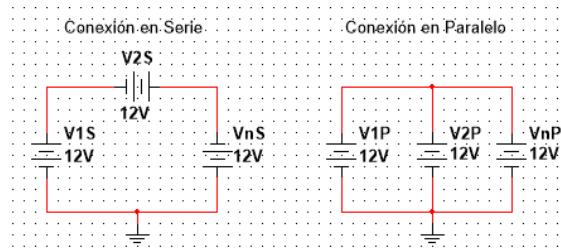
$$R_{Total} = R1 + R2 + \dots + R_n$$

- **Conexión en paralelo:** Se suma la corriente entregada y la tensión se iguala a la más baja de todas las fuentes conectadas en paralelo.

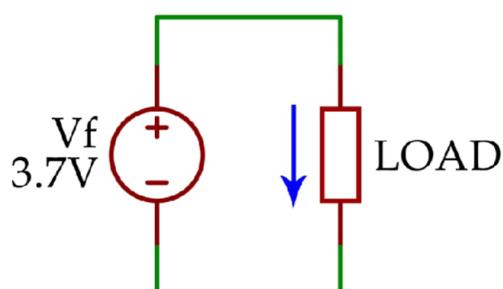
$V = Constante = La menor tensión entradada por las fuentes$

$$I_{Total} = I1 + I2 + \dots + I_n$$

$$R_{Total} = \frac{1}{\frac{1}{R1} + \frac{1}{R2} + \dots + \frac{1}{R_n}}$$

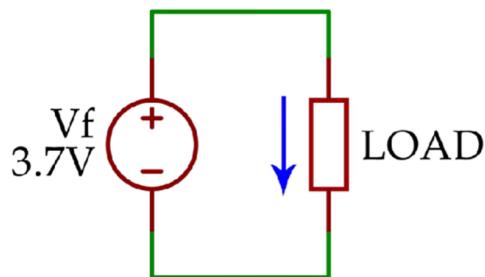


- **Capacidad de la batería:** Es una medida dada en miliamperes hora [mAh], la cual indica cuanta corriente puede entregar la batería en 1 hora.
  - Por ejemplo, si en la batería se indica que puede dar 3000 mAh, esto significa que, si conectamos una carga que exige una corriente constante de 3A y la batería está totalmente cargada, en 1 hora se habrá descargado completamente.



$$3A=1h$$

- Si ahora con la misma batería conecto una carga que no demanda 3A, sino 1.5A, la batería durará 2 horas y así consecutivamente.



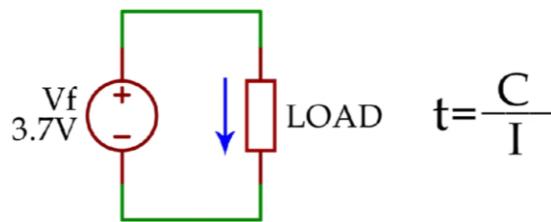
$$3A=1h \quad 1.5A=2h \quad 6A=0.5h$$

- Por lo que se puede afirmar que, **la capacidad de la batería no implica que ésta forzosamente entrega cierta cantidad de corriente**, sino que **indica cuanto puede durar entregando cierta corriente en un lapso de tiempo**, pero **la corriente que se consume depende totalmente de la carga** (lo que se conecte a la batería), esto se realiza siguiendo la ecuación descrita a continuación:

$t = \text{Tiempo de duración de la batería [horas]}$

$C = \text{Capacidad de la batería [mAh]}$

$I = \text{Corriente que demanda la carga [mA]}$



Los mAh entregados por la **capacidad de la batería no afectan al circuito**, pero la **tensión nominal** sí.

- **Tasa de descarga:** También conocida como **discharge rate**, es una cantidad que indica cuanta corriente máxima se le puede exigir a la batería sin que esta explote. **La tasa de descarga se indica con un número seguido de una C en las indicaciones de la batería** y se usa para manejar picos de corriente solicitados por motores o para demandar mucha corriente a baterías pequeñas, como se hace en drones para reducir su peso, aunque esto implica que se puede usar menos tiempo.
  - Para obtener la corriente máxima que se le puede exigir a una batería se multiplica la tasa de descarga por su capacidad, pero en su unidad solo se considera la corriente, no la hora.

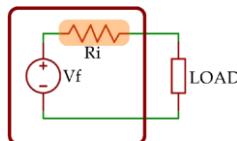
$$\text{Corriente Máxima} = \frac{\text{Capacidad [mAh]}}{1h} * \text{Tasa de Descarga [adimensional]}$$

- Por ejemplo, si en una batería con **capacidad** de **3000 mAh** se tiene una **tasa de descarga** de **8C**, entonces la **corriente máxima** que puede entregar es de 24000 mA, pero como se explicó anteriormente, esto reduce el tiempo de duración de la batería.

$$\text{Corriente Máxima} = \frac{3000 \text{ [mAh]}}{1h} * 8 \text{ [adimensional]} = 24,000 \text{ [mA]}$$

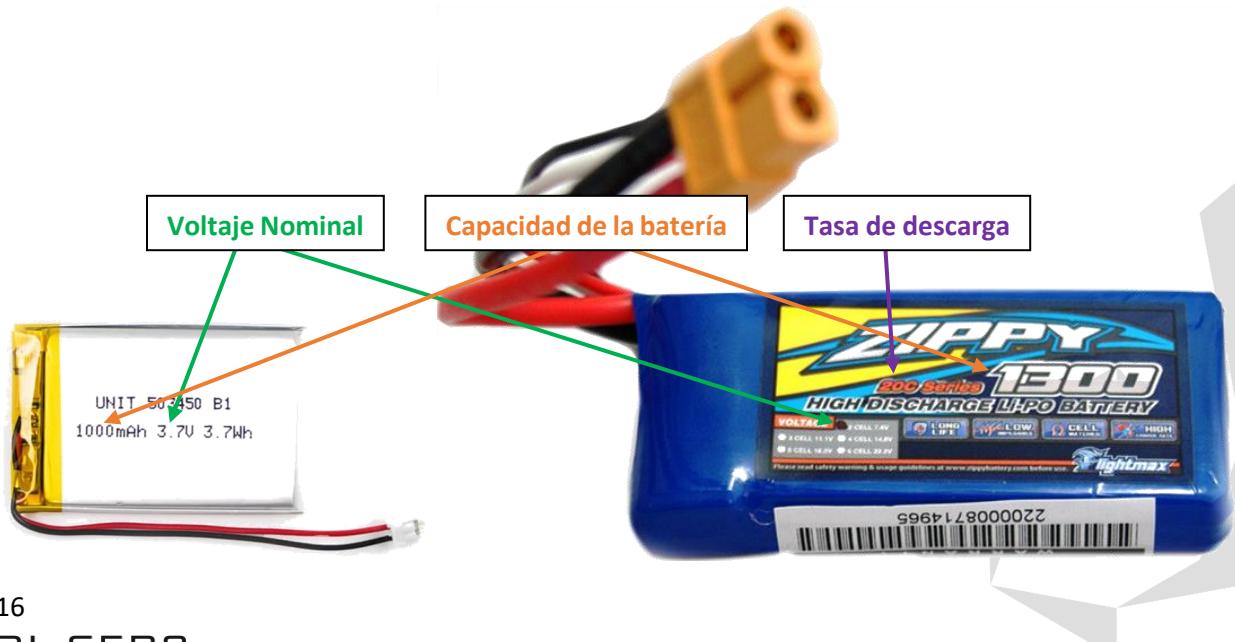
$$t = \frac{C}{I} = \frac{3000 \text{ [mAh]}}{24,000 \text{ [mA]}} = 0.125 \text{ [h]} = 0.125 \text{ [h]} \left( \frac{60 \text{ [min]}}{1 \text{ [h]}} \right) = 7.5 \text{ [min]}$$

- Es muy importante mencionar que, **aunque en teoría la tasa de descarga indica la corriente máxima que puede entregar una batería, no se debe tomar como una absoluta realidad**, ya que la misma batería tiene una resistencia interna y si se le exige una corriente muy grande, esa resistencia se calentará, **causando que la batería explote**.



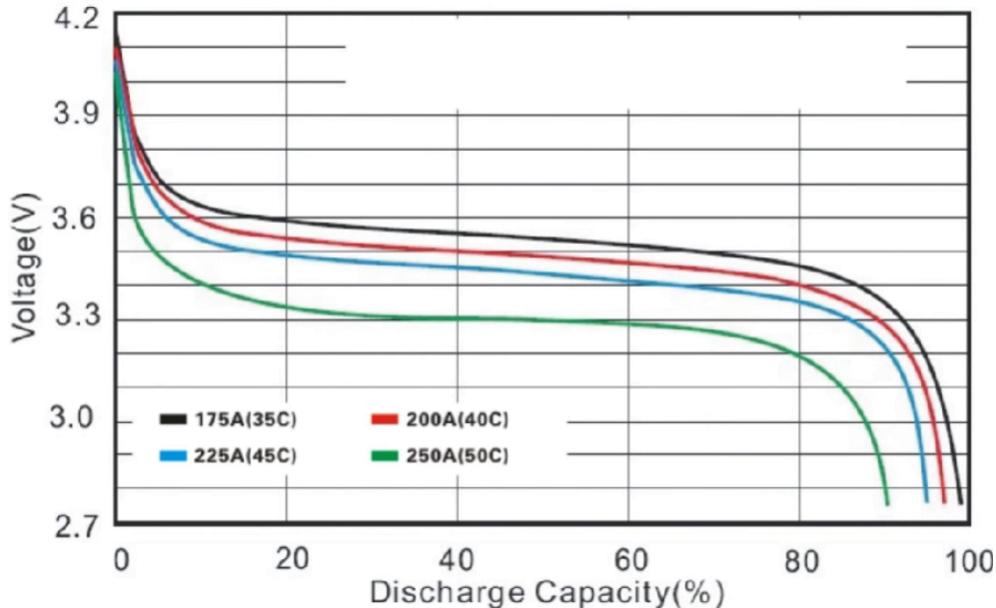
**Diagrama - Baterías LiPo y sus Datos Importantes de Uso: Voltaje Nominal, Capacidad y Tasa**

*En las baterías LiPo pequeñas, el número que viene en el renglón superior indica sus dimensiones y siempre son de 1S.*



## Carga y Descarga de las Baterías LiPo: Curvas de Descarga y Tasa de Carga

- **Curva de descarga:** Mencionamos previamente que una sola batería LiPo cuenta con una **tensión nominal de 3.7V**, una **tensión mínima de 3.3V** y **máxima de 4.2V**, pero ésta no se descarga de forma continua, sino que lo hace con un comportamiento descrito por una gráfica llamada **curva de descarga**, donde se ve el cambio de la tensión entregada por la batería, dependiendo de su nivel de carga y la corriente que se le esté demandando (**Capacidad de la batería \* Tasa de descarga**).
  - Al **inicio del funcionamiento de la batería**, la tensión baja drásticamente **de 4.2 a 3.6V** aproximadamente, **luego se mantiene** con una tensión que baja poco a poco **de 3.6 a 3.3V** y **finalmente** la tensión baja agresivamente **de 3.3V a 2.8V**.
    - En la gráfica se tienen diferentes curvas de descarga, que dependen del valor de la **tasa de descarga (C)** que se esté utilizando.
    - Es recomendable **no dejar que las baterías se descarguen a menos de 3.3V**, ya que si su tensión baja de **2.8V** se dañará y no se podrá volver a cargar, quedando inutilizable.



- **Curva de carga:** Los cargadores de baterías LiPo utilizan un método de carga llamado **CC/CV (Constant Current/Constant Voltage)**, este consiste en **primero aplicar una corriente constante de carga** con una tensión que suba poco a poco hasta alcanzar la **tensión máxima de 4.2V** y **luego aplicar esa tensión de forma constante** con una corriente que baje poco a poco hasta alcanzar un **valor llamado corriente de corte**, donde la batería ya se habrá cargado completamente.
  - Si la batería se carga con más de **4.2V** es muy probable que **explote**, por lo cual no es recomendable irse cuando se está cargando una batería Lipo, esta debe ser monitoreada, también debemos contar con un extintor cerca y de preferencia hacerlo al aire libre.

Tanto la **corriente constante de carga** como la **corriente de corte** se obtienen de la siguiente forma:

- La **corriente constante de carga** se calcula con un dato dado en el datasheet de la batería LiPo, donde al igual que antes se tenía la **tasa de descarga (C)**, ahora se tendrá la **tasa de**

carga (**C**) o **charge rate**, que nuevamente es un número que se encuentra alado de la letra **C** y se multiplica por la capacidad de la batería para calcular la **corriente de carga**.

- Por ejemplo, si en una batería con **capacidad** de **3000 mAh** se tiene una **tasa de carga** de **2C**, entonces la **corriente constante** de carga es de  $6000 \text{ mA} = 6 \text{ A}$ .

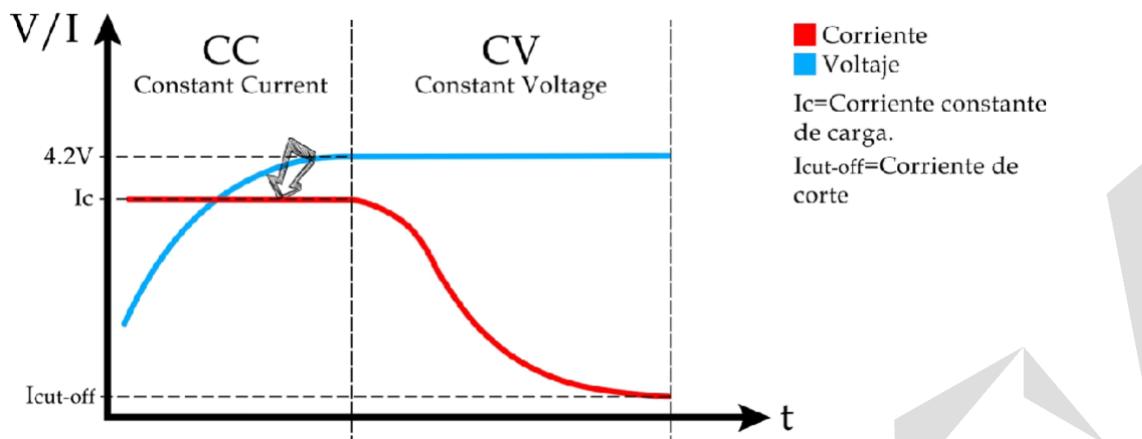
$$\text{Corriente de Carga} = \frac{\text{Capacidad [mAh]}}{1\text{h}} * \text{Tasa de Carga [adimensional]}$$

$$\text{Corriente de Carga} = \frac{3000 \text{ [mAh]}}{1\text{h}} * 2 \text{ [adimensional]} = 6000 \text{ [mA]} = 6 \text{ [A]}$$

- La **corriente constante de carga** obtenida después de realizar el cálculo **se debe considerar como la corriente máxima con la que se puede cargar la batería de forma más rápida**, pero **es recomendable cargar la batería con una corriente menor**, ya que, **aunque se cargue más lentamente, será más seguro realizarlo de esta forma**.
- Si no se puede encontrar la **tasa de carga** en el datasheet, se considera de **1C**, porque si cargamos la batería con una **corriente de carga** mayor a la que admite, **esta explotará**.
  - Cuando se elija la tasa de **1C**, normalmente se llegará al 100% de carga en un tiempo de espera de entre 1:00 y 1:30.
- La **corriente de corte** también llamada **cut-off current**, indica cuando la batería LiPo se ha cargado completamente, ésta se muestra en el datasheet o simplemente se calcula obteniendo el 10% de la **Capacidad de la batería**:
  - Por ejemplo, en una batería con **capacidad** de **3000 mAh** la **corriente de corte** es de **300 mA**, cuando el cargador alcance esta corriente se considera que la batería LiPo está 100% cargada.

$$\text{Corriente de Corte} = \text{Cut - off Current} = 0.1 * \frac{\text{Capacidad [mAh]}}{1\text{h}}$$

$$\text{Corriente de Corte} = 0.1 * \frac{3000 \text{ [mAh]}}{1\text{h}} = 300 \text{ [mA]}$$



Este proceso lo realiza el cargador de baterías, el más recomendado es el **IMAX B6AC**, aunque también existe un módulo llamado **TP4056**, pero este es difícil de configurar al cargar baterías distintas.

## Cargador de Baterías LiPo IMAX B6AC

**Número de celdas:** Recordemos que las baterías LiPo pueden estar compuestas de varias **celdas** denotadas por la letra **S** por su conexión en serie, indicando el **número de unidades de 3.7V** que están conectadas para aumentar la tensión final entregada, pero manteniendo una corriente constante:

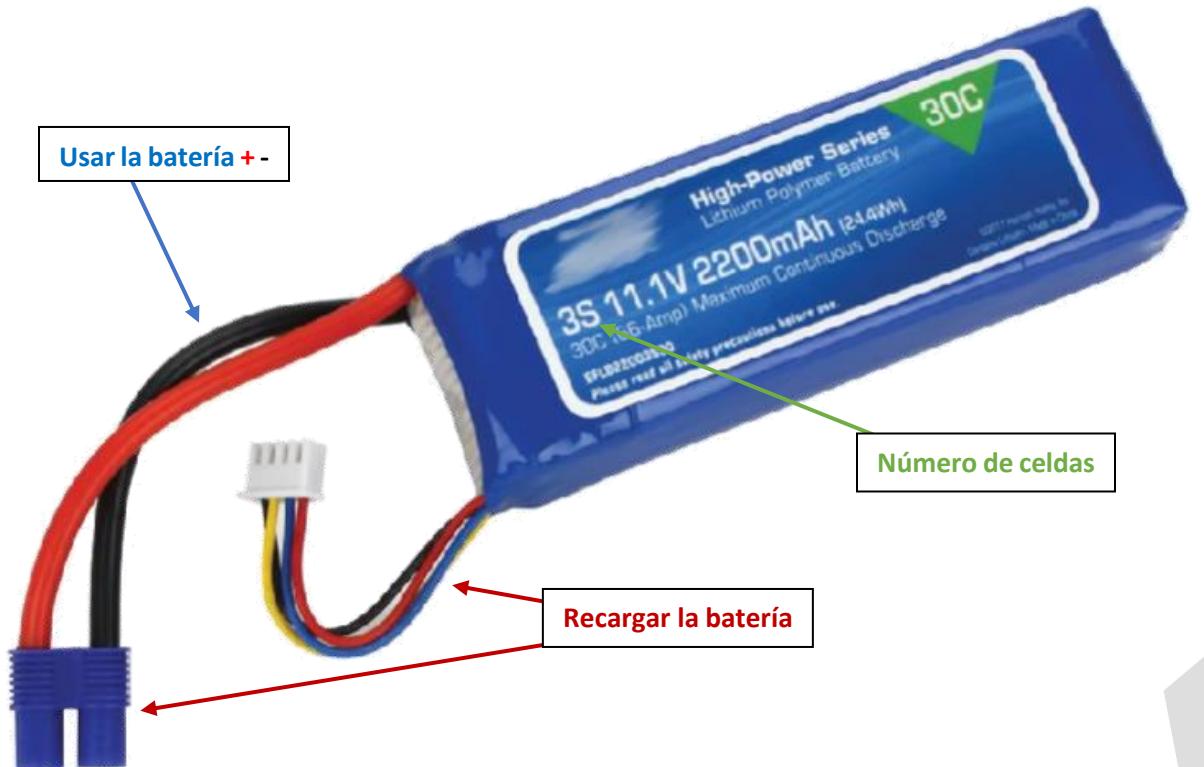
- **Conexión en serie:** Se suma la tensión entregada y la corriente se iguala a la más baja de todas las fuentes conectadas en serie.

$$V_{Total} = V1 + V2 + \dots + V_n$$

*I = Constante = La menor corriente entradada por las fuentes*

Las baterías LiPo cuentan con 2 conjuntos de cables, sus usos se describen a continuación:

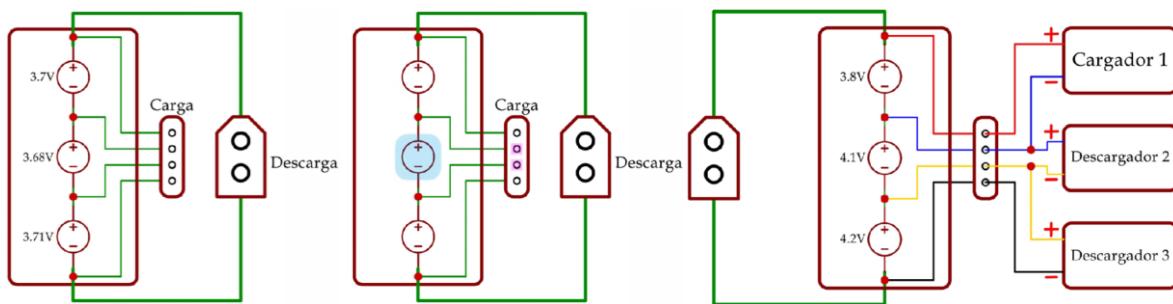
- **Usar la batería:** El conjunto de cables más gruesos (**rojo positivo + y negro negativo -**) llamado **conector de descarga** sirve para entregar corriente y tensión al dispositivo electrónico que se quiera alimentar.
- **Recargar la Batería:** El conjunto de cables más delgados llamado **conector de balanceo** y los dos gruesos previamente mencionados sirven para cargar la batería en un modo de balanceo.



La importancia del modo de **balanceo** en las baterías LiPo es que, **para que la capacidad de la batería no se vea reducida**, todas las **celdas** que se encuentran en su interior deben tener niveles de tensión iguales (o muy parecidos), porque si alguna de ellas no se encuentra cargada totalmente o cuenta con una tensión distinta a las demás, **debido a la conexión en serie** de las **celdas** dentro de la batería LiPo, **la corriente entregada será menor a la indicada en el empaque**, ya que se entregará la **capacidad** de la **celda** menos cargada.

A continuación, se muestra el diagrama esquemático de una batería **3S**, demostrando el propósito del conjunto de cables más gruesos llamado **conector de descarga (descarga)** y el conjunto de los cables más delgados llamado **conector de balanceo (carga)**, que gestiona la carga de cada **celda** individualmente:

- **Conector de descarga:** Indica la tensión total entregada, que es la multiplicación del **número de celdas** por la **tensión** de una sola de ellas, el valor de dicha tensión depende de su nivel de carga.
- **Conector de balanceo:** Todos sus cables están conectados a las terminales positiva y negativa de cada una de las **celdas**, si se mide la tensión entre dos cables seguidos, se obtendrá la tensión de una sola de las **celdas**. Todas ellas deben tener el mismo voltaje para que la corriente entregada por la batería no se vea reducida, por lo que el cargador se encarga de medir sus niveles de tensión y asegurarse que sean iguales, cargándolas y descargándolas de forma individual.



El cargador **IMAX B6AC** puede alimentar **baterías LiPo de 2 a 6 celdas** realizando su **balanceo**, en su lado derecho cuenta con dos puertos de conexión para realizar su recarga:

- En las conexiones circulares grandes roja y negra se conecta el conjunto de cables más gruesos (**rojo positivo + y negro negativo -**) del **conector de descarga**.
- En los conectores blancos que tiene alado (llamados conectores JST) se introduce el conjunto de cables más delgados del **conector de balanceo** que admite **baterías LiPo de 2 a 6 celdas**.

En su lado izquierdo, el **cargador IMAX B6AC** se puede alimentar directamente en la **toma de corriente de la pared (100 a 240VAC)** o se puede conectar a una **fuente de alimentación DC de 11 a 18V/1.5A**, función que resulta muy útil cuando se quiere recargar una batería en exteriores, dando la posibilidad de, por ejemplo, recargarla con el cargador del coche.



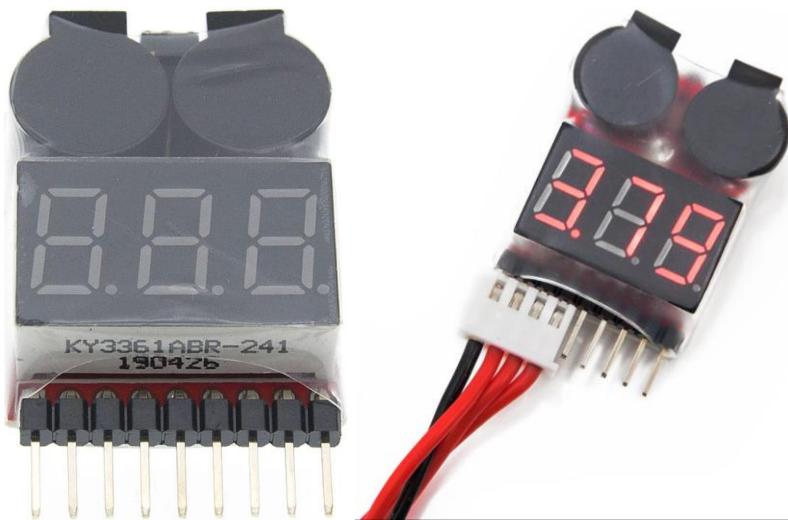
## Proceso de Carga de una Batería LiPo con el Cargador IMAXB6AC

### Medidor de Tensión para Baterías LiPo

Este dispositivo electrónico sirve para indicar la tensión de cada una de las **celdas** contenidas en una **batería LiPo**, puede medir las tensiones **de hasta 8 celdas (8S)** y se conecta al **conjunto de cables más delgados (conector de balanceo)**; sus pines significan lo siguiente:

- **Pin 1 (de hasta la izquierda):** A este pin se conecta el **cable negro** del **conector de balanceo**, que corresponde a la **terminal negativa de tierra (-)**.
- **Pin 2:** Se conecta al positivo (+) de la **celda 1** para medir su tensión.
- **Pin 3:** Se conecta al positivo (+) de la **celda 2** para medir su tensión.
- **Pin 4:** Se conecta al positivo (+) de la **celda 3** para medir su tensión.
- **Pin 5:** Se conecta al positivo (+) de la **celda 4** para medir su tensión.
- **Pin 6:** Se conecta al positivo (+) de la **celda 5** para medir su tensión.
- **Pin 7:** Se conecta al positivo (+) de la **celda 6** para medir su tensión.
- **Pin 8:** Se conecta al positivo (+) de la **celda 7** para medir su tensión.
- **Pin 9:** Se conecta al positivo (+) de la **celda 8** para medir su tensión.

Obviamente si la batería no tiene las **8 celdas (8S)**, algunos de los pines del medidor de voltaje quedan sin conectarse a nada.



Cuando el lector de tensión se conecte al **conjunto de cables más delgados de la batería LiPo (conector de balanceo)** hará un sonido su buzzer y posteriormente se ejecutará una secuencia, donde se mostrará la siguiente información en los 3 displays de 7 segmentos:

1. **Número de celdas en la batería:** Primero se muestra el número de **celdas** con las que cuenta la batería que se está analizando en el siguiente formato: #celdas C E.
2. **Luego se muestra un letrero con la palabra ALL en los 3 displays de 7 segmentos.**
3. **Voltaje total entregado por la batería:** Despues se indica la **tensión total** que entrega la batería en su conjunto de cables más gruesos (**rojo positivo + y negro negativo -**) llamado **conector de descarga**, este voltaje será el resultado de la suma de las tensiones entregadas por cada **celda**, **debido a su conexión en serie**.

- a. Aquí es importante tomar en cuenta que si una sola **celda (1S)** puede entregar una tensión máxima de **4.2V**, una mínima de **3.3V** y una nominal de **3.7V**, en una batería por ejemplo de **2S**, todas esas tensiones son multiplicadas por el número de **celdas**, por lo tanto, la tensión máxima sería de  **$4.2 * 2 = 8.4V$** , la mínima sería de  **$3.3 * 2 = 6.6V$**  y la nominal de  **$3.7 * 2 = 7.2V$** .
4. **Celda a la que se le medirá la tensión:** Luego se muestra un letrero que indica cuál es el número de la celda a la cual se le leerá su tensión: **n o #celda**.
5. **Tensión individual de cada celda:** Esta es la parte importante, ya que **la tensión de todas las celdas debe ser muy parecida** para que se haya cumplido el objetivo de **balancearlas** al cargar la batería, pudiendo diferir solo por decimales, y además se deben cumplir las siguientes características, dependiendo de si la batería se utilizará o almacenará:
  - a. **Uso de Batería LiPo:** Cuando se vaya a utilizar una batería, esta **deberá estar 100% cargada**, por lo que **todas sus celdas deben contar con un mismo voltaje**, que deberá ser **la tensión máxima**, osea **4.2V**.
  - b. **Almacenamiento de Batería LiPo:** Cuando se vaya a almacenar una batería, **todas sus celdas deben contar con un mismo voltaje**, que deberá ser la tensión nominal, osea **3.7V**.
    - i. Si una batería se almacena con su tensión máxima de **4.2V** o mínima de **3.3V**, esta se puede inflar o **explorar**.
6. **Al terminar de mostrar la tensión individual de cada una de las celdas, se volverá a mostrar el mismo letrero con la palabra ALL en los 3 displays de 7 segmentos y se repetirá la secuencia desde el paso 2.**

Este es el primer paso que se debe realizar antes de siquiera pensar en cargar una batería LiPo, para así asegurarnos de su nivel de carga, ya que, **si su tensión es menor a 2.8V, corremos el riesgo de que, al intentarla cargar, esta explote**.

### Cuidados de una Batería LiPo

Las baterías LiPo pueden ser muy frágiles, por lo cual:

- No deben ser picadas con nada.
- Debemos tener cuidado de no causar un corto circuito entre sus **conectores de descarga**.
- No deben ser utilizadas en entornos sometidos a grandes temperaturas.
- Se deben dejar de alimentar cuando su carga se haya completado al 100%, osea cuando se llegue a la **corriente de corte**.
- Se deben meter en cajas sólidas para protegerlas cuando se introduzcan en proyectos que serán de uso rudo.

*Si no se toman en cuenta estos cuidados, las baterías se pueden inflar y quedar inutilizables o explotar.*

**Cuando una batería LiPo sea recargada, transportada o almacenada**, cada una de ellas debe ser introducida en una **bolsa ignífuga** diferente, que las aísla y protege, ya que, si alguna de ellas llegara a explotar y se encuentra al lado de otra, se creará una acción en cadena que las hará explotar a todas.

- Esto no significa que sean las **bolsas ignífugas** totalmente seguras, ya que el fuego de la explosión se puede escapar por sus aberturas, por eso también es mejor separarlas una de otra cuando se almacenen.

- Siempre debemos tomar en cuenta que **cuando se almacene una batería, esta no puede estar totalmente cargada (4.2V) o totalmente descargada (3.3V)**, sino que **se debe encontrar en su tensión nominal (3.7V)**. Si esto no se cumple, la batería puede inflarse o hasta **explosionar**.
- Además, esa condición de que se debe tener una tensión máxima de **4.2V**, una mínima de **3.3V** y una nominal de **3.7V** es para las baterías de 1 **celda (1S)**, pero si se tiene por ejemplo una batería de tres **celdas (3S)**, todas esas tensiones son multiplicadas por el número de **celdas**, por lo tanto, la tensión máxima sería de  $4.2 * 3 = 12.6V$ , la mínima sería de  $3.3 * 3 = 9.9V$  y la nominal de  $3.7 * 3 = 11.1V$ .
- Es una buena práctica que, durante el proceso de carga una batería LiPo, esta se meta en una **bolsa ignífuga**, porque si llegara a ocurrir algún accidente, se reduce la destrucción que podrían ocasionar al **explosionar**. Para ello se debe conseguir una extensión del **conector de balanceo**, ya que el de la batería siempre es muy corto.



Ya que se haya comprobado que **el nivel de tensión de cada celda de la batería esté arriba de 2.8V** y se haya introducido en una bolsa ignífuga **para prevenir su explosión**, se podrá realizar su proceso de carga:



La secuencia que se debe ejecutar con los 4 botones del cargador IMAXB6AC para configurar la recarga de una batería LiPo es la siguiente:

- **Batt. type:** Al encender el cargador se debe dar clic en el botón de **Batt. type** para elegir el tipo de batería que se quiere alimentar, ya que el **IMAXB6AC** puede cargar baterías LiPo, LiFe, Lilon y LiHV. Una vez elegido el tipo, se debe dar clic en el botón de **Start**.
  - **LiPo Batt:** Esta opción permite alimentar o descargar baterías LiPo a través de las diferentes formas enlistadas a continuación, las cuales se visualizan al dar clic a los botones **Dec.** e **Inc.**:
    - **LiPo CHARGE:** Permite cargar una batería **de forma directa**, donde **NO se gestiona la carga de cada celda individualmente**, sino que **sólo se administra tensión y corriente** a través de su **conector grueso de descarga**.
      - **Este modo sólo se utiliza cuando se quiere cargar baterías LiPo pequeñas de 1S para drones**, pero como estas no tienen **conector de balanceo**, se necesita de dos adaptadores para poder conectar sus cables delgados a los **conectores de descarga** del **cargador IMAXB6AC**.
    - **LiPo BALANCE:** Permite cargar la batería en **modo de balanceo**, el cual gestiona la carga de cada **celda** individualmente para que todas tengan el mismo nivel de tensión, que será el máximo de **4.2V** cuando se quiera recargar la batería al 100%.
      - **Este modo se utiliza cuando se quiere cargar baterías LiPo de 2, 3, 4, 5 y 6S**, conectando sus **conectores de balanceo** y **descarga** al **cargador IMAXB6AC** a través de un adaptador.
    - **LiPo STORAGE:** Permite cargar y descargar la batería en **modo de almacenamiento**, el cual, no importando la tensión individual de cada **celda**, las descargará y/o cargará continuamente hasta que todas tengan el mismo nivel de tensión, que será el nominal de **3.7V** por **celda**.
  - Una vez elegido un **modo de carga o descarga**, se da clic en el botón de **Start** para indicar **la corriente** de la batería y después **otra vez se da clic en el mismo botón** para ingresar **la tensión correspondiente a las características de la batería y el modo elegido**:
    - **Corriente de carga:** Esta se indica en Amperes y se calcula al multiplicar la **capacidad** de la batería por su **tasa de carga**, donde si no se tiene ese dato, se considera de **1S**, por lo cual se cargará la batería con la misma corriente de su capacidad.
      - El **nivel de la corriente** se puede subir con el botón **Inc.**, bajar con el botón **Dec.** y elegir un valor al dar clic en el botón de **Start**.
    - **Número de celdas de la batería:** No se indica la tensión de carga porque **al elegir un número de celdas, esta se asigna automáticamente** por el **cargador**.
      - El **número de celdas** se puede subir con el botón **Inc.**, bajar con el botón **Dec.** y elegir un valor con el botón de **Start**.
  - En esta parte de la configuración es donde **se conecta la batería al cargador IMAXB6AC**, hay que tener mucho cuidado con la polaridad de los cables gruesos (**rojo positivo + y negro negativo -**) del **conector de descarga**, ya que, si estos se conectan al revés, se occasionaría un cortocircuito, dañando permanentemente al cargador.
    - Una vez seleccionado el modo de carga/descarga, corriente y número de celdas, se debe mantener presionado el botón de **Start** para comenzar la recarga de la batería LiPo.
      - Al haber iniciado la carga se mostrará en la pantalla del cargador:

Li#Celdas	Corriente	Tensión
ModoCarga/Descarga	Temporizador	00000
○ <b>Li#Celdas:</b>		
<ul style="list-style-type: none"> <li>▪ <b>Tipo de Batería:</b> <ul style="list-style-type: none"> <li>• Li: Batería Lipo.</li> </ul> </li> <li>▪ <b>Número de Celdas:</b> <ul style="list-style-type: none"> <li>• 1S: Batería de 1 celda.</li> <li>• nS: Batería de <math>n = 2, 3, 4, 5</math> o máximo 6 celdas.</li> </ul> </li> </ul>		
○ <b>Modo de Carga/Descarga:</b>		
<ul style="list-style-type: none"> <li>▪ <b>CHG:</b> Modo LiPo CHARGE de carga directa (1S).</li> <li>▪ <b>BAL:</b> Modo LiPo BALANCE con carga de balanceo.           <ul style="list-style-type: none"> <li>• <b>Corriente:</b> En los modos CHG y BAL la corriente entregada es resultado de la multiplicación de la capacidad de la batería por su tasa de carga, aunque siempre es más seguro alimentar con 1C, porque si se alimenta la batería con mucha corriente, esta puede explotar.</li> <li>• <b>Tensión:</b> En los modos CHG y BAL la tensión proporcionada en teoría es la máxima de 4.2V entregada a cada celda, pero en el IMAXB6AC esto no se indica directamente, sino que es proporcionado de forma automática por el cargador, nosotros solo indicamos el número de celdas de la batería (2, 3, 4, 5 y 6S), el cual se puede conocer al contar los cables del conector de balanceo y restarle 1.</li> </ul> </li> <li>▪ <b>STO:</b> Modo LiPo STORAGE de almacenamiento.           <ul style="list-style-type: none"> <li>• <b>Corriente:</b> En el modo STO la corriente entregada debe ser del 75 al 100% de la capacidad de la batería, pudiendo como máximo entregar 1A.</li> <li>• <b>Tensión:</b> En el modo STO las celdas de la batería se cargan y descargan individualmente para que todas lleguen a la tensión nominal de 3.7V, en el cargador esto no se señala directamente, solo se ingresa el número de celdas de la batería.</li> </ul> </li> </ul>		
○ <b>Temporizador:</b> Mide el tiempo transcurrido de la carga/descarga. Cuando se carga una batería con un charge rate de 1C, normalmente se llegará al 100% de carga en un tiempo de espera de entre 1 hora y 1:30. Si la batería está dañada o ya muy gastada tardará más tiempo en cargarse, lo que la puede calentar y ocasionar que explote, para evitar esta situación, el		

cargador cuenta con un límite de tiempo, donde si la batería tarda mucho en cargarse, se cortará su alimentación.

- Si durante la carga/descarga se presiona el botón de **Inc.**, se podrá visualizar el nivel de tensión de cada una de las celdas de la batería:

Tensión Celda1

Tensión Celda2

Tensión Celda3

Tensión Celda4

Tensión Celda5

Tensión Celda6

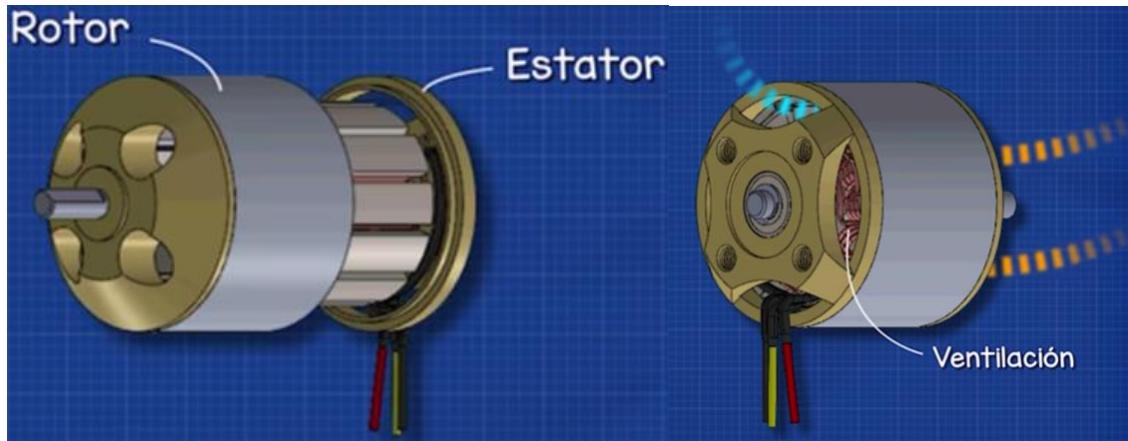
- Al terminar de cargar la batería, se debe dar clic en el botón de **Batt. type** antes de desconectar la batería, sino podría ocurrir un chispazo en sus conectores.



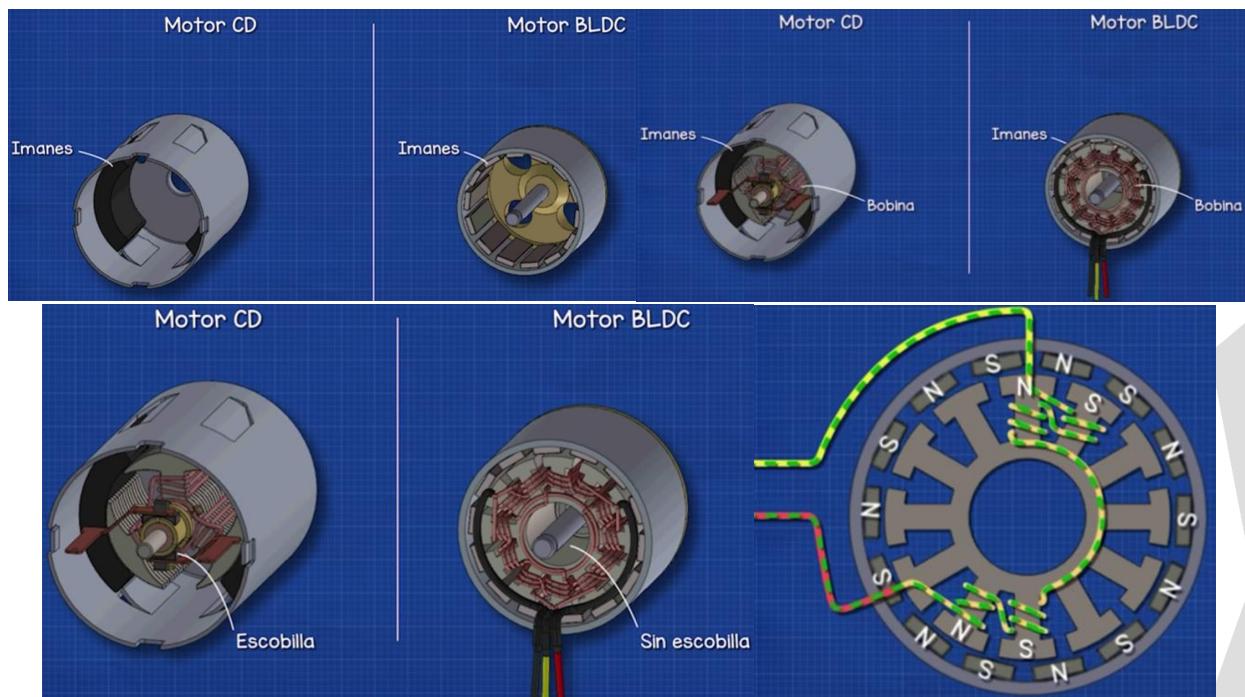
# Motor Brushless, BLDC o Sin Escobillas

El motor sin escobillas, BLDC o brushless es un motor DC, **aunque se acciona a través de una señal trifásica**. La mayoría de ellos es de tipo **Outrunner**, donde su estator (parte fija) se encuentra en su centro y su rotor (parte que gira) en su perímetro; este proporciona un mayor torque. Aunque de igual manera existen los BLDC tipo **Inrunner**, en este caso el estator se encuentra en su perímetro y su rotor en su centro, girando de forma similar a los motores DC con escobillas; este tipo de motor proporciona mayor velocidad. La velocidad y torque son inversamente proporcionales debido a la fórmula de la potencia mecánica rotacional, donde:  $Pot = \tau * w = Torque [N \cdot m] * velocidad angular [\frac{rad}{s}] = Potencia [Watts]$ .

El rotor y el estator tienen orificios para ventilar el motor y evitar el calentamiento de sus bobinas.

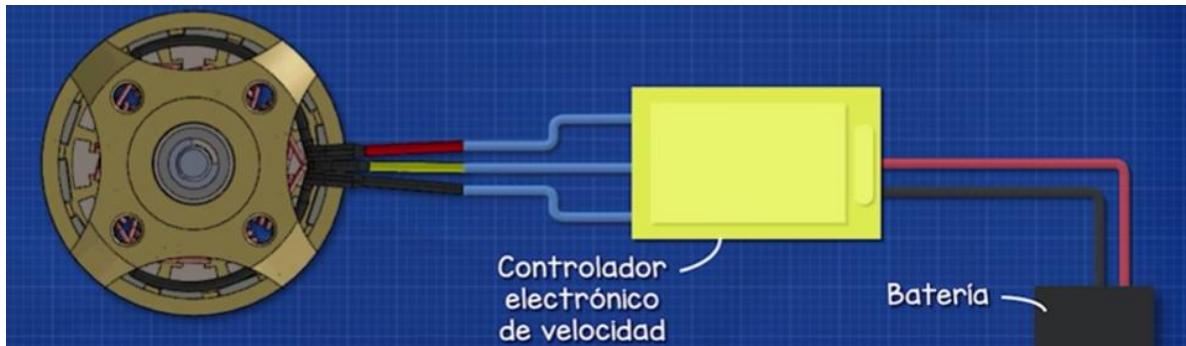


Internamente el BLDC cuenta con **electroimanes estáticos** para **mover los imanes permanentes de su perímetro a la posición donde se requiera**, para ello se encienden y apagan sus 3 bobinas **A, B y C**, siguiendo la secuencia dada por el **controlador externo del motor**, comúnmente llamado **ESC**.

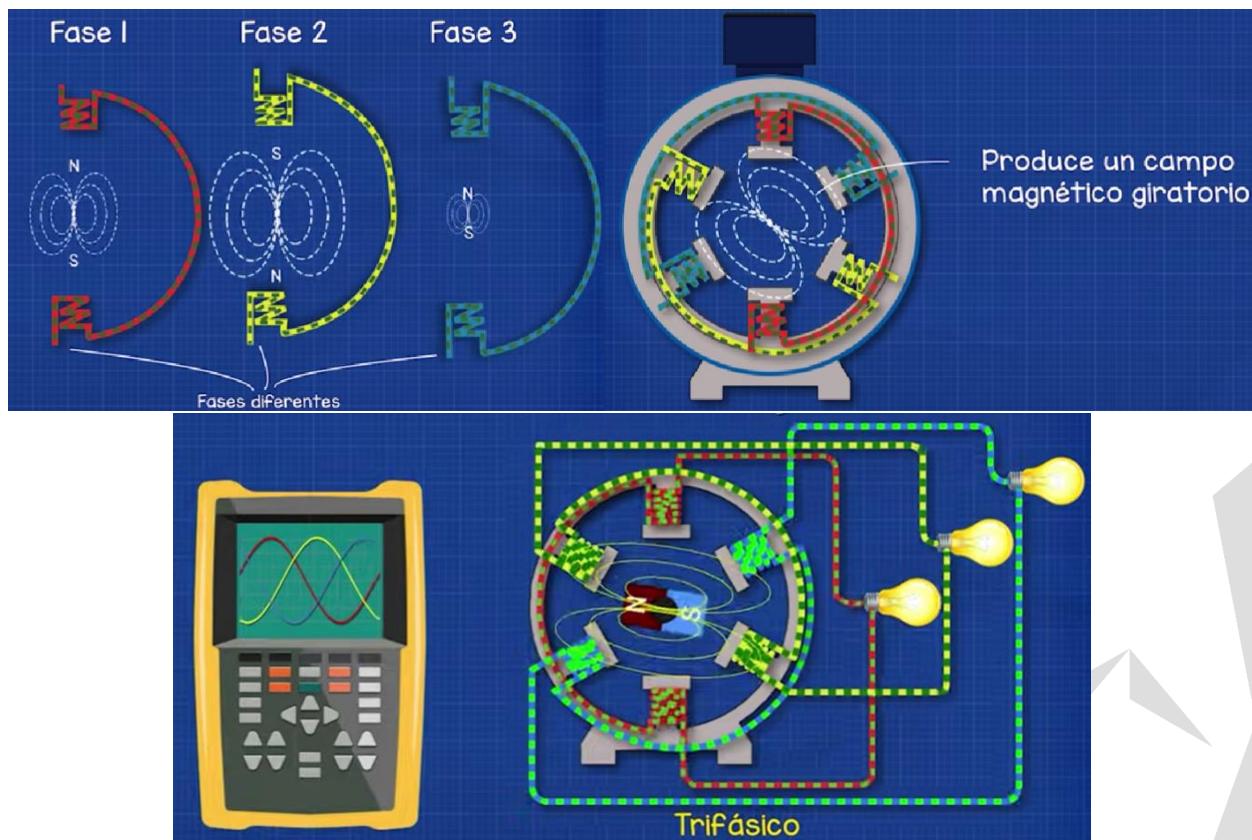


La mayor diferencia que tienen los **BLDC** contra los motores **DC convencionales** es que **la conmutación de sus bobinas se realiza sin las terminales llamadas escobillas**, sino que se hace a través de **controladores externos** y por esa razón el desgaste durante su uso es menor; como no tienen escobillas, no se deterioran al generar fricción cuando giran, por lo que son más rápidos, fuertes y mejores al usarse como generadores eléctricos, por ello es que son ampliamente utilizados en drones, autos a radiocontrol, taladros, turbinas, ventiladores, presas, molinos, robots que soporten mucho peso, etc.

Para mover al motor sin escobillas, **el controlador ESC (Electronic Speed Controller)** convierte la fuente constante **DC** que recibe de una batería en una **señal trifásica** que accione sus bobinas A, B y C.

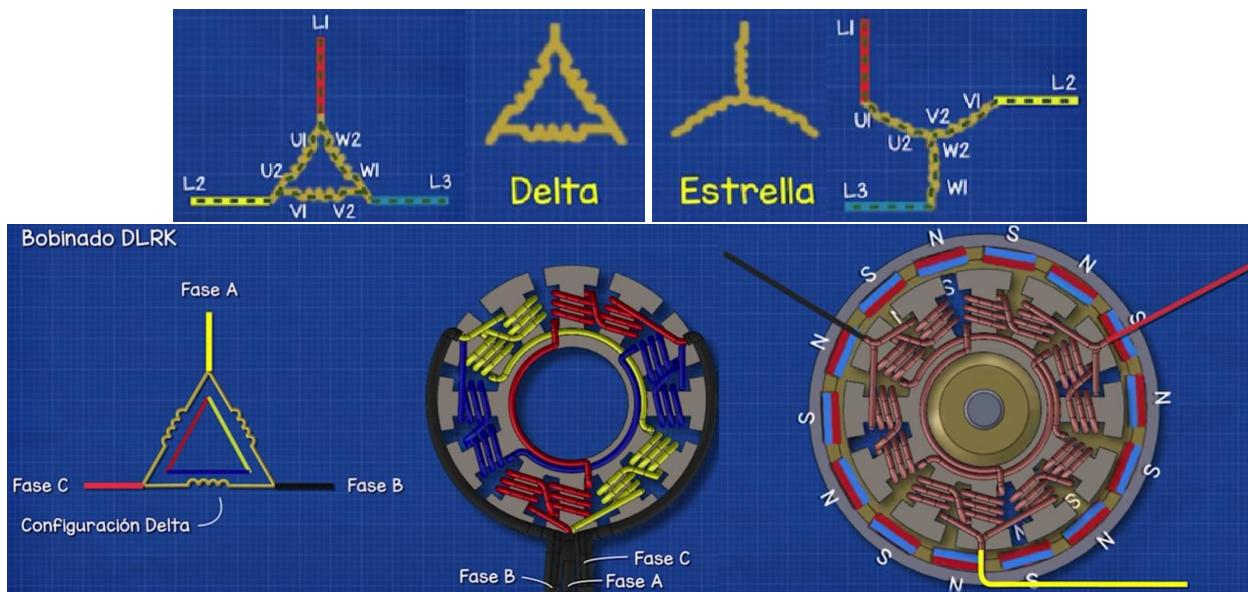


Dentro del motor las 3 bobinas se encuentran separadas físicamente a **120° de distancia de forma equitativa**; cada una de ellas recibe una **señal senoidal** (o fase) diferente, perteneciente a una **señal trifásica**, cuyas señales individuales de igual manera se encuentra desfasadas en el tiempo 120°, estas crean el campo giratorio que mueve los imanes permanentes del rotor perteneciente al motor BLDC.

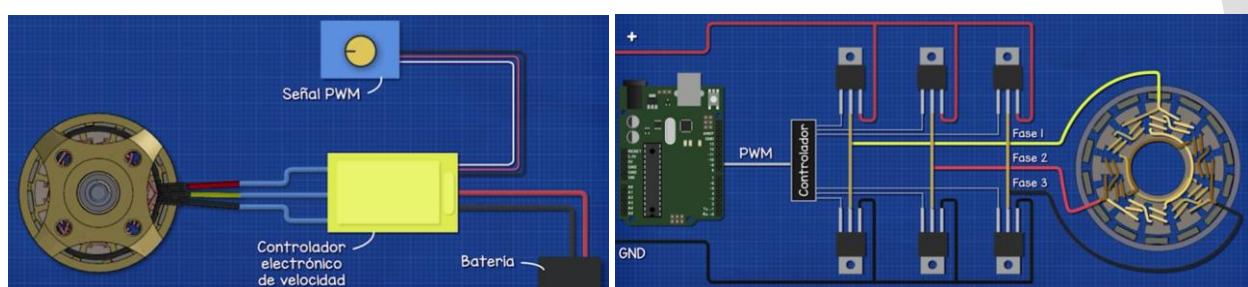


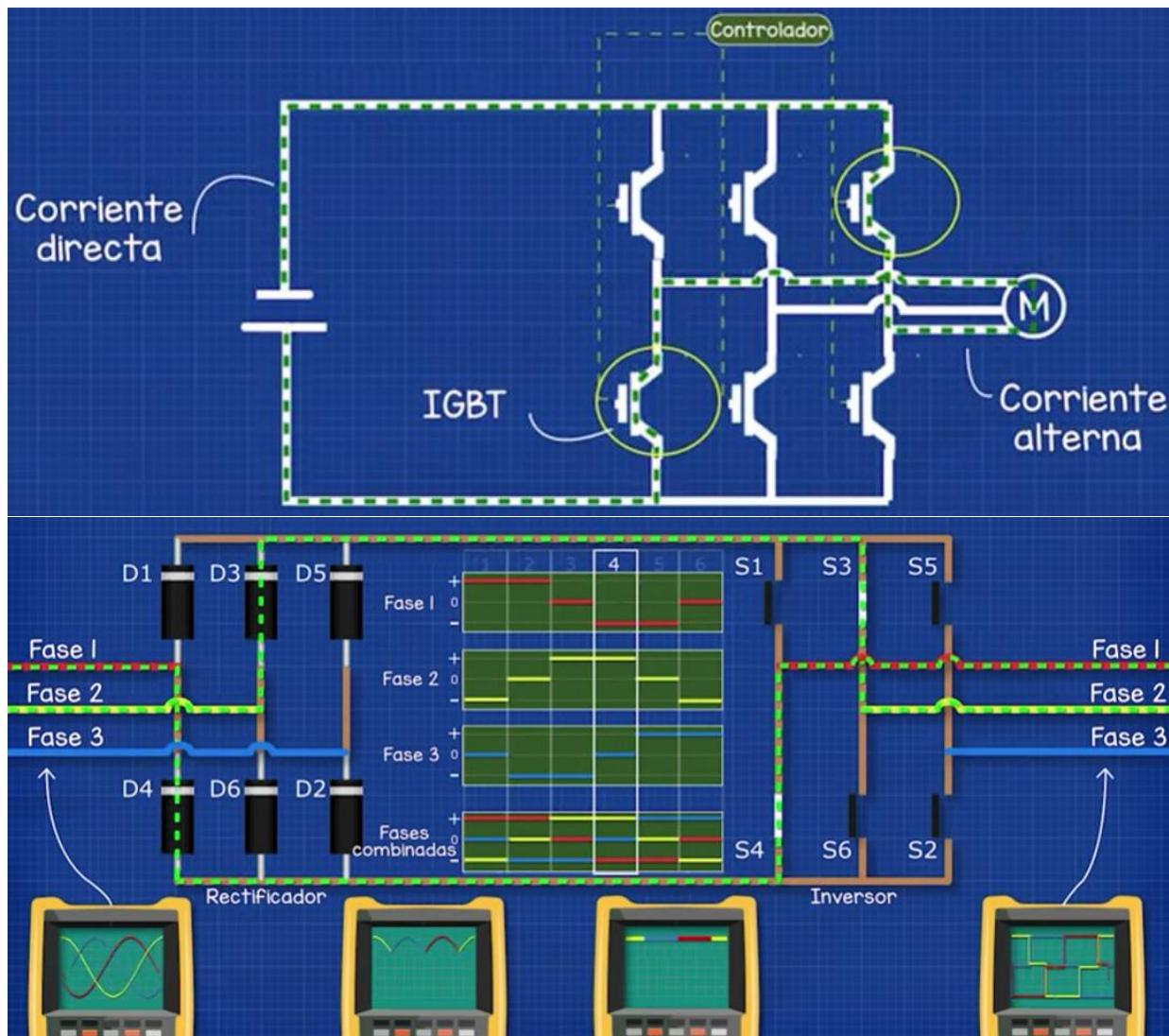
Las 3 bobinas del motor se pueden conectar entre sí de dos maneras, los BLDC utilizan una configuración DLRK, y a continuación, se explicarán las características de los dos tipos de **conexiones trifásicas**:

- **Conexión delta, en triángulo o DLRK:** En esta configuración los dos extremos de las 3 bobinas se conectan entre sí y en sus conexiones comunes se alimenta al motor con las 3 señales trifásicas, debido a esta conexión la corriente fluye de una fase a otra, lo que aumenta la tensión y corriente que recibe cada bobina, que es alimentada no por una fase sino por dos a la vez, esto causa que el motor demande mucha potencia al arrancar, pero posea una alta velocidad y torque. Por esa razón es que ésta es la conexión ideal para el motor brushless.
- **Conexión en estrella:** En esta configuración solo una de las puntas de las 3 bobinas se conecta entre sí, mientras que la punta restante de cada una se alimenta con las 3 señales trifásicas, dando como resultado un mismo nodo común, llamado punto neutro; debido a esta conexión la corriente de cada fase (señal senoidal) fluye solamente a través de una de las 3 bobinas, lo que reduce la tensión y corriente que reciben en comparación con la conexión DLRK, bajando el torque entregado por el motor al 33%, su única ventaja es que demanda menos potencia para el funcionamiento del motor.

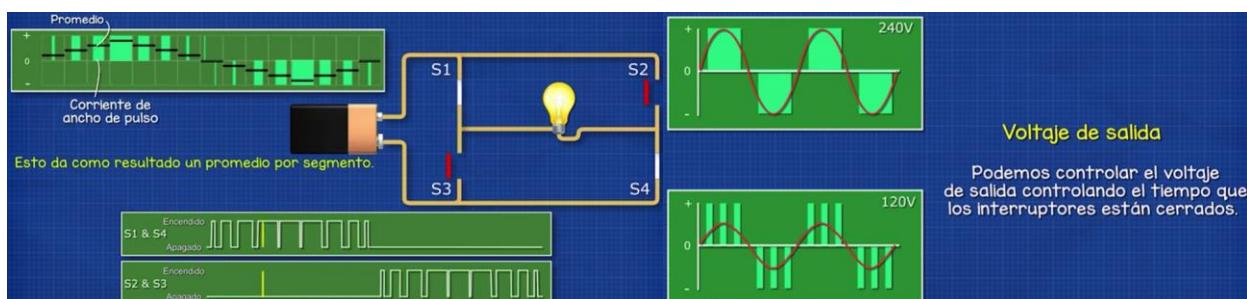


Ya conociendo el tipo de conexión del motor, la forma en la que el **controlador de velocidad (ESC)** transforma **una señal DC** en una **trifásica** es a través de un circuito llamado **Inversor**, el cual commuta 6 transistores MOSFET (asignando 2 para cada fase) por medio de un controlador interno que ejecuta una **configuración de puente H** para simular de forma digital las 3 señales senoidales que conforman una **señal trifásica**, creando así el campo giratorio que genera el movimiento del BLDC en 6 etapas.



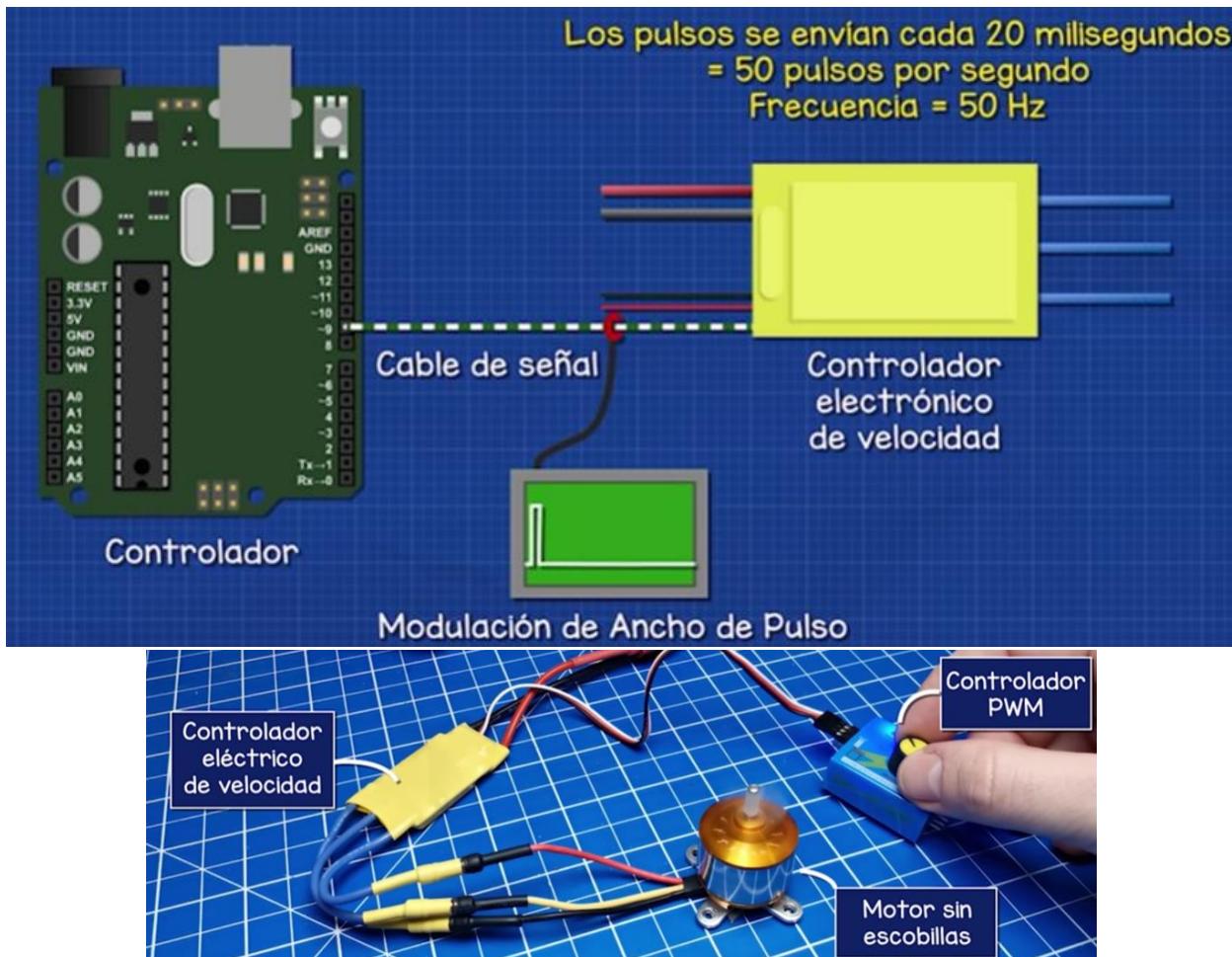


Si solo se usa el circuito inversor, la señal trifásica obtenida será totalmente cuadrada, para mejorar esta situación lo que se hace es ingresar una señal que varíe su tiempo de duración y pulsaciones por segundo en una cierta secuencia con el fin de obtener una mejor definición de la señal senoidal, esto lo realiza el controlador interno a través de una señal PWM, la cual al variar su ancho de pulso (duty cycle), cambia la amplitud y frecuencia de la señal senoidal generada, controlando así la velocidad de giro del motor.

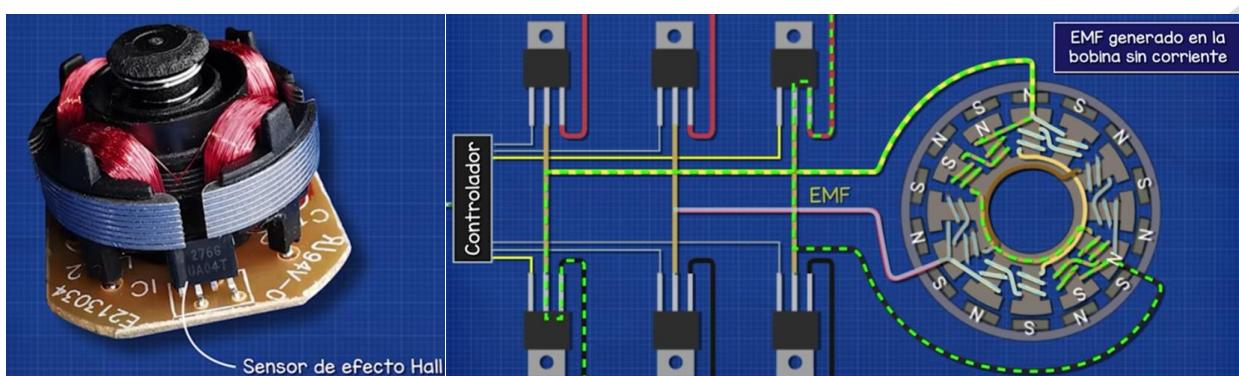


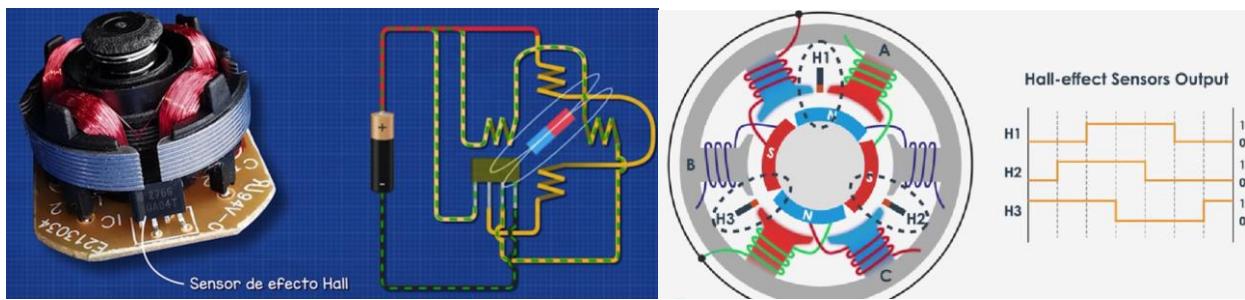
En esta parte es donde entra en juego el segundo controlador, que puede ser un [Arduino](#), [FPGA](#), [Raspberry Pi](#), [circuito especializado probador de servomotores](#), etc. ya que estos son los que generan la

señal PWM (Pulse Width Module) que controla la velocidad de giro del motor sin escobillas, que en específico se deberá mandar a una frecuencia de 50 Hz ( $T = \frac{1}{f} = \frac{1}{50} = 0.02 [s] = 20 [ms]$ ), pudiendo variar su duty cycle entre 1 y 2 milisegundos, de la misma forma como se controlan los servomotores.

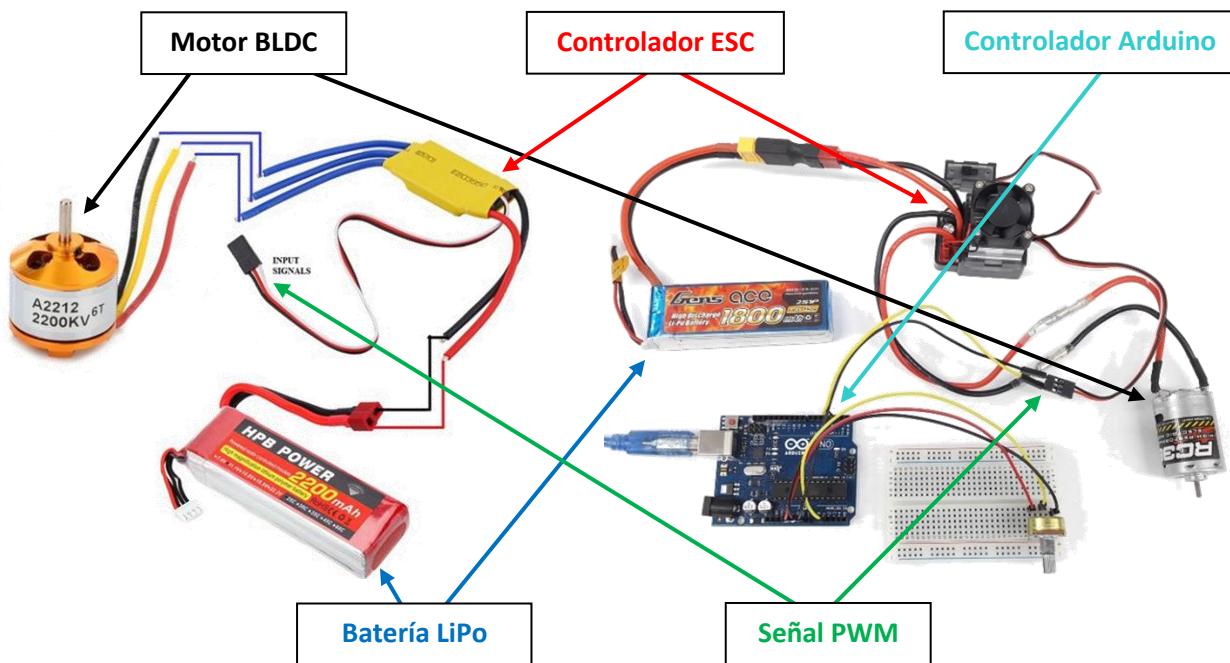


Cuando arranca el motor sin escobillas, este hace algunos ruidos y gira poco, esto lo hace para determinar la posición del rotor, ya que para que el **controlador interno del ESC** pueda **variar la velocidad del motor**, **necesita saber la posición de su rotor**, esto en algunos BLDC se realiza a través de sensores de efecto hall, pero cuando no los tiene, se realiza al medir la fem causada en las bobinas al dejar de energizarlas durante la secuencia de **alimentación trifásica**.





En conclusión, los motores BLDC son alimentados por una señal trifásica que controla sus bobinas **A**, **B** y **C**, la cual es generada por el **controlador ESC (Electronic Speed Controller)**, que es alimentado por una **fuente DC**, usualmente proveniente de una **batería LiPO (Litio Polímero)**, pero además el **ESC a su vez recibe una señal PWM (Pulse Width Module)** por medio de un **segundo controlador** que puede ser un **Arduino, FPGA, Raspberry Pi, etc.** Donde a través de la variación del **duty cycle de una señal PWM**, puede **cambiar la velocidad de rotación del motor brushless**, que girará de forma continua.



La tensión y corriente que recibe el motor no es proporcionada por la **batería LiPo directamente**, sino por el **ESC**, ahí es donde se comprueba que el motor y el **controlador** sean compatibles:

- Debemos asegurarnos de que **la tensión entregada por el ESC y la consumida por el motor sea igual** y que **la corriente entregada por el ESC supere al menos por 5A la requerida por el motor**.
  - **Para aumentar el torque del motor:** Los motores dan un rango de tensión de alimentación, cuanto más alto sea **el voltaje proporcionado**, mayor será el torque generado, aun así, se tiene un límite en el peso que soporta y **si al motor se le somete a esa fuerza máxima que se opone a su movimiento, se bloqueará**, dejando de girar, y al suceder esto, **la corriente que demandan sus bobinas aumentará drásticamente**, a esto se le llama **corriente de bloqueo**, por esa razón de igual manera debemos asegurarnos

que se tenga una alta **tasa de descarga** en la **batería LiPo** que pueda manejar esos picos de corriente demandados por el motor, una tasa recomendada es arriba de **20C**.

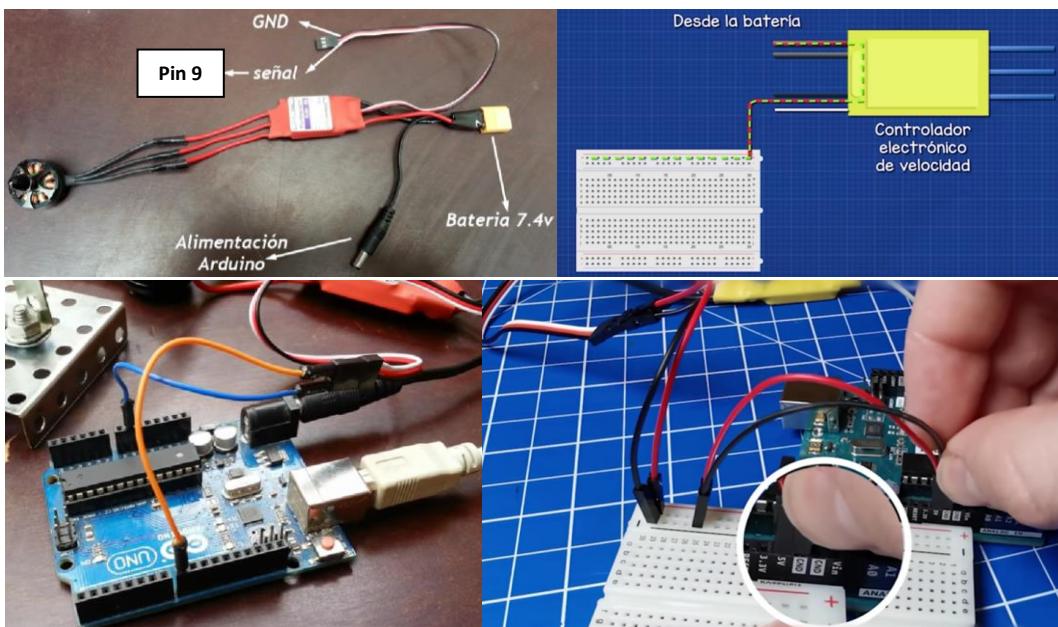
- Recordemos que los mAh entregados por la **capacidad de la batería Lipo no afectan al circuito**, pero la **tensión nominal** sí.
- *La corriente y tensión que demande el BLDC se podrá encontrar en su data sheet.*
- **Para aumentar la velocidad del motor:** El **duty cycle de la señal PWM debe ser mayor**.

Además, uno pensaría que los **cables rojo y negro** de los **pines de señal PWM**, serían para alimentar al **controlador ESC**, pero pasa el caso contrario, **esos pines no son para alimentar el ESC**, sino que proporcionan **5V para alimentar otro dispositivo**, que usualmente es el segundo controlador que proporciona la señal PWM, ya sea un **Arduino, FPGA, Raspberry Pi, etc.** este pin que proporciona alimentación es llamado **BEC (Battery Eliminator Circuit)**.

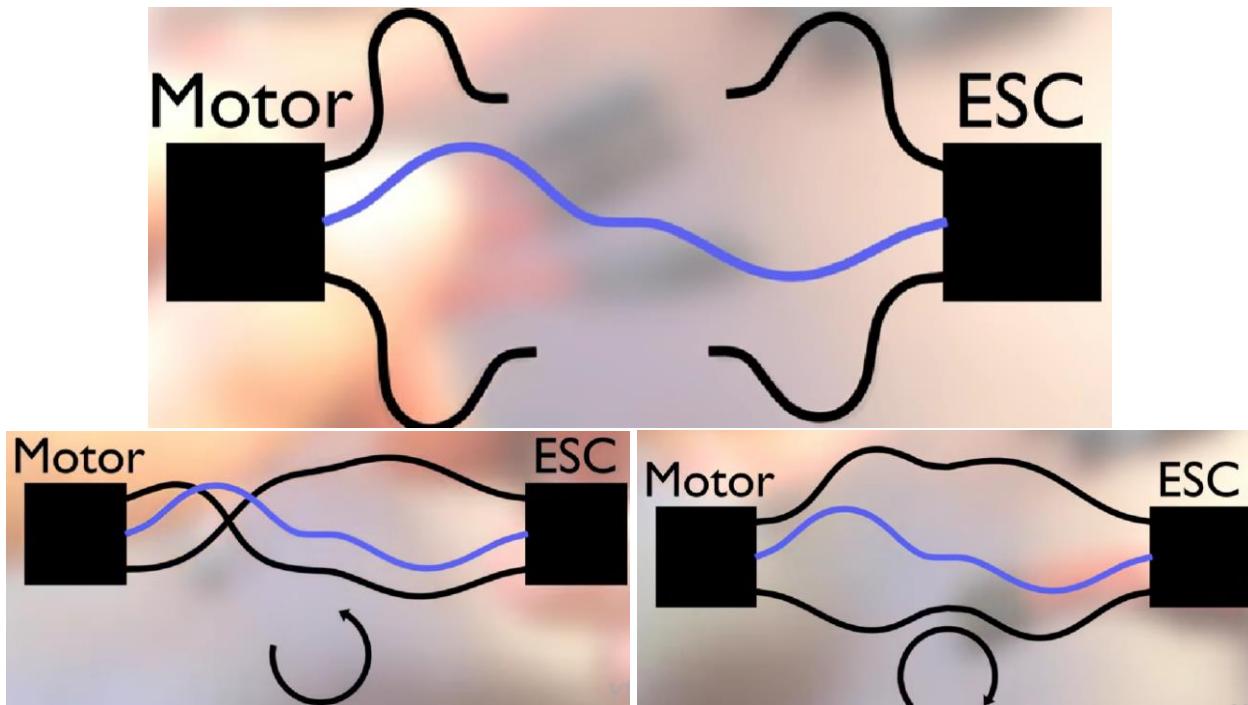
- **El controlador ESC entrega dos distintas formas de alimentación**, una es la adicional dada a otros dispositivos por el pin **BEC** y la otra es la proporcionada por la **señal trifásica**, que tiene el mismo valor de tensión y corriente que la entregada por la **batería LiPo**.



- **El Arduino podrá ser alimentado por medio de sus pines GND y 5V o a través de su conector Jack**, cualquiera de las dos opciones es viable y no necesitará estar conectado al ordenador, ya que su energía no la estará obteniendo del puerto serial, sino de la **batería LiPo**.



La dirección de giro del BLDC dependerá de la forma en la que se conecten sus 3 cables trifásicos con los 3 cables de salida del ESC, el de en medio siempre se conecta con el otro de en medio, pero si se invierte la conexión de los 2 cables restantes, usualmente se invertirá el sentido de giro del motor, aunque algunos controladores de velocidad ya tienen incluido un controlador interno que permite hacer ese cambio de sentido en la rotación sin necesidad de invertir la conexión de los cables.



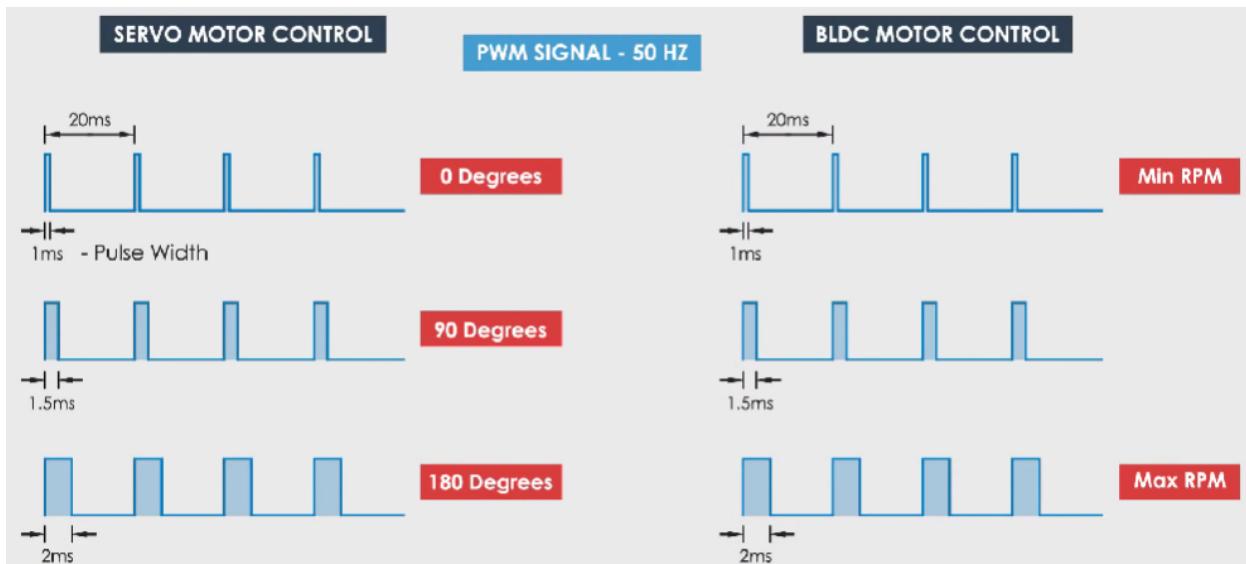
Un último aspecto por explicar de los motores BLDC es su valor de KV, el cual se multiplica por la tensión recibida de la señal trifásica del ESC para indicar la velocidad de rotación máxima en revoluciones por minuto (rpm) que puede alcanzar el motor.

$$rpm = KV * V_{ESC} = KV * V_{LiPo} = KV * 3.7 * \#S = KV * 3.7 * \#Celdas [rpm]$$

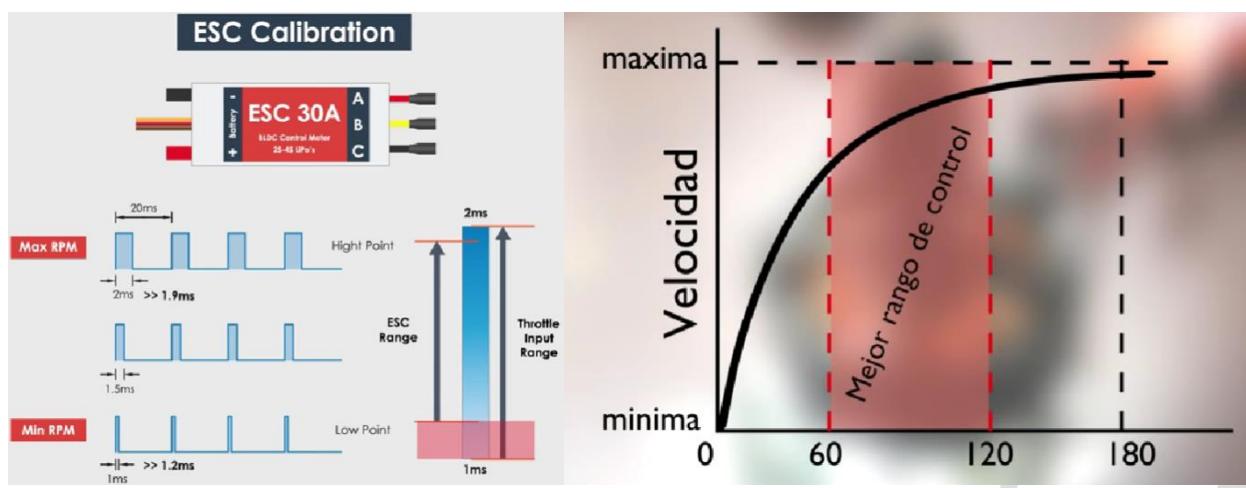


Un dato curioso de los BLDC y los servomotores es que como ambos se controlan de la misma manera, se puede utilizar el cable de alimentación alternativa **BEC (Battery Eliminator Circuit)** para controlar ambos usando la señal PWM de un mismo **controlador ESC (Electronic Speed Controller)**, solo se tiene que considerar que la misma señal PWM controla aspectos distintos en ambos tipos de motores y que estas se calibran de forma individual para cada motor, no importando si es del mismo tipo y modelo:

- **Servomotor:** A través de **una señal PWM con frecuencia de 50Hz, variando su duty cycle entre duraciones de 1 a 2ms, se controla la posición del eje del motor posicionándolo de 0 a 180°.**
- Motor BLDC: Con una **señal PWM con frecuencia de 50Hz, variando su duty cycle entre duraciones de 1 a 2ms, se controla la velocidad de rotación**, variando de 0 a  $KV * V_{ESC}$  [rpm].



Cabe mencionar, que al igual que pasaba con el movimiento de los servomotores, donde en teoría su señal se mueve de 0 a 180° con una variación de **duty cycle entre duraciones de 1 a 2ms**, esto en la realidad no siempre se cumple, por lo que deberemos **calibrar el estado de duración inicial y final para poder tener un correcto control con la señal PWM**, además es importante tomar en cuenta que ni el movimiento del servo ni la velocidad del BLDC se controla de forma lineal, se debe hacer pruebas para determinar qué porcentaje de la señal PWM causa un efecto distinto en el motor.



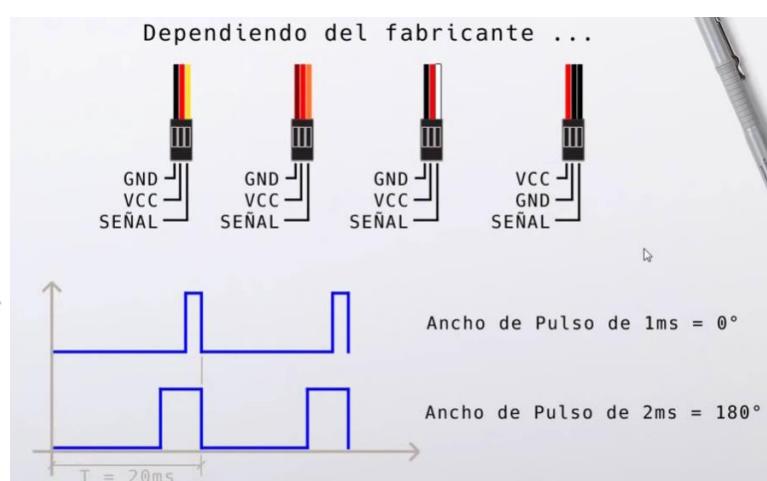
Existen varios tipos de motores brushless, donde más que nada varía:

- Su tamaño.
- El rango de tensión de alimentación que reciben del **ESC**, que a su vez lo recibe de la **batería LiPo**.
- La **velocidad en rpm (revoluciones por minuto)** que pueden alcanzar dependiendo de su constante **KV**.

Aunque tienen sus diferencias, los BLDC comparten algunas características entre ellos y con los servomotores, como que la mayoría es controlada por una **señal moduladora de pulsos o PWM (Pulse Width Module)** con una **frecuencia de 50 Hz** ( $T = \frac{1}{f} = \frac{1}{50} = 0.02 [s] = 20 [ms]$ ), esta frecuencia se refiere al periodo que hay en la señal al enviar un pulso, pero **después de mandar ese pulso, se puede declarar un tiempo de espera** para permitir que el servomotor varíe su velocidad de un valor a otro. Además, en todos fácilmente se puede determinar su velocidad con su valor de **KV**, pero **para conocer su torque** se deben realizar pruebas y para aumentarlo se pueden adaptar sistemas de engranes en su punta.

## Motor Brushless MT2204 2300 Kv

- El BLDC **se alimenta con baterías Lipo de 2 a 3S** en sus cables trifásicos **A, B y C**.
- Demanda una **corriente de 12A** ya con carga y es recomendable que la **batería Lipo** utilizada para su alimentación tenga una **tasa de descarga** arriba de **20C**.
- Su velocidad es controlada por una **señal moduladora de pulsos o PWM (Pulse Width Module)** con una **frecuencia de 50 Hz** ( $T = \frac{1}{f} = \frac{1}{50} = 0.02 [s] = 20 [ms]$ ) y su constante **KV = 2300**.
  - El rango de velocidades que abarca el motor sin escobillas como máximo, alimentándolo con una batería de **3S** es de:
    - 0 a  $KV * V_{ESC} [rpm] = 2300 * 3.7 * 3S = 25,530 [rpm]$ .

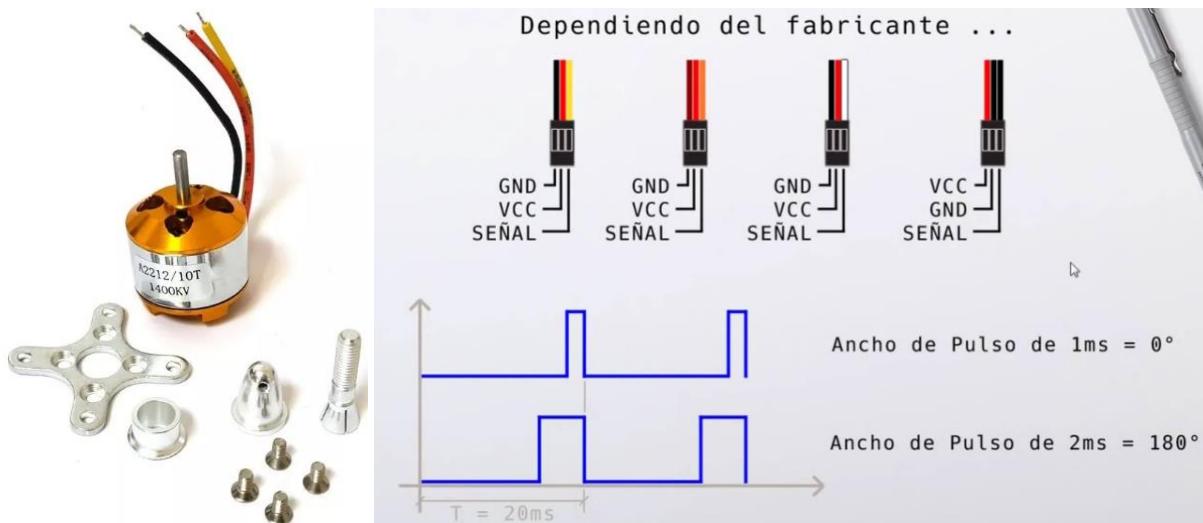


## Motor Brushless A2212 1400 Kv

- El BLDC **se alimenta con baterías Lipo de 2 a 3S** en sus cables trifásicos **A, B y C**.
- Demanda una **corriente de 12A** ya con carga y es recomendable que la **batería Lipo** utilizada para su alimentación tenga una **tasa de descarga** arriba de **20C**.

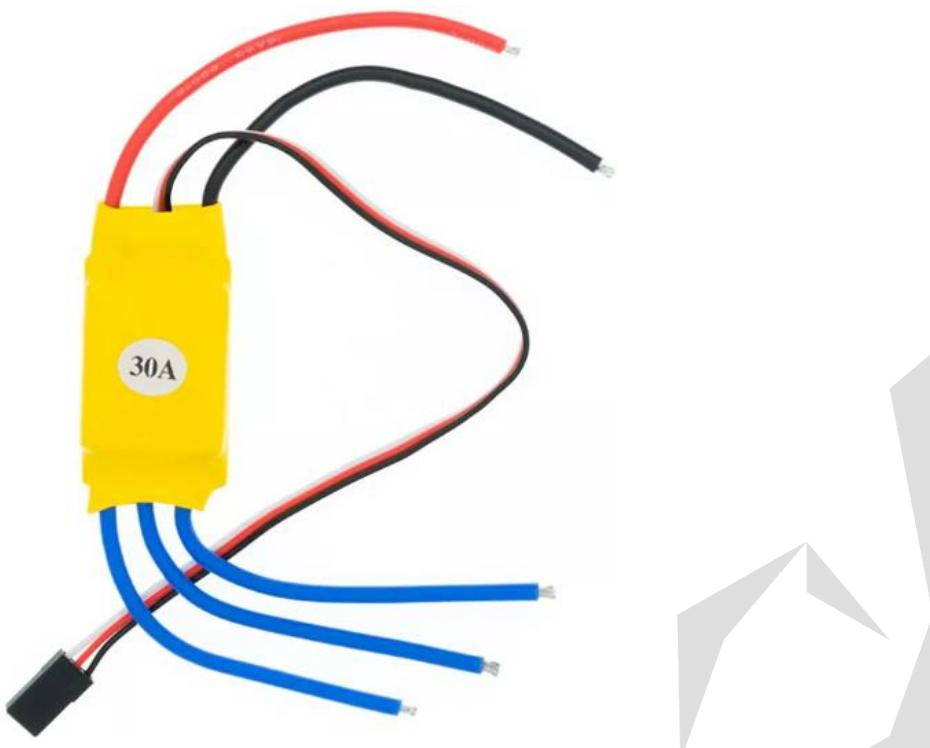


- Su velocidad es controlada por una **señal moduladora de pulsos o PWM** (**Pulse Width Module**) con una **frecuencia de 50 Hz** ( $T = \frac{1}{f} = \frac{1}{50} = 0.02 [s] = 20 [ms]$ ) y su constante  **$KV = 1400$** .
  - El rango de velocidades que abarca el motor sin escobillas como máximo, alimentándolo con una batería de **3S** es de:
    - 0 a  **$KV * V_{ESC}$  [rpm]** =  $1400 * 3.7 * 3S = 15,540 [rpm]$ .



## Electronic Speed Controller (ESC) 30A

- El **controlador ESC (Electronic Speed Controller)** se alimenta con **baterías Lipo de 2 a 3S**.
- Tiene una salida de alimentación alternativa **BEC (Battery Eliminator Circuit)** de 5V/2A.
- Entrega una **corriente de 30A** a los motores sin escobillas que alimenta.



## Electronic Speed Controller (ESC) 30A BLHeli

- El **controlador ESC (Electronic Speed Controller)** se alimenta con **baterías Lipo de 2 a 4S**.
- Tiene una salida de alimentación alternativa **BEC (Battery Eliminator Circuit)** de 5V/2A.
- Entrega una **corriente de 30A** a los motores sin escobillas que alimenta.

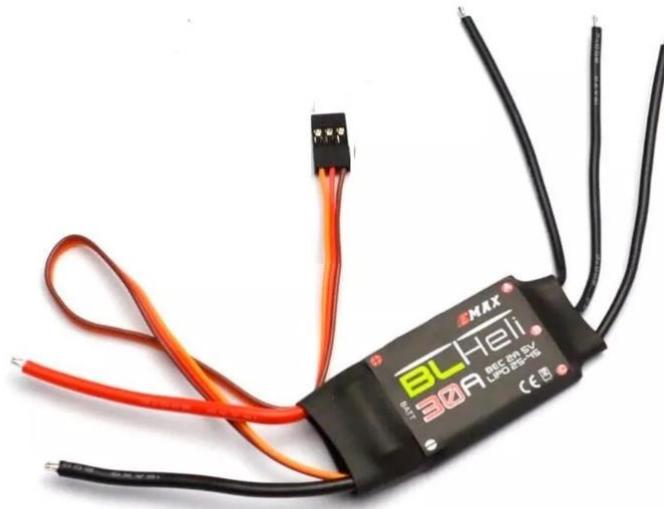
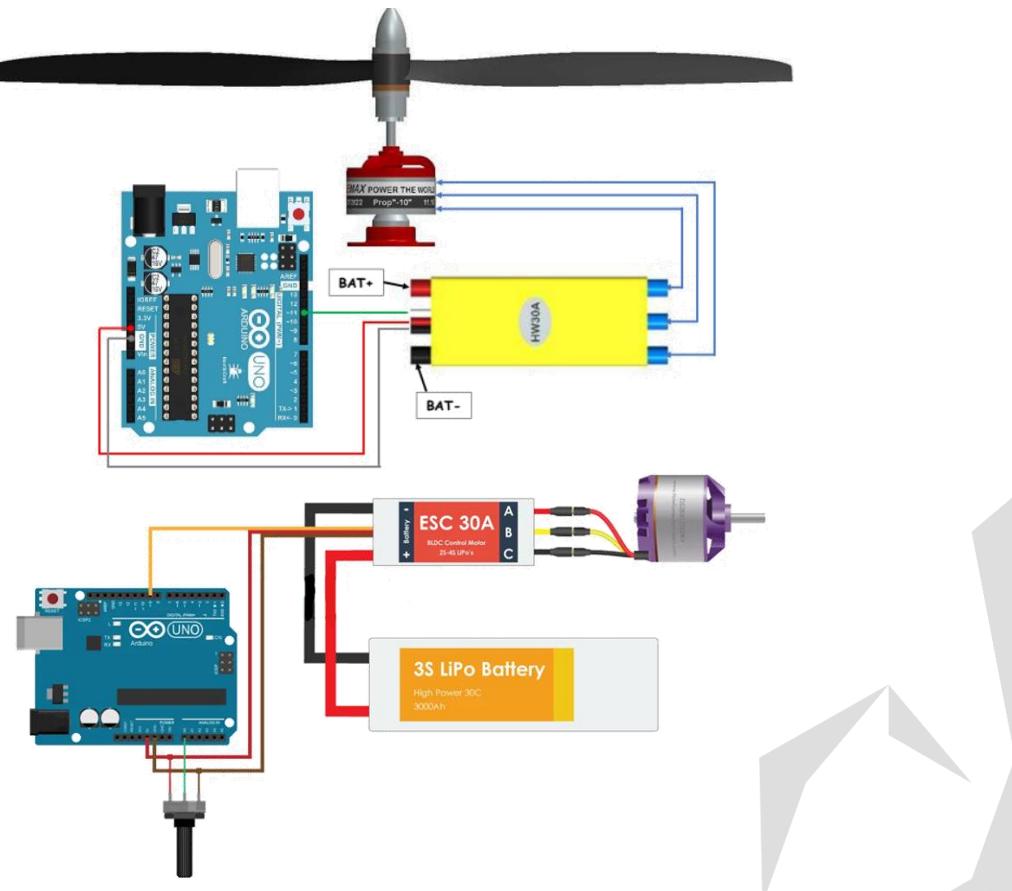


Diagrama de conexión del controlador (Arduino o FPGA), la batería LiPo, el ESC y el motor BLDC:



## Código Arduino - Control de Velocidad con Código:

```
/*33.1.-Control de velocidad de un motor brushless (BLDC) por medio de código: Para la conexión de este motor se usa un controlador externo llamado ESC (Electronic Speed Controller), el cual es alimentado por una batería LiPo, misma que alimenta al Arduino a través del cable rojo del ESC por medio de su pin de 5V, a la tierra del ESC igual se conectarán el pin GND del Arduino y a través de uno de los pines digitales del Arduino se controlará su velocidad de rotación, a través de una señal PWM con frecuencia de 50 Hz, que varíe su duty cycle de 1 a 2ms. El control del BLDC es idéntico al del servomotor, pero la única diferencia es que esa señal PWM controla la posición de 0 a 180 y la de este tipo de motor controla la velocidad, pero su funcionamiento y librería de uso es la misma.

Motor brushless MT2204 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 2300, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:
0 a KV * V_ESC [rpm] = 2300 * 3.7 * 3S = 25,530 [rpm].
La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).-----

Motor brushless A2212 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 1400, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:
0 a KV * V_ESC [rpm] = 1400 * 3.7 * 3S = 15,540 [rpm].
La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).*/
//IMPORTACIÓN DE LIBRERÍAS
#include <Servo.h> //Librería de control de servomotores.

//DECLARACIÓN DE VARIABLES Y/O CONSTANTES
/*Objeto de la clase Servo, este no recibe ningún parámetro, solamente sirve para poder usar los métodos de la librería.*/
Servo motorBLDC;
int PINPWM SERVO = 3;           //Pin 2 = Cable de señal PWM conectada al PIN2 digital del Arduino.
int velocidadMotor = 5000;      //Este delay de 5,000 ms = 5s solamente hace que la velocidad varie.

//CONFIGURACIÓN DE LOS PINES Y LA COMUNICACIÓN SERIAL
void setup() {
    /*servomotor.attach(): Método que sirve para inicializar el objeto del servomotor:
     - El primer parámetro indica el Pin del Arduino al que se conectó el cable de la señal PWM.*/
    motorBLDC.attach(PINPWM SERVO);
}

//EJECUCIÓN DEL PROGRAMA EN UN BUCLE INFINITO
void loop() {
    /*servomotor.write(0): Método usado para controlar la velocidad de giro del eje perteneciente a un motor sin escobillas. Recibe como argumento un número que representa la velocidad deseada.*/
    motorBLDC.write(60);          //Valor mínimo = 60; Valor máximo = 150.
    /*delay(ms): Método que detiene la ejecución del programa un cierto tiempo dado en milisegundos.*/
    delay(velocidadMotor);
    motorBLDC.write(150);
    delay(velocidadMotor);
}
```

## Código Arduino - Control de Velocidad con Código y Duty Cycle Personalizado:

```
/*33.2.-Control de velocidad de un motor brushless (BLDC) por medio de código, pero calibrando el porcentaje del duty cycle de la señal PWM para adaptarla al motor sin escobillas: Para la conexión de este motor se usa un controlador externo llamado ESC (Electronic Speed Controller), el cual es alimentado por una batería LiPo, misma que alimenta al Arduino a través del cable rojo del ESC por medio de su pin de 5V, a la tierra del ESC igual se conectarán el pin GND del Arduino y a través de uno de los pines digitales del Arduino se controlará su velocidad de rotación, a través de una señal PWM con frecuencia de 50 Hz, que varíe su duty cycle de 1 a 2ms. El control del BLDC es idéntico al del servomotor, pero la única diferencia es que esa señal PWM controla la posición de 0 a 180 y la de este tipo de motor controla la velocidad, pero su funcionamiento y librería de uso es la misma.

Motor brushless MT2204 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 2300, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:
0 a KV * V_ESC [rpm] = 2300 * 3.7 * 3S = 25,530 [rpm].
La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).-----
```

```
Motor brushless A2212 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 1400, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:
0 a KV * V_ESC [rpm] = 1400 * 3.7 * 3S = 15,540 [rpm].
La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).*/
//IMPORTACIÓN DE LIBRERÍAS
#include <Servo.h> //Librería de control de servomotores.
```

```
//DECLARACIÓN DE VARIABLES Y/O CONSTANTES
/*Objeto de la clase Servo, este no recibe ningún parámetro, solamente sirve para poder usar los métodos de la librería.*/
Servo motorBLDC;
int PINPWM_BLDC = 3;           //Pin 3 = Cable de señal PWM conectada al PIN3~ digital del Arduino.
/*Cuando el valor bajo del duty cycle en la señal PWM dura 1000 µs = 1 ms, la velocidad del BLDC está en 0 [rpm]. Esta es la teoría, pero si el motor no gira, este valor se debe calibrar para encontrar el óptimo, además al editar esto se puede cambiar el alcance de la velocidad de giro del motor.*/
int DUTYCYLEMIN = 0;           //Calibración BLDC: High Duty Cycle Mínimo ≈ 0 µs = 0 s
/*Cuando el valor alto del duty cycle en la señal PWM dura 2000 µs = 2 ms, la velocidad del BLDC está en KV * V_ESC [rpm]. Esta es la teoría, pero si el motor deja de girar, este valor se debe calibrar para encontrar el óptimo, además al editar esto se puede cambiar el alcance de la velocidad de giro del motor.*/
int DUTYCYLEMAX = 3000;         //Calibración BLDC: High Duty Cycle Máximo ≈ 3000 µs = 3 ms
int velocidadMotor = 900;       //Este delay de 900 ms solamente hace que la velocidad varie.

//CONFIGURACIÓN DE LOS PINES Y LA COMUNICACIÓN SERIAL
void setup() {
    /*servomotor.attach(): Método que sirve para inicializar el objeto del servomotor:
```



```

    - El primer parámetro indica el Pin del Arduino al que se conectó el cable de la señal PWM.
    - El segundo parámetro indica la duración mínima en us de la señal PWM y el tercero la máxima.*/
  motorBLDC.attach(PINPWM_BLDC, DUTYCYLEMIN, DUTYCYLEMAX);
}

//EJECUCIÓN DEL PROGRAMA EN UN BUCLE INFINITO
void loop() {
  /*servomotor.write(0): Método usado para controlar la velocidad de giro del eje perteneciente a
  un motor sin escobillas. Recibe como argumento un número que representa la velocidad deseada.*/
  motorBLDC.write(20);           //Valor mínimo = 20; Valor máximo = 180.
  /*delay(ms): Método que detiene la ejecución del programa un cierto tiempo dado en milisegundos.*/
  delay(velocidadMotor);
  motorBLDC.write(180);
  delay(velocidadMotor);
}

```

## Código Arduino - Control de Movimiento con Potenciómetro:

**/\*33.3.-Control de velocidad de un motor brushless (BLDC) con un potenciómetro y calibrando el porcentaje del duty cycle de la señal PWM para adaptarla al motor sin escobillas:** Para la conexión de este motor se usa un controlador externo llamado ESC (Electronic Speed Controller), el cual es alimentado por una batería LiPo, misma que alimenta al Arduino a través del cable rojo del ESC por medio de su pin de 5V, a la tierra del ESC igual se conectarán el pin GND del Arduino y a través de uno de los pines digitales del Arduino se controlará su velocidad de rotación, a través de una señal PWM con frecuencia de 50 Hz, que varíe su duty cycle de 1 a 2ms. El control del BLDC es idéntico al del servomotor, pero la única diferencia es que esa señal PWM controla la posición de 0 a 180 y la de este tipo de motor controla la velocidad, pero su funcionamiento y librería de uso es la misma.

Motor brushless MT2204 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 2300, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:

0 a KV \* V<sub>ESC</sub> [rpm] = 2300 \* 3.7 \* 3S = 25,530 [rpm].

La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).-----

Motor brushless A2212 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 1400, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:

0 a KV \* V<sub>ESC</sub> [rpm] = 1400 \* 3.7 \* 3S = 15,540 [rpm].

La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).\*/
//IMPORTACIÓN DE LIBRERÍAS

```
#include <Servo.h> //Librería de control de servomotores.
```

```
//DECLARACIÓN DE VARIABLES Y/O CONSTANTES
/*Objeto de la clase Servo, este no recibe ningún parámetro, solamente sirve para poder usar los
métodos de la librería.*/
Servo motorBLDC;
int PINPWM_BLDC = 3;           //Pin 3 = Cable de señal PWM conectada al PIN3~ digital del Arduino.
/*Cuando el valor bajo del duty cycle en la señal PWM dura 1000 ps = 1 ms, la velocidad del BLDC
está en 0 [rpm]. Esta es la teoría, pero si el motor no gira, este valor se debe calibrar para
encontrar el óptimo, además al editar esto se puede cambiar el alcance de la velocidad de giro
del motor.*/
int DUTYCYLEMIN = 0;           //Calibración BLDC: High Duty Cycle Mínimo ≈ 0 ps = 0 s
/*Cuando el valor alto del duty cycle en la señal PWM dura 2000 ps = 2 ms, la velocidad del BLDC
está en KV * VESC [rpm]. Esta es la teoría, pero si el motor deja de girar, este valor se debe
calibrar para encontrar el óptimo, además al editar esto se puede cambiar el alcance de la
velocidad de giro del motor.*/
int DUTYCYLEMAX = 3000;         //Calibración BLDC: High Duty Cycle Máximo ≈ 3000 ps = 3 ms
int velocidadMotor = 900;        //Este delay de 900 ms solamente hace que la velocidad varíe.
int PINPOT = A0;                //Pin A0 = Cable de la terminal central del pot (patita de en medio).
```

//CONFIGURACIÓN DE LOS PINES Y LA COMUNICACIÓN SERIAL

```
void setup() {
  /*servomotor.attach(): Método que sirve para inicializar el objeto del servomotor:
  - El primer parámetro indica el Pin del Arduino al que se conectó el cable de la señal PWM.
  - El segundo parámetro indica la duración mínima en us de la señal PWM y el tercero la máxima.*/
  motorBLDC.attach(PINPWM_BLDC, DUTYCYLEMIN, DUTYCYLEMAX);
}
```

```
//EJECUCIÓN DEL PROGRAMA EN UN BUCLE INFINITO
void loop() {
  /*analogRead(): Método que lee un valor analógico por medio del ADC del Arduino que consta de 10
  bits, por lo que convertirá los valores de tensión de 0 a 5V que reciba a un equivalente de 0 a
  210 - 1 = 1023.*/
  int LECTURAPOT = analogRead(PINPOT);
  /*map(): Método que realiza una regla de 3, convirtiendo un número de un rango a otro que sea
  equivalente al primero, en sus parámetros recibe lo siguiente:
  - Primer parámetro: Recibe la variable de donde proviene el número a convertir.
  - Segundo parámetro: Recibe el valor mínimo del primer rango de valores.
  - Tercer parámetro: Recibe el valor máximo del primer rango de valores.
  - Cuarto parámetro: Recibe el valor mínimo del segundo rango de valores, al que queremos llegar.
  - Quinto parámetro: Recibe el valor máximo del segundo rango de valores, al que queremos llegar.
  De esta manera se convierten los valores de 0 a 1023 entregados por el método analogRead() a
  valores de ángulos de rotación de 20 a 180 que varíe la velocidad del BLDC de 0 a KV * VESC [rpm].*/
  int VELOCIDAD = map(LECTURAPOT, 0, 1023, 20, 180);
  /*Servo.write(): Método usado para controlar la posición del eje perteneciente a un servomotor.
  Recibe como argumento un valor numérico que representa el ángulo de giro deseado.*/
  motorBLDC.write(VELOCIDAD);
}
```

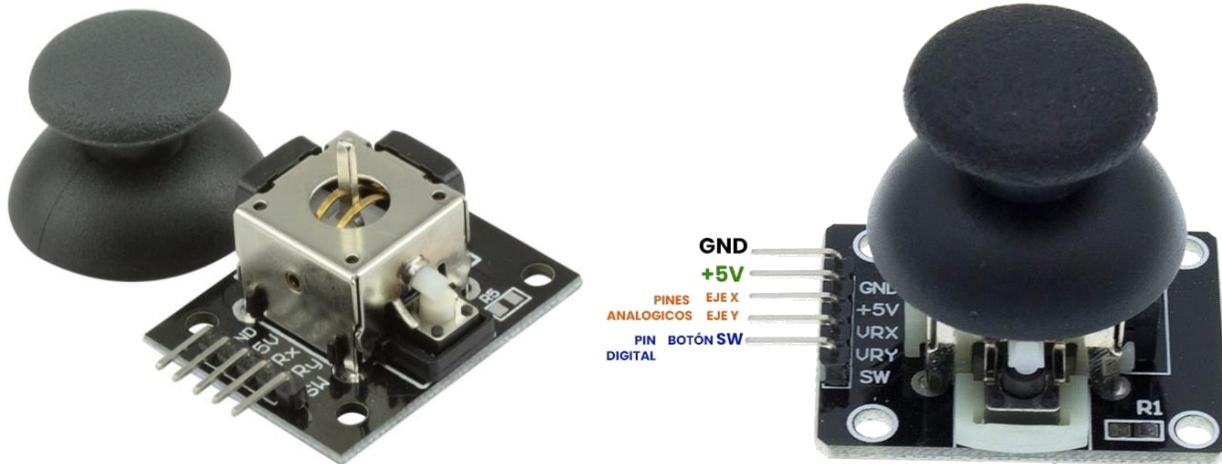


## Código Arduino - Control de Movimiento con JoyStick:

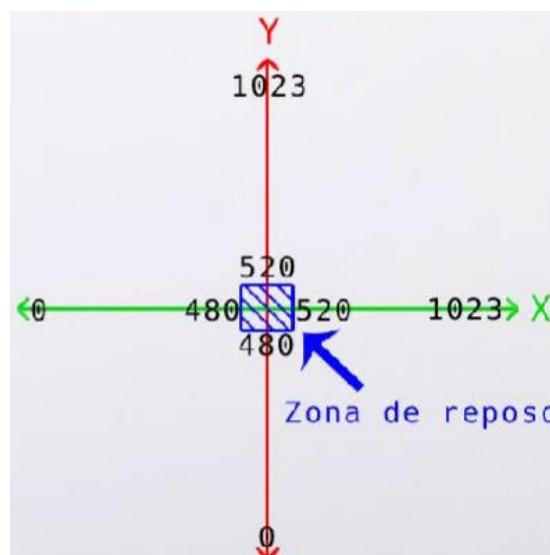
### Módulo KY-023: Joystick

El módulo KY-023 del joystick cuenta con los siguientes 5 pines de conexión:

- **VCC (+5V)** y **GND**: Son los pines de alimentación del encoder que reciben de 3.3 a 5V.
- **VRX** y **VRY**: Se encuentran conectados a las direcciones horizontal (**VRX**) y vertical (**VRY**) de la palanca perteneciente al joystick, generando a través de sus resistencias variables dos señales analógicas que **después de haber pasado por el ADC del Arduino, pueden adoptar valores de 0 a 1023**, abarcando completamente el plano XY.
- **SW**: El pin SW es un interruptor normalmente abierto, que al ser presionado se conecta al **pin GND** y adopta el valor de 0 lógico, para ello este pin siempre se debe conectar a una resistencia pull-up de  $10k\Omega$ , osea que esté conectada entre el pin y la alimentación **VCC**.

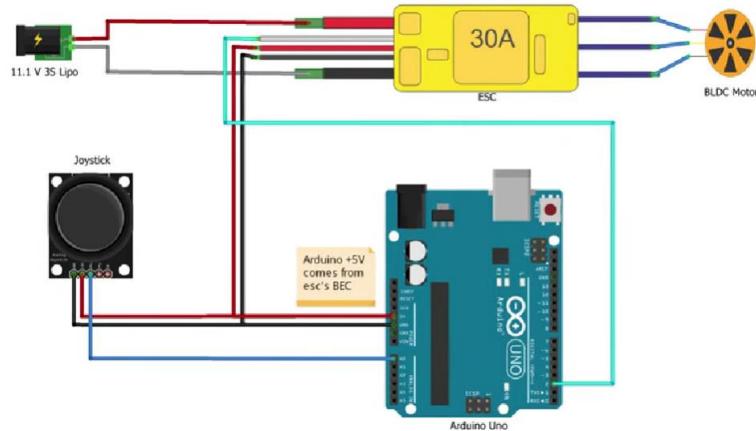


Aunque idealmente la coordenada inicial se encuentra en el punto:  $Posicion_{inicial} = (511.5, 511.5)$ , este no siempre es el caso, por lo cual se declarará un **cuadrado que considere el posible error, llamado zona de reposo**; a partir de este punto la palanca del control analógico podrá ser movida a cualquier posición del plano XY, y al hacerlo adoptará los siguientes valores de tensión **de 0 a 5V**, osea de **0 a 1023**:



## Diagrama de conexión del Arduino o FPGA, la batería LiPo, el ESC, Joystick y el motor BLDC:

Para este circuito no es tan necesario conectar el pin BEC con el de 5V del Arduino, pero si se debe conectar necesariamente la tierra de ambos para que el circuito funcione.



/\*33.4.-Control de velocidad de un motor brushless (BLDC) con un potenciómetro y calibrando el porcentaje del duty cycle de la señal PWM para adaptarla al motor sin escobillas: Para la conexión de este motor se usa un controlador externo llamado ESC (Electronic Speed Controller), el cual es alimentado por una batería LiPo, misma que alimenta al Arduino a través del cable rojo del ESC por medio de su pin de 5V, a la tierra del ESC igual se conectará el pin GND del Arduino y a través de uno de los pines digitales del Arduino se controlará su velocidad de rotación, a través de una señal PWM con frecuencia de 50 Hz, que varíe su duty cycle de 1 a 2ms. El control del BLDC es idéntico al del servomotor, pero la única diferencia es que esa señal PWM controla la posición de 0 a 180 y la de este tipo de motor controla la velocidad, pero su funcionamiento y librería de uso es la misma.

Motor brushless MT2204 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 2300, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:

$$0 \text{ a } KV * V_{ESC} [\text{rpm}] = 2300 * 3.7 * 3S = 25,530 [\text{rpm}].$$

La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).

-----

Motor brushless A2212 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 1400, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:

$$0 \text{ a } KV * V_{ESC} [\text{rpm}] = 1400 * 3.7 * 3S = 15,540 [\text{rpm}].$$

La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]). Acerca del Joystick cabe mencionar que cuando este está conectado al protoboard, el sentido Y es el horizontal y el vertical es el X.\*/

//IMPORTACIÓN DE LIBRERÍAS

```
#include <Servo.h> //Librería de control de servomotores.
```

//DECLARACIÓN DE VARIABLES Y/O CONSTANTES

/\*Objeto de la clase Servo, este no recibe ningún parámetro, solamente sirve para poder usar los métodos de la librería.\*/

Servo motorBLDC;

int PINPWM\_BLDC = 3; //Pin 3 = Cable de señal PWM conectada al PIN3~ digital del Arduino.

/\*Cuando el valor bajo del duty cycle en la señal PWM dura 1000 µs = 1 ms, la velocidad del BLDC está en 0 [rpm]. Esta es la teoría, pero si el motor no gira, este valor se debe calibrar para encontrar el óptimo, además al editar esto se puede cambiar el alcance de la velocidad de giro del motor.\*/

int DUTYCYLEMIN = 0; //Calibración BLDC: High Duty Cycle Mínimo ≈ 0 µs = 0 s

/\*Cuando el valor alto del duty cycle en la señal PWM dura 2000 µs = 2 ms, la velocidad del BLDC está en KV \* V<sub>ESC</sub> [rpm]. Esta es la teoría, pero si el motor deja de girar, este valor se debe calibrar para encontrar el óptimo, además al editar esto se puede cambiar el alcance de la velocidad de giro del motor.\*/

int DUTYCYLEMAX = 3000; //Calibración BLDC: High Duty Cycle Máximo ≈ 3000 µs = 3 ms.

#define VRY A0 //Pin VRY que da las coordenadas verticales Y del Joystick.

int Y; //Coordenada Y que adopta valores de 0 a 1023 por el ADC de 10 bits.

int CoordenadaY; //Coordenada Y después de la transformación de 0 a 1023 -> 0 a 5V.

//CONFIGURACIÓN DE LOS PINES Y LA COMUNICACIÓN SERIAL

void setup() {

/\*servomotor.attach(): Método que sirve para inicializar el objeto del servomotor:

- El primer parámetro indica el Pin del Arduino al que se conectó el cable de la señal PWM.

- El segundo parámetro indica la duración mínima en µs de la señal PWM y el tercero la máxima.\*/

motorBLDC.attach(PINPWM\_BLDC, DUTYCYLEMIN, DUTYCYLEMAX);

/\*pinMode(): Método que indica cuales pines del Arduino son entradas y cuales son salidas:

- primer parámetro: Indica el pin de Arduino que será asignado como salida o entrada.

- segundo parámetro: Usa la instrucción OUTPUT para indicar que el pin es una salida o

INPUT para indicar que el pin es una entrada.

El número del pin que recibe este método como primer parámetro se puede declarar directamente como un número o se puede declarar al inicio del programa como una variable.\*/

pinMode(VRY, INPUT); //Constante VRX asignada al pin A1.

/\*Serial.begin(baudRate): Este método inicializa la comunicación serial entre la placa Arduino y la computadora, además de que configura su velocidad de transmisión dada en unidad de baudios (bit transmitido por segundo) que recibe como su único parámetro:

- En general, 9600 baudios es una velocidad de transmisión comúnmente utilizada y es

```

compatible con la mayoría de los dispositivos y programas.
- Sin embargo, si se necesita una transferencia de datos más rápida y el hardware/software
lo admiten, se puede optar por velocidades más altas como 115200 o 57600 baudios.
Es importante asegurarse de que la velocidad de transmisión especificada coincida con la
velocidad de comunicación del otro dispositivo al que se conecta el Arduino. Si la velocidad de
transmisión no coincide, los datos pueden no transmitirse o recibirse correctamente.*/
Serial.begin(9600); //Comunicación serial de 9600 baudios, para recibir los pasos del encoder.
/*Serial.println(): Método que escribe un mensaje en la consola serial de Arduino, a la cual se puede
acceder en la esquina superior derecha donde se encuentra una lupa.*/
Serial.println("Control Joystick de Motor Brushless");
}

//EJECUCIÓN DEL PROGRAMA EN UN BUCLE INFINITO
void loop() {
/*analogRead(): Método que lee un valor analógico por medio del ADC del Arduino que consta de 10
bits, por lo que convertirá los valores de tensión de 0 a 5V que reciba a un equivalente de 0 a
2^10 - 1 = 1023.*/
Y = analogRead(VRY);
/*En el Joystick se considera una zona de error de ±20 más o menos, por eso en vez de considerar
rangos de 0 a 511.5 y de 511.5 a 1023, se considera de 0 a 480 y de 520 a 1023.*/
if(Y >= 0 && Y < 480){ //Obtención de coordenadas -Y: Velocidad de 20 a 20+(180-20)/2 = 100.
CoordenadaY = map(Y, 0, 480, 20, 100);
}else if(Y >= 520 && Y <= 1023){ //Obtención de coordenadas +Y: Velocidad de 20+(180-20)/2 = 100 a 180.
CoordenadaY = map(Y, 520, 1023, 100, 180);
}else{
CoordenadaY = 80;
}
/*Serial.println(): Método que escribe un mensaje en la consola serial de Arduino, a la cual se puede
acceder en la esquina superior derecha donde se encuentra una lupa. Si queremos concatenar esto con
un valor se debe usar el método String() para convertir su contenido en una cadena de caracteres.*/
Serial.println("Velocidad Joystick = " + String(CoordenadaY) + " rpm");
/*Servo.write(): Método usado para controlar la posición del eje perteneciente a un servomotor.
Recibe como argumento un valor numérico que representa el ángulo de giro deseado.*/
motorBLDC.write(CoordenadaY); //Velocidad mínima = 20; Velocidad máxima = 180.
}

```

## Control de Motores Brushless en Verilog y VHDL

Se realizarán 2 programas individuales de Verilog y VHDL donde se controlará el circuito primero con solamente código y después se usarán los switches de la tarjeta de desarrollo Nexys 2 para indicarle al BLDC a qué velocidades debe girar. Es importante mencionar que la conexión es la misma con la batería LiPo, lo único que varía es que la señal PWM no proviene del Arduino, sino de la tarjeta de desarrollo Nexys 2. El cable de la **señal PWM** será conectado directamente al puerto **JA1** de la Nexys 2 y la alimentación del servomotor será conectada a los puertos **+JA6 o +JA12** y **-JA5 o -JA11**.

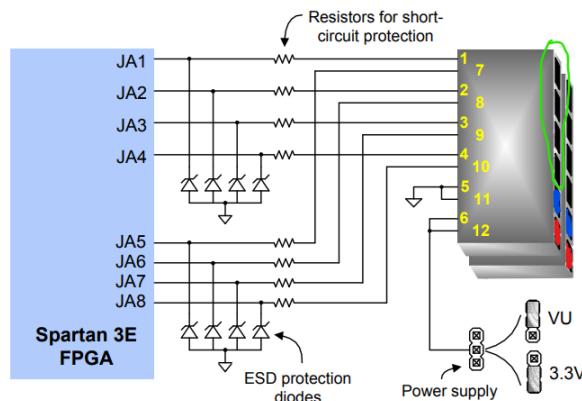


Figure 23: Nexys2 Pmod connector circuits

Table 3: Nexys2 Pmod Connector Pin Assignments							
Pmod JA		Pmod JB		Pmod JC		Pmod JD	
JA1: L15	JA7: K13	JB1: M13	JB7: P17	JC1: G15	JC7: H15	JD1: J13	JD7: K14 <sup>1</sup>
JA2: K12	JA8: L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2: M18	JD8: K15 <sup>2</sup>
JA3: L17	JA9: M14	JB3: R15	JB9: T18	JC3: G13	JC9: G16	JD3: N18	JD9: J15 <sup>3</sup>
JA4: M15	JA10: M16	JB4: T17	JB10: U18	JC4: H16	JC10: J12	JD4: P18	JD10: J14 <sup>4</sup>

Notes: <sup>1</sup> shared with LD3    <sup>2</sup> shared with LD3    <sup>3</sup> shared with LD3    <sup>4</sup> shared with LD3

## Código Verilog – Control de Movimiento con Switches:

### Divisor de Reloj y Creación de Señal PWM:

```
//Control de velocidad de un motor brushless (BLDC) por medio de código, pero calibrando el porcentaje del duty cycle
//de la señal PWM para adaptarla al motor sin escobillas: Para la conexión de este motor se usa un controlador
//externo llamado ESC (Electronic Speed Controller), el cual es alimentado por una batería LiPo, misma que puede
//alimentar dispositivos externos a través del cable rojo del ESC llamado BEC (Battery Eliminator Controller) por medio
//de su pin de 5V y GND, a la tierra del ESC igual se conectaría el pin GND de la Nexys 2 y a través del puerto J1 se
//controlaría su velocidad de rotación, por medio de una señal PWM con frecuencia de 50 Hz, que varíe su duty cycle de
//1 a 2ms. El control del BLDC es idéntico al del servomotor, pero la única diferencia es que esa señal PWM controla la
//posición de 0 a 180 y la de este tipo de motor controla la velocidad de giro, pero su funcionamiento es el mismo, solo
//que se debe calibrar.
//-----
//Motor brushless MT2204 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 2300, por lo que el rango de
//velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:
//0 a KV * V_ESC [rpm] = 2300 * 3.7 * 3S = 25,530 [rpm].
//La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).
//-----
//Motor brushless A2212 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 1400, por lo que el rango de
//velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:
//0 a KV * V_ESC [rpm] = 1400 * 3.7 * 3S = 15,540 [rpm].
//La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).

module controlSwitchesPWM(
    input relojNexys2,           //Reloj de 50MHz proporcionado por la NEXYS 2 en el puerto B8.
    input rst,                   //Botón de reset.
    input [2:0] selectPos,       //3 switches para elegir una velocidad de 0, media y máxima.
    //Las salidas usadas dentro de un condicional o bucle se declaran como reg.
    output reg [2:0] ledAngulo,
    output reg PWM              //Reloj que quiero con una frecuencia menor a 50MHz.
);

//REG: No es ni una entrada ni una salida porque no puede estar vinculada a ningún puerto de la NEXYS 2, solo sirve
//para almacenar y usar valores que sobrevivirán durante la ejecución del código y que además se deben usar dentro de
//un condicional o bucle.
reg [25:0] divisorDeReloj;          //Vector con el que se obtiene el divisor de reloj.
//Este reg sirve para que podamos obtener una gran gama de frecuencias indicadas en la tabla del divisor de reloj,
//dependiendo de la coordenada que elijamos tomar del vector, se asignara al contador que dicta la velocidad del
//motor brushless para que se cree una secuencia entre dos o más velocidades distintas.
integer conteoFrecuenciaPWM = 1;    //Integer para el contador de uno a 1,000,000.
//Variable que obtiene un conteo para obtener la señal de 50 Hz que manda pulsos al motor brushless con el fin de que
//llegue a velocidades de 0 a KV * V_ESC [rpm], esta tiene un periodo de T = 1/50 = 0.02s = 2ms.
//El valor del conteo se obtiene al dividir el periodo de 20ms sobre el periodo de la señal de 50MHz proveniente del
//reloj de la Nexys 2: T50Mz = 1/50,000,000 = 20e-9s = 20ns. Por lo tanto, al realizar la operación se obtiene que:
//T50Hz = 0.02s = 2ms; T50Mz = 20e-9s = 20ns; ConteоФrecuenciaPWM = T50Hz/T50Mz = 2ms/20ns = 2e-3/2e-9 = 1,000,000.
//Al realizar la calibración, el valor real de los pasos para alcanzar la posición 0 grados es de = 57000.
integer posicion0Grados = 57000;   //Integer que dicta los pasos para situar el eje del motor en 0 grados.
//Variable asignada al integer conteoFrecuenciaPWM para situar el estado alto de la señal PWM en 1ms, posicionando al
//motor en una posición de 0 grados inicialmente.
integer movServo = 0;             //Integer que dicta los pasos que debe dar el motor para llegar a una velocidad.
//El valor del integer movServo sera modificado dependiendo del valor ingresado por los switches del vector
//conteoFrecuenciaPWM, para así hacer girar el eje del motor a una velocidad específica, para ello la variable
//selectorDutyCycle solo puede abarcar 1 valor.

//POSEDGE: La instrucción posedge() solo puede tener una entrada o reg dentro de su paréntesis y a fuerza se debe
//declarar en el paréntesis del always@(), además hace que los condicionales o bucles que estén dentro del always@()
//se ejecuten por si solos cuando ocurra un flanco de subida en la entrada que tiene posedge() dentro de su
//paréntesis, el flanco de subida ocurre cuando la entrada pasa de valer 0 lógico a valer 1 lógico y el hecho de
//que la instrucción posedge() haga que el código se ejecute por si solo, significa que yo directamente no debo
//indicarlo con una operación lógica en el paréntesis de los condicionales o bucles, si lo hago me dará error,
//aunque si quiero que se ejecute una acción en específico cuando se dé el flanco de subida en solo una de las
//entradas que usan posedge(), debo meter el nombre de esa entrada en el paréntesis del condicional o bucle.
//Si uso posedge() en el paréntesis de un always@(), todas las entradas de ese always@() deben ser activadas igual
//por un posedge().
always@(posedge(relojNexys2), posedge(rst))
begin : clkdiv //El nombre del always@() es clkdiv.
    if(rst) begin
        //En VHDL se puede declarar a un número hexadecimal para evitar poner muchos bits de un número
        //binario grande, pero en Verilog si hago esto el programa se confunde en el tipo de dato que esta
        //recibiendo y como consecuencia obtendremos un error.
        divisorDeReloj = 26'b00000000000000000000000000000000;
    end else if(conteoFrecuenciaPWM > 1000000) begin
        conteoFrecuenciaPWM = 1;
    end else begin
        divisorDeReloj = divisorDeReloj + 1;           //Esto crea al divisor de reloj.
        conteoFrecuenciaPWM = conteoFrecuenciaPWM + 1; //Esto crea la frecuencia de 50 Hz para la señal PWM.
    end
end

//Debo asignar el contenido de una coordenada de la señal divisorDeReloj a salidaReloj para obtener una
//frecuencia en específico, cada coordenada del vector corresponde a una frecuencia en la tabla del divisor
//de reloj y asignara cierta velocidad de respuesta al control de velocidad por medio de switches.
always@(posedge(divisorDeReloj[12]))
begin : highDutyCyclePWM //El nombre del always@() es highDutyCyclePWM.
    //Cuando el valor del selector sea distinto de cero, vera cual es la velocidad que le corresponde y la
    //asignará a la variable movServo, que hará girar al motor en su velocidad 0, media y máxima.
    if(selectPos != 3'b000) begin
        case(selectPos)
            //CASE se usa para evaluar los diferentes valores de la variable que tenga en su paréntesis
            //Los switches del selector se levantan para que el integer movimientoServo adopte diferentes
            //números que muevan el motor sin escobillas a diferentes velocidades, que son las siguientes:

```

```

    3'b001 : begin
        movServo = 0;      //Bit menos significativo del selector = 0 = velocidad cero.
        ledAngulo = 3'b001;
    end
    3'b010 : begin
        movServo = 25000; //Bit 2 del selector = movServo = 25,000 = velocidad media.
        ledAngulo = 3'b010;
    end
    3'b100 : begin
        movServo = 50000; //Bit 3 del selector = 50,000 = vel máxima = 25,530 [rpm].
        ledAngulo = 3'b100;
    end
    default: begin
        movServo = 0;
        ledAngulo = 3'b000;
    end
endcase
end
endmodule

```

//Para mantener el estado alto de la señal PWM de 1 a 2 ms con el fin de mover el servomotor de 0 a 180 grados, se debe realizar un conteo como el que se realizó para crear la frecuencia de 50 Hz en la señal PWM, para ello se realiza el siguiente calculo:

//T1ms = 1ms; T50Mz = 20e-9s = 20ns; ConteoDutyCycleAlto = 1ms/20ns = 1e-3/20e-9 = 50,000; velocidad 0.  
//T1ms = 1.5ms; T50Mz = 20e-9s = 20ns; ConteoDutyCycleAlto = 1.5ms/20ns = 1.5e-3/20e-9 = 75,000; velocidad media.  
//T1ms = 2ms; T50Mz = 20e-9s = 20ns; ConteoDutyCycleAlto = 2ms/20ns = 2e-3/20e-9 = 100,000; velocidad máxima.  
//Con este cálculo podemos saber que, para que el pulso en alto abarque un rango inicial de 1ms se debe usar 50e3  
//pasos, a este ancho inicial de pulso se le debe sumar el valor de la signal DutyCycle\_1a2ms para que el pulso paulatinamente vaya aumentando de 1 a 2ms, creando así la secuencia de 3 velocidades.  
//Pero recordemos que esta es la teoría, para alcanzar dichas posiciones se debe realizar una calibración, donde veremos el número de pasos reales para alcanzar cada velocidad.  
//AL REALIZAR LA CALIBRACION DEL SERVOMOTOR SE OBTIENE LO SIGUIENTE:  
//El motor sin escobillas MT2204 alcanza un rango de 1ms con un paso inicial de 57e3.

**always@**(conteoFrecuenciaPWM, posicionGrados, movServo)  
begin : pwmSignal //El nombre del always@() es pwmSignal.  
if (conteoFrecuenciaPWM <= (posicionGrados + movServo)) begin  
 PWM = 1'b1;  
end else begin  
 PWM = 1'b0;  
end
end

## Código UCF:

```

//ENTRADAS DEL MODULO divisorDeReloj:
net "relojNexys2" loc = "B8"; //El reloj de 50MHz viene del puerto B8.
net "rst" loc = "H13"; //El botón de Reset está conectado al push button BTN3.
//Bit más significativo del selectPos conectado al switch SW2 en el puerto K18.
net "selectPos[2]" loc = "K18";
//Bit 2 del selectPos conectado al switch SW1 en el puerto H18.
net "selectPos[1]" loc = "H18";
//Bit menos significativo del selectPos conectado al switch SW0 en el puerto G18.
net "selectPos[0]" loc = "G18";

//CABLE DE SENAL PWM DEL SERVOMOTOR CONECTADO A LOS PUERTOS JA:
net "PWM" loc = "L15"; //La señal PWM del servomotor está conectada al puerto JA1.
//Bit más significativo del ledAngulo conectado al led LD2 en el puerto K15.
net "ledAngulo[2]" loc = "K15";
//Bit 2 del ledAngulo conectado al led LD1 en el puerto J15.
net "ledAngulo[1]" loc = "J15";
//Bit menos significativo del ledAngulo conectado al led LD0 en el puerto J14.
net "ledAngulo[0]" loc = "J14";

```

## Código VHDL – Control de Movimiento con Switches:

### Divisor de Reloj y Creación de Señal PWM:

--Control de velocidad de un motor brushless (BLDC) por medio de código, pero calibrando el porcentaje del duty cycle de la señal PWM para adaptarla al motor sin escobillas: Para la conexión de este motor se usa un controlador externo llamado ESC (Electronic Speed Controller), el cual es alimentado por una batería LiPo, misma que puede alimentar dispositivos externos a través del cable rojo del ESC llamado BEC (Battery Eliminator Controller) por medio de su pin de 5V y GND, a la tierra del ESC igual se conectará el pin GND de la Nexys 2 y a través del puerto JA1 se controlará su velocidad de rotación, por medio de una señal PWM con frecuencia de 50 Hz, que varíe su duty cycle de 0 a 2ms. El control del BLDC es idéntico al del servomotor, pero la única diferencia es que esa señal PWM controla la posición de 0 a 180 y la de este tipo de motor controla la velocidad de giro, pero su funcionamiento es el mismo, solo que se debe calibrar.

-----

--Motor brushless MT2204 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 2300, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:  
--0 a KV \* V\_ESC [rpm] = 2300 \* 3.7 \* 3S = 25,530 [rpm].  
--La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).

-----

--Motor brushless A2212 que se alimenta con baterías LiPo de 2 a 3S, cuenta con una KV = 1400, por lo que el rango de velocidades que abarca como máximo, alimentándolo con una batería de 3S es de:  
--0 a KV \* V\_ESC [rpm] = 1400 \* 3.7 \* 3S = 15,540 [rpm].  
--La cual es controlada por una señal PWM (Pulse Width Module) con una frecuencia de 50 Hz (20 [ms]).

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Librerías para poder usar el lenguaje VHDL.
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--Librería declarada para poder hacer operaciones matemáticas sin considerar el signo.

entity controlSwitchesPWM is
    Port ( relojNexys2 : in STD_LOGIC; --Reloj de 50MHz proporcionado por la NEXYS 2 en el puerto B8.
           rst : in STD_LOGIC; --Botón de reset.
           selectPos : in STD_LOGIC_VECTOR (2 downto 0); --Selector de posición.
           ledAngulo : out STD_LOGIC_VECTOR (2 downto 0); --Leds del selector de position.
           PWM : out STD_LOGIC); --Señal PWM.
end controlSwitchesPWM;

--Arquitectura: Aquí se hará el divisor de reloj y la señal PWM haciendo uso de una signal y condicionales if.
architecture frecuenciaNueva of controlSwitchesPWM is
--SIGNAL: No es ni una entrada ni una salida porque no puede estar vinculada a ningún puerto de la NEXYS 2, solo
--existe durante la ejecución del código y sirve para poder almacenar algún valor, se debe declarar dentro de la
--arquitectura y antes de su begin, se le asignan valores con el simbolo :=
signal divisorDeReloj : STD_LOGIC_VECTOR (25 downto 0); --Vector con el que se obtiene el divisor de reloj.
--Esta signal sirve para que podamos obtener una gran gama de frecuencias indicadas en la tabla del divisor de reloj
--dependiendo de la coordenada que elijamos tomar del vector, se asignara al contador que dicta la velocidad del
--motor brushless para que se cree una secuencia entre dos o más velocidades distintas.
signal conteoFrecuenciaPWM: integer range 1 to 1e6 := 1; --Signal para el contador de uno a 1,000,000.
--Variable que obtiene un conteo para obtener la señal de 50 Hz que manda pulsos al motor brushless con el fin de que
--llegue a velocidades de 0 a KV * V_ESC [rpm], esta tiene un periodo de T = 1/50 = 0.02s = 2ms.
--El valor del conteo se obtiene al dividir el periodo de 20ms sobre el periodo de la señal de 50MHz proveniente del
--reloj de la Nexys 2: T50Mz = 1/50,000,000 = 20e-9s = 20ns. Por lo tanto, al realizar la operación se obtiene que:
--T50Hz = 0.02s = 2ms; T50Mz = 20e-9s = 20ns; ConteоФrecuenciaPWM = T50Hz/T50Mz = 2ms/20ns = 2e-3/2e-9 = 1,000,000.
--Al realizar la calibración, el valor real de los pasos para alcanzar la posición 0 grados es de = 57000.
--Variable asignada al integer conteoFrecuenciaPWM para situar el estado alto de la señal PWM en 1ms, posicionando al
--motor en una posición de 0 grados inicialmente.
signal posicion0Grados: integer := 57e3; --Integer que dicta los pasos para situar el eje del motor en 0 grados.
--Variable asignada al integer conteoFrecuenciaPWM para situar el estado alto de la señal PWM en 1ms, posicionando al
--motor en una posición de 0 grados inicialmente.
signal movServo: integer range 0 to 50000 := 0; --Dicta los pasos que debe dar el motor para llegar a una posición.
--El valor del integer movServo será modificado dependiendo del valor ingresado por los switches del vector
--conteoFrecuenciaPWM, para así hacer girar el eje del motor a una velocidad específica, para ello la variable
--selectorDutyCycle solo puede abarcar 1 valor.

begin
    --A los process se les puede dar un nombre diferente para diferenciarlos entre sí, se hace esto porque hay
    --programas de VHDL donde se usan varios process y de esta forma es más fácil no confundir uno con otro, la sintaxis
    --es la siguiente:
    --nombreProcess : process(Entradas o signals que se usan en el bucle, condicional u operación matemática) begin
    --        Contenido del process.
    --end process;
    clkdiv: process(relojNexys2, rst) begin
        if(rst = '1') then --Cuando el botón Reset sea presionado valdrá 1 lógico y el divisor de reloj se reiniciara.
            divisorDeReloj <= "00000000000000000000000000000000";
        elsif(conteoFrecuenciaPWM > 1e6) then
            --Se reinicia el conteo para obtener la frecuencia de 50 Hz de la señal PWM cuando se llegue al tope
            --calculado anteriormente, donde: ConteоФrecuenciaPWM = T50Hz/T50Mz = 2ms/20ns = 2e-3/2e-9 =
            --1,000,000 = 1e6.
            conteoFrecuenciaPWM <= 1;
        elsif(rising_edge(relojNexys2)) then
            --La instrucción rising_edge() hace que este condicional solo se ejecute cuando ocurra un flanco de
            --subida en la señal de reloj clkNexys2 proveniente de la NEXYS 2.
            divisorDeReloj <= divisorDeReloj + 1; --Esto crea al divisor de reloj.
            conteoFrecuenciaPWM <= conteoFrecuenciaPWM + 1;--Esto crea la frecuencia de 50 Hz para la señal PWM.
        end if;
    end process clkdiv;
    --Debo asignar el contenido de una coordenada de la signal divisorDeReloj a salidaReloj para obtener una
    --frecuencia en específico, cada coordenada del vector corresponde a una frecuencia en la tabla del divisor de
    --reloj y asignara cierta velocidad de respuesta al control de velocidad por medio de switches.
    highDutyCyclePWM: process(divisorDeReloj(12)) begin
        if(rising_edge(divisorDeReloj(12))) then
            --Cuando el valor del selector sea distinto de cero, vera cual es la posición que le corresponde y
            --la asignará a la variable movServo, que hará girar al motor en su velocidad 0, media y máxima.
            if(selectPos /= "000") then
                case(selectPos) is
                    --CASE se usa para evaluar los diferentes valores de la variable que tenga en su
                    Paréntesis.
                    --Los switches del selector se levantan para que el integer movimientoServo
                    --adote diferentes números que muevan el motor sin escobillas a diferentes
                    --velocidades, que son las siguientes:
                    when "001" =>
                        movServo <= 0;
                        --Bit menos significativo del selector = 0 = velocidad cero.
                        ledAngulo <= "001";
                    when "010" =>
                        movServo <= 25000;
                        --Bit 2 del selector = movServo = 25,000 = velocidad media.
                        ledAngulo <= "010";
                    when "100" =>
                        movServo <= 50000;
                        --Bit 3 del selector = 50,000 = vel máxima = 25,530 [rpm].
                        ledAngulo <= "100";
                        --Si no se ha seleccionado nada o se mete con los switches cualquier otra cosa,
                        --el BLDC se detiene.
                    when others =>
                        movServo <= 0;
                        ledAngulo <= "000";
                end case;
            end if;
        end if;
    end process;

```

```

        end if;
    end if;
end process highDutyCyclePWM;

--Para mantener el estado alto de la señal PWM de 1 a 2 ms con el fin de mover el servomotor de 0 a 180 grados, se
--debe realizar un conteo como el que se realizó para crear la frecuencia de 50 Hz en la señal PWM, para ello se
--realiza el siguiente calculo:
--T1ms = 1ms; T50Mz = 20e-9s = 20ns; ConteoDutyCycleAlto = 1ms/20ns = 1e-3/20e-9 = 50,000; velocidad 0.
--T1ms = 1.5ms; T50Mz = 20e-9s = 20ns; ConteoDutyCycleAlto = 1.5ms/20ns = 1.5e-3/20e-9 = 75,000; velocidad media.
--T1ms = 2ms; T50Mz = 20e-9s = 20ns; ConteoDutyCycleAlto = 2ms/20ns = 2e-3/20e-9 = 100,000; velocidad máxima.
--Con este cálculo podemos saber que, para que el pulso en alto abarque un rango inicial de 1ms se debe usar 50e3
--pasos, a este ancho inicial de pulso se le debe sumar el valor de la señal DutyCycle_1a2ms para que el pulso
--paulatinamente vaya aumentando de 1 a 2ms, creando así la secuencia de 3 velocidades.
--Pero recordemos que esta es la teoría, para alcanzar dichas posiciones se debe realizar una calibración, donde
--veremos el número de pasos reales para alcanzar cada velocidad.
--AL REALIZAR LA CALIBRACION DEL SERVOMOTOR SE OBTIENE LO SIGUIENTE:
--El motor sin escobillas MT2204 alcanza un rango de 1ms con un paso inicial de 57e3.
pwmSignal: process(conteoFrecuenciaPWM, posicion0Grados, movServo)begin
    --Este if crea el estado en alto y bajo de la señal PWM que crea la secuencia, yendo de 0 a 25,530 [rpm].
    if(conteoFrecuenciaPWM <= (posicion0Grados + movServo)) then
        PWM <= '1';
    else
        PWM <= '0';
    end if;
end process pwmSignal;
end frecuenciaNueva;

```

### Código UCF:

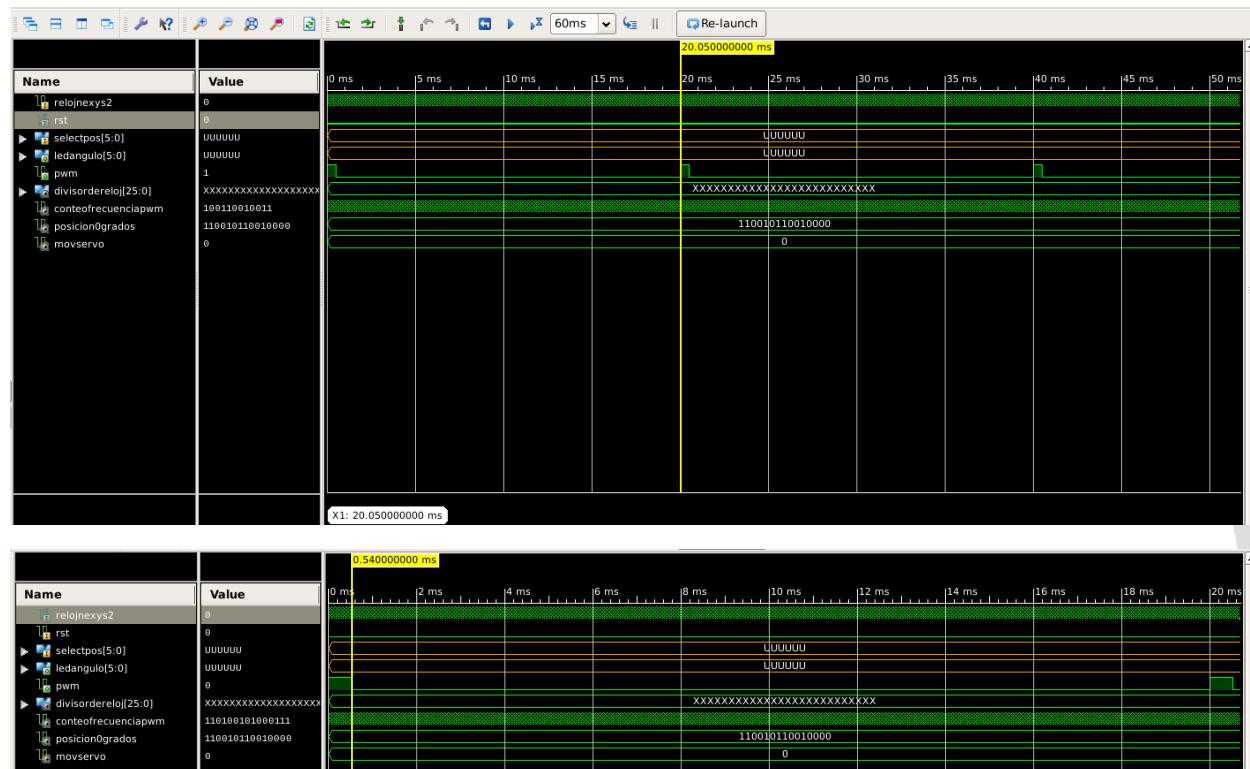
```

//ENTRADAS DEL MODULO divisorDeReloj:
net "relojNexys2" loc = "B8"; //El reloj de 50MHz viene del puerto B8.
net "rst" loc = "H13"; //El botón de Reset está conectado al push button BTN3.
//Bit más significativo del selectPos conectado al switch SW2 en el puerto K18.
net "selectPos[2]" loc = "K18";
//Bit 2 del selectPos conectado al switch SW1 en el puerto H18.
net "selectPos[1]" loc = "H18";
//Bit menos significativo del selectPos conectado al switch SW0 en el puerto G18.
net "selectPos[0]" loc = "G18";

//CABLE DE SENAL PWM DEL SERVOMOTOR CONECTADO A LOS PUERTOS JA:
net "PWM" loc = "L15"; //La señal PWM del servomotor está conectada al puerto JA1.
//Bit más significativo del ledAngulo conectado al led LD2 en el puerto K15.
net "ledAngulo[2]" loc = "K15";
//Bit 2 del ledAngulo conectado al led LD1 en el puerto J15.
net "ledAngulo[1]" loc = "J15";
//Bit menos significativo del ledAngulo conectado al led LD0 en el puerto J14.
net "ledAngulo[0]" loc = "J14";

```

### Simulación PWM:



Al realizar la comparación de los programas de control de un servomotor realizados con Arduino y VHDL o Verilog, podremos ver que su mayor diferencia es que los pasos necesarios para mover el BLDC a una velocidad en específico varían mucho, pero no se notó gran diferencia entre el control que proporcionan ya aplicado en la realidad, ya que el duty cycle en ambos casos es personalizable para cada motor brushless.

## Referencias:

Jared Owen, “¿Cómo funciona un Motor Eléctrico? (Motor de corriente continua)”, 2020 [Online], Available: <https://www.youtube.com/watch?v=CWuIQ1ZSE3c>

vt en línea, “Cómo funciona un motor brushless o sin escobillas”, 2019 [Online], Available: <https://www.youtube.com/watch?v=NnUiAgUundw&t=487s>

Mundo Electrónica, “Todo sobre baterías Li-Po | Usos, carga, cuidados, etc”, 2018 [Online], Available: <https://www.youtube.com/watch?v=N5byJMoEzbA&t=861s>

Toalti RC, “BATERIAS LIPO en 2022 Todo lo que debes saber”, 2021 [Online], Available: <https://www.youtube.com/watch?v=sWhJZFbQhXc&t=538s>

Mentalidad de Ingeniería, “Motor BLDC”, 2022 [Online], Available: <https://www.youtube.com/watch?v=3qy4NtYli8o>

Mentalidad de Ingeniería, “Estrella Delta Explicada”, 2022 [Online], Available: [https://www.youtube.com/watch?v=4657K\\_XtgMQ&t=1s](https://www.youtube.com/watch?v=4657K_XtgMQ&t=1s)

Mentalidad de Ingeniería, “Variador de Frecuencia Explicado - Conceptos Básicos del VFD Inversor de IGBT”, 2021 [Online], Available: <https://www.youtube.com/watch?v=Q2LCqornDvE>

vt en línea, “Cómo usar un motor brushless con arduino”, 2019 [Online], Available: <https://www.youtube.com/watch?v=6WsvDX3MKwg&t=1s>

How To Mechatronics, “How Brushless Motor and ESC Work and How To Control them using Arduino”, 2019 [Online], Available: <https://www.youtube.com/watch?v=uOQk8SJso6Q&t=2s>

Arduino.CC, “UNO R3”, 2023 [Datasheet Producto], Available: <https://docs.arduino.cc/hardware/uno-rev3>

Digilent, “Digilent Nexys2 Board Reference Manual”, 2008 [Datasheet Producto], Available: [https://digilent.com/reference/\\_media/reference/programmable-logic/nexys-2/nexys2\\_rm.pdf](https://digilent.com/reference/_media/reference/programmable-logic/nexys-2/nexys2_rm.pdf)

Readytosky, “Motor Brushless Mt2204 2300kv 2s 3s Drone Racer Fpv”, 2023 [Datasheet Producto], Available: <https://articulo.mercadolibre.com.mx/MLM-775735243-motor-brushless-mt2204-2300kv-2s->

3s-drone-racer-fpv-\_JM#position=2&search\_layout=grid&type=item&tracking\_id=9a8c1316-3800-4263-bead-8924e09ad334

CDMX ELECTRÓNICA, “Motor Brushless A2212 Kv1400 / Kv2200”, 2023 [Datasheet Producto], Available: [https://articulo.mercadolibre.com.mx/MLM-827704923-motor-brushless-a2212-kv1400-kv2200-\\_JM](https://articulo.mercadolibre.com.mx/MLM-827704923-motor-brushless-a2212-kv1400-kv2200-_JM)

Tecneu, “Controlador De Velocidad Electrónico Esc 30a Multiaxis Drone”, 2023 [Datasheet Producto], Available: [https://articulo.mercadolibre.com.mx/MLM-1578311587-controlador-de-velocidad-electronico-esc-30a-multiaxis-drone-\\_JM#position=1&search\\_layout=grid&type=item&tracking\\_id=78d7c099-c409-4386-818b-64c2eb02d143](https://articulo.mercadolibre.com.mx/MLM-1578311587-controlador-de-velocidad-electronico-esc-30a-multiaxis-drone-_JM#position=1&search_layout=grid&type=item&tracking_id=78d7c099-c409-4386-818b-64c2eb02d143)

Tecneu, “Esc 30a Brushless Control Drone F450 F550 Fpv Emax 3s 4s”, 2023 [Datasheet Producto], [https://articulo.mercadolibre.com.mx/MLM-759879779-esc-30a-brushless-control-drone-f450-f550-fpv-emax-3s-4s-\\_JM](https://articulo.mercadolibre.com.mx/MLM-759879779-esc-30a-brushless-control-drone-f450-f550-fpv-emax-3s-4s-_JM)

