ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA ELÉCTRICA

ÁREA DE INGENIERÍA ELECTRÓNICA

ENSAYO Y CONTROL DE UN MOTOR CC

CRESPO MURILLO, LYDIA
TUTOR:
MARTIN RAMOS, JUAN ANTONIO

FECHA: FEBRERO 2024

Índice de contenidos

Lista de figuras 4
Lista de tablas 7
Lista de gráficas 8
Palabras clave 9
1. Hipótesis de partida y alcance 10
2. Motor de corriente continua 11
2.1 Introducción y principio de funcionamiento 11
2.2 Componentes del motor CC 12
2.3 Características del motor CC 13
3. Descripción del hardware 16
3.1 Motor de corriente continua 16
3.2 Encoder 17
3.3 Driver 18
4. Modelado estático 21
4.1 Modelado estático del motor en vacío 21
4.2 Modelado estático del motor con un generador acoplado en vacío 25
4.3 Modelado estático del motor con un generador acoplado en carga 29
4.4 Cálculo de los parámetros del motor de CC 33
5. Empleo del driver y Arduino 36
5.1 Qué es arduino 36
5.2 Porqué utilizar arduino 38
5.3 Entorno arduino 39
5.4 Control de velocidad y dirección de un motor CC 42
6. Diseño del driver 46
6.1 Qué es altium designer 46
6.2 Diseño del driver 46
6.3 Montaje del driver 51

7. Simulink y Arduino 58
8. Incorporación del Encoder 60
9. Control de velocidad 85
10. Presupuesto 95
10.1 Presupuesto del material principal 95
10.2 Presupuesto de la Placa PCB 95
10.3 Presupuesto mano de obra 96
10.4 IVA (21%) 96
11. Trabajo futuro y conclusiones 97
11.1 Trabajo futuro 97
11.2 Conclusiones 97
12. Bibliografía 99
Apéndice I: Datos del ensayo estático del motor con generador acoplado en vacio. 102
Apéndice II: Datos del ensayo estático del motor con generador acoplado en carga. 104
Apéndice III: Hojas de características del Hardware 106

Lista de figuras

Figura 2.1- Esquema de funcionamiento básico del motor de corriente continua. 11
Figura 2.2 -Partes principales de un motor de corriente continua. 13
Figura 2.3 - Ejemplos de modulación por ancho de pulso. 14
Figura 3.1 - Motor de CC modelo 89890911 utilizado. 16
Figura 3.2 - Encoder HEDM-5600-B12 utilizado. 17
Figura 3.3 - Driver L298N utilizado.....18
Figura 3.4 - Descripción del driver L298N utilizado. 19
Figura 3.5 - Esquema de conexión de dos motores de corriente continua. 20
Figura 3.6 - Esquema de conexión de un motor paso a paso.....20
Figura 4.1 - Esquema del modelo estático del motor en vacío. 22
Figura 4.2 - Esquema del montaje del modelo estático del motor en vacío. 22
Figura 4.3 - Montaje en el laboratorio del modelo estático del motor en vacío. 23
Figura 4.4 - Esquema del modelo estático del motor con generador acoplado en vacío.....25
Figura 4.5 - Esquema del montaje del modelo estático del motor con un generador acoplado en vacío. 26
Figura 4.6 - Montaje en el laboratorio del modelo estático del motor con generador acoplado en vacío. 27
Figura 4.7 – Esquema del modelo del modelo estático del motor con un generador acoplado en carga.....29
Figura 4.8 – Esquema del montaje del modelo estático del motor con un generador acoplado en carga.....30
Figura 4.9 – Montaje en el laboratorio del modelo estático del motor con generador acoplado en carga.....31
Figura 5.1 – Placa de circuito impreso de Arduino. 38
Figura 5.2 – Esquema de montaje para el control de velocidad y dirección del motor CC. 44
Figura 5.3 – Montaje para el control de velocidad y dirección del motor CC realizado en el laboratorio.....44
Figura 6.1 - Esquema de puente completo. 47

Figura 6.2 - Tensión entre puerta y surtidor los MOSFETs en función del tiempo. 47
Figura 6.3- Esquema de puente completo cuando M1 está abierto. 48
Figura 6.4 - Esquema de puente completo cuando M2 está abierto. 48
Figura 6.5 - Esquema eléctrico en Altium. 50
Figura 6.6 - Cara inferior y superior de la placa PCB junto a sus pistas y planos de masa correspondientes. 51
Figura 6.7 Cara superior e inferior de la PCB impresa. 52
Figura 6.8- Ejemplo de taladrar la PCB. 52
Figura 6.9 - Imagen de como quedan los componentes soldados. 53
Figura 6.10- Imagen de la placa finalizada. 53
Figura 6.11 - Puntos para la comprobación del funcionamiento del driver. 54
Figura 6.12 -Arriba tensión de entrada al amplificador operacional y abajo la tensión salida de Arduino con niveles de 0 – 5V. 54
Figura 6.13. Arriba tensión de entrada al amplificador operacional con niveles de 0 – 5V. Abajo tensión de salida del amplificador operacional con niveles de 0 – 15V. 55
Figura 6.14- Arriba tensión de entrada al amplificador operacional con niveles de 0 – 5V. Abajo la tensión de salida a la señal “LO” con niveles de 0 – 15V. 56
Figura 6.15 - Arriba tensión de entrada al amplificador operacional con niveles de 0 – 5V. Abajo la tensión en el punto medio con niveles de 0 – 24V. 56
Figura 8.1 - Modelo en Simulink para comprobar el correcto funcionamiento del encoder. 60
Figura 8.2 - Partes de un bloque S-Function Builder. 61
Figura 8.3- Panel de configuración de un bloque S-Function Builder. 62
Figura 8.4 -Tabla de librerías del PWM. 63
Figura 8.5 -Tabla de parámetros y puertos del PWM. 63
Figura 8.6 - Construcción del wrapper. 67
Figura 8.7 - Barra de herramientas del enc_oder. 70
Figura 8.8 -Tabla de librerías del enc_oder. 70
Figura 8.9 -Tabla de parámetros y puertos del enc_oder. 71
Figura 8.10 - Configuración del tiempo de paso. 82
Figura 8.11 - Indicación del puerto de Arduino. 82
Figura 8.12 - Montaje en el laboratorio de la prueba del encoder. 83

Figura 8.13 - Lectura en el osciloscopio de la frecuencia de la prueba del encoder. 84
Figura 8.14 - Lectura en el Scope de las rpm de la prueba del encoder. 84
Figura 9.1 - Modelo en Simulink para simular el control de velocidad. 86
Figura 9.2 - Parámetros del motor pare el modelo en Simulink. 87
Figura 9.3 - Subsistema del modelo en Simulink para simular el control de velocidad. 87
Figura 9.4 - Parámetros del PID. 88
Figura 9.5 - Modelo en Simulink para simular el control de velocidad real. 88
Figura 9.6 - Lectura de ambos osciloscopios para motor en vacio con velocidad 2000. 89
Figura 9.7 - Lecturas del osciloscopio para motor en vacio con velocidad 2000.....89
Figura 9.8 - Lectura del Scope para motor en vacio con velocidad 2000. 90
Figura 9.9 - Lectura de ambos osciloscopios para motor en vacio con velocidad 1500. 90
Figura 9.10 - Lectura del osciloscopio para motor en vacio con velocidad 1500. 91
Figura 9.11 - Lectura del Scope para motor en vacio con velocidad 1500. 91
Figura 9.12 - Lectura de ambos osciloscopios para motor en carga con velocidad 2000. 92
Figura 9.13 - Lectura del osciloscopio para motor en carga con velocidad 2000. 92
Figura 9.14 - Lectura del Scope para motor en carga con velocidad 2000. 93
Figura 9.15 - Lectura de ambos osciloscopios para motor en carga con velocidad 1500. 93
Figura 9.16 - Lectura del osciloscopio para motor en carga con velocidad 1500. 93
Figura 9.17- Lectura del Scope para motor en carga con velocidad 1500. 94

Lista de tablas

Tabla 4.1. Ensayo estático del motor de CC en vacío. 24
Tabla 4.2. Ensayo estático de un motor de CC con un generador acoplado en vacío. 28
Tabla 4.3. Ensayo estático de un motor de CC con un generador acoplado en carga [1]. 32
Tabla 4.4. Ensayo estático de un motor de CC con un generador acoplado en carga [2]. 32
Tabla 10.1 - Presupuesto material principal. 95
Tabla 10.2- Presupuesto de la placa PCB. 96
Tabla 10.3- Presupuesto material total. 96
Tabla 10.4 - Presupuesto mano de obra. 96
Tabla 10.5 - Presupuesto total del prototipo. 96
Tabla 13.1- Tabla de datos del modelo estático del motor con generador acoplado en vacío [1]. 103
Tabla 13.2- - Tabla de datos del modelo estático del motor con generador acoplado en vacío [2]. 103
Tabla 14.1 - Tabla de datos del modelo estático del motor con generador acoplado en carga [1]. 105
Tabla 14.2- Tabla de datos del modelo estático del motor con generador acoplado en carga [2]. 105

Lista de gráficas

Gráfica 4.1 – Distribución de la corriente del motor frente a la velocidad. 24
Gráfica 4.2 – Distribución de la corriente del motor frente a la velocidad. 28
Gráfica 4.3 – Distribución de la corriente del motor frente a la velocidad. 33
Gráfica 6.1 - Relación de tensiones de entrada y salida de las señales HO y LO. 49

Palabras clave

- **f.e.m** (*Fuerza electromotriz*): Se refiere a la diferencia de potencial entre dos puntos.

- **f.m.m** (*Fuerza magnetomotriz*): Fuerza ejercida por la interacción de campos magnéticos.

- **PWM** (*Phase Width Modulation*): Modulación de ancho de pulso.

- **CC**: Corriente continua.

- **Duty**: Ciclo de trabajo.

- **PID** (Proporciona, Integral y Derivativo): Algoritmo de control.

- **CPR**: Ciclos por revolución.

- **rpm**: Revoluciones por minuto.

- **GND**: Toma a tierra.

1. Hipótesis de partida y alcance

Este trabajo final de grado, parte con la premisa de servir como plataforma educacional de la asignatura "Alimentación y accionamiento de prototipos mecatrónicos". La cual forma parte del plan de estudios del Máster en Ingeniería Mecatrónica, ofertado por la Escuela Politécnica de Ingeniería de Gijón.

Durante el desarrollo de este trabajo, se podrán encontrar todos los pasos a seguir para reproducir el prototipo que representará el control de velocidad de un motor de corriente continua. Además, se repasarán todas las dificultades que surgen durante el proceso. Encontrando las soluciones oportunas para logar un control de velocidad estable, completo y seguro.

Este prototipo será también un equipo reproducido para prácticas lectivas y se explicarán cómo funcionan todos sus componentes, así como el software involucrado. Se realizarán todos los cálculos necesarios para elaborar el modelo estático del motor, detallando el funcionamiento de Arduino y su implementación. Al igual que se detallará el funcionamiento de un Encoder, para finalmente, analizar cómo se realiza el control de velocidad del motor modelado.

Con este proyecto se pretende presentar un banco de ensayos útil para la mayoría de las aplicaciones industriales, que puede ser de vital importancia. De hecho, habitualmente se trabaja con simulaciones y no se puede observar el comportamiento real de la máquina. De esta forma, con el contenido de este T.F.G detallado en los siguientes capítulos del documento, se podrá ver una comparativa entre los resultados obtenidos mediante una simulación y los obtenidos mediante ensayos físicos.

Todo ello se acompañará con una bibliografía donde se citarán las fuentes consultadas para la realización del proyecto, así como apéndices en los que se encontrarán las tablas completas con los datos obtenidos en los ensayos estáticos del motor y las hojas de características del Hardware utilizado.

2. Motor de corriente continua

2.1 INTRODUCCIÓN Y PRINCIPIO DE FUNCIONAMIENTO

Un motor de corriente continua es un dispositivo electromecánico que convierte la energía eléctrica en energía mecánica mediante la interacción de campos magnéticos. El principio de funcionamiento se basa en la ley de Lorentz, que establece que una corriente eléctrica que circula por un conductor, en presencia de un campo magnético, experimenta una fuerza (F).

$$F = B * L * I \tag{2.1.}$$

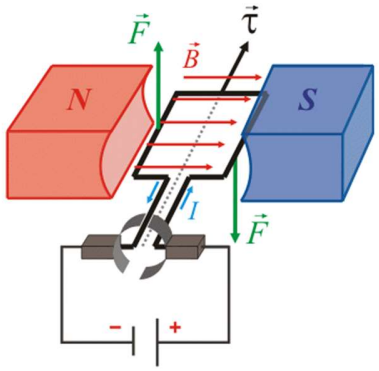


Figura 2.1- Esquema de funcionamiento básico del motor de corriente continua.

Donde:

- B: es la inducción del campo magnético (T).
- L: longitud del conductor (m).
- I: es la intensidad que circula por el conductor (A).
- F: es la fuerza que se produce sobre el conductor (N).

2.2 COMPONENTES DEL MOTOR CC

Este tipo de motores está compuesto principalmente por:

- Estator: Es la parte fija del motor, el cual es un núcleo de hierro con devanados de alambre enrollados alrededor de él que generan un campo magnético estacionario cuando se le aplica una corriente eléctrica. Estas bobinas están dispuestas en forma de anillo alrededor del rotor.
- Rotor: Es la parte móvil del motor, consta de un eje central que puede estar compuesto por bobinas de alambre (para motores de CC sin escobillas) o un devanado de alambre enrollado alrededor de un núcleo de hierro (para motores de CC con escobillas). Cuando la corriente eléctrica fluye a través de estas bobinas, se genera un campo magnético que interactúa con el campo magnético del estator haciendo que el rotor gire.
- Carcasa: Es la estructura externa del motor, protege a éste y aloja todas las partes internas. Generalmente está hecha de materiales como hierro fundido o aluminio para aportar resistencia y disipar el calor generado durante el funcionamiento del motor.
- Conmutador: Es un dispositivo mecánico que se encuentra en motores de CC con escobillas, se encuentra montado en el eje del rotor, éste permite la conmutación de la corriente eléctrica en las bobinas del rotor. Está compuesto por segmentos de cobre que están aislados entre sí y se conectan a las bobinas del rotor. A medida que el rotor gira, el conmutador se asegura que la corriente eléctrica se suministre en la dirección correcta para mantener el movimiento del motor.
- Escobillas: Las escobillas son dispositivos conductores que están en contacto directo con el conmutador en los motores de CC con escobillas. Son responsables de transferir la corriente eléctrica desde una fuente externa al rotor y permiten cambiar la dirección de la corriente a medida que gira el motor.
- Imanes: En algunos motores de CC se utilizan imanes permanentes para crear el campo magnético estacionario en el estator, éstos pueden estas ubicados en el estator o en el rotor, dependiendo del diseño del motor.

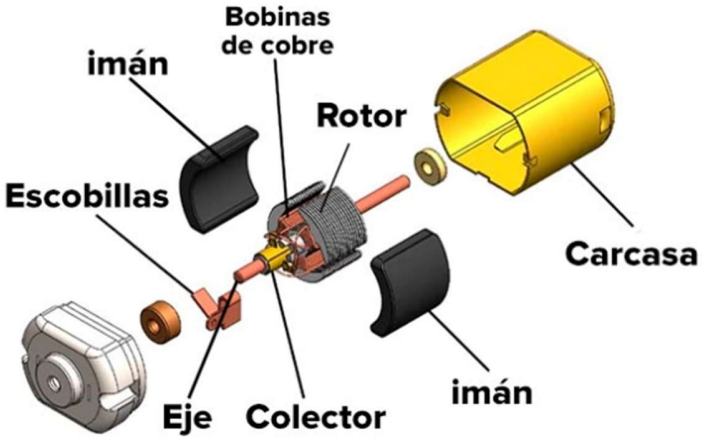


Figura 2.2 -Partes principales de un motor de corriente continua.

2.3 CARACTERÍSTICAS DEL MOTOR CC

Este tipo de motores son compactos, ligeros y se utilizan en procesos que requieren arranques y paradas frecuentes. Son ampliamente utilizados en una gran variedad de aplicaciones debido a su capacidad de proporcionar un control preciso de la velocidad y de la dirección del movimiento.

Las características de control de este tipo de motores han dado como resultado la posibilidad de obtener un inmenso número de usos y aplicaciones, por lo que, su control de velocidad es de gran importancia. En estos motores la velocidad de giro puede regularse de forma sencilla, controlando la corriente por el inducido o inductor, o ambas. Esta característica, unida a los altos pares de arranque que se pueden obtener, ha hecho que este tipo de motor sea insustituible en las aplicaciones que necesitan variaciones de velocidad, como trenes de laminación y tracción eléctrica.

Para que este tipo de motor funcione correctamente requieren la utilización de un driver, ya que su corriente y tensión deben de ser controlados. Un driver, actúa como un amplificador de potencia que proporciona la corriente necesaria para el motor, en función de las señales de control que recibe. Además, estos drivers incluyen características de protección para el motor,

como protección contra sobrecorriente, sobrevoltaje y sobrecalentamiento. Estas características ayudan a prolongar la vida útil del motor y a prevenir daños. Por lo que es esencial para controlar un motor de CC de manera eficiente, segura y precisa. Proporcionando la potencia y las características de control específicas para su funcionamiento.

Para el control de velocidad y de posición se trabaja con señales de pulsos PWM (modulación por ancho de pulso). Un PWM se utiliza para controlar el ancho de pulso de una señal digital, controlando a su vez la potencia entregada a los dispositivos. Funciona como un interruptor, que se activa y desactiva constantemente, regulando así la cantidad de corriente y potencia que se desea entregar. Con este tipo de sistemas el motor recibe corriente por un tiempo y deja de recibirlo por otro, repitiéndose de forma continua. Si se aumenta el tiempo en que el pulso está a nivel alto, se entregará más potencia y si se reduce el tiempo entregará menos potencia.

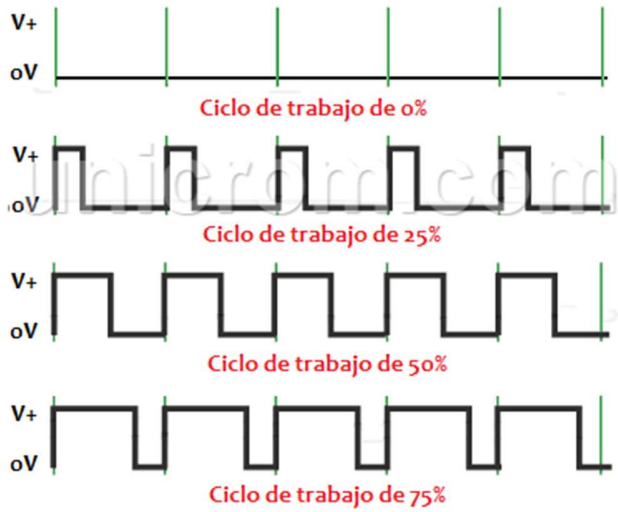


Figura 2.3 - Ejemplos de modulación por ancho de pulso.

En la ilustración anterior podemos ver que:

- Si tenemos un ciclo de trabajo del 0%, esto significaría que no tendríamos tiempo con pulso en nivel alto y el motor estará apagado.

- Si tenemos un ciclo de trabajo del 25%, entonces nuestro motor estaría un 25% del periodo en nivel alto, dT=0,25T, siendo d el ciclo de trabajo y T el periodo, y un 75% en nivel bajo, 1-dT=1-0,75T.
- Si tenemos un ciclo de trabajo del 75%, entonces nuestro motor estaría un 75% del periodo en nivel alto, dT=0,75T y un 25% en nivel bajo, 1-dT=1-0,75T.
- En el caso de que tuviéramos un ciclo de trabajo del 100%, esto significaría que estaríamos todo el tiempo a nivel alto, por tanto, el motor estaría trabajando al máximo porque recibe corriente todo el tiempo.

La principal ventaja del PWM es la eficiencia energética que obtenemos, ya que el circuito entrega a la carga una cantidad de potencia proporcional a la necesaria para que realice el trabajo. Por lo que, si se necesita aumentar la velocidad del motor se incrementaría la potencia entregada a éste, aumentando un ciclo de trabajo mayor. En el caso contrario de querer disminuir la velocidad, el ciclo de trabajo sería menor.

3. Descripción del hardware

En este apartado describiremos el hardware utilizado para realizar los diferentes ensayos.

3.1 MOTOR DE CORRIENTE CONTINUA

El motor utilizado ha sido el modelo 89890911. En el APÉNDICE III encontramos las hojas de características dadas por el fabricante. En ellas nos indica la velocidad en rpm, en vacío, que será de 4000 con una corriente absorbida de 0,34 A. En condiciones nominales el motor tendrá una velocidad de 3430 rpm con una potencia de salida de 104W y una eficacia del 80%. Las características generales dadas por el fabricante indican que tendremos un motor con aislamiento Clase E, con una potencia de salida de 209 W, un par inicial de 2000 mNm, y una resistencia e inductancia inducidas de 0,7 Ω y 0,73 mH.

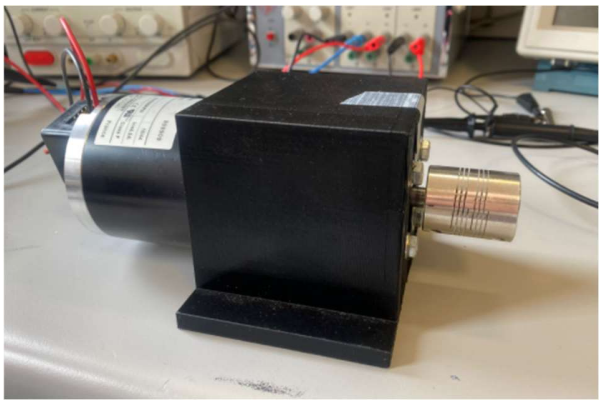


Figura 3.1 - Motor de CC modelo 89890911 utilizado.

3.2 ENCODER

Un encoder es un componente que se le añade a un motor de CC para poder convertir el movimiento mecánico en pulsos digitales y ser interpretados por el sistema de electrónica de control integrado. Está compuesto por un disco conectado a un eje giratorio, pudiendo ser de plástico o vidrio. Se codifica combinando zonas transparentes y opacas que bloquean el paso de la luz. Esto produce una secuencia de información pudiendo así controlar en nuestro caso la velocidad.

El encoder utilizado es el HEDM-5600-B12. En el APÉNDICE III encontramos las hojas de características dadas por el fabricante. En ella podemos encontrar la siguiente descripción referente al nombre del encoder.

HEDM: Encoder con ruedas de código de película.
5600: Tipo de montaje, orejas de montaje externas. 2 canales de salida.
B: Resolución de 1000 CPR.
12: Diámetro del eje, 12-6 mm.

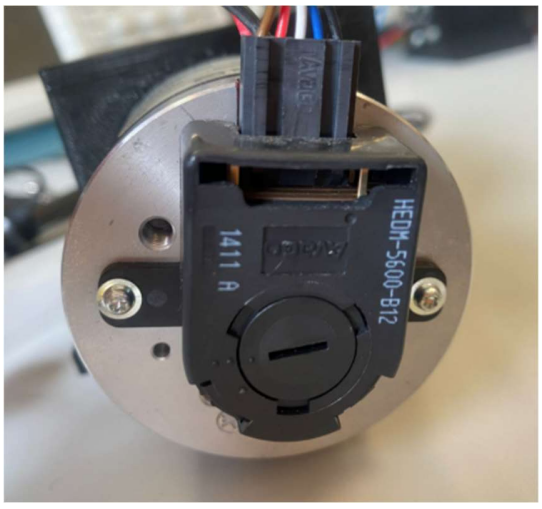


Figura 3.2 - Encoder HEDM-5600-B12 utilizado.

3.3 DRIVER

El módulo controlador nos permite controlar la velocidad y dirección de un motor de CC. Utilizaremos el L298N, este controlador permite trabajar con dos motores en el caso de que sean de corriente continua o uno si este es paso a paso. Esto se debe a que está formado por 4 transistores permitiendo así invertir el sentido de la corriente para poder invertir el de giro del motor.

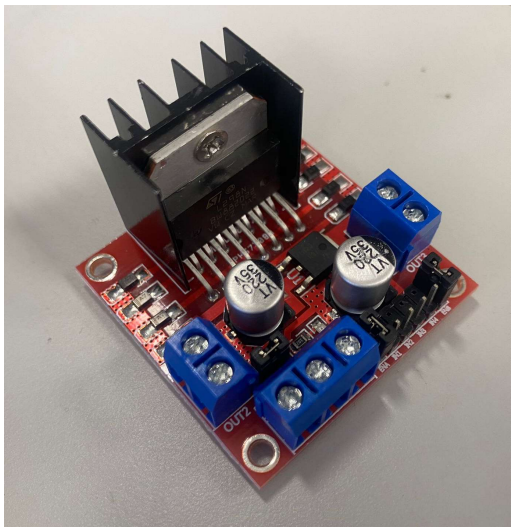


Figura 3.3 - Driver L298N utilizado.

En el APÉNDICE III se encuentra la hoja de características del fabricante en la que podemos observar que este módulo trabaja con una tensión entre 3 y 35V, con una intensidad de corriente máxima de 2A.

Para alimentarse tiene una bornera con 3 pines (+12V, GND, +5V) y tiene dos maneras de alimentarse dependiendo del voltaje que necesiten nuestros motores para funcionar. Para escoger estas formas se utiliza el Jumper que controla el regulador, de manera que tendríamos dos formas principales de alimentación:

La primera forma sería con el Jumper puesto, que significa que el regulador estará funcionando y estaría controlando motores de entre 5-12V. Esto significa, que si por ejemplo nuestro motor funciona a 9V entonces conectaríamos dicho voltaje en el pin 12V, pin GND al pin GND, y el pin 5V funcionaría de salida, por lo que podríamos alimentar otro componente desde este pin. Esta salida será de 5V y 500mA.

La segunda forma de alimentarse será sin el Jumper, lo que significa que el regulador no está funcionando, sirve para controlar motores de entre 12 y 35V, por lo que si nuestro motor funciona con 20V conectamos dicho voltaje en el pin +12V, GND al pin GND y como el regulador está desconectado, debemos conectar 5V en el pin +5V para alimentar la parte lógica del driver. Si se conectase 5V con el Jumper puesto se generaría un cortocircuito y se quemará el integrado.

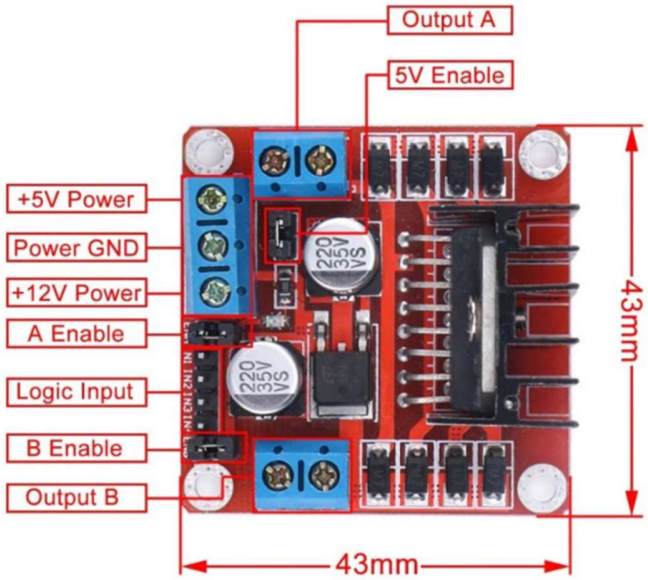


Figura 3.4 - Descripción del driver L298N utilizado.

Las dos borneras de los pines sirven para conectar los dos motores de corriente continua o un motor paso a paso y son controlados por los pines IN1, IN2, IN3 e IN4. En las bornas para controlar los motores se encuentran las marcas OUT1, OUT2, OUT3 y OUT4. Por lo que IN1

controla OUT1, IN2 controla OUT2, IN3 controla OUT3 e IN4 controla OUT4. Los pines ENABLE A y ENABLE B serán los encargados de controlar el ancho de pulso (PWM).

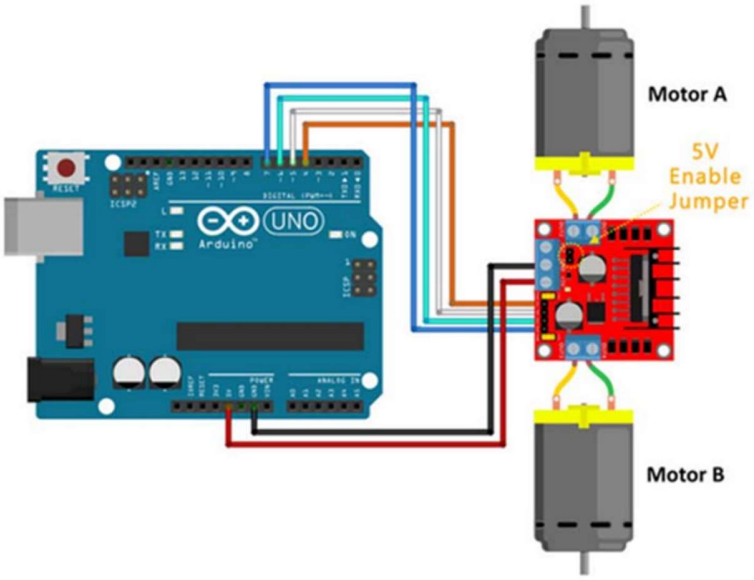


Figura 3.5 - Esquema de conexión de dos motores de corriente continua.

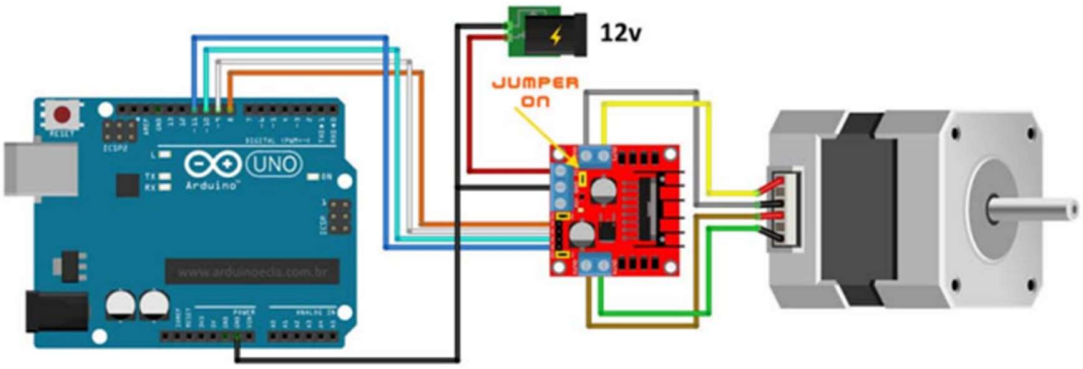


Figura 3.6 - Esquema de conexión de un motor paso a paso.

4. Modelado estático

Para realizar el modelado estático realizaremos tres ensayos, el primero será un motor en vacío, el segundo será el modelo estático del motor acoplándole un generador en vacío y el tercero será el modelo estático del motor acoplándole un generador con una carga de 12 Ω.

4.1 MODELADO ESTÁTICO DEL MOTOR EN VACÍO

Para realizar el modelado del motor de CC tendremos en cuenta que este esté en vacío. Lo primero que haremos será conectar este a una fuente de tensión en serie con una sonda de corriente de efecto hall, alimentada a ± 15 V, para poder medir la corriente media del motor según vayamos variando la fuente de tensión hasta 24V. Para minimizar el rizado de la corriente lo que haremos será añadir un condensador en paralelo con la fuente.

Para poder medir la velocidad del motor alimentaremos el encoder a una fuente de tensión de 5V y a su vez lo conectaremos a un osciloscopio para poder visualizar los cambios de velocidad según varie la tensión de alimentación del motor. En el osciloscopio podremos leer en kHz la frecuencia del motor, sabiendo que según las hojas del fabricante del encoder este funciona a 1000 rpm, únicamente tendremos que multiplicar la frecuencia por 60 para obtener la velocidad en rpm. Si a su vez queremos la velocidad angular del motor en rad/seg deberemos hacer la siguiente conversión.

$$Velocidad\left(\frac{rad}{seg}\right) = Velocidad\,(rpm) * \frac{2*\pi}{60} \qquad (4.1.)$$

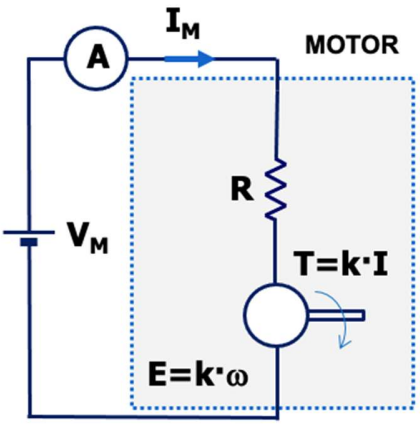


Figura 4.1 - Esquema del modelo estático del motor en vacío.

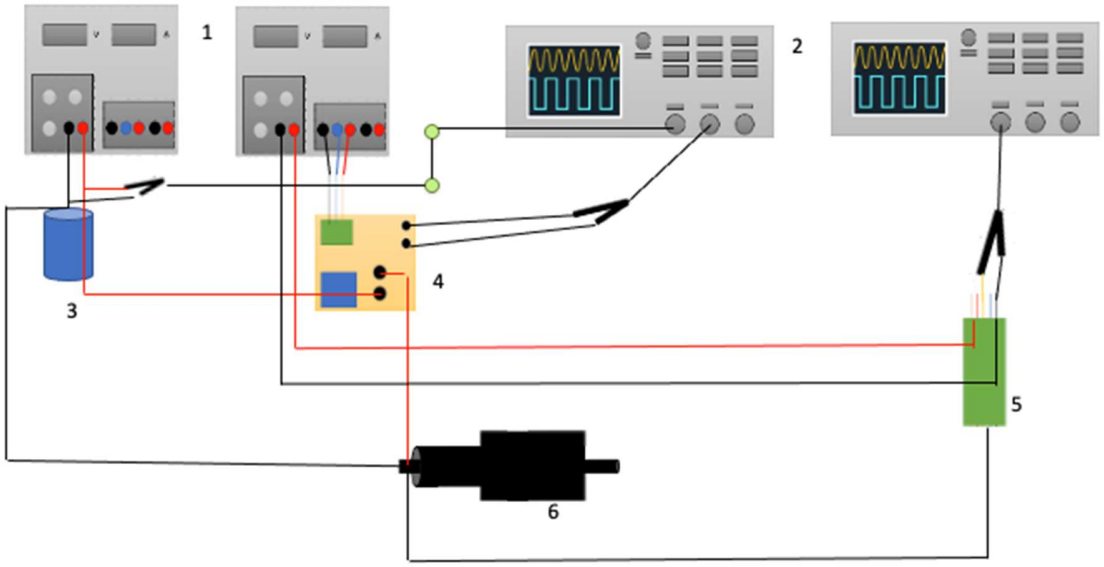


Figura 4.2 - Esquema del montaje del modelo estático del motor en vacío.

Donde:

1. Fuentes de tensión.
2. Osciloscopios.
3. Condensador.
4. Sonda de corriente.

5. Encoder.
6. Motor.

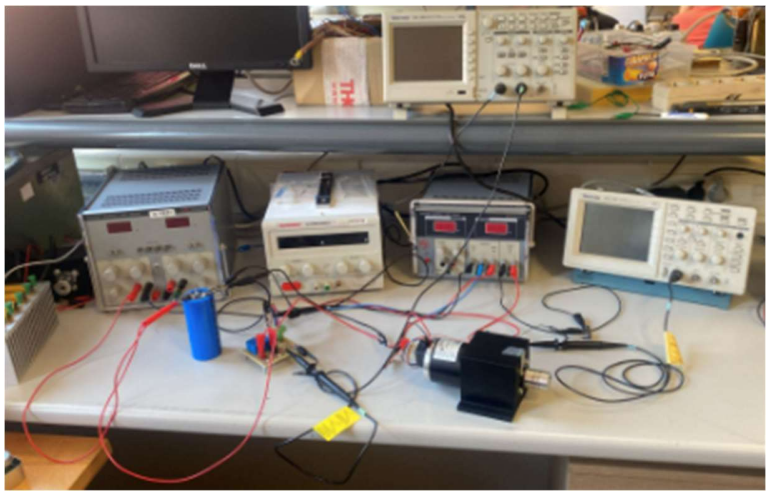


Figura 4.3 - Montaje en el laboratorio del modelo estático del motor en vacío.

Una vez realizado el montaje anterior procederemos a obtener los valores de las diferentes corrientes y velocidades.

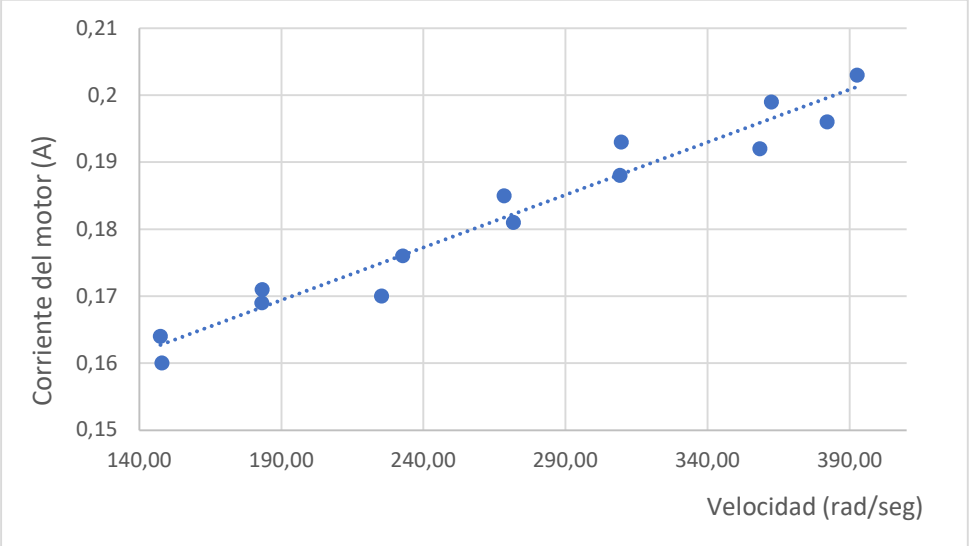
Tensión (V)	Corriente (mA)	Frecuencia (kHz)	Velocidad (rpm)	Velocidad angular (rad/s)
5,4	0,161	12,83	769,8	80,61
6,3	0,169	15,16	909,6	95,25
7,2	0,174	17,48	1048,8	109,83
9,0	0,179	21,90	1314,0	137,60
11,3	0,183	28,12	1687,2	176,68
13,5	0,190	33,73	2023,8	211,93
16,2	0,194	40,31	2418,6	253,28
18,1	0,196	46,29	2777,4	290,85
19,3	0,198	49,03	2941,8	308,06
20,1	0,201	51,04	3062,4	320,69

21,9	0,207	55,56	3333,6	349,09
22,3	0,209	56,78	3406,8	356,76
23,4	0,213	59,54	3572,4	374,10
24,0	0,215	60,99	3659,4	383,21

Tabla 4.1. Ensayo estático del motor de CC en vacío.

Conociendo el valor de la corriente media por cada punto y el valor de la corriente angular podemos conocer el valor de los parámetros a y b de la corriente del motor, que obtendremos mediante mínimos cuadrados representando la velocidad en rad/seg frente a la corriente media del motor.

$$I_M = a + b * \omega \tag{4.1.}$$



Gráfica 4.1 – Distribución de la corriente del motor frente a la velocidad.

De esta regresión por el método de mínimos cuadrados obtenemos los valores de a y b necesarios para el par inicial del motor, por tanto, obtendremos un valor de 0,1547 para a y un valor de 0,0001517 para b.

$$I_M = 0{,}1395 + 0{,}0001572x \tag{4.2.}$$

Para los dos siguientes ensayos el montaje realizaremos el mismo procedimiento, añadiendo una carga en serie con el motor para el segundo experimento. Para ello utilizaremos dos motores de CC idénticos, uno lo usaremos como motor y el otro como generador. Realizaremos el mismo montaje que para el modelado del motor del apartado anterior, una vez lo tengamos montado mediante el eje del motor le conectaremos a el otro motor el cual usaremos como generador.

4.2 MODELADO ESTÁTICO DEL MOTOR CON UN GENERADOR ACOPLADO EN VACÍO

Una vez conectado el motor y generador en este ensayo conectaremos las bornas del generador a un osciloscopio para poder medir la tensión del generador, en este caso será la misma que la fuerza electromotriz, no usaremos el osciloscopio para medir la corriente de éste ya que tendrá un valor nulo para cualquier tensión.

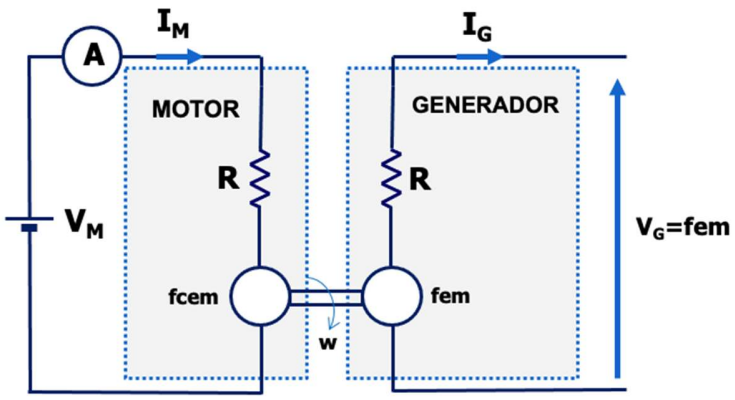


Figura 4.4 - Esquema del modelo estático del motor con generador acoplado en vacío.

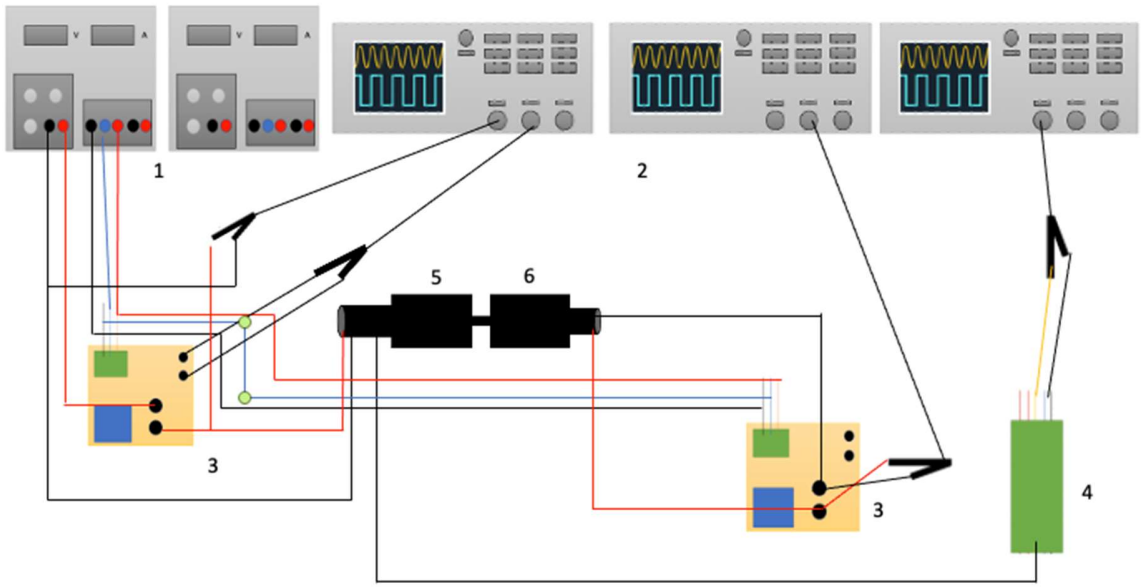


Figura 4.5 - Esquema del montaje del modelo estático del motor con un generador acoplado en vacío.

Donde:

1. Fuentes de tensión.
2. Osciloscopios.
3. Sonda de corriente.
4. Encoder.
5. Motor.
6. Generador.

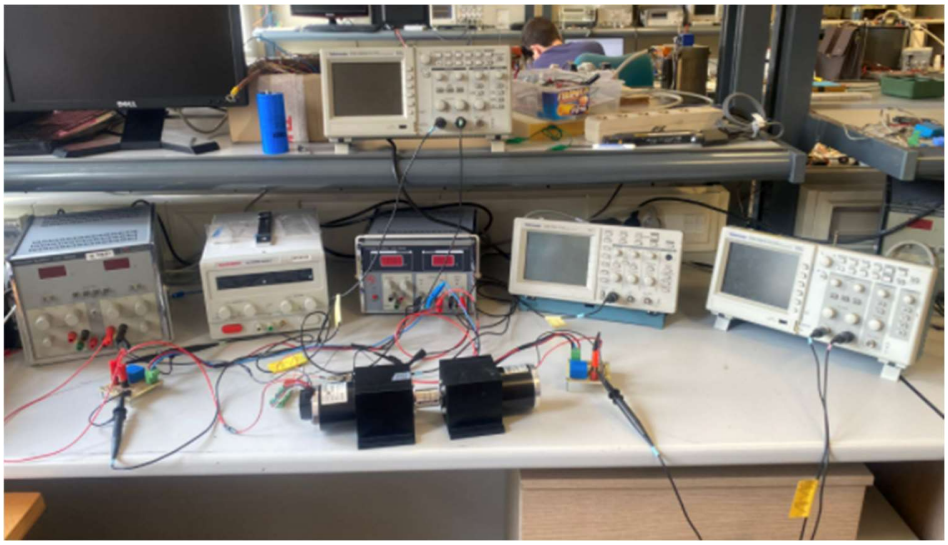


Figura 4.6 - Montaje en el laboratorio del modelo estático del motor con generador acoplado en vacío.

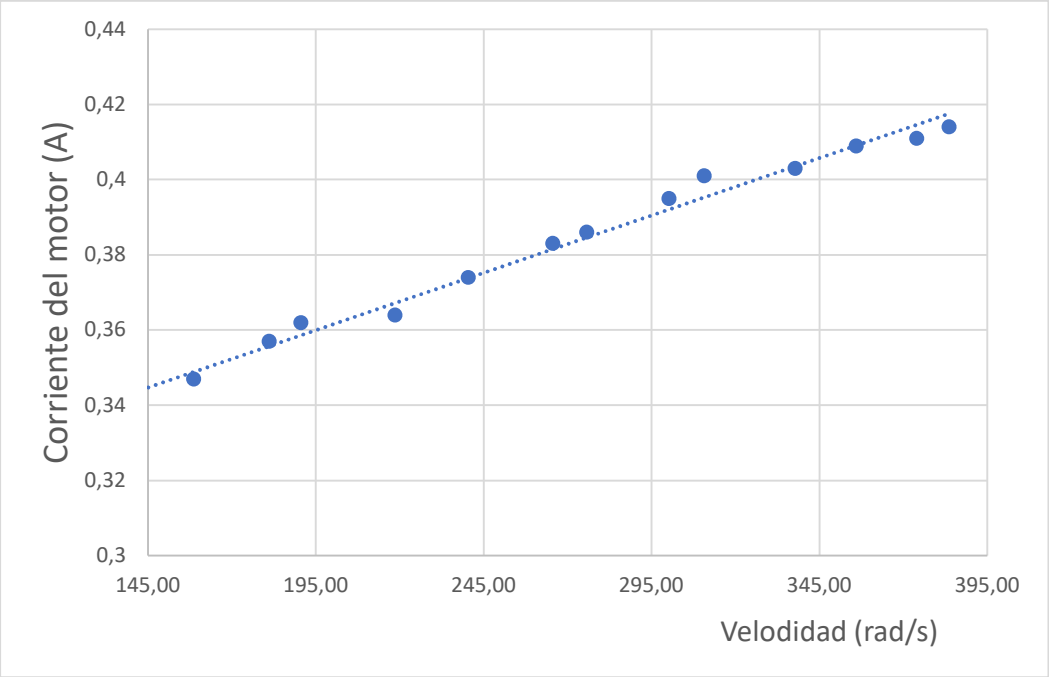
Una vez realizado el montaje iremos variando la tensión de alimentación del motor mediante la fuente de alimentación para poder conocer la corriente del motor $I_{m.}$ y poder así mediante la ecuación 4.3 saber el valor de la potencia de entrada. En el osciloscopio podremos leer la tensión de entrada en el generador V_g sabiendo que, la corriente del generador I_g es nula ya que está en vacio y no circula corriente por él, por lo que su potencia de salida también es nula. En el osciloscopio podemos leer también el valor de la frecuencia y como ya se ha explicado antes poder calcular el valor de la corriente angular en rad/s en cada punto.

$$P_{ent} = V_m * I_m \tag{4.3.}$$

Tensión del motor (V)	Corriente del motor (A)	Potencia de entrada (W)	Tensión del generador (V)	Frecuencia (kHz)	Velocidad (rad/s)
9,3	0,340	3,162	8,6	22,75	142,94
10,1	0,347	3,305	9,2	25,24	158,59
11,6	0,357	4,141	10,9	28,82	181,08

12,3	0,362	4,453	11,3	30,33	190,57
13,7	0,364	4,987	12,7	34,78	218,53
15,1	0,374	5,647	14,2	38,25	240,33
16,7	0,383	6,396	16,0	42,25	265,46
17,4	0,386	6,716	16,5	43,87	275,64
18,7	0,395	7,387	17,1	47,76	300,08
19,6	0,401	7,860	17,9	49,43	310,58
21,1	0,403	8,503	19,2	53,74	337,66
22,0	0,409	8,998	20,1	56,64	355,88
23,1	0,411	9,494	21,7	59,50	373,85
23,8	0,414	9,853	22,2	61,04	383,53

Tabla 4.2. Ensayo estático de un motor de CC con un generador acoplado en vacío.



Gráfica 4.2 – Distribución de la corriente del motor frente a la velocidad.

De esta regresión por el método de mínimos cuadrados obtenemos los valores de a y b necesarios para el par inicial del motor, por tanto, obtendremos un valor de 0,3004 para a y un valor de 0,0003 para b.

$$I_M = 0{,}3004 + 0{,}0003x \qquad (4.4.)$$

4.3 MODELADO ESTÁTICO DEL MOTOR CON UN GENERADOR ACOPLADO EN CARGA

Una vez conectado el motor y generador en este ensayo conectaremos las bornas del generador en serie con una resistencia de 12Ω y una sonda de corriente que conectaremos a un osciloscopio para poder medir la corriente del generador, para medir la tensión del condensador conectaremos el osciloscopio en paralelo a la resistencia, que en este caso será directamente proporcional al valor de la resistencia por la corriente, siendo la fuerza electromotriz la suma de los valores de la resistencia y la resistencia del generador, de valor 0,54 Ω calculada en el apartado 4, directamente proporcional a la corriente del generador.

$$fem = E = (R + R_L) * I_G \qquad (4.5.)$$

$$V_G = R_L * I_G \qquad (4.6.)$$

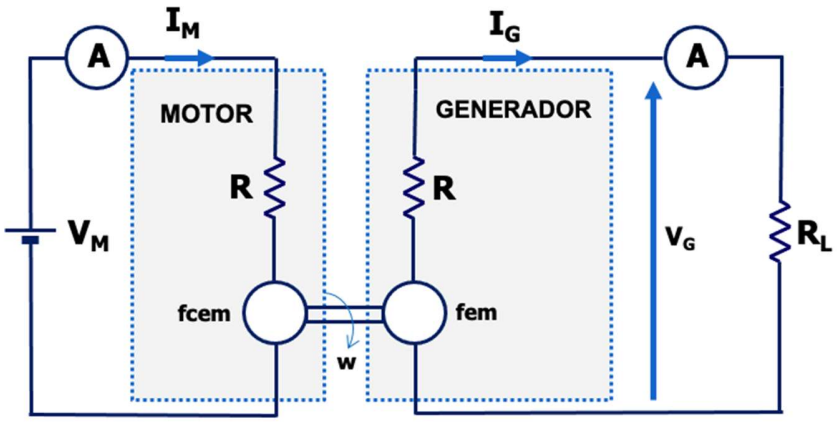


Figura 4.7 – Esquema del modelo del modelo estático del motor con un generador acoplado en carga.

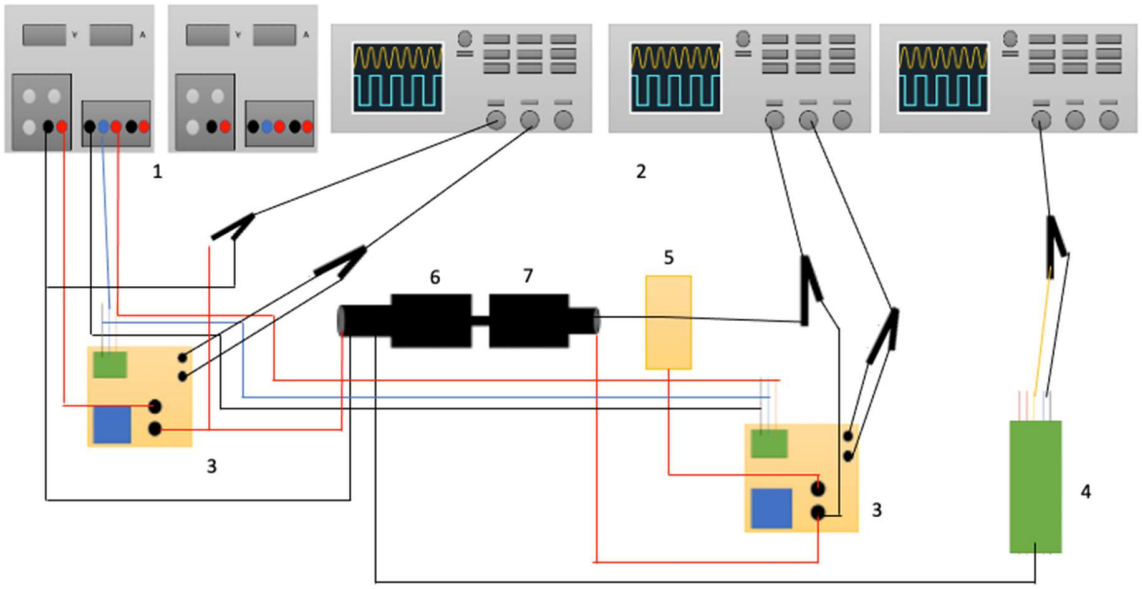


Figura 4.8 – Esquema del montaje del modelo estático del motor con un generador acoplado en carga.

Donde:

1. Fuentes de tensión.
2. Osciloscopios.
3. Sonda de corriente.
4. Encoder.
5. Resistencia.
6. Motor.
7. Generador.

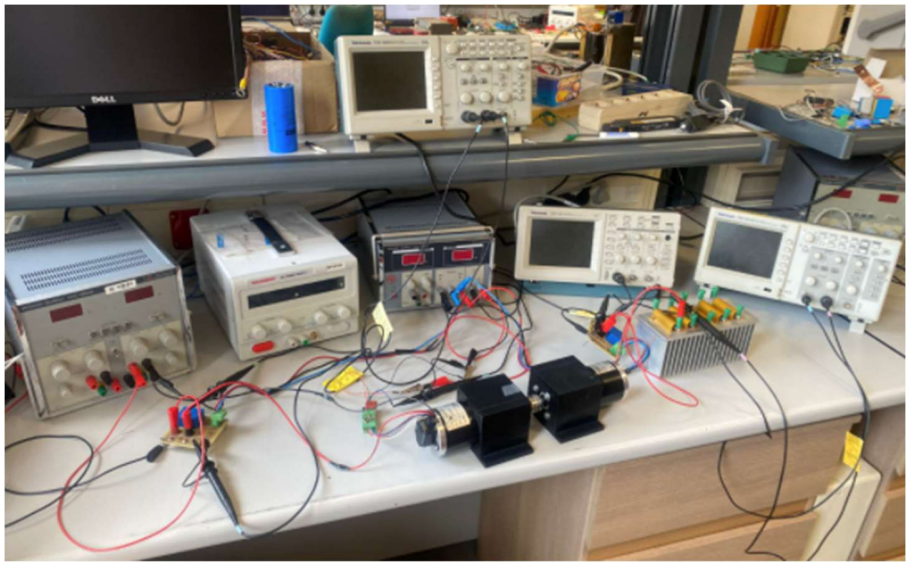


Figura 4.9 – Montaje en el laboratorio del modelo estático del motor con generador acoplado en carga.

Una vez realizado el montaje iremos variando la tensión de alimentación del motor mediante la fuente de alimentación para poder conocer la corriente del motor I_m, y poder así mediante la ecuación 4.3 saber el valor de la potencia de entrada. En el osciloscopio podemos leer la tensión en el generador V_g. En este caso también necesitaremos conocer la corriente por el generador I_g ya que no está en vacío y por el circulará ahora una corriente, por lo que la potencia de salida tampoco será nula. En el osciloscopio podemos leer también el valor de la frecuencia y como ya se ha explicado antes poder calcular el valor de la corriente angular en rad/s en cada punto.

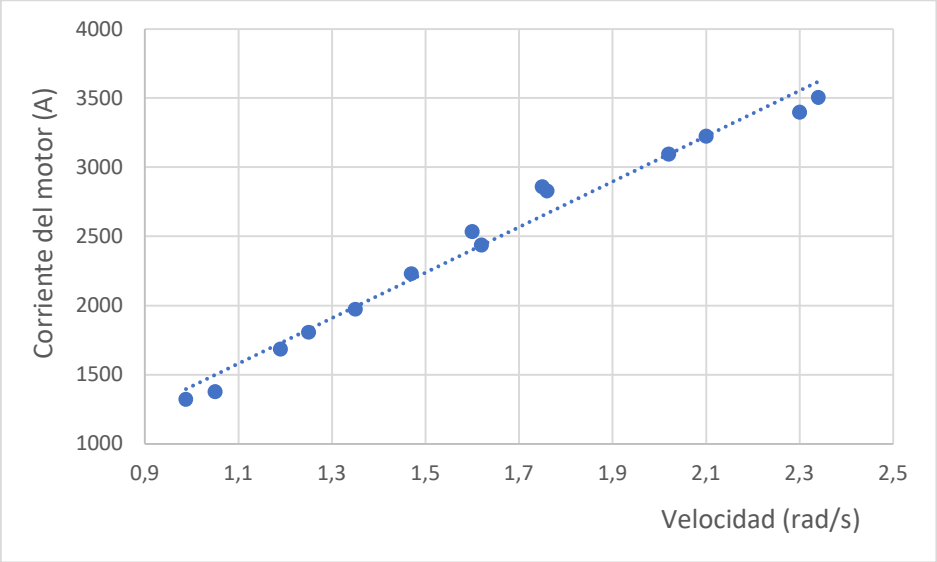
Tensión del motor (V)	Corriente del motor (A)	Potencia de entrada (W)	Tensión del generador (V)	Corriente del generador (A)	Potencia de salida (W)
9,3	0,988	9,190	7,98	0,649	5,18
10,1	1,05	10,61	8,08	0,664	5,37
11,6	1,19	13,80	10,4	0,817	8,50
12,3	1,25	15,38	10,8	0,853	9,21
13,7	1,35	18,50	12,2	0,967	11,80

15,1	1,47	22,20	13,2	1,050	13,86
16,7	1,62	27,05	14,3	1,220	17,45
17,4	1,60	27,84	14,7	1,280	18,82
18,7	1,76	32,91	16,8	1,440	24,19
19,6	1,75	34,30	17,5	1,460	25,55
21,1	2,02	42,62	18,9	1,530	28,92
22,0	2,10	46,20	19,1	1,640	31,32
23,1	2,30	53,13	20,7	1,730	35,81
23,8	2,34	55,69	21,3	1,820	38,77

Tabla 4.3. Ensayo estático de un motor de CC con un generador acoplado en carga [1].

Tensión del motor (V)	Corriente del motor (A)	Potencia de entrada (W)	Tensión del generador (V)	Frecuencia (kHz)	Velocidad (rad/s)
9,3	0,988	9,190	8,6	22,03	138,42
10,1	1,05	10,61	9,2	22,94	144,14
11,6	1,19	13,80	10,9	28,08	176,43
12,3	1,25	15,38	11,3	30,12	189,25
13,7	1,35	18,50	12,7	32,89	206,65
15,1	1,47	22,20	14,2	37,16	233,48
16,7	1,62	27,05	16,0	40,63	255,29
17,4	1,60	27,84	16,5	42,23	265,34
18,7	1,76	32,91	17,1	47,17	296,38
19,6	1,75	34,30	17,9	47,63	299,27
21,1	2,02	42,62	19,2	51,59	324,15
22,0	2,10	46,20	20,1	53,73	337,60
23,1	2,30	53,13	21,7	56,63	355,82
23,8	2,34	55,69	22,2	58,42	367,06

Tabla 4.4. Ensayo estático de un motor de CC con un generador acoplado en carga [2].



Gráfica 4.3 – Distribución de la corriente del motor frente a la velocidad.

De esta regresión por el método de mínimos cuadrados obtenemos los valores de a y b necesarios para el par inicial del motor, por tanto, obtendremos un valor de -299 para a y un valor de 1644,8 para b.

$$I_M = -299 + 1644{,}8x \qquad (4.7.)$$

4.4 CÁLCULO DE LOS PARÁMETROS DEL MOTOR DE CC

Una vez realizado el ensayo experimental y recogidos los valores de tensión de entrada del motor, corriente de entrada del motor, y los valores velocidad de giro, analizaremos el comportamiento del motor de CC con el generador acoplado en vacío ajustándolo por un modelo conocido que se representa en la siguiente ecuación.

$$V = R * I + E = R * I + k * \omega \qquad (4.8.)$$

Donde:

- V: es la tensión que se aplica al inducido.

- R: es la resistencia del devanado inducido.
- I: es la corriente por el inducido.
- E: es la fuerza contraelectromotriz.
- k: es la constante del motor.
- ω: es la velocidad angular en rad/seg.

Para obtener los valores de R y k más favorables se emplea el algoritmo de ajuste mediante mínimos cuadrados, mediante el cual obtendremos las dos siguientes ecuaciones.

$$k = \frac{\sum_{j=1}^{n}(V_j * I_j) * \sum_{j=1}^{n}(\omega_j * I_j) - \sum_{j=1}^{n}(V_j * \omega_j) * \sum_{j=1}^{n}(I_j{}^2)}{\left[\sum_{j=1}^{n}(\omega_j * I_j)\right]^2 - \sum_{j=1}^{n}(I_j{}^2) * \sum_{j=1}^{n}(\omega_j{}^2)} \quad (4.9.)$$

$$R = \frac{\sum_{j=1}^{n}(V_j * \omega_j) - k\sum_{j=1}^{n}(\omega_j{}^2)}{\sum_{j=1}^{n}(I_j * \omega_j)} \quad (4.10.)$$

Por lo que con los datos de la Tabla 4.2. obtendremos unos valores de k = 0,0622 y un valor de R de 0,469 Ω.

Del esquema del modelo estático sabemos,

$$fem = \varepsilon = k * w \quad (4.11.)$$

$$P_{ent} = V_m * I_m \quad (4.12.)$$

$$P_{sal} = V_g * I_g \quad (4.13.)$$

$$T_o = k * I_M \quad (4.14.)$$

$$T = 2 * T_o + k * I_g = 2 * T_o + T_{util} \quad (4.15.)$$

$$\eta_m = \frac{w * T_{util}}{V * I_m} \quad (4.15.)$$

$$\eta_g = \frac{V_g * I_g}{w * T_{util}} \quad (4.16.)$$

Por tanto, con los datos de la Tabla 3, podemos calcular el valor del rendimiento medio del motor que tiene un valor del 81,9% y el generador tendrá un rendimiento del 82,2%. La tabla completa se encuentra en el APÉNDICE II.

5. Empleo del driver y Arduino

5.1 QUÉ ES ARDUINO

Arduino es una plataforma que nos facilita el uso de la electrónica en todo tipo de proyectos. Se fundamenta en la filosofía del software libre y código abierto.

Se trata de una placa con un microcontrolador, pieza clave del dispositivo, que es una versión reducida de un ordenador. A su vez, se encuentra formado por: la unidad central de procesamiento, microprocesador, memoria y periféricos.

Consiste en un circuito integrado programable destinado a realizar una serie de tareas específicas relacionadas con el control: E/S (entrada-salida) y gestión de interrupciones, para poder interactuar con el mundo exterior.

Hay muchos tipos diferentes de placas disponibles en el mercado, cada una con su propio conjunto de características. Éstas se diferencian en cuanto a la memoria, puertos E/S, conectividad y velocidad de procesamiento, pero su funcionalidad es la misma. Algunas de ellas son:

- Arduino Mega
- Arduino Due
- Arduino Uno
- Arduino Mini
- Arduino Robot

Nosotros para realizar este proyecto utilizaremos el Arduino Mega.

Hay que tener en cuenta que cuando estemos hablando de Arduino estaremos haciendo referencia a tres cosas diferentes: Arduino Software, Arduino Hardware y Arduino Code. Hoy en día existen distintas clases de Arduino que se diferencian entre sí por la potencia, el número de entradas o la realización de funciones específicas.

Para indicar al hardware la tarea especifica que queremos realizar necesitaremos enviarle una serie de instrucciones, a través del lenguaje de programación. Esto lo haremos a través de un entorno de desarrollo integrado o IDE.

- Software:

Como se acaba de indicar el IDE nos proporcionará las herramientas que necesitaremos para desarrollar el código. Al ser una aplicación libre de multiplataforma es accesible desde Windows, Linux y macOS.

Arduino cuenta con su propio lenguaje de programación, Arduino Code basado en C++ incluyendo funciones y métodos específicos para el entorno.

- Hardware:

Será la placa de circuito impresa que incluye el microcontrolador junto con diferentes pines y conectores, como se puede observar en la siguiente ilustración. Para que ésta funcione será necesario conectarla a la corriente mediante un plug de alimentación. La entrada USB nos permitirá conectarla al dispositivo donde tenemos instalado el IDE para volcar el código en la placa.

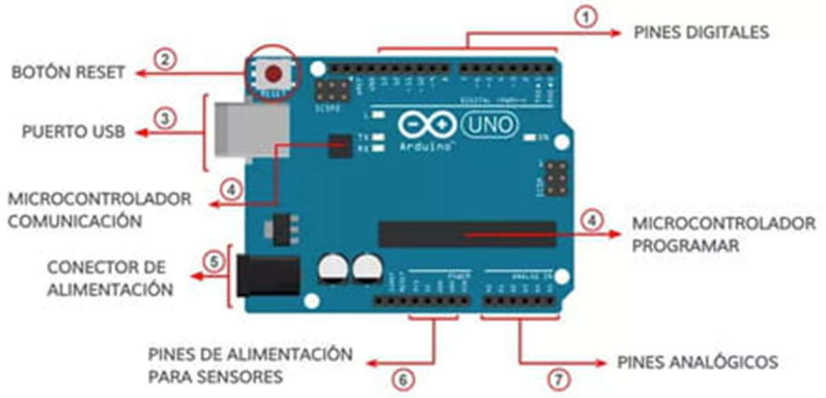


Figura 5.1 – Placa de circuito impreso de Arduino.

5.2 PORQUÉ UTILIZAR ARDUINO

Arduino nos permite realizar proyectos electrónicos con distintos grados de dificultad y a su vez es sencillo de manejar, tiene un coste relativamente bajo y existe una amplia gama de documentación en la web. Por lo que la facilidad y accesibilidad para aprender a usar la herramienta y el sencillo desarrollo de su software permite crear proyectos que posteriormente se pueden comercializar.

Una de sus ventajas más importantes es que permite cargar directamente los programas en el microcontrolador, sin la necesidad de un componente de hardware aparte. Únicamente tendremos que realizar las conexiones necesarias, desarrollar un programa a través del IDE y seleccionar la placa conectada en el puerto en el que se encuentra.

A continuación, tendrá lugar la compilación del programa y se cargará en el hardware correspondiente. Tras esto podremos desconectar la placa del dispositivo y únicamente tendremos que mantener la alimentación.

5.3 ENTORNO ARDUINO

El IDE está constituido por un editor de texto, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes y una serie de menús.

Arduino utiliza para escribir el código fuente o programa de aplicación lo que denomina *sketch*, programa. Estos programas son escritos en el editor de texto. También existe la posibilidad de cortar/pegar y buscar/remplazar texto.

En el área de mensajes se muestra la información mientras se cargan los programas y también muestra los errores de éste. La consola muestra el texto de salida, incluyendo los mensajes de error completos entre otras informaciones. La barra de herramienta permite verificar el proceso de carga, creación, apertura y guardado de programas:

- Verify/Compile: chequea el código en busca de errores.
- New: crea un nuevo sketch.
- Stop: finaliza la monitorización serie y oculta otros botones.
- Open: presenta un menú de todos los programas de tu librería de sketch.
- Save: guarda el programa.
- Serial Monitor: inicia la monitorización serie.
- Upload to I/O Board: compila el código y lo vuelca en la placa E/S de Arduino.

Las partes principales de un programa hecho en Arduino son:

- Bloque de inclusión de módulos y declaración de variables.
- Bloque de configuración void setup(), donde se indica el modo de funcionamiento de los pines (entrada y salida).
- Comunicación serie, que se utiliza para inicializar la configuración de la placa.

Algunas de las funciones más utilizadas en la programación de Arduino son:

- pinMode: establece el modo de pinde entrada o salida.
- analogRead: lee un voltaje analogico de un pin de entrada analógica.
- analogWrite: escribe un voltaje analogico a un pin de salida analógica.
- digitalRead: lee un valor de un pin de entrada digital.
- digitalWrite: establece el valor de un pin de salida digital para alta o baja.

Como ejemplo, se ha realizado un programa para atenuar un LED desde Arduino.

```
const int LED=5; // Definición de variable y se la asignamos al pin 5
void setup () // Programa de configuración de entradas y salidas
{
        pinMode (LED, OUTPUT); // Hacer que el pin del LED sea salida
}
void loop () // Programa ejecutable
{
        for (int i=0; i<256; i++)
                {
                        analogWrite (LED, i);
                        delay (10);
                }
                for (int i=255; i>=0; i=--)
                {
                        analogWrite (LED, i);
                        delay (10);
                }
}
```

Este programa aumenta y disminuye gradualmente el brillo de un LED utilizando modulación de ancho de pulso (PWM) a través de la función analogwrite ().

1. const int LED=5; declara una variable entera constante denominada LED conectada al pin número 5.
2. void setup (); es la función que únicamente se realiza una vez cuando se enciende el Arduino. En este caso, establece el modo del pin LED en OUTPUT utilizando la función pinMode (), que permite controlar el voltaje en el pin.
3. void loop (); es la función que se ejecutara de forma infinita siempre que el Arduino esté encendido.
4. El bucle for de las líneas 8-12 aumenta gradualmente el brillo del LED de 0 a 255, utilizando la función analogWrite(). El primer argumento es el número de pin (en este caso, LED), y el segundo argumento es el ciclo de trabajo PWM, que determina el brillo del LED. Un ciclo de trabajo 0 no produce voltaje de salida (el LED está apagado). Mientras que un ciclo de trabajo 255 produce un voltaje de salida constante (el LED está completamente encendido).
5. La llamada a la función delay (10) en la línea 10 provoca un retraso de 10 milisegundos entre cada llamada analogWrite (), lo que ralentiza la velocidad de cambio y hace que el brillo del LED cambie más suavemente.
6. El bucle for de las líneas 14-18 disminuye el brillo del LED gradualmente de 255 a 0, en pases de 1, utilizando las mismas funciones analogWrite () y delay ().
7. La función loop () se repite desde el principio, por lo que el brillo del LED aumenta y disminuye gradualmente en un bucle.

Hay que tener en cuenta que la función analogWrite () solo funciona en ciertos pines de la placa Arduino que soportan PWM. En la mayoría de las placas estos son los pines 3,5,6,9,10 y 11, los que pueden usarse para la salida PWM.

5.4 CONTROL DE VELOCIDAD Y DIRECCIÓN DE UN MOTOR CC

Para controlar la velocidad y dirección del motor CC se ha utilizado el driver L298N, un motor, una fuente de alimentación y una placa de Arduino Mega.

Lo primero que se debe hacer es crear un programa en lenguaje Arduino que controle la velocidad y dirección del motor para posteriormente volcarla en la placa. Este programa será el siguiente.

```
int IN1=11;
int IN2=12;
void setup ()
{
        pinMode (11, OUTPUT);
        pinMode (12, OUTPUT);
        TCCR1B = (TCCR1B&0xF8)|1;
}
void loop ()
{
        analogWrite (11, 200);
        analogWrite (12, 0);
        delay (5000);

        analogWrite (11, 0);
        analogWrite (12, 0);
        delay (5000);

        analogWrite (11, 0);
        analogWrite (12, 125);
        delay (5000);
}
```

El motor estará conectado a los pines IN1 e IN2 del driver. Las variables IN1 e IN2 se definen como enteros y se establecen en 11 y 12, respectivamente. Estas variables se utilizan más adelante en el código para referirse a los pines conectados al módulo del driver.

1. La función setup () se llama una vez cuando se inicia el programa y se utiliza para inicializar los modos de pin. En este caso se establece los modos de pin de IN1 e IN2 como salidas utilizando la función pinMode ().
2. La línea TCCR1B=(TCCR1B&0xF8)|1; establece la frecuencia PWM de los pines 11 y 12 a 31,25 kHz, que es la frecuencia más alta que se puede lograr con la placa Arduino Mega.
3. La función loop () se llama repetidamente después de la función setup () y se utiliza para ejecutar la lógica principal del programa. En este caso, se utilizan tres funciones analogWrite () para controlar la velocidad y dirección del motor mediante modulación de ancho de pulso (PWM).
4. La función analogWrite () toma dos argumentos: el primero es el numero pin y el segundo es el ciclo de trabajo. El ciclo de trabajo es un valor entre 0 y 255, donde 0 representa el 0% del ciclo de trabajo (motor apagado) y 255 representa el 100% del ciclo de trabajo (motor a máxima velocidad). El valor pasado a la función analogWrite () determina la velocidad y la dirección del motor.
5. La primera función analogWrite () establece el valor de IN1 en 200 y el valor de IN2 en 0. Lo que hará que el motor gire en una dirección a una velocidad moderada durante 5 segundos. La segunda función analogWrite () establece el valor de IN1 e IN2 en 0, lo que detendrá el motor durante 5 segundos. La tercera analogWrite () establece el valor IN1 en 125. Lo que hará que el motor gire en la dirección opuesta a una velocidad más lenta durante 5 segundos.

Una vez tenemos el programa de Arduino comenzamos con el montaje, Ilustración 5.4.2. Como se va a alimentar el motor a una tensión inferior a 12V, el Jumper del driver debe estar conectado.

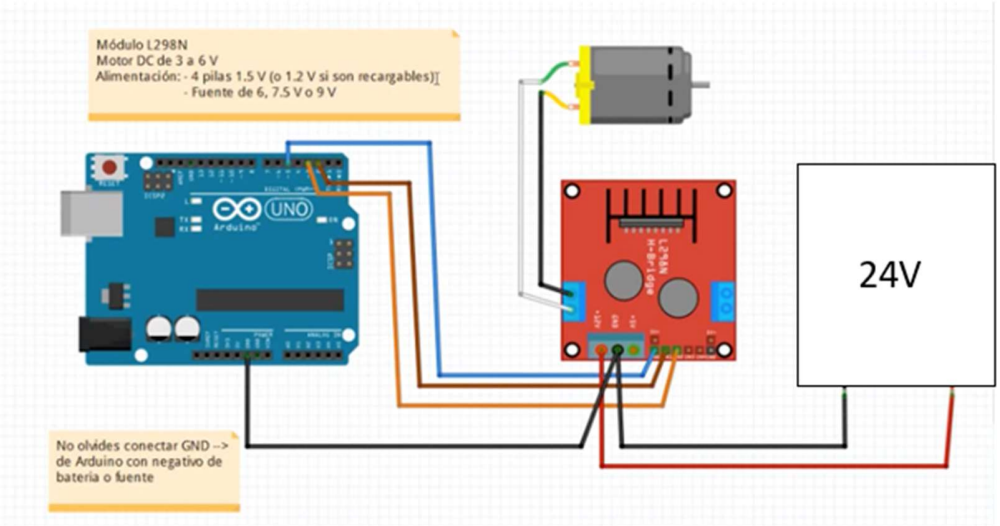


Figura 5.2 – Esquema de montaje para el control de velocidad y dirección del motor CC.

Conectaremos el motor a la borna OUTPUT A del driver, éste será alimentado desde una fuente externa de tensión a los pines +12V y GND. Una vez tengamos alimentado el driver lo conectaremos a la placa Arduino, por lo que el GND de ésta debe ir con el GND de la alimentación del driver. Conectaremos el pin11 a IN1 y el pin12 a IN2.

Una vez realizado el montaje se vuelca el programa sobre la placa y podemos observar como el motor gira a la velocidad indicada el tiempo deseado, realiza una parada y posteriormente gira en sentido contrario.

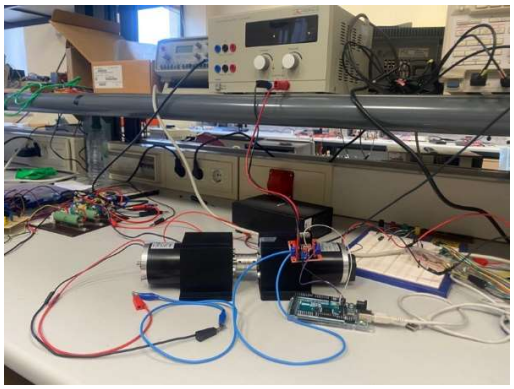


Figura 5.3 – Montaje para el control de velocidad y dirección del motor CC realizado en el laboratorio.

Al realizar el montaje y poner a prueba el motor en vacío con una corriente de 0,2 A, nos damos cuenta de que el driver tiene una temperatura muy elevada. Por lo que, cuando alimentemos nuestro motor a la corriente de 0,5 A, nuestro driver se quemará. Concluimos que debemos diseñar nuestro propio driver para la aplicación deseada.

6. Diseño del driver

6.1 QUÉ ES ALTIUM DESIGNER

El diseño del accionamiento del motor o driver lo realizaremos con el programa Altium Designer ya que es una potente herramienta de diseño electrónico que permite crear esquemas electrónicos utilizando una amplia biblioteca de componentes y facilita la conexión entre ellos mediante conexiones eléctricas definidas. También permite diseñar placas de circuito impreso, PCB, mediante la disposición de componentes y el enrutamiento de conexiones eléctricas.

6.2 DISEÑO DEL DRIVER

Para el diseño del driver debemos tener en cuenta que éste actúa como un amplificador de potencia, que puede proporcionar la corriente necesaria al motor en función de las señales de control que recibe.

Nuestro driver constará de un puente completo, el cual está compuesto por cuatro interruptores controlados, MOSFETs, dispuestos de manera que permitan dirigir la corriente a través de la carga en ambas direcciones. Estos interruptores se disponen en dos ramas, las cuales se controlan de manera coordinada para invertir la polaridad o aplicar la corriente en la dirección deseada.

Cada rama está compuesta de dos MOSFETs de manera que cuando entre puerta (G) y surtidor (S) tengamos una tensión de 15V el MOSFET estará cerrado y en el caso de tener una tensión de 0V estaría abierto. De tal manera que si representamos gráficamente como quedaría la tensión respecto del tiempo,

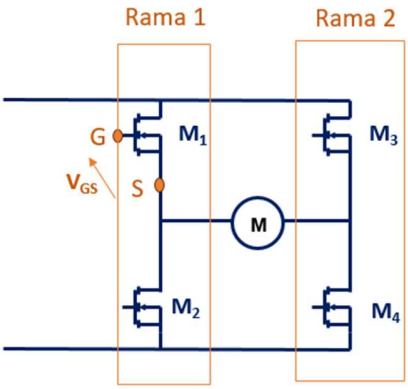


Figura 6.1 - Esquema de puente completo.

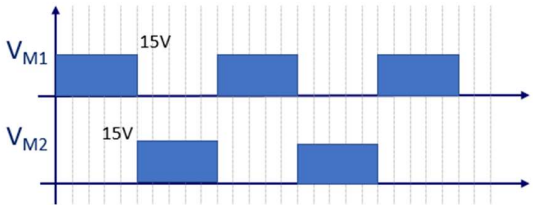


Figura 6.2 - Tensión entre puerta y surtidor los MOSFETs en función del tiempo.

Sabemos que abren y cierra primero uno y después el otro, analizamos una de las ramas, sabiendo que la otra tiene el mismo comportamiento. Por lo tanto, sabemos que cuando M1 esté abierto, M2 estará cerrado y al contrario. Comenzamos analizando el primer caso que es cuando M1 esté abierto.

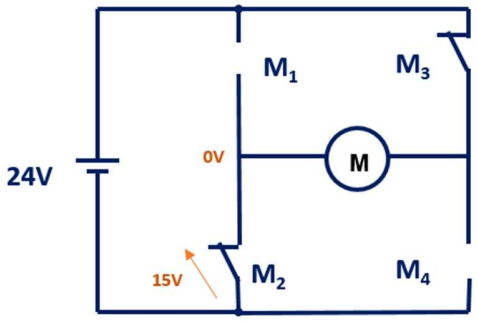


Figura 6.3- Esquema de puente completo cuando M1 está abierto.

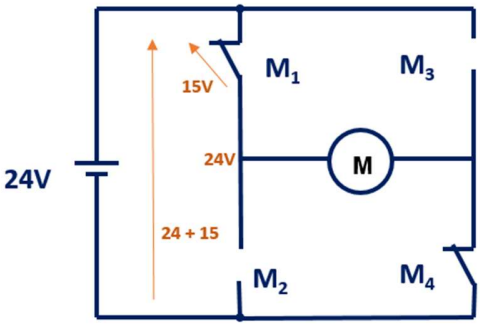
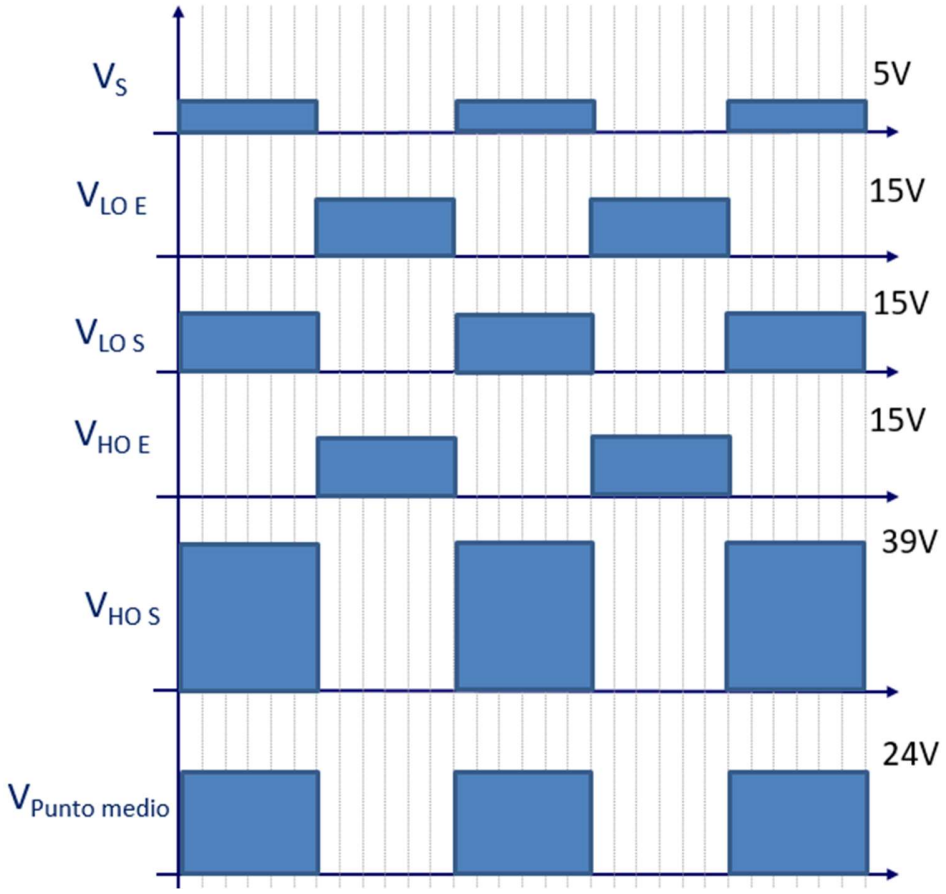


Figura 6.4 - Esquema de puente completo cuando M2 está abierto.

El driver necesita que la tensión de entrada sea de al menos 9V, sin embargo, al estar alimentándose desde Arduino (5V) la tensión de alimentación es insuficiente para el driver. De manera que, necesitaremos un amplificador operacional que trabaje en régimen de saturación para elevar la tensión de alimentación a 15V. Este sistema se replicará por cada rama del puente completo.

Seguidamente el driver nos proporcionará dos señales: una “HO” y otra “LO”. Esto significa que en nuestro esquema eléctrico necesitaremos un MOSFET IRF2111 para cada una de estas señales.

Los distintos valores de tensión de entrada y salida que proporcionan cada una de las señales se reflejan en la siguiente gráfica:



Gráfica 6.1 - Relación de tensiones de entrada y salida de las señales HO y LO.

Donde:

- Vs corresponde con la tensión de salida del amplificador operacional
- V_{LO} corresponde con la tensión de la señal LO.
- V_{HO} corresponde con la tensión de la señal HO.
- $V_{Punto\ medio}$ corresponde con la tensión de la señal del punto medio.

Los subíndices S y E corresponde a salida y entrada.

Al estar utilizando un puente completo podemos controlar el sentido de giro del motor dependiendo del duty con el que trabajemos. Por lo que, si conocemos el duty utilizado sabremos en qué sentido está girando el motor.

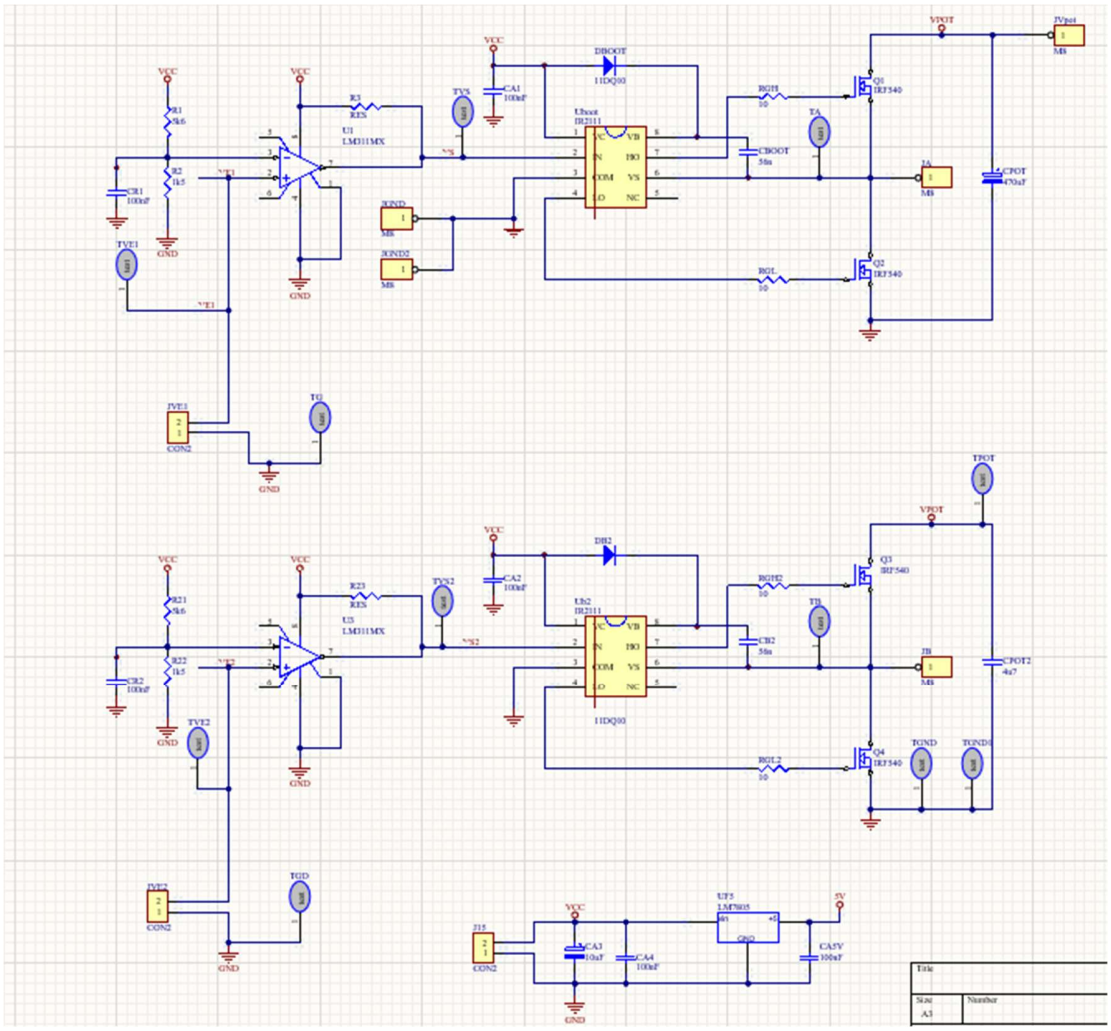


Figura 6.5 - Esquema eléctrico en Altium.

Una vez finalizado el esquema eléctrico comenzaremos realizando la PCB, el programa nos proporciona un ejemplo de colocación de los componentes, que tendremos que organizar de la manera más óptima posible y realizar las pistas por las que posteriormente fluirá la corriente eléctrica, al igual que realizar los planos de masa correspondientes.

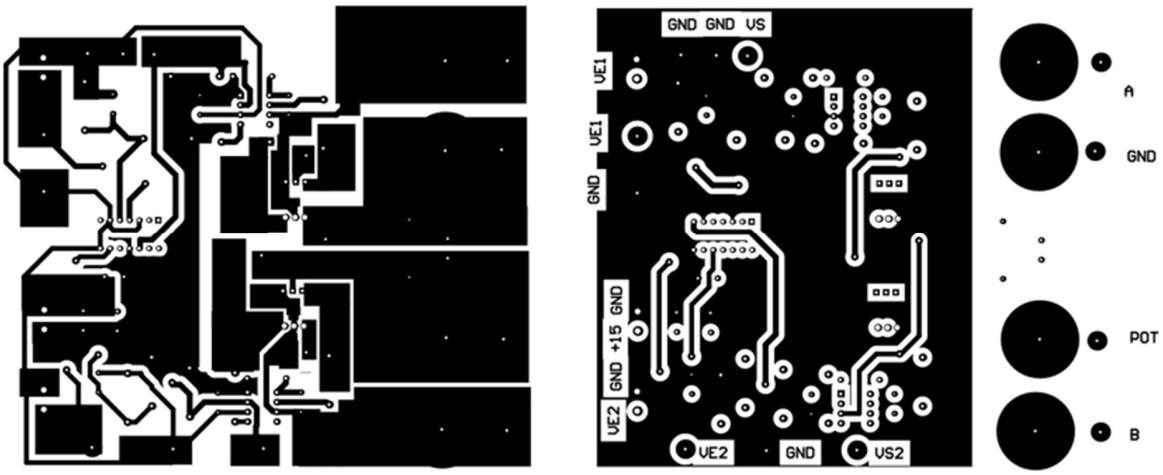


Figura 6.6 - Cara inferior y superior de la placa PCB junto a sus pistas y planos de masa correspondientes.

6.3 MONTAJE DEL DRIVER

Una vez que tenemos diseñado el esquemático y la PCB se comenzará a construir el driver para lo cual lo primero será imprimir la placa.

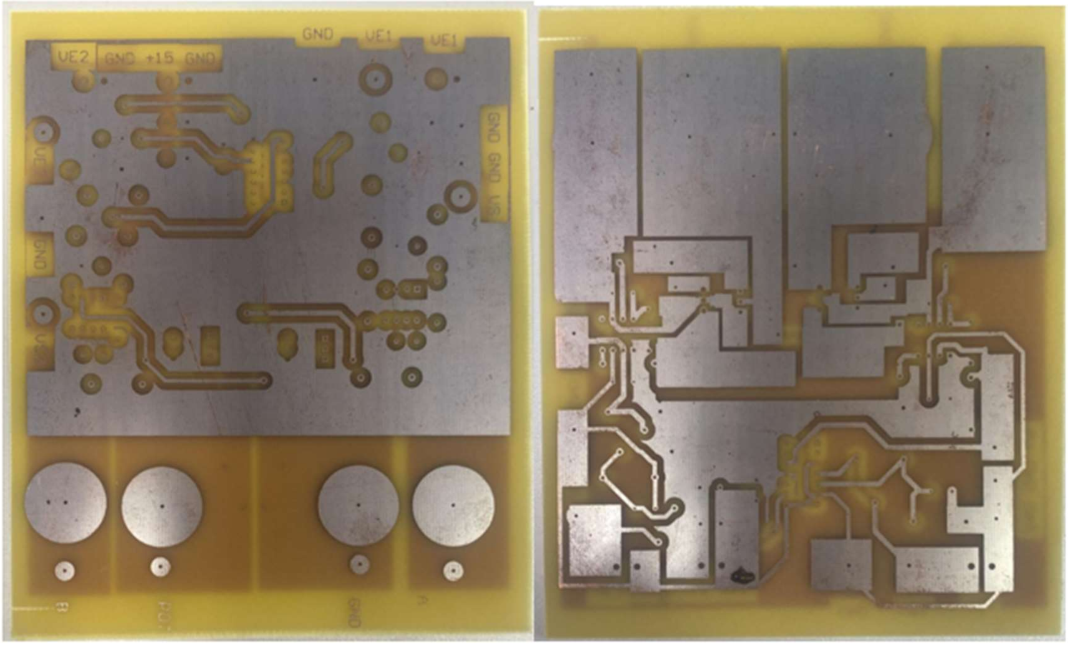


Figura 6.7 Cara superior e inferior de la PCB impresa.

Una vez que eliminemos el barniz que protege al cobre de todas las pistas de la placa para permitir que fluya la corriente eléctrica por toda la placa comenzaremos a taladrar ésta.

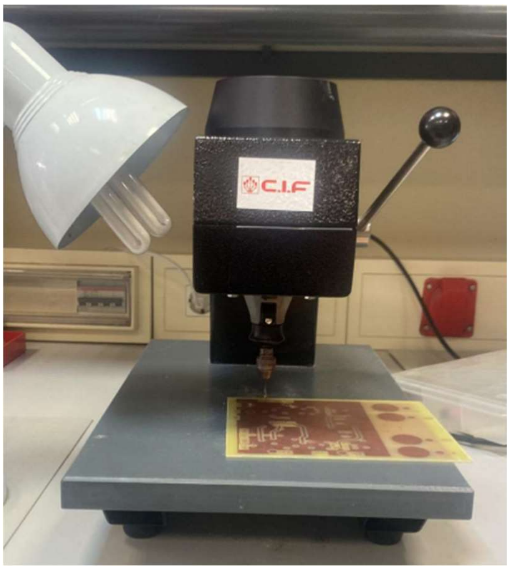


Figura 6.8- Ejemplo de taladrar la PCB.

Para poder entonces soldar con el soldador de estaño componente a componente.

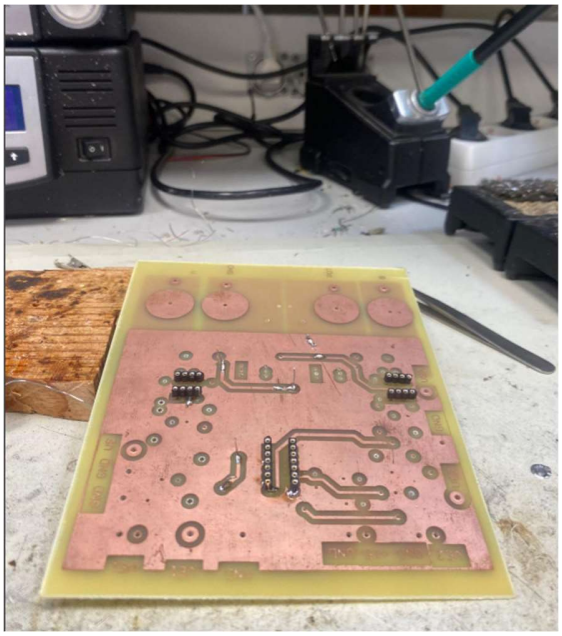


Figura 6.9 - Imagen de como quedan los componentes soldados.

Por último, una vez que tengamos todos los componentes soldados comprobaremos con ayuda de un osciloscopio el funcionamiento de ésta.

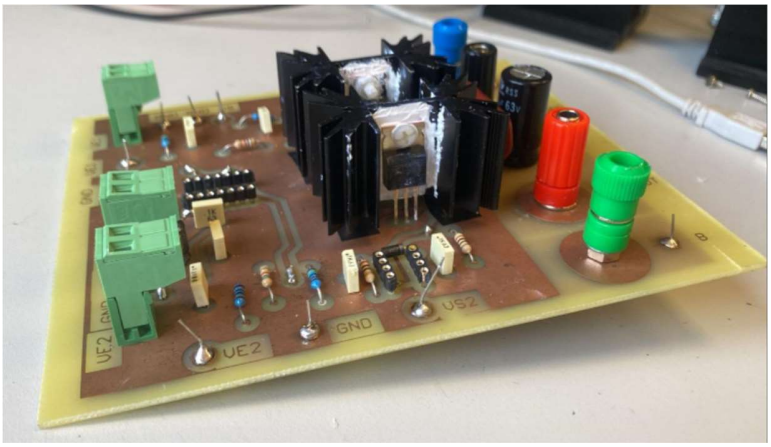


Figura 6.10- Imagen de la placa finalizada.

Comenzaremos estudiando el comportamiento de la parte inferior del esquema. De manera que leeremos en el osciloscopio las patillas 4, 7, el punto medio, la tensión de entrada y salida.

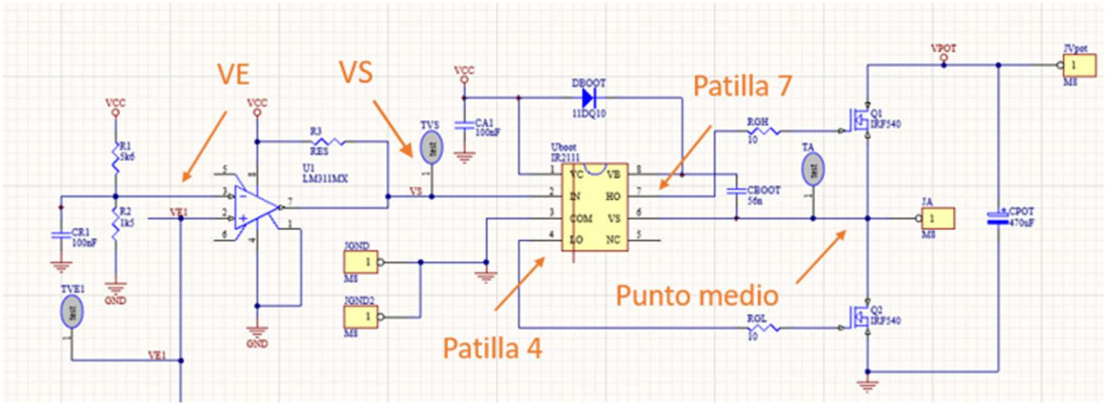


Figura 6.11 - Puntos para la comprobación del funcionamiento del driver.

En la forma de onda inferior podemos observar que la tensión de alimentación de entrada al operacional es de 5V como esperábamos ya que es alimentado por el Arduino.

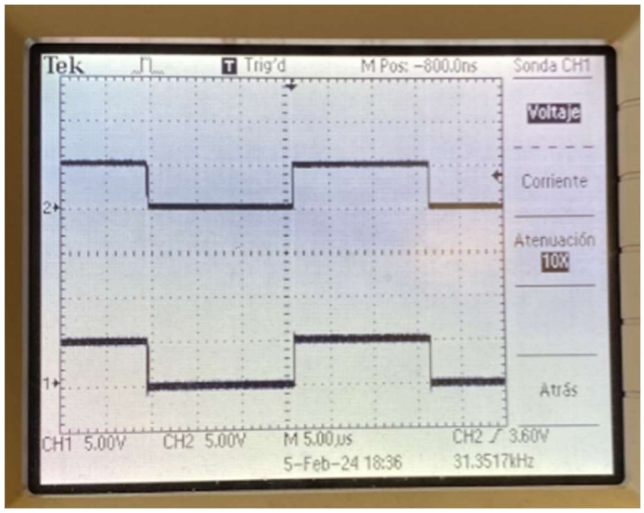


Figura 6.12 -Arriba tensión de entrada al amplificador operacional y abajo la tensión salida de Arduino con niveles de 0 – 5V.

Como ya se ha explicado anteriormente la tensión en la salida, VS, del amplificador debe de ser de 0-15V, como se puede comprobar en el osciloscopio.

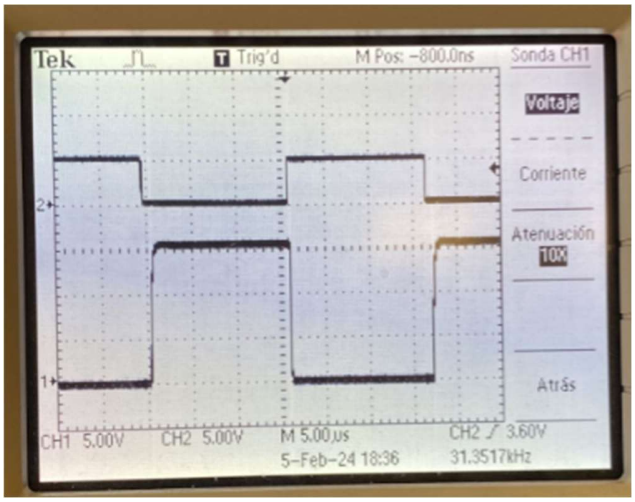
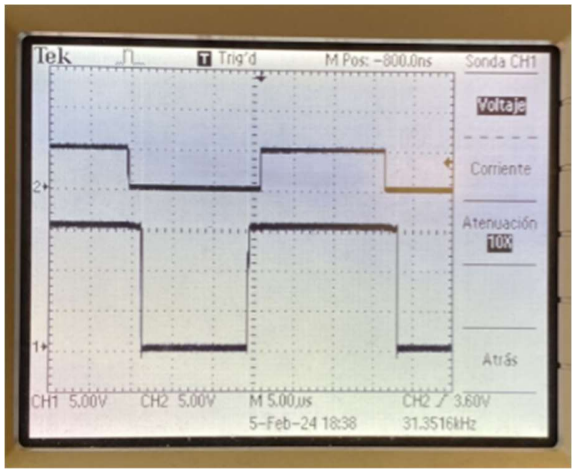


Figura 6.13. Arriba tensión de entrada al amplificador operacional con niveles de 0 – 5V. Abajo tensión de salida del amplificador operacional con niveles de 0 – 15V.

En la patilla 4 corresponde con la entrada a la señal “LO” por lo que se espera una tensión de 0-15V contraria a la tensión de salida del amplificador.



6.14- Arriba tensión de entrada al amplificador operacional con niveles de 0 – 5V. Abajo la tensión de entrada a la señal “LO” con niveles de 0 – 15V.

En la patilla 7 corresponde con la salida de la señal “LO” por lo que se espera una tensión de 0-15V igual a la tensión de salida del amplificador.

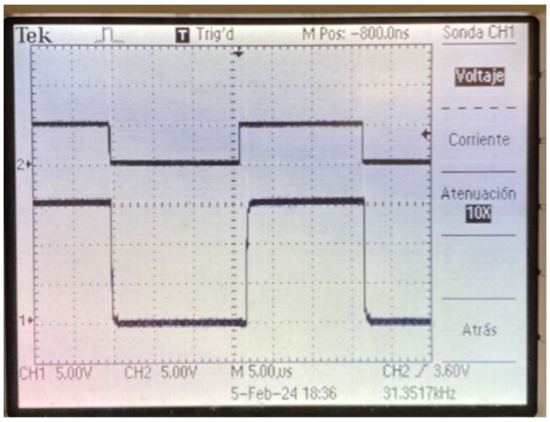


Figura 6.14- Arriba tensión de entrada al amplificador operacional con niveles de 0 – 5V. Abajo la tensión de salida a la señal "LO" con niveles de 0 – 15V.

En el punto medio se espera una tensión de 0-24V, en el cual se puede apreciar ruido generado por la sonda del osciloscopio principalmente.

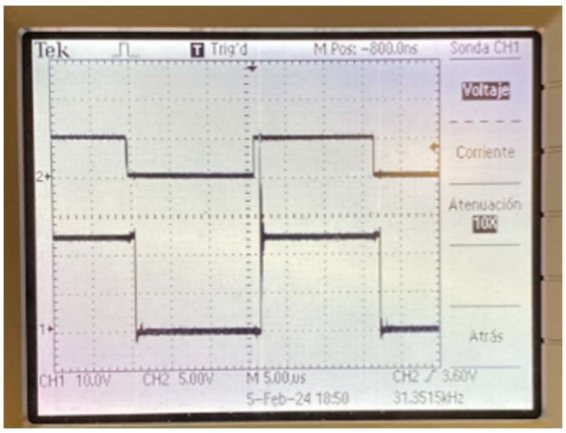


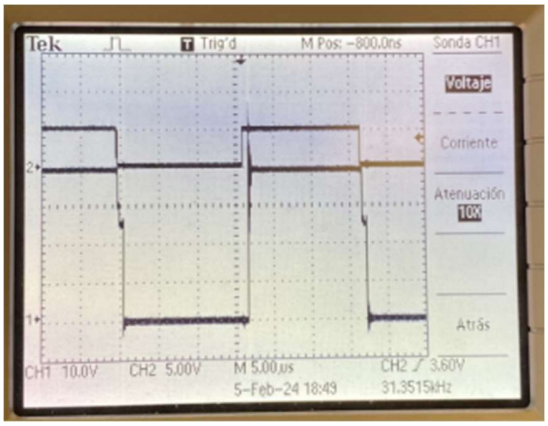
Figura 6.15 - Arriba tensión de entrada al amplificador operacional con niveles de 0 – 5V.

Abajo la tensión en el punto medio con niveles de 0 – 24V.

De la misma manera comprobaremos las tensiones en la parte superior del esquema. Donde en la patilla 4 corresponde con la entrada a la señal "HO" por lo que se espera una tensión de 0-15V contraria a la tensión de salida del amplificador.

6.16- Arriba tensión de entrada al amplificador operacional con niveles de 0 – 5V. Abajo la tensión de entrada a la señal "HO" con niveles de 0 – 15V.

En la patilla 7 corresponde con la salida de la señal "HO" por lo que se espera una tensión de 0-39V como se ha explicado anteriormente.



6.17- Arriba tensión de entrada al amplificador operacional con niveles de 0 – 5V. Abajo la tensión de salida a la señal "HO" con niveles de 0 – 39V.

En conclusión, las lecturas realizadas con el osciloscopio corresponden con los valores teóricos esperados, tal y como se muestra en la Gráfica 6.1.

7. Simulink y Arduino

Para el modelado y simulación del modelo dinámico del motor utilizaremos la herramienta Simulink de MATLAB.

MATLAB es una plataforma de programación y cálculo numérico para analizar datos, desarrollar algoritmos y crear modelos matemáticos. Utilizaremos este software para el diseño de controladores, simulación de sistemas dinámicos y análisis de circuitos con su herramienta Simulink.

Simulink ofrece el soporte para la programación y simulación de nuestro motor. Además de contar con unas características generales muy útiles para nuestro proyecto:

- Modelado gráfico: Permite construir modelos de sistemas dinámicos utilizando bloques gráficos, éstos representan los componentes del sistema y las conexiones que existen entre ellos, facilitando así la representación visual del sistema.
- Simulación en tiempo real: Permite simular el comportamiento de nuestro sistema a lo largo del tiempo, pudiendo observar cómo evoluciona el sistema si variamos las condiciones.
- Análisis y visualización: Proporciona herramientas para analizar y visualizar los resultados de la simulación, incluyendo gráficos en 2D y 3D.
- Biblioteca de bloques: Ofrece una amplia biblioteca de bloques predefinidos que representan componentes comunes de sistemas de control mecánico, y electrónicos. Aunque sean bloques predefinidos podemos personalizarlos y combinarlos para construir nuestro modelo.
- Desarrollo de Sistemas de Control: Clave para realizar el diseño y la simulación de nuestro sistema de control. Además, podemos modelar y simular sistemas de control PID o sistemas dinámicos.
- Integración con MATLAB: Como comentábamos al inicio del apartado, Simulink se integra estrechamente con MATLAB, permitiéndonos combinar el poder del modelado

gráfico de Simulink con las capacidades de programación y análisis numérico de MATLAB.

Otra de las características principales de Simulink es la generación de código y hardware en tiempo real. Esto quiere decir, que nos ofrece el soporte necesario para la programación y simulación de algoritmos diseñados para plataformas de hardware, como Arduino, a través de su interfaz con MATLAB. La integración de Arduino y Simulink nos permite desarrollar modelos y algoritmos en Simulink y luego implementarlos directamente en una placa Arduino para su ejecución en hardware.

El funcionamiento de Arduino pasa por interactuar con los pines de entrada y salida, realizar las lecturas analógicas y generar las salidas digitales necesarias de nuestro control. Para interpretar esta información desde Simulink, es necesario utilizar bloques como “Digital Input” o “Analog Output”.

Por último, una vez generado el código necesario para el control de velocidad, se descargará directamente a la placa Arduino. Seguidamente, se podrá desconectar del ordenador y alimentarla de manera autónoma para ejecutar el algoritmo, realizando el monitoreo y depuración desde el propio Simulink.

En conclusión, Simulink es una herramienta muy útil para facilitarnos el modelado y la simulación de sistemas dinámicos complejos mediante un enfoque gráfico e intuitivo. Permitiéndonos además operar con la plataforma Arduino para la ejecución del algoritmo en un hardware independiente.

8. Incorporación del Encoder

Como comentábamos en el apartado 3 “Descripciones del Hardware”, añadiremos un Encoder HEDM-5600-B12 a nuestro motor de corriente continua.

En un motor de CC el encoder nos proporcionará la información precisa de la velocidad angular del rotor, lo que nos permite ajustar la velocidad de manera óptima. Para ello, hemos desarrollado el siguiente modelo para comprobar el correcto funcionamiento del Encoder. Comparando las rpm leídas con un osciloscopio y las leídas en el osciloscopio digital de Simulink. Este paso es fundamental para asegurar el correcto control de velocidad del motor.

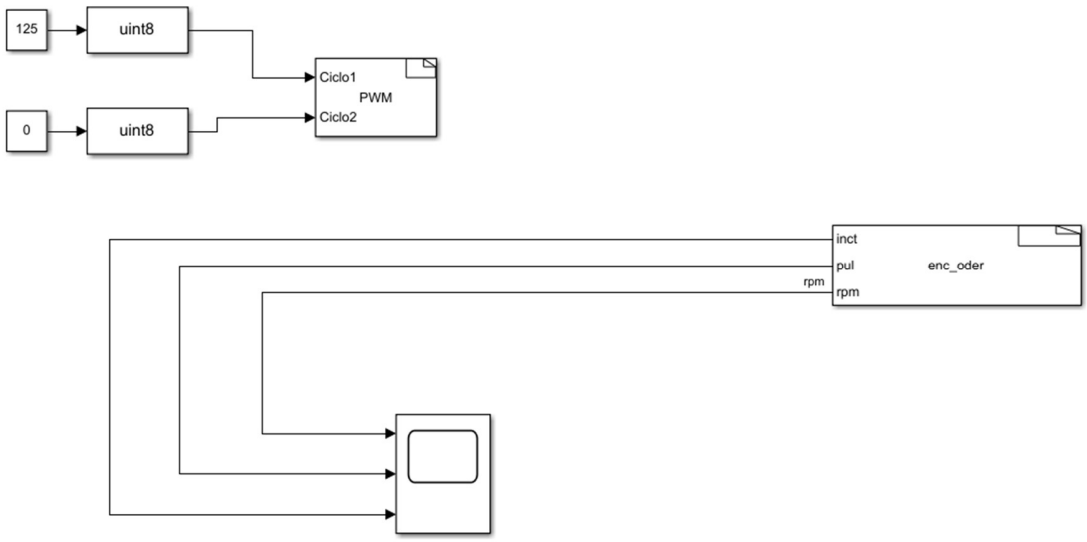


Figura 8.1 - Modelo en Simulink para comprobar el correcto funcionamiento del encoder.

Este modelo está formado por dos partes, el bloque de PWM y el bloque encoder. El bloque PWM consta de dos constantes, en este caso 125 y 0, las cuales son las constantes del duty que queremos en nuestro motor y de un bloque S-Function Builder.

Un bloque S-Function Builder o Constructor de funciones S es una herramienta de Simulink la cual nos permite crear bloques de función personalizados utilizando en nuestro caso código de Arduino. Este bloque nos proporcionará una interfaz gráfica para definir y configurar manualmente todo el código, ya que no hay ningún bloque estándar en Simulink que realice la función que queremos.

Este bloque consta de 5 partes diferenciadas:

- Barra de herramientas
- Panel de configuración
- Tabla de parámetros y puertos
- Tabla de librerías
- Editor de código

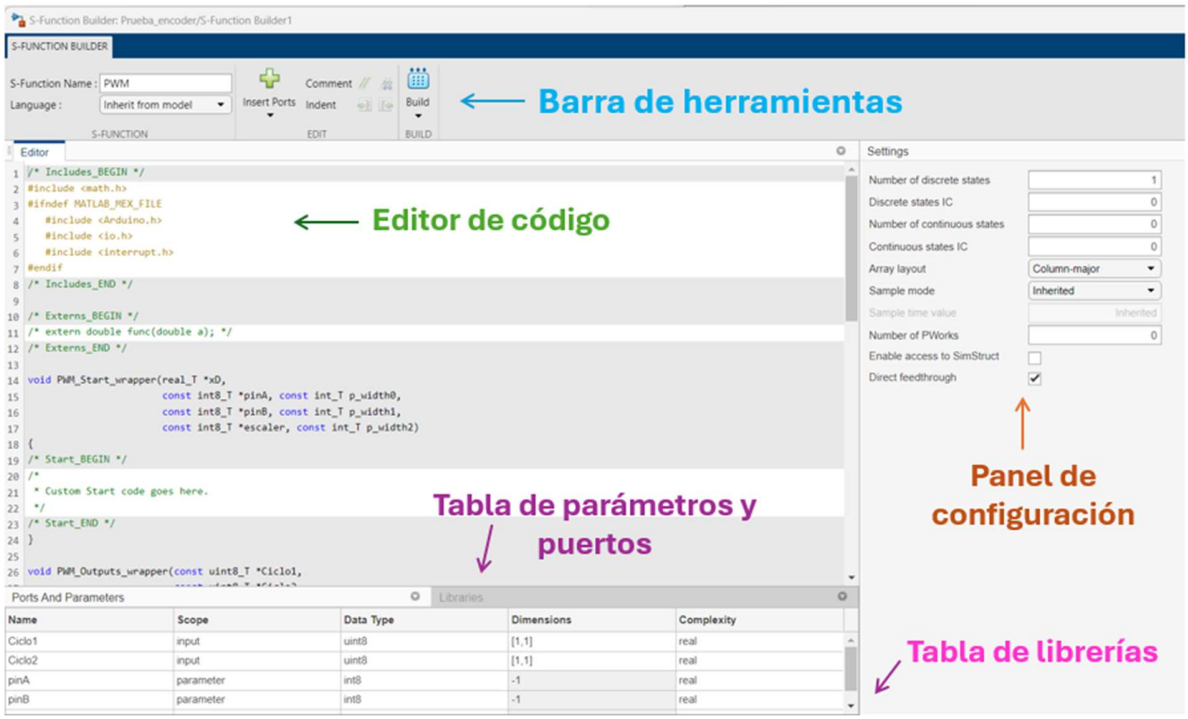


Figura 8.2 - Partes de un bloque S-Function Builder.

Por lo que para diseñar nuestro bloque PWM lo primero que haremos será ponerle un nombre al bloque, en este caso será PWM. A continuación, en el panel de configuración indicaremos

que tendremos un estado discreto cuyo valor inicial será 0. No tendremos estados continuos por lo que este punto y el siguiente los dejaremos a 0. El diseño de matrices indicaremos que será del tipo columna-mayor, ya que no hay motivo para cambiar cómo se almacenan las matrices. El modo de muestra, indicaremos que es del tipo heredado ya que se ejecutará con el tiempo de paso general. El número de `PWorks` y el acceso a `SimStruct` son innecesarios en un bloque sencillo, sin embargo debemos activar las salidas del bucle algebraico. Quedando así el panel de configuración como se indica a continuación.

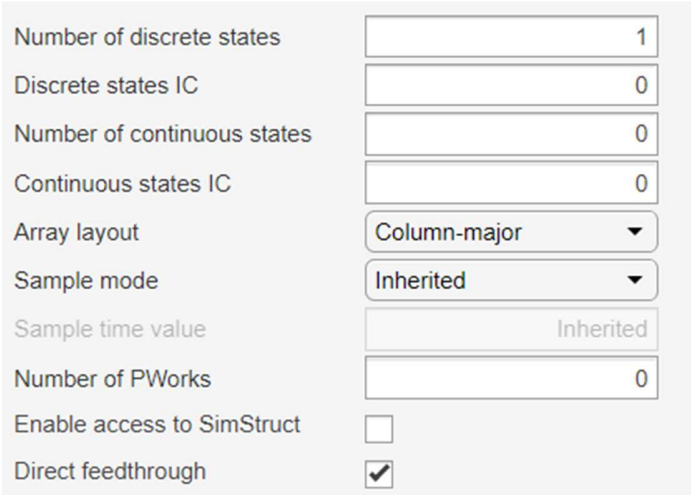


Figura 8.3- Panel de configuración de un bloque S-Function Builder.

El siguiente paso es la definición de la tabla de librerías la cual en nuestro bloque estará formada por 5 parámetros, el primero será `Ciclo1` al que indicaremos que es una entrada (input) de tipo `unit8`, ya que será un numero entero no negativo entre 0 y 255, su dimensión será la matriz [1,1], y será real. El siguiente será `Ciclo2` el cual se definirá de la misma manera que `Ciclo1`. Los siguientes parámetros para definir serán `pinA`, `pinB` y `escaler` los cuales los describiremos como parámetros de tipo `int8`, ya que será un numero entero con 8 bits de longitud, cuya dimensión es -1, la cual significa heredada y real. Por lo que nuestra tabla de parámetros y puertos quedará de la siguiente manera:

Ports And Parameters | Libraries

Name	Scope	Data Type	Dimensions	Complexity
Ciclo1	input	uint8	[1,1]	real
Ciclo2	input	uint8	[1,1]	real
pinA	parameter	int8	-1	real
pinB	parameter	int8	-1	real
escaler	parameter	int8	-1	real

Figura 8.4 -Tabla de librerías del PWM.

A continuación, en la tabla de parámetros y puertos indicamos los valores de los parámetros que acabamos de definir de tal manera que a `pinA` le daremos el valor 11, ya que indica la patilla asignada a `Ciclo1` en Arduino, a `pinB` le daremos el valor 12 ya que es la patilla asignada a `Ciclo2` en Arduino y a escaler le asignaremos el 1 ya que es el divisor de frecuencia para el Timer.

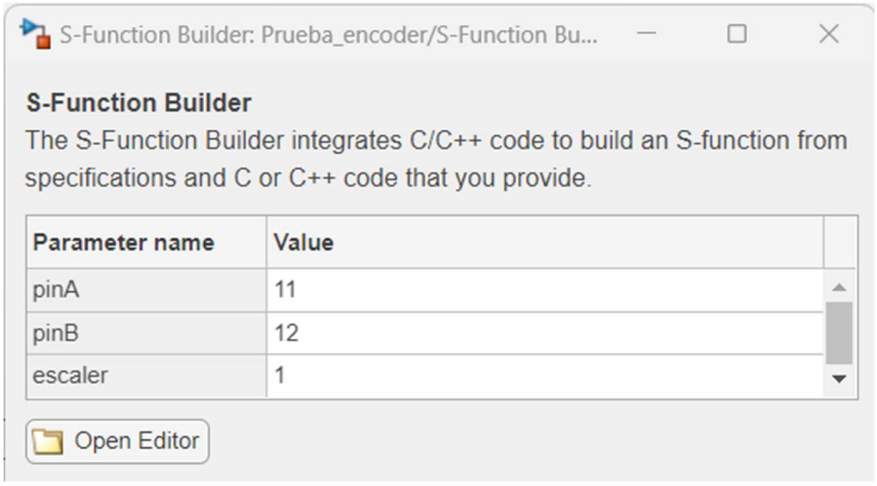


Figura 8.5 -Tabla de parámetros y puertos del PWM.

Por último, diseñaremos el código predeterminado que ya nos da el bloque, de tal manera que nuestro código será el siguiente.

```
/* Includes_BEGIN */
#include <math.h>
#ifndef MATLAB_MEX_FILE
```

```
  #include <Arduino.h>
  #include <io.h>
  #include <interrupt.h>
#endif
/* Includes_END */

/* Externs_BEGIN */
/* extern double func(double a); */
/* Externs_END */

void PWM_Start_wrapper(real_T *xD,
                       const int8_T *pinA, const int_T p_width0,
                       const int8_T *pinB, const int_T p_width1,
                       const int8_T *escaler, const int_T p_width2)
{
/* Start_BEGIN */
/*
 * Custom Start code goes here.
 */
/* Start_END */
}

void PWM_Outputs_wrapper(const uint8_T *Ciclo1,
                         const uint8_T *Ciclo2,
                         const real_T *xD,
                         const int8_T *pinA, const int_T p_width0,
                         const int8_T *pinB, const int_T p_width1,
                         const int8_T *escaler, const int_T p_width2)
{
/* Output_BEGIN */
if (xD[0]==1){
```

```
  #ifndef MATLAB_MEX_FILE
    analogWrite(pinA[0],Ciclo1[0]);
    analogWrite(pinB[0],Ciclo2[0]);
  #endif
}
/* Output_END */
}

void PWM_Update_wrapper(const uint8_T *Ciclo1,
                        const uint8_T *Ciclo2,
                        real_T *xD,
                        const int8_T *pinA, const int_T p_width0,
                        const int8_T *pinB, const int_T p_width1,
                        const int8_T *escaler, const int_T p_width2)
{
/* Update_BEGIN */
if(xD[0]!=1){
  #ifndef MATLAB_MEX_FILE
    pinMode(pinA[0], OUTPUT);
    pinMode(pinB[0], OUTPUT);
    TCCR1B=(TCCR1B&0xF8)|escaler[0];
  #endif
 xD[0]=1;
 }//Fin del primer if
/* Update_END */
}

void PWM_Terminate_wrapper(real_T *xD,
                           const int8_T *pinA, const int_T p_width0,
                           const int8_T *pinB, const int_T p_width1,
                           const int8_T *escaler, const int_T p_width2)
```

```
{
/* Terminate_BEGIN */
/*
 * Custom Terminate code goes here.
 */
/* Terminate_END */
}
```

En el encabezado se incluyen las bibliotecas utilizadas para general el código, `math.h`, `Arduino.h`, `io.h`e `interrupt.h`

A continuación, se describe la función `PWM_Start_wrapper` la cual es el punto de entrada para el inicio de la generación de la señal PWM, que en este caso es una función vacía y no realiza ninguna operación específica al inicio. Seguido a ésta, la señal de salida del PWM cuyo nombre es `PWM_Outputs_wrapper` que verifica el valor de `xD [0]`y si es igual a 1, utiliza la función `analogWrite`de Arduino para establecer el ciclo de trabajo de las señales PWM en los pines especificados por `pinA[ ]` y `pinB[ ]`.

La función `PWM_Update_wrapper` maneja la actualización de la generación de señales PWM y si `xD [0]` no es igual a 1, configura los pines especificados por `pinA [0]` y `pinB [0]` como salidas digitales y configura el registro de control de temporización (TCCR1B) con el valor especificado por `escaler [0]`. Además, establece `xD [0]` en 1.

Por último, la función `PWM_Terminate_wrapper` sirve como punto de terminación, aunque en este código está vacía.

Una vez finalizamos el código debemos construir el bloque para generar el código de forma que iremos a `Build` e indicamos que queremos generar un wrapper TLC.

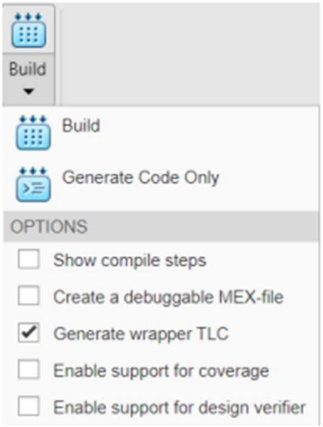


Figura 8.6 - Construcción del wrapper.

El último paso será abrir el archivo `PWM_wrapper.c` en MATLAB que acabamos de generar y añadiremos `extern "C"` a las funciones `Outputs_wrapper` y `Update_wrapper` . Guardaremos el archivo indicando la extensión cpp. De manera que nuestro archivo `PWM_wrapper.cpp` será el siguiente.

```
/*
 * Include Files
 *
 */
#if defined(MATLAB_MEX_FILE)
#include "tmwtypes.h"
#include "simstruc_types.h"
#else
#include "rtwtypes.h"
#endif



/* %%%-SFUNWIZ_wrapper_includes_Changes_BEGIN --- EDIT HERE TO _END */
#include <math.h>
```

```
#ifndef MATLAB_MEX_FILE
  #include <Arduino.h>
  #include <io.h>
  #include <interrupt.h>
#endif
/* %%%-SFUNWIZ_wrapper_includes_Changes_END --- EDIT HERE TO _BEGIN */
#define u_width 1

/*
 * Create external references here.
 *
 */
/* %%%-SFUNWIZ_wrapper_externs_Changes_BEGIN --- EDIT HERE TO _END */
/* extern double func(double a); */
/* %%%-SFUNWIZ_wrapper_externs_Changes_END --- EDIT HERE TO _BEGIN */

/*
 * Output function
 *
 */
extern "C" void PWM_Outputs_wrapper(const uint8_T *Ciclo1,
                        const uint8_T *Ciclo2,
                        const real_T *xD,
                        const int8_T *pinA, const int_T p_width0,
                        const int8_T *pinB, const int_T p_width1,
                        const int8_T *escaler, const int_T p_width2)
{
/* %%%-SFUNWIZ_wrapper_Outputs_Changes_BEGIN --- EDIT HERE TO _END */
if (xD[0]==1){
   #ifndef MATLAB_MEX_FILE
      analogWrite(pinA[0],Ciclo1[0]);
```

```
      analogWrite(pinB[0],Ciclo2[0]);
   #endif
}
/* %%%-SFUNWIZ_wrapper_Outputs_Changes_END --- EDIT HERE TO _BEGIN */
}

/*
 * Updates function
 *
 */
extern "C" void PWM_Update_wrapper(const uint8_T *Ciclo1,
                        const uint8_T *Ciclo2,
                        real_T *xD,
                        const int8_T *pinA, const int_T p_width0,
                        const int8_T *pinB, const int_T p_width1,
                        const int8_T *escaler, const int_T p_width2)
{
/* %%%-SFUNWIZ_wrapper_Update_Changes_BEGIN --- EDIT HERE TO _END */
if(xD[0]!=1){
   #ifndef MATLAB_MEX_FILE
      pinMode(pinA[0], OUTPUT);
      pinMode(pinB[0], OUTPUT);
      TCCR1B=(TCCR1B&0xF8)|escaler[0];
   #endif
  xD[0]=1;
 } //Fin del primer if
/* %%%-SFUNWIZ_wrapper_Update_Changes_END --- EDIT HERE TO _BEGIN */
}
```

El siguiente bloque que hemos creado es otro `S-Function Builder` llamado encoder para poder visualizar mediante el Scope que las rpm que podemos leer en el osciloscopio real serán las mismas que nuestro programa está leyendo. Además, tendremos que hacer lo mismo que en el bloque anterior con los siguientes cambios. Nuestro nuevo bloque lo hemos llamado `enc_oder` en la barra de herramientas.

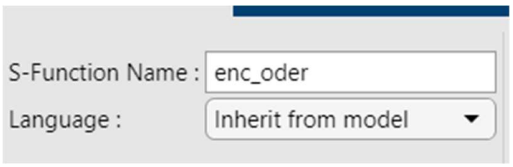


Figura 8.7 - Barra de herramientas del enc_oder.

El panel de conjuración será igual que el anterior. En la tabla de librerías tendremos las entradas `inct` y rpm de tipo `double` ya que serán números que pueden tener una parte decimal y pueden representar un amplio rango de presión, cuya dimensión es [1,1] y reales, y otra entrada `pul` la cual será de tipo `unit16` ya que será un numero entero de 16 bits de longitud cuya dimensión será también [1,1] y real. También tendremos los parámetros `enc`, `pinA` y `pinB` de tipo `unit8`, dimensión -1 y reales y un último parámetro de tipo `unit16` y dimensión -1.

Ports And Parameters | Libraries

Name	Scope	Data Type	Dimensions	Complexity
inct	output	double	[1,1]	real
pul	output	uint16	[1,1]	real
rpm	output	double	[1,1]	real
enc	parameter	int8	-1	real
pinA	parameter	int8	-1	real
pinB	parameter	int8	-1	real
pulsosvuelta	parameter	uint16	-1	real

Figura 8.8 -Tabla de librerías del enc_oder.

En la tabla de parámetros y puertos indicamos los valores de los parámetros que acabamos de definir de tal manera que a `enc` le daremos el valor 0, a `pinA` le daremos el valor 18, ya que indica la patilla asignada en Arduino, a `pinB` le daremos el valor 24 ya que es la patilla asignada en Arduino y a `pulsosvuelta` le asignaremos el valor 1000.

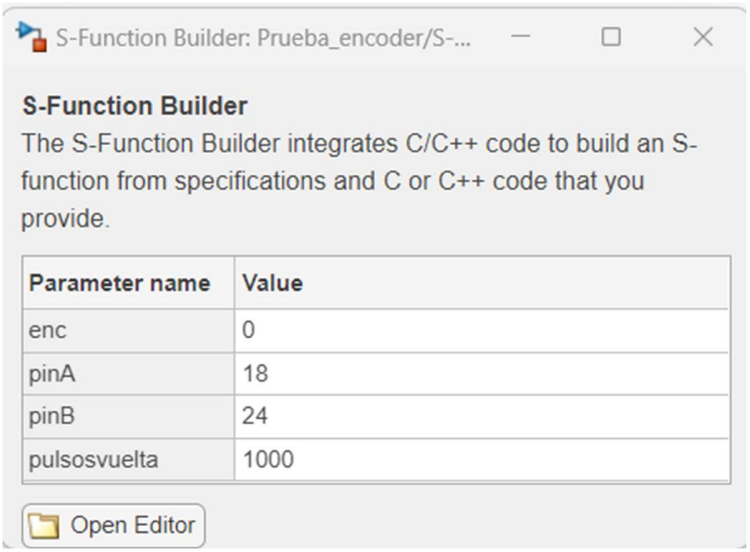


Figura 8.9 -Tabla de parámetros y puertos del enc_oder.

El siguiente paso al igual que en el bloque anterior es generar un `S-Function Builder` para el Encoder que realice mediciones relacionadas con la velocidad angular y los pulsos generados por el mismo encoder.

```
/* Includes_BEGIN */
#include <math.h>
#ifndef MATLAB_MEX_FILE
  #include <Arduino.h>
  //#include "I2Cdev.h"a
  //#define OUTPUT_READABLE_rpm
 typedef struct Lectura { int pulsos; double tviejo;} L_t;
 L_t vec_tor[3]={{0,0},{0,0},{0,0}};
 int getIntNum(int pin) {
 /* returns the interrupt number for a given interrupt pin
  see http://arduino.cc/it/Reference/AttachInterrupt        */
 switch(pin) {
   case 2:
     return 0;
```

```
    case 3:
      return 1;
    case 21:
      return 2;
    case 20:
      return 3;
    case 19:
      return 4;
    case 18:
      return 5;
    default:
      return -1;
    }//Fin del switch
  }//Fin de la función auxiliar
//Función de la interrupción
void cuenta0()
{
  vec_tor[0].pulsos++;
}
void cuenta1()
{
  vec_tor[1].pulsos++;
}
void cuenta2()
{
  vec_tor[2].pulsos++;
}
#endif
/* Includes_END */

/* Externs_BEGIN */
```

```
/* extern double func(double a); */
/* Externs_END */

void enc_oder_Start_wrapper(real_T *xD,
                            const int8_T *enc, const int_T p_width0,
                            const int8_T *pinA, const int_T p_width1,
                            const int8_T *pinB, const int_T p_width2,
                            const uint16_T *pulsosvuelta, const int_T p_width3)
{
/* Start_BEGIN */

/* Start_END */
}

void enc_oder_Outputs_wrapper(real_T *inct,
                              uint16_T *pul,
                              real_T *rpm,
                              const real_T *xD,
                              const int8_T *enc, const int_T p_width0,
                              const int8_T *pinA, const int_T p_width1,
                              const int8_T *pinB, const int_T p_width2,
                              const uint16_T *pulsosvuelta, const int_T p_width3)
{
/* Output_BEGIN */
if (xD[0]==1){
   #ifndef MATLAB_MEX_FILE
   //L_t* pcod2= (L_t*)pW[0];
   pul[0]=vec_tor[enc[0]].pulsos;
   inct[0]=micros()-vec_tor[enc[0]].tviejo;
         if (inct[0]>10000)
         {
```

```
                noInterrupts(); //durante el cálculo
                inct[0]=micros()-vec_tor[enc[0]].tviejo;
        pul[0]=vec_tor[enc[0]].pulsos;


        vec_tor[enc[0]].tviejo=micros();
        vec_tor[enc[0]].pulsos=0;
        interrupts();


        double atr=1e6/inct[0];
        rpm[0]=(60*atr)/pulsosvuelta[0]*pul[0];


          }
    #endif
}
/* Output_END */
}

void enc_oder_Update_wrapper(real_T *inct,
                        uint16_T *pul,
                        real_T *rpm,
                        real_T *xD,
                        const int8_T *enc, const int_T p_width0,
                        const int8_T *pinA, const int_T p_width1,
                        const int8_T *pinB, const int_T p_width2,
                        const uint16_T *pulsosvuelta, const int_T p_width3)
{
/* Update_BEGIN */
/*Coloco el set-up*/
```

```
if(xD[0]!=1){
  #ifndef MATLAB_MEX_FILE
    /* set encoder pins as inputs   */
    pinMode(pinA[0], INPUT);
    pinMode(pinB[0], INPUT);

    /* attach interrupts    */
    switch(enc[0]) {
     case 0:
      attachInterrupt(getIntNum(pinA[0]), cuenta0, RISING);
     break;
     case 1:
      attachInterrupt(getIntNum(pinA[0]), cuenta1, RISING);
     break;
     case 2:
      attachInterrupt(getIntNum(pinA[0]), cuenta2, RISING);
     break;
     }
  //L_t* pcod= (L_t*)pW[0];
   Vec_tor[enc[0]].pulsos=0;
   Vec_tor[enc[0]].tviejo=micros(); //Actualización del tiempo
  #endif
  xD[0]=1;
 } //Fin del primer if
/*Update_END */
}
void enc_oder_Terminate_wrapper (real_T *xD,
                  const int8*enc, const int_T p_width0,
                  const int8*pniA, const int_T p_width1,
                  const int8*pinB, const int_T p_width2,
                  const uint16_T*pulsosvuelta, const int_T p_width3,
```

```
{
/* Terminate_BEGIN */
/* Terminate_END */
}
```

Empezamos incluyendo las bibliotecas que vamos a necesitar, las cuales son `math.h` y `Arduino.h`. A continuación, definimos la estructura `Lectura` que contiene un contador de pulsos y un tiempo anterior. Este bloque esta diseñado para trabajar con tres encoders por si fuera necesario más de uno, aunque nosotros únicamente utilizaremos uno.

La función `getIntNum` devuelve el número de interrupción asociado a un número de pin de interrupción en Arduino. Tenemos tres funciones de interrupción (`cuenta0`, `cuenta1` y `cuenta3`) que se ejecutarán cuando se detecte un flanco de subida en los pines de los encoder, incrementando así el contador de pulsos correspondiente en `vec_tor`.

Tenemos cuatro funciones wrapper, la primera `enc_oder_Start_wrapper` ejecuta el inicio de la simulación, que en nuestro caso está vacía. La `enc_oder_Outputs_wrapper` maneja las salidas de la simulación calculando la diferencia de tiempo y la velocidad angular en rpm basándose en los pulsos registrados. La `enc_oder_Update_wrapper` configura los pines del encoder como entradas y adjunta las interrupciones correspondientes inicializando las variables y el tiempo anterior. La última será la `enc_oder_Terminate_wrapper` que ejecuta el final de la simulación, está vacía.

Al igual que en el bloque anterior debemos construir el bloque para generar en MATLAB el archivo `enc_oder_wrapper.c` y añadir a este `extern “C”` a las funciones `Outputs_wrapper` y `Update_wrapper` y guardaremos el archivo indicando la extensión `cpp`. De manera que nuestro archivo `enc_oder_wrapper.cpp` será el siguiente:

```
/*
```

```
 * Include Files
 *
 */
#if defined(MATLAB_MEX_FILE)
#include "tmwtypes.h"
#include "simstruc_types.h"
#else
#include "rtwtypes.h"
#endif
/* %%%-SFUNWIZ_wrapper_includes_Changes_BEGIN --- EDIT HERE TO _END */
#include <math.h>
#ifndef MATLAB_MEX_FILE
  #include <Arduino.h>
  //#include "I2Cdev.h"a
  //#define OUTPUT_READABLE_rpm
 typedef struct Lectura { int pulsos; double tviejo;} L_t;
 L_t vec_tor[3]={{0,0},{0,0},{0,0}};
 int getIntNum(int pin) {
 /* returns the interrupt number for a given interrupt pin
  see http://arduino.cc/it/Reference/AttachInterrupt       */
 switch(pin) {
   case 2:
     return 0;
   case 3:
     return 1;
   case 21:
     return 2;
   case 20:
     return 3;
   case 19:
     return 4;
```

```
    case 18:
      return 5;
    default:
      return -1;
    } //Fin del switch
  } //Fin de la función auxiliar
//Función de la interrupción
void cuenta0()
{
  vec_tor[0].pulsos++;
}
void cuenta1()
{
  vec_tor[1].pulsos++;
}
void cuenta2()
{
  vec_tor[2].pulsos++;
}
#endif
/* %%%-SFUNWIZ_wrapper_includes_Changes_END --- EDIT HERE TO _BEGIN */
#define y_width 1
/*
 * Create external references here.
 *
 */
/* %%%-SFUNWIZ_wrapper_externs_Changes_BEGIN --- EDIT HERE TO _END */
/* extern double func(double a); */
/* %%%-SFUNWIZ_wrapper_externs_Changes_END --- EDIT HERE TO _BEGIN */
/*
 * Output function
```

```
 *
 */
extern "C" void enc_oder_Outputs_wrapper(real_T *inct,
                        uint16_T *pul,
                        real_T *rpm,
                        const real_T *xD,
                        const int8_T *enc, const int_T p_width0,
                        const int8_T *pinA, const int_T p_width1,
                        const int8_T *pinB, const int_T p_width2,
                        const uint16_T *pulsosvuelta, const int_T p_width3)
{
/* %%%-SFUNWIZ_wrapper_Outputs_Changes_BEGIN --- EDIT HERE TO _END */
if (xD[0]==1){
   #ifndef MATLAB_MEX_FILE
   //L_t* pcod2= (L_t*)pW[0];
   pul[0]=vec_tor[enc[0]].pulsos;
   inct[0]=micros()-vec_tor[enc[0]].tviejo;
        if (inct[0]>10000)
        {
                noInterrupts(); // durante el cálculo
                inct[0]=micros()-vec_tor[enc[0]].tviejo;
      pul[0]=vec_tor[enc[0]].pulsos;
      vec_tor[enc[0]].tviejo=micros();
      vec_tor[enc[0]].pulsos=0;
      interrupts();
      double atr=1e6/inct[0];
      rpm[0]=(60*atr)/pulsosvuelta[0]*pul[0];
        }
   #endif
}
/* %%%-SFUNWIZ_wrapper_Outputs_Changes_END --- EDIT HERE TO _BEGIN */
```

```
}
/*
 * Updates function
 *
 */
extern "C" void enc_oder_Update_wrapper(real_T *inct,
                        uint16_T *pul,
                        real_T *rpm,
                        real_T *xD,
                        const int8_T *enc, const int_T p_width0,
                        const int8_T *pinA, const int_T p_width1,
                        const int8_T *pinB, const int_T p_width2,
                        const uint16_T *pulsosvuelta, const int_T p_width3)
{
/* %%%-SFUNWIZ_wrapper_Update_Changes_BEGIN --- EDIT HERE TO _END */
/*Coloco el set-up*/
if(xD[0]!=1){
   #ifndef MATLAB_MEX_FILE
      /* set encoder pins as inputs   */
      pinMode(pinA[0], INPUT);
      pinMode(pinB[0], INPUT);

      /* attach interrupts    */
      switch(enc[0]) {
        case 0:
          attachInterrupt(getIntNum(pinA[0]), cuenta0, RISING);
        break;
        case 1:
          attachInterrupt(getIntNum(pinA[0]), cuenta1, RISING);
        break;
        case 2:
```

```
        attachInterrupt(getIntNum(pinA[0]), cuenta2, RISING);
      break;
      }
  //L_t* pcod= (L_t*)pW[0];
  vec_tor[enc[0]].pulsos=0;
  vec_tor[enc[0]].tviejo=micros(); //Actualización del tiempo

  #endif


 xD[0]=1;
} //Fin del primer if
/* %%%-SFUNWIZ_wrapper_Update_Changes_END --- EDIT HERE TO _BEGIN */
}
```

Una vez que tenemos nuestro modelo de Simulink diseñado, configuraremos el tiempo de paso en la simulación, para ello iremos a `Modeling`, `Model Settings` y ahí podremos configurar los parámetros indicando un paso fijo de 0,2ms.

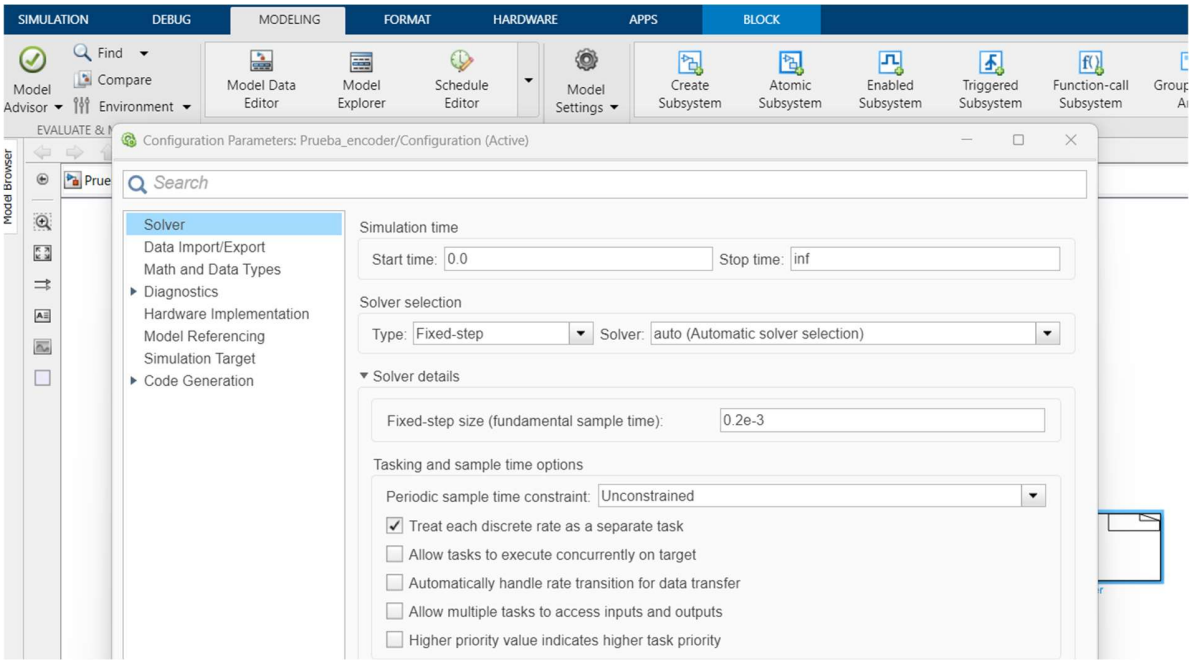


Figura 8.10 - Configuración del tiempo de paso.

A continuación, debemos indicar al Simulink el tipo de Arduino que vamos a utilizar y en qué puerto se encuentra, para ello en la misma pestaña del programa iremos a `Hardware Implementation` y le indicaremos que vamos a usar un Arduino Mega 2560 y que éste se encuentra en el puerto 3.

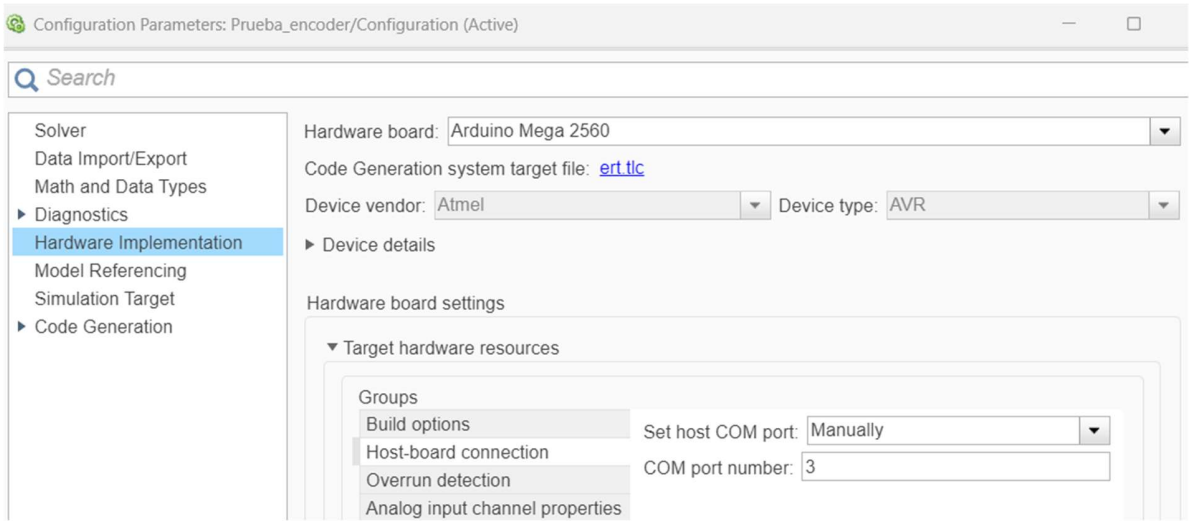


Figura 8.11 - Indicación del puerto de Arduino.

Una vez definido nuestro modelo realizaremos el montaje en el laboratorio. Para probar si éste es correcto alimentaremos el Arduino a nuestro ordenador. Seguidamente, conectaremos el encoder por los pines 18 y 24 al Arduino, también a GND y a Vin para alimentarlo. Conectaremos el motor a la fuente de tensión y lo dejaremos en vacío para comprobar si la lectura del osciloscopio y la del Scope son la misma.

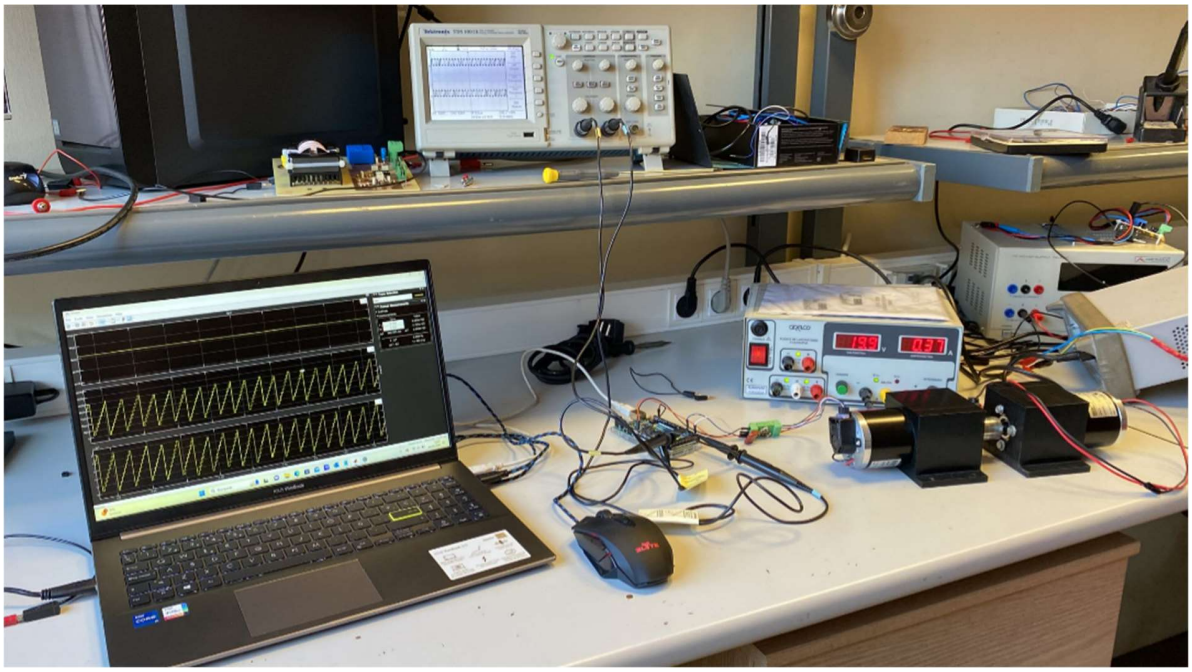


Figura 8.12 - Montaje en el laboratorio de la prueba del encoder.

Figura 8.13 - Lectura en el osciloscopio de la frecuencia de la prueba del encoder.

De manera que en osciloscopio podemos leer una frecuencia de 52,8kHz lo cual equivale a $(52{,}8x60) = 3164\ rpm$.

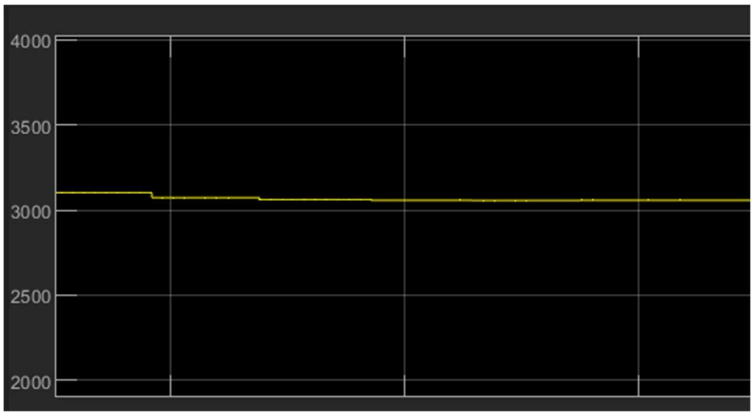


Figura 8.14 - Lectura en el Scope de las rpm de la prueba del encoder.

Si leemos las rpm que nos indica el Scope podemos decir que nuestro modelo es correcto. Ya que, aproximadamente las rpm del Scope son 3100.

9. Control de velocidad

Un control de velocidad en un motor de CC se refiere a la capacidad de variar la velocidad de rotación del motor de manera controlada y precisa. Para lo cual se ha realizado una implementación de algoritmos y modelos para controlar la velocidad de un sistema dinámico utilizando el entorno de simulación de Simulink.

Para realizar un correcto funcionamiento del control de velocidad es necesario crear un controlador PID. Éste es un dispositivo utilizado en sistemas de control automático para regular y mantener una variable controlada en un valor deseado. Sus tres términos fundamentales son:

- Proporcional (D): Proporciona una respuesta proporcional al error actual entre la variable controlada y el setpoint. A mayor error, mayor será la salida del controlador proporcional. Este término es el responsable de la acción inmediata del controlador ante cambios en el error.
- Integral (I): Acumula el error a lo largo del tiempo y produce una salida proporcional a la suma acumulada de los errores pasados. Es útil para corregir errores residuales o estacionarios que no se corrigen solo con la acción del proporcional.
- Derivativo (D): Produce una respuesta proporcional a la tasa de cambio del error. Ayuda a prevenir oscilaciones excesivas y a mejorar la estabilidad del sistema al anticiparse a cambios en el error. Es útil para sistemas con inercia o retardos en la respuesta.

El controlador PID combina estos tres términos para generar una señal de control que se aplica al sistema para ajustar su comportamiento y mantener la variable controlada en el setpoint deseado. La ecuación que representa un controlador PID es:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad (9.1)$$

Donde:

- U(t) es la señal de control aplicada al sistema en el tiempo t.
- K_p, K_i y K_d son los parámetros del controlador proporcional, integral y derivativo.
- $e(t)$ es el error, definido como la diferencia entre la variable controlada y el setpoint en el tiempo t.

El ajuste adecuado de estos parámetros es crucial para el rendimiento óptimo del controlador PID en términos de estabilidad, tiempo de respuesta y rechazo de perturbaciones. El PID necesario para nuestro modelo puede generarse con Simulink de la siguiente forma.

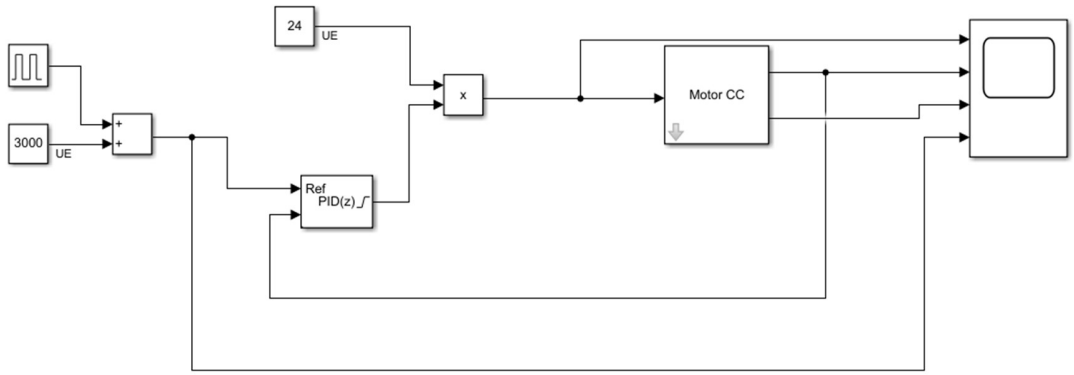


Figura 9.1 - Modelo en Simulink para simular el control de velocidad.

Realizamos el esquema en el Simulink de manera que creamos un subsistema que representa a nuestro motor real, introduciendo en él todos aquellos parámetros que definen nuestro motor como la constante k del motor, la inductancia, resistencia, momento de inercia, coeficiente de rozamiento, par del eje y par útil, los cuales se pueden encontrar en la hoja de características del motor en el APENDICE.

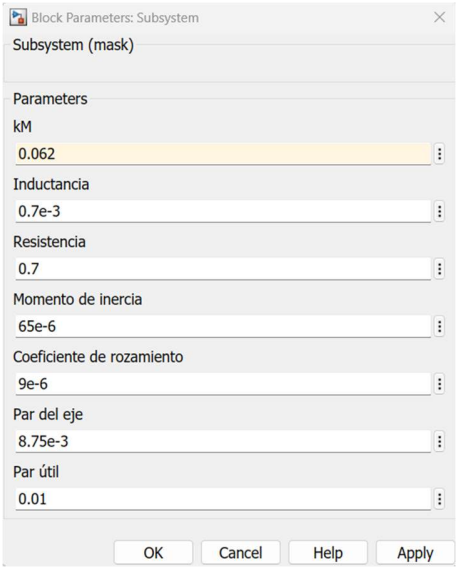


Figura 9.2 - Parámetros del motor pare el modelo en Simulink.

Una vez definidos todos los parámetros del sistema, nuestro subsistema queda de la siguiente manera.

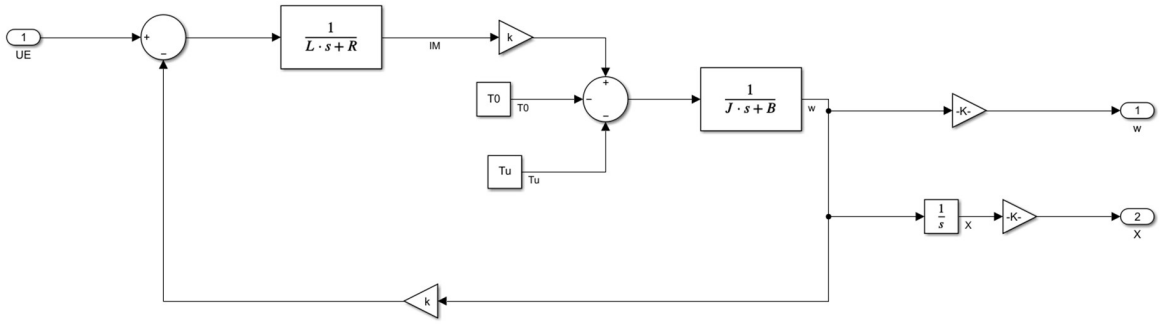


Figura 9.3 - Subsistema del modelo en Simulink para simular el control de velocidad.

Para generar nuestro PID debemos tunear el bloque del PID Controller indicando que utilizaremos el método de `clamping´ la cual es una técnica utilizada para limitar los valores de la salida del controlador para evitar que exceda los límites. Una vez tuneado propio bloque de Simulink nos indica los valores de los tres parámetros del PID necesarios para nuestro motor.

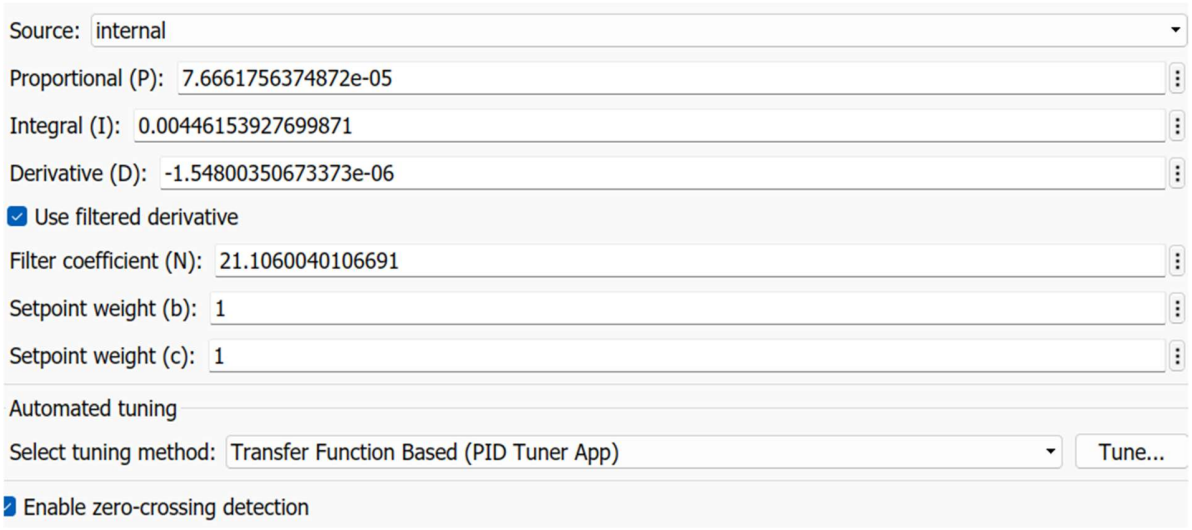


Figura 9.4 - Parámetros del PID.

Una vez definidos los parámetros necesarios para nuestro modelo crearemos el modelo que utilizaremos para controlar nuestro motor real.

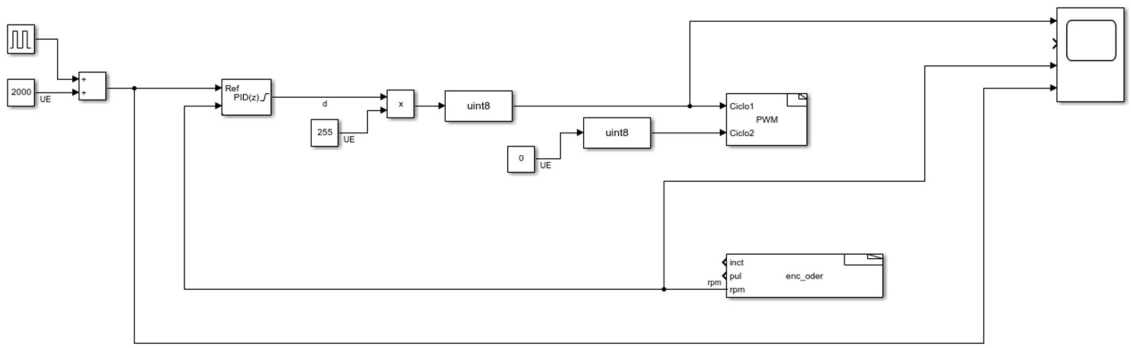


Figura 9.5 - Modelo en Simulink para simular el control de velocidad real.

Para el cuál utilizamos los bloques de PWM y enc_oder que ya habíamos creado anteriormente. Para comprobar que el control de velocidad es correcto hemos probado su funcionamiento en cuatro casos.

El primer caso será el motor en vacio a indicándole una velocidad de 2000.

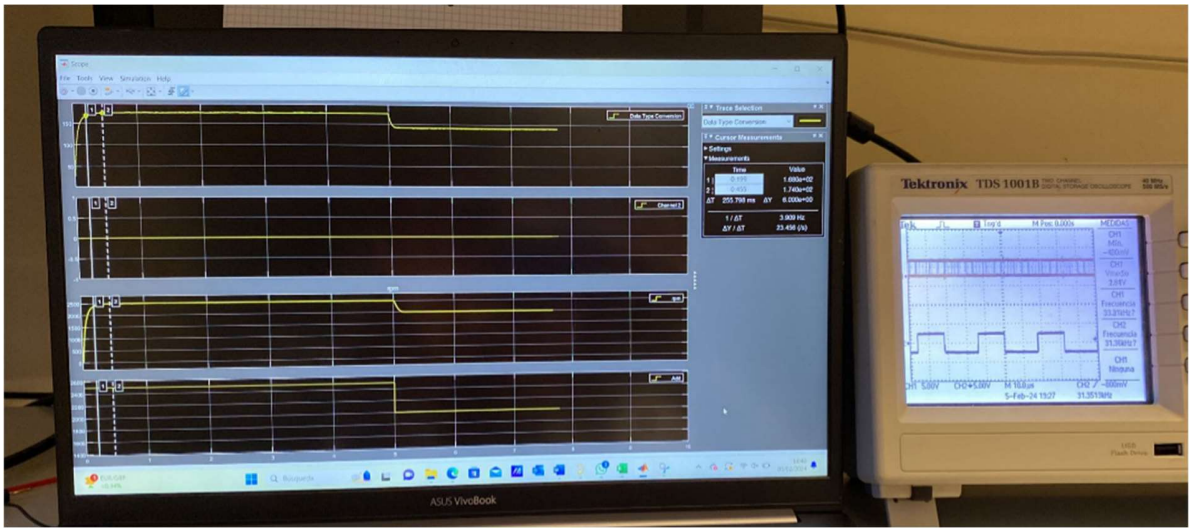


Figura 9.6 - Lectura de ambos osciloscopios para motor en vacio con velocidad 2000.

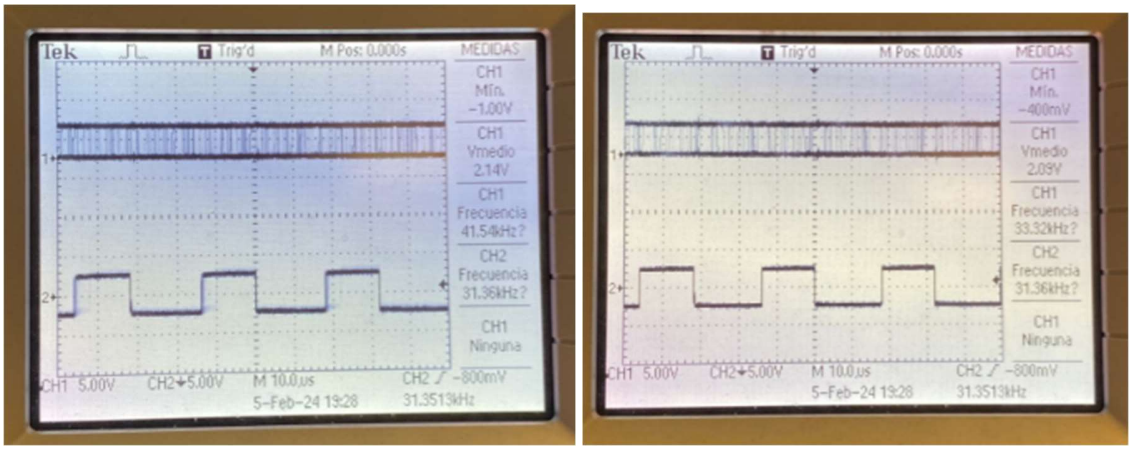


Figura 9.7 - Lecturas del osciloscopio para motor en vacio con velocidad 2000.

En la imagen 9.7 puede leerse que el motor tiene una frecuencia 41,54kHz, lo cual equivale a (41,54 x 60) = 2492,42 rpm, al igual puede leerse que el motor tiene una frecuencia de 33,32kHz, lo cual equivale a (33,32 x 60) = 1999,2 rpm.

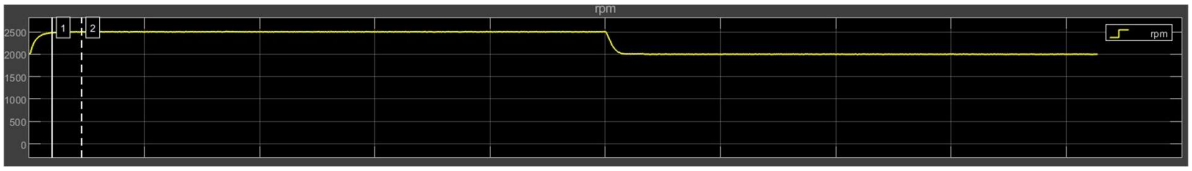


Figura 9.8 - Lectura del Scope para motor en vacio con velocidad 2000.

En el Scope podemos leer que inicialmente el motor gira a 2500 rpm y posteriormente a 2000rpm, al igual que podíamos leer en el osciloscopio.

El segundo caso será el motor en vacio a indicándole una velocidad de 1500rpm.

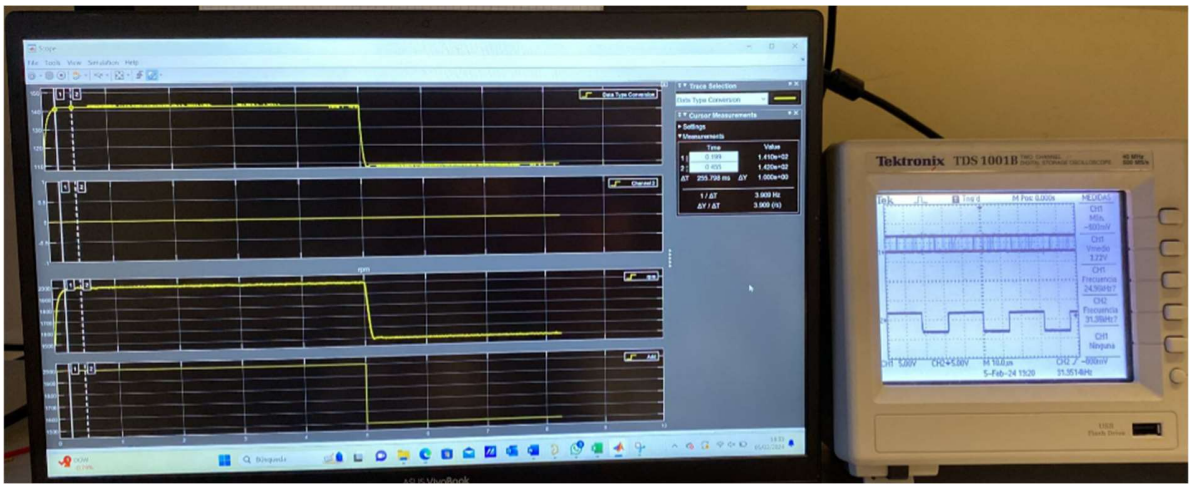


Figura 9.9 - Lectura de ambos osciloscopios para motor en vacio con velocidad 1500.

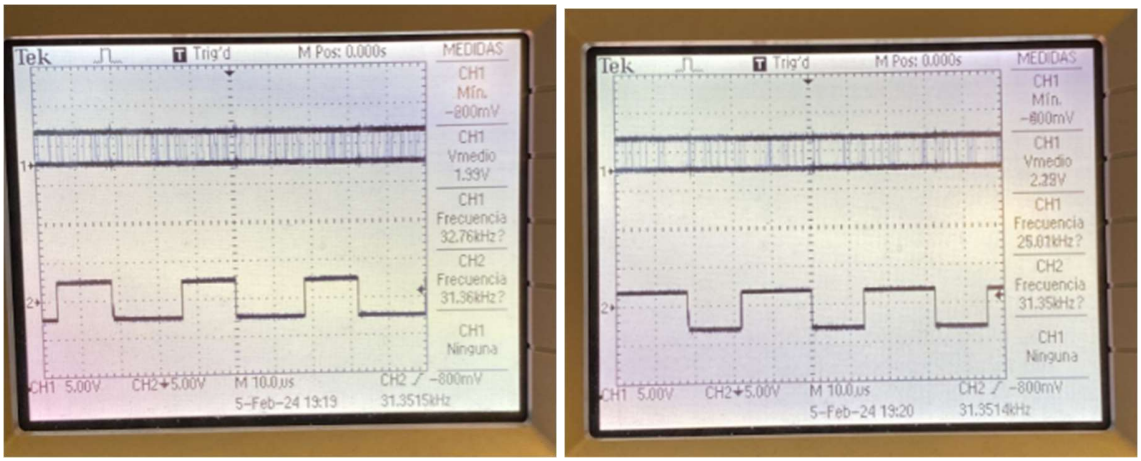


Figura 9.10 - Lectura del osciloscopio para motor en vacio con velocidad 1500.

En la imagen 9.10 puede leerse que el motor tiene una frecuencia de 32,76kHz, lo cual equivale a (32,76 x 60) = 1965,6 rpm, al igual puede leerse que el motor tiene una frecuencia de 25,81kHz, lo cual equivale a (25,81 x 60) = 1548,6 rpm.

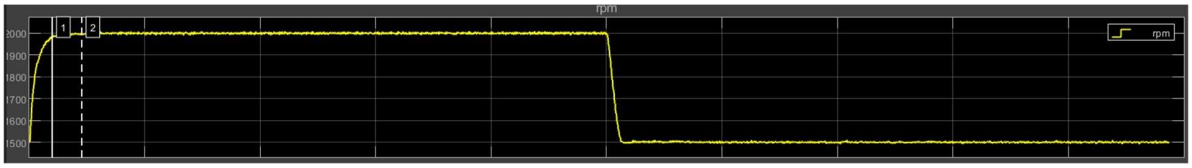


Figura 9.11 - Lectura del Scope para motor en vacio con velocidad 1500.

En el Scope podemos leer que inicialmente el motor gira a 2000 rpm y posteriormente a 1500rpm, al igual que podíamos leer en el osciloscopio.

El tercer caso será el motor en carga e indicándole una velocidad de 2000rpm.

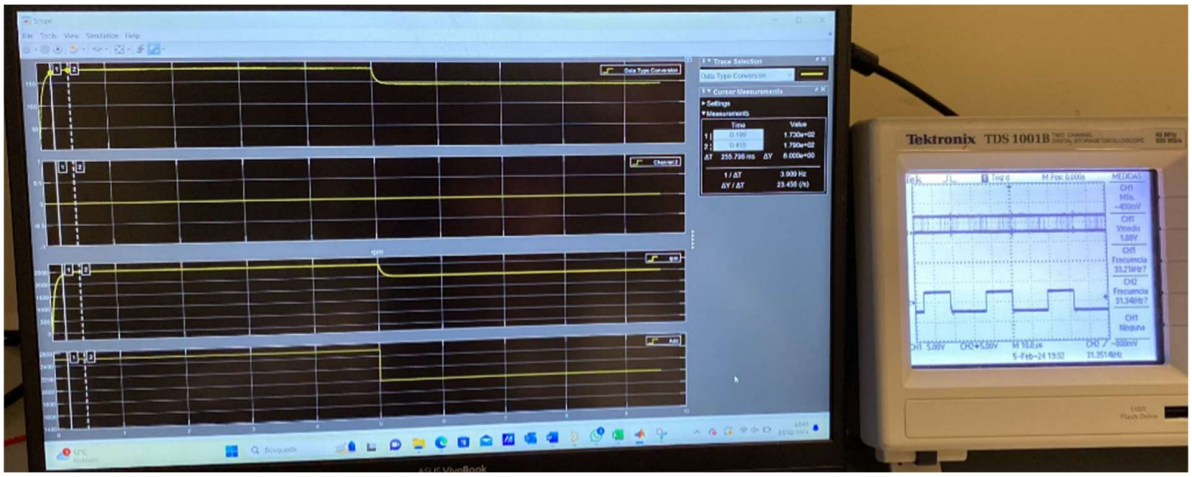


Figura 9.12 - Lectura de ambos osciloscopios para motor en carga con velocidad 2000.

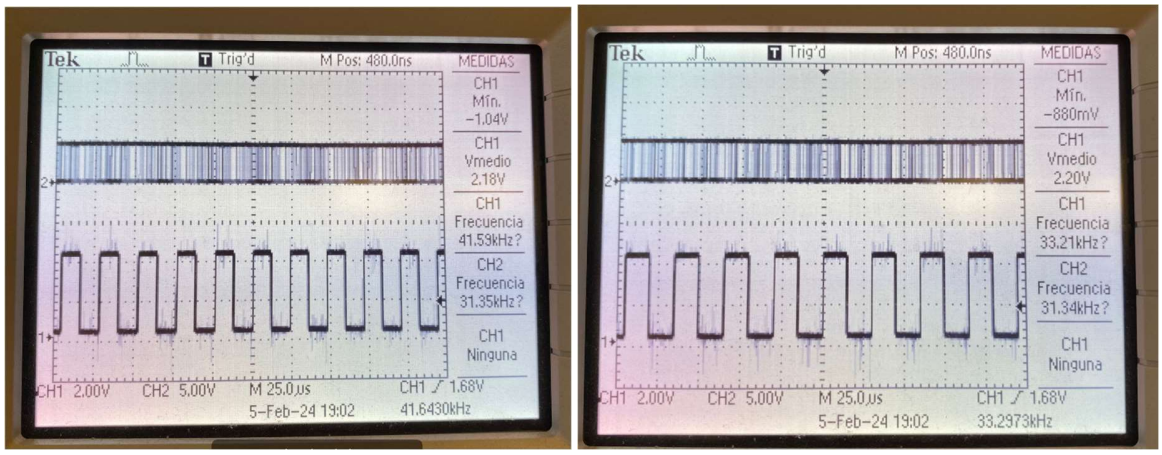


Figura 9.13 - Lectura del osciloscopio para motor en carga con velocidad 2000.

En la imagen 9.13 puede leerse que el motor tiene una frecuencia de 41,59kHz, lo cual equivale a (41,59 x 60) = 2495,4 rpm, al igual puede leerse que el motor tiene una frecuencia de 33,21kHz, lo cual equivale a (33,21 x 60) = 1992,6 rpm.

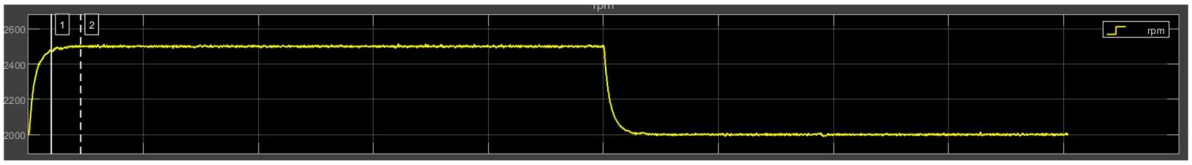


Figura 9.14 - Lectura del Scope para motor en carga con velocidad 2000.

En el Scope podemos leer que inicialmente el motor gira a 2500 rpm y posteriormente a 2000rpm, al igual que podíamos leer en el osciloscopio.

El último caso será el motor en carga e indicándole una velocidad de 1500rpm.

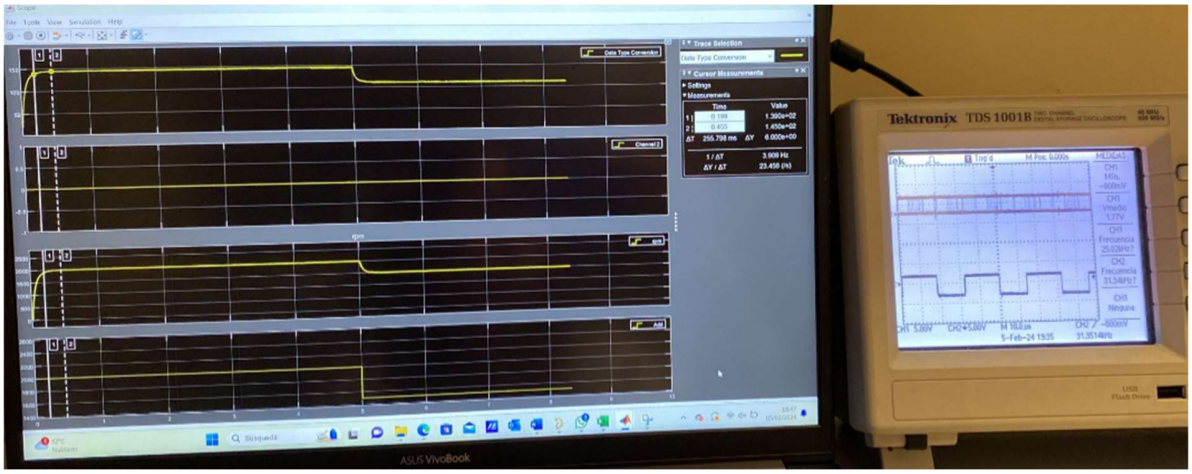


Figura 9.15 - Lectura de ambos osciloscopios para motor en carga con velocidad 1500.

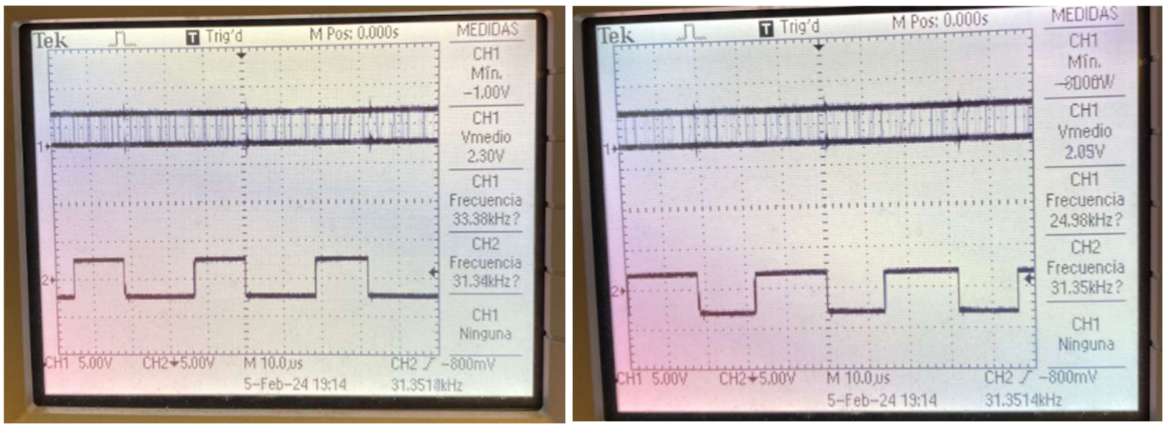


Figura 9.16 - Lectura del osciloscopio para motor en carga con velocidad 1500.

En la imagen 9.18 puede leerse que el motor tiene una frecuencia de 33,38kHz, lo cual equivale a (33,38 x 60) = 2002,8 rpm, al igual puede leerse que el motor tiene una frecuencia de 24,98kHz, lo cual equivale a (24,98 x 60) = 1498,8 rpm.

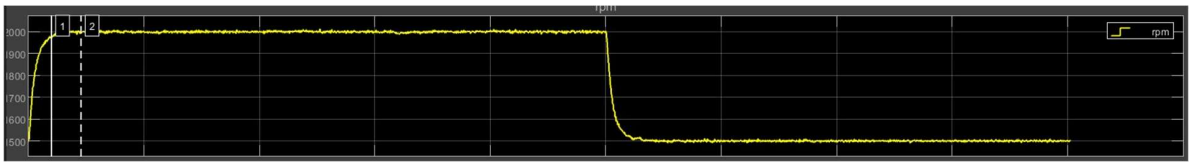


Figura 9.17- Lectura del Scope para motor en carga con velocidad 1500.

En el Scope podemos leer que inicialmente el motor gira a 2000 rpm y posteriormente a 1500rpm, al igual que podíamos leer en el osciloscopio.

10. Presupuesto

10.1 PRESUPUESTO DEL MATERIAL PRINCIPAL

En este apartado se recoge el presupuesto del material principal necesario para la elaboración del presente T.F.G.

Descripción	Cantidad	Precio unitario	Valor total
Motor 89890911	2	39,02	78,04
Encoder HEDM 5600 B12	1	41,79	41,79
Arduino Mega	1	40,95	40,95
Driver L298N	1	5,32	5,32
Total	166,10 €		

Tabla 10.1 - Presupuesto material principal.

10.2 PRESUPUESTO DE LA PLACA PCB

Descripción	Cantidad	Precio Unitario	Valor total
Disipador Fischer	2	5,60	11,2
Condensador 100nF	4	0,87	3,48
Condensador 56nF	2	0,77	1,54
Condensador 470μF	1	0,41	0,41
Resistencia 5k6	6	0,21	0,72
Resistencia 1k5	2	0,15	0,30
MOSFET IR2111	2	3,83	7,66
Comparador LM319N	1	2,33	2,33
Diodo	2	0,74	1,48
Regleta Clemar	3	0,95	2,85
MOSFET IRF5040	4	3,21	12,84

Total	44,91€

Tabla 10.2- Presupuesto de la placa PCB.

PRESUPUESTO TOTAL MATERIAL PRINCIPAL	166,10 €
PRESUPUESTO TOTAL MATERIAL PLACA PCB	44,91€
PRESUPUESTO TOTAL MATERIAL EN CONJUNTO	211,01 €

Tabla 10.3- Presupuesto material total.

10.3 PRESUPUESTO MANO DE OBRA

En este apartado se tiene en cuenta la mano de obra necesaria para realizar el proyecto.

MANO DE OBRA			
PROFESION	PRECIO/H	NÚMERO HORAS	Precio Total
Ingeniero	14,36 €	336	4824,96 €

COSTE MANO DE OBRA	4824,96 €

Tabla 10.4 - Presupuesto mano de obra.

10.4 IVA (21%)

A todo proyecto hay que aplicarle los impuestos donde se implementa que serán del 21%.

IVA	
TOTAL PRESUPUESTO	5035,97 €
IVA(21%)	1057,56 €

TOTAL PRESUPUESTO	6093,53 €

Tabla 10.5 - Presupuesto total del prototipo.

El presupuesto final del prototipo teniendo en cuenta el material, la mano de obra y el IVA será de 6093,53 €

11.Trabajo futuro y conclusiones

11.1 TRABAJO FUTURO

Con este proyecto se ha conseguido realizar un prototipo para el control de velocidad de un motor de CC. Se ha realizado un trabajo en el modelo del motor que nos ha permitido conocer su comportamiento. Por otra parte, también hemos visto cómo utilizar Arduino en MATLAB, lo que nos ha permitido abrir una ventana de conocimiento muy versátil para proyectos similares de control y modelado de motores.

Todas estas aptitudes adquiridas a lo largo de la realización del proyecto son de gran utilidad y servirán de base para futuros ensayos, como por ejemplo: ampliación del control de velocidad para poder leer en qué dirección se está moviendo el motor, control del sentido del movimiento que deseemos el movimiento, regulación del par motor o control de posición.

11.2 CONCLUSIONES

Visto todos los puntos sobre los que se ha trabajado a lo largo de este proyecto, se pueden obtener las siguientes conclusiones:

En primer lugar, podemos constatar que los valores de los parámetros del motor obtenidos mediante el modelo estático corresponden con los valores indicados por el fabricante en su hoja de características.

Por otra parte, hemos conseguido implementar un código en Arduino capaz de realizar correctamente el control de la velocidad y de la dirección del motor.

Seguidamente hemos comprobado que la lectura del Encoder utilizado, refleja correctamente las lecturas obtenidas por el osciloscopio. Lo que nos permite realizar con fiabilidad el control de velocidad del motor de CC.

Finalmente, con la realización de este prototipo, junto con sus detalles técnicos, cálculos, esquemas y toda la información que se encuentra en este trabajo. Se concluye una guía y plataforma educacional adecuada y útil para la realización de las prácticas de la asignatura "Alimentación y accionamiento de prototipos mecatrónicos".

12. Bibliografía

[1] Federico Vargas-Machuca Saldarriaga; *Máquinas Eléctricas Rotativas*; Ed. Megaprint; 1ª Edición; Perú, 1990.

[2] Salvador Martínez García, Juan Gualda Gil; *Electrónica de potencia. Componentes, topologías y equipos*; Ed. Paraninfo; 5ª Edición; Madrid; 2006.

[3] Stephen J. Chapman; *Máquinas Eléctricas*; Ed. McGraw Hill; 5ª Edición; México, DF; 2012.

[4] Jesús Fraile Ardanuy, Jesús Fraile Mora; *Accionamientos eléctricos*; Ed. Garceta; 2ª Edición; Madrid; 2019

[5] Jesús Fraile Mora; *Máquinas eléctricas*; Ed. McGraw Hill; 5ª Edición; Madrid; 2003.

[6] Pedro Ponce Cruz; *Máquinas Eléctricas. Técnicas Modernas de Control*; Ed. AlfaOmega; 2ª Edición; México 2016.

[7] Formación para la industria, https://automatismoindustrial.com/curso-carnet-instalador-baja-tension/motores/1-3-5-motores-de-corriente-continua/constitucion-motor-corriente-continua/

[8] Harmonic Drive SE, https://harmonicdrive.de/es/glosario/motor-dc
https://clr.es/blog/es/tipos-de-encoders-aplicaciones-motores/

[9] Implementar funciones de MATLAB en Simulink con bloques MATLAB Función, Implementar funciones de MATLAB en Simulink con bloques MATLAB Function - MATLAB & Simulink - MathWorks España

[10] Buil S-Functions Automatically Using S-Function Builder, Build S-Functions Automatically Using S-Function Builder - MATLAB & Simulink - MathWorks España

[11] Control de velocidad de motores, Motor Speed Control - MATLAB & Simulink (mathworks.com)

[12] Como trabajar con Arduino, Arduino - Home

[13] Ned Mohan; *Advanced Electric Drives. Analysis, Control and Modeling Using MATLAB/Simulink;* Ed. Wiley; 1ª Edición; Estados Unidos 2014.

Apéndice I: Datos del ensayo estático del motor con generador acoplado en vacio.

Tensión del motor (V)	Corriente del motor (A)	Tensión del generador (V)	Corriente del generador (A)	Potencia de entrada (W)	Potencia de salida (W)	Par inicial (Nm)	Constante motor	Par útil motor (Nm)	Par (Nm)
9,3	0,34	8,6	0	3,162	0	0,0109	0,062	0,0109	0,0328
10,1	0,35	9,2	0	3,505	0	0,0111	0,062	0,0111	0,0332
11,6	0,36	10,9	0	4,141	0	0,0113	0,062	0,0113	0,0339
12,3	0,36	11,3	0	4,453	0	0,0114	0,062	0,0114	0,0342
13,7	0,36	12,7	0	4,987	0	0,0116	0,062	0,0116	0,0349
15,1	0,37	14,2	0	5,647	0	0,0119	0,062	0,0119	0,0356
16,7	0,38	16,0	0	6,396	0	0,0121	0,062	0,0121	0,0363
17,4	0,39	16,5	0	6,716	0	0,0122	0,062	0,0122	0,0366
18,7	0,39	17,1	0	7,387	0	0,0124	0,062	0,0124	0,0372
19,6	0,40	17,9	0	7,860	0	0,0125	0,062	0,0125	0,0375
21,1	0,40	19,2	0	8,503	0	0,0128	0,062	0,0128	0,0388
22,0	0,41	20,1	0	8,998	0	0,0129	0,062	0,0129	0,0383

23,1	0,41	21,7	0	9,494	0	0,0131	0,062	0,0131	0,0388
23,8	0,41	22,2	0	9,853	0	0,0132	0,062	0,0132	0,0393

Tabla 13.1- Tabla de datos del modelo estático del motor con generador acoplado en vacío [1].

Tensión del motor (V)	Frecuencia (kHz)	Velocidad (rpm)	Velocidad (rad/s)	Potencia útil (Nm)	E (V)	Rendimiento motor	Rendimiento generador
9,3	22,75	1365,0	142,94	4,69	8,60	0	0
10,1	25,24	1514,4	158,59	5,27	9,20	0	0
11,6	28,82	1729,2	181,08	6,14	10,90	0	0
12,3	30,33	1819,8	190,57	6,51	11,3	0	0
13,7	34,78	2086,8	218,53	7,64	12,7	0	0
15,1	38,25	2295,0	240,33	8,55	14,2	0	0
16,7	42,25	2535,0	265,46	9,63	16,0	0	0
17,4	43,87	2632,2	275,64	10,08	16,5	0	0
18,7	47,76	2865,6	300,08	11,18	17,1	0	0
19,6	49,43	2969,8	310,58	11,66	17,9	0	0
21,1	53,74	3224,4	337,66	12,93	19,2	0	0
22,0	56,64	3398,4	355,88	13,81	20,1	0	0
23,1	59,50	3570,0	373,85	14,70	21,7	0	0
23,8	61,04	3662,4	383,53	15,19	22,2	0	0

Tabla 13.2 - Tabla de datos del modelo estático del motor con generador acoplado en vacío [2].

Apéndice II: Datos del ensayo estático del motor con generador acoplado en carga.

Tensión del motor (V)	Corriente del motor (A)	Tensión del generador (V)	Corriente del generador (A)	Potencia de entrada (W)	Potencia de salida (W)	Par inicial (Nm)	Constante motor	Par útil motor (Nm)	Par (Nm)
9,3	0,99	7,98	0,649	9,19	5,18	0,0109	0,062	0,051	0,0729
10,1	1,05	8,08	0,664	10,61	5,37	0,0109	0,062	0,052	0,0740
11,6	1,19	10,4	0,817	13,80	8,50	0,0113	0,062	0,062	0,0844
12,3	1,25	10,8	0,853	15,38	9,21	0,0114	0,062	0,064	0,0870
13,7	1,35	12,2	0,967	18,50	11,80	0,0115	0,062	0,071	0,0946
15,1	1,47	13,2	1,05	22,20	13,86	0,0118	0,062	0,077	0,1005
16,7	1,62	14,3	1,22	27,05	17,45	0,0120	0,062	0,088	0,1116
17,4	1,60	14,7	1,28	27,84	18,82	0,0121	0,062	0,091	0,1156
18,7	1,76	16,8	1,44	32,91	24,19	0,0124	0,062	0,102	0,1264
19,6	1,75	17,5	1,46	34,30	25,55	0,0124	0,062	0,103	0,1277
21,1	2,02	18,9	1,53	42,62	28,92	0,0126	0,062	0,108	0,1328
22,0	2,10	19,1	1,64	46,20	31,32	0,0128	0,062	0,114	0,1400

23,1	2,30	20,7	1,73	53,13	35,81	0,0129	0,062	0,120	0,1461
23,8	2,34	21,3	1,82	55,69	38,77	0,0130	0,062	0,126	0,1520

Tabla 14.1 - Tabla de datos del modelo estático del motor con generador acoplado en carga [1].

Tensión del motor (V)	Frecuencia (kHz)	Velocidad (rpm)	Velocidad (rad/s)	Potencia útil (Nm)	fem (V)	Rendimiento motor	Rendimiento generador
9,3	22,03	1321,8	138,42	10,09	8,09	0,770	0,766
10,1	22,94	1376,4	144,14	10,67	8,28	0,708	0,758
11,6	28,08	1684,8	176,43	14,89	10,19	0,791	0,824
12,3	30,12	1807,2	189,25	16,46	10,64	0,791	0,821
13,7	32,89	1973,4	206,65	19,54	12,06	0,799	0,840
15,1	37,16	2229,6	233,48	23,46	13,09	0,809	0,840
16,7	40,63	2437,8	255,29	28,49	15,21	0,827	0,807
17,4	42,23	2533,8	265,34	30,68	15,96	0,872	0,796
18,7	47,17	2830,2	296,38	37,47	17,96	0,915	0,819
19,6	47,63	2857,8	299,27	38,23	18,20	0,898	0,843
21,1	51,59	3095,4	324,15	43,04	19,08	0,818	0,869
22,0	53,73	3223,8	337,60	47,26	20,45	0,836	0,828
23,1	56,63	3397,8	355,82	51,98	21,57	0,805	0,854
23,8	58,42	3505,2	367,06	55,78	22,69	0,830	0,841

Tabla 14.2- Tabla de datos del modelo estático del motor con generador acoplado en carga [2].

Apéndice III: Hojas de características del Hardware

DCmind: DC DIRECT-DRIVE BRUSH MOTORS

Ø 63 mm - 209 W

- Silent motor
- 24 V built in EMC filter class B
- Excellent efficiency
- Long life
- IP65
- In accordance with UL - CE - ROHS regulations

Part numbers

	24 V	48 V	90 V	120 V
Type	89890	89890	89890	89890
Voltage	24 V⎓	48 V⎓	90 V⎓	120 V⎓
References				
Option: IP65 level	**89890011**	**89890003**	**89890004**	**89890005**
Option: holding brake 0.5 Nm, 24 V⎓	**89890511**	**89890503**	**89890504**	**89890505**
Option: 2 channels encoder 1000 pulses/ revolution, 5 V⎓	**89890911**	**89890903**	**89890904**	**89890905**
No-load characteristics				
Speed (rpm)	4000	3780	3700	3730
Absorbed current (A)	0.34	0.16	0.09	0.07
Nominal characteristics				
Speed (rpm)	3430	3370	3320	3350
Torque (mNm)	290	290	290	290
Output power (W)	104	102	101	102
Absorbed current (A)	5.4	2.53	1.34	1.01
Efficiency (%)	80	84	84	84
Maximum efficiency characteristics				
Speed (rpm)	3660	3480	3410	3430
Torque (mNm)	179	207	218	230
Output power (W)	69	75	78	83
Absorbed current (A)	3.5	1.9	1	0.82
Efficiency (%)	82	84	84	84
General characteristics				
Insulation conforming to IEC60085	Class E	Class E	Class E	Class E
Noise level (dBA)	35	35	35	35
Max. output power (W)	209	265	269	281
Starting torque (mNm)	2000	2680	2780	2875
Starting current (A)	35.3	22.2	12.1	9.4
Resistance (Ω)	0.7	2.2	7.4	12.8
Inductance (mH)	0.73	3.3	12	21
Torque constant (mNm/A)	57	122	232	308
Electrical time constant (ms)	1.1	1.5	1.6	1.6
Mechanical time constant (ms)	13	9	9	9
Inertia (g.cm²)	650	650	650	650
Weight (g)	1600	1600	1600	1600
Commutator segments	12	12	12	12
Service life (h)	5000	5000	5000	5000
Wires length (mm)	200	200	200	200
Ball bearing	✓	✓	✓	✓
Comments				
IP65 level except for the output shaft. Encoder and brake options are IP20.				

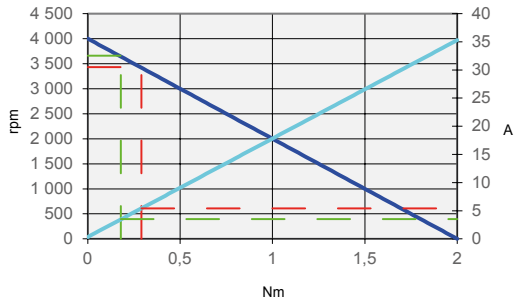
Product adaptations, contact us

- Special output shaft
- Shaft with pinion, pulley, worm gear
- Special supply voltage
- Other wire length
- Optical or Hall effect encoder - 1 or 2 channels
- Specific motor mounting flange
- Special motor connectors
- IP67, IP69K

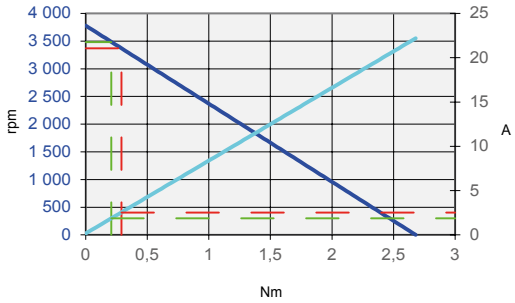
Product made to order

Curves

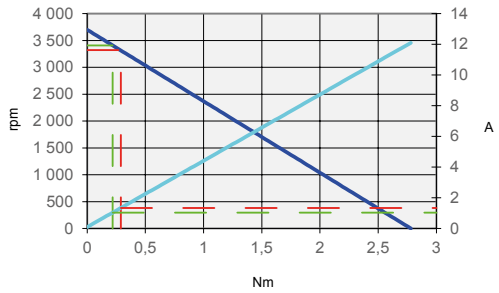
89890011 - 89890511 - 89890911



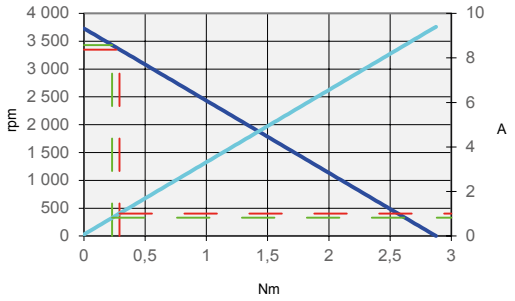
89890003 - 89890503 - 89890903



89890004 - 89890504 - 89890904

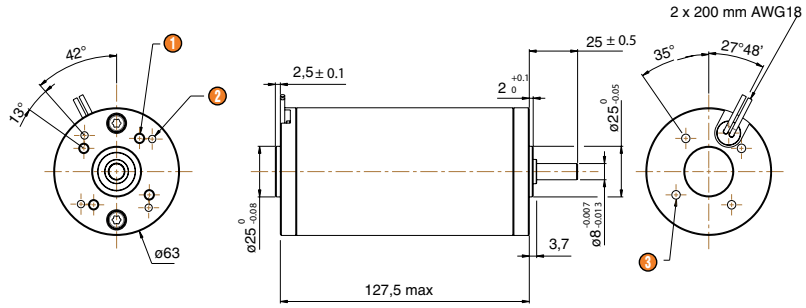


89890005 - 89890505 - 89890905



- **Speed (rpm)**
- **Current (A)**
- **Torque at nominal**
- **Torque at maximum efficiency**

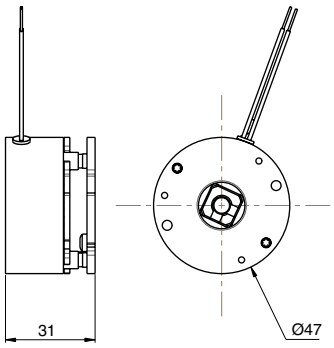
Dimensions (mm)



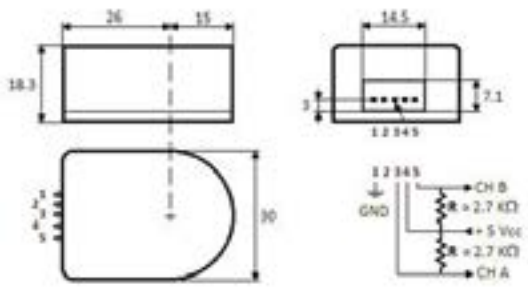
1. 4 x M5 at 90° depth 10 over Ø 40
2. 4 x Ø 3.65 at 90° depth 8 over Ø 48
3. 4 x M5 at 90° depth 7 over Ø 40

Options

Holding brake 0.5 Nm



Encoder



Agilent Technologies

Quick Assembly Two and Three Channel Optical Encoders

Technical Data

HEDM-550x/560x
HEDS-550x/554x
HEDS-560x/564x

Features

- **Two Channel Quadrature Output with Optional Index Pulse**
- **Quick and Easy Assembly**
- **No Signal Adjustment Required**
- **External Mounting Ears Available**
- **Low Cost**
- **Resolutions Up to 1024 Counts Per Revolution**
- **Small Size**
- **-40°C to 100°C Operating Temperature**
- **TTL Compatible**
- **Single 5 V Supply**

Description

The HEDS-5500/5540, HEDS-5600/5640, and HEDM-5500/5600 are high performance, low cost, two and three channel optical incremental encoders. These encoders emphasize high reliability, high resolution, and easy assembly.

Each encoder contains a lensed LED source, an integrated circuit with detectors and output circuitry, and a codewheel which rotates between the emitter and detector IC. The outputs of the HEDS-5500/5600 and HEDM-5500/ 5600 are two square waves in quadrature. The HEDS-5540 and 5640 also have a third channel index output in addition to the two channel quadrature. This index output is a 90 electrical degree, high true index pulse which is generated once for each full rotation of the codewheel.

The HEDS series utilizes metal codewheels, while the HEDM series utilizes a film codewheel allowing for resolutions to 1024 CPR. The HEDM series is nont available with a third channel index.

These encoders may be quickly and easily mounted to a motor. For larger diameter motors, the HEDM-5600, and HEDS-5600/5640 feature external mounting ears.

The quadrature signals and the index pulse are accessed through five 0.025 inch square pins located on 0.1 inch centers.

Standard resolutions between 96 and 1024 counts per revolution are presently available. Consult local Agilent sales representatives for other resolutions.

Applications

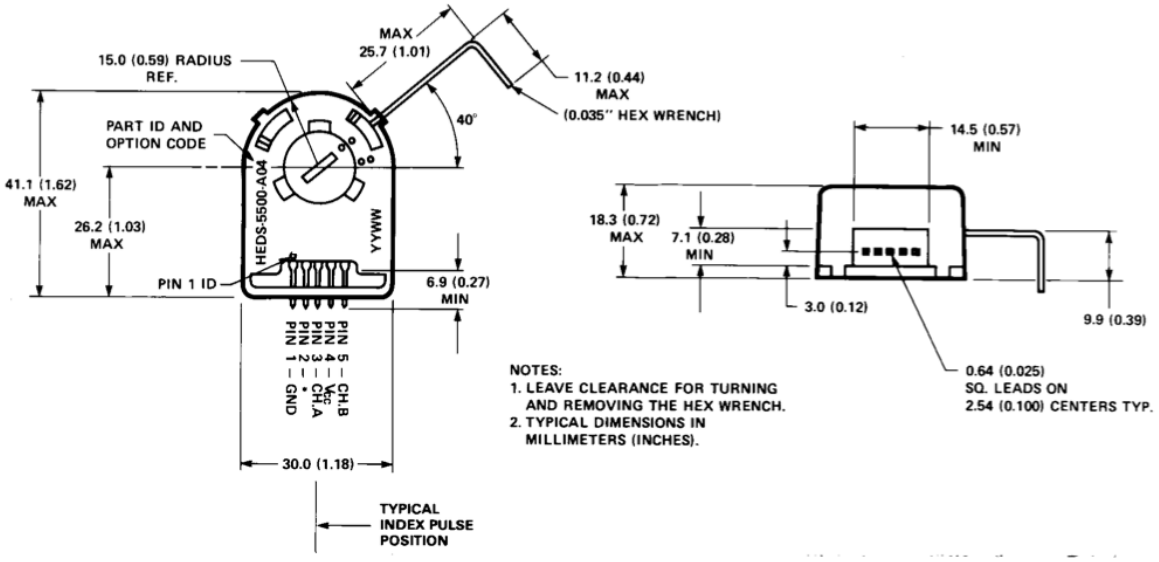
The HEDS-5500, 5540, 5600, 5640, and the HEDM-5500, 5600 provide motion detection at a low cost, making them ideal for high volume applications. Typical applications include printers, plotters, tape drives, positioning tables, and automatic handlers.

Note: Agilent Technologies encoders are not recommended for use in safety critical applications. Eg. ABS braking systems, power steering, life support systems and critical care medical equipment. Please contact sales representative if more clarification is needed.

ESD WARNING: NORMAL HANDLING PRECAUTIONS SHOULD BE TAKEN TO AVOID STATIC DISCHARGE.

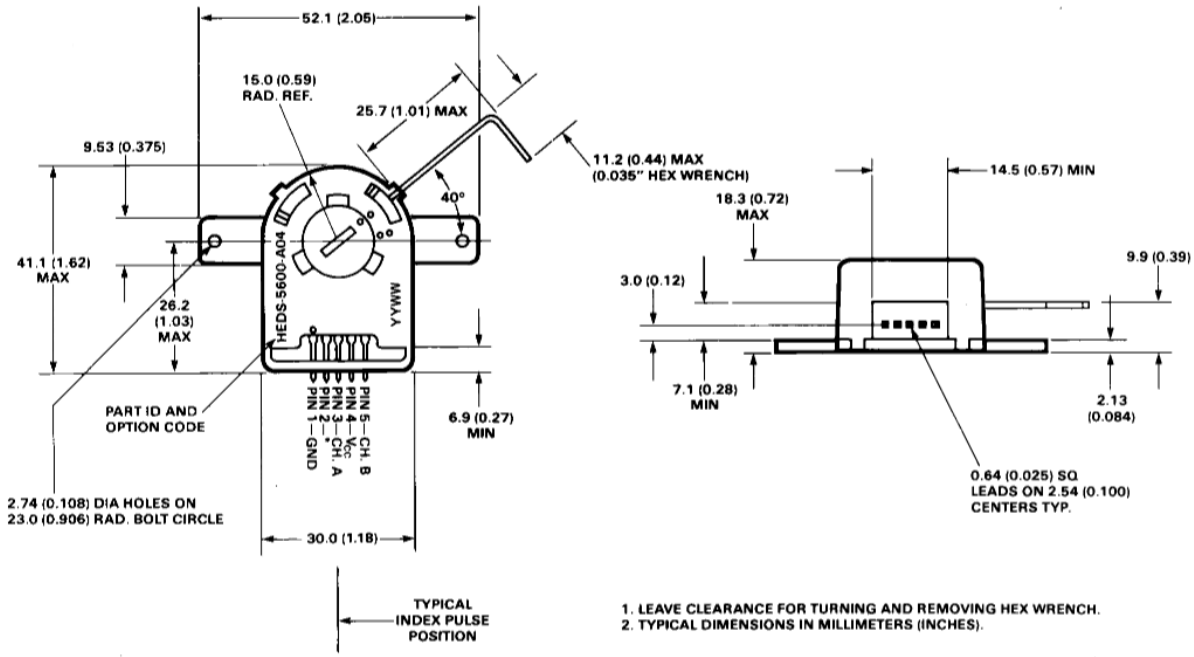
Package Dimensions

HEDS-5500/5540, HEDM-5500



***Note:** For the HEDS-5500 and HEDM-5500, Pin #2 is a No Connect. For the HEDS-5540, Pin #2 is CH. I, the index output.

HEDS-5600/5640, HEDM-5600



***Note:** For the HEDS-5600 and HEDM-5600, Pin #2 is a No Connect. For the HEDS-5640, Pin #2 is CH. I, the index output.

Theory of Operation

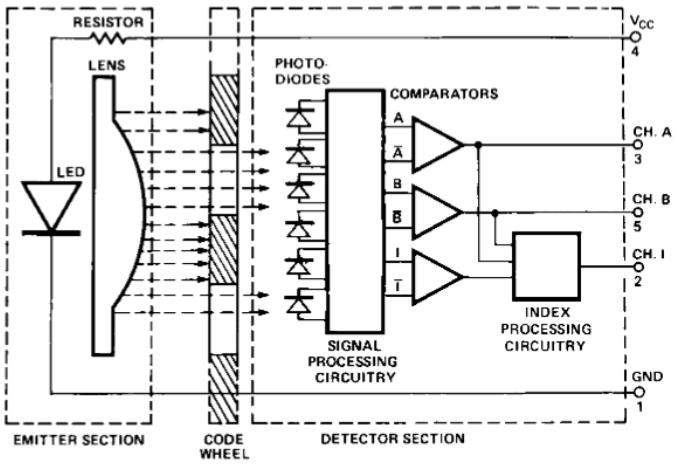
The HEDS-5500, 5540, 5600, 5640, and HEDM-5500, 5600 translate the rotary motion of a shaft into either a two- or a three-channel digital output.

As seen in the block diagram, these encoders contain a single Light Emitting Diode (LED) as its light source. The light is collimated into a parallel beam by means of a single polycarbonate lens located directly over the LED. Opposite the emitter is the integrated detector circuit. This IC consists of multiple sets of photodetectors and the signal processing circuitry necessary to produce the digital waveforms.

The codewheel rotates between the emitter and detector, causing the light beam to be interrupted by the pattern of spaces and bars on the codewheel. The photodiodes which detect these interruptions are arranged in a pattern that corresponds to the radius and design of the codewheel. These detectors are also spaced such that a light period on one pair of detectors corresponds to a dark period on the adjacent pair of detectors. The photodiode outputs are then fed through the signal processing circuitry resulting in A, $\overline{A}$, B and $\overline{B}$ (also I and $\overline{I}$ in the HEDS-5540 and 5640). Comparators receive these signals and produce the final outputs for channels A and B. Due to this integrated phasing technique, the digital output of channel A is in quadrature with that of channel B (90 degrees out of phase).

In the HEDS-5540 and 5640, the output of the comparator for I and $\overline{I}$ is sent to the index processing circuitry along with the outputs of channels A and B.

Block Diagram



NOTE: CIRCUITRY FOR CH. I IS ONLY IN HEDS-5540 AND 5640 THREE CHANNEL ENCODERS.

The final output of channel I is an index pulse P_O which is generated once for each full rotation of the codewheel. This output P_O is a one state width (nominally 90 electrical degrees), high true index pulse which is coincident with the low states of channels A and B.

Definitions

Count (N): The number of bar and window pairs or counts per revolution (CPR) of the codewheel.

One Cycle (C): 360 electrical degrees (°e), 1 bar and window pair.

One Shaft Rotation: 360 mechanical degrees, N cycles.

Position Error ($\Delta\Theta$): The normalized angular difference between the actual shaft position and the position indicated by the encoder cycle count.

Cycle Error (ΔC): An indication of cycle uniformity. The difference between an observed shaft angle which gives rise to one electrical cycle, and the nominal angular increment of 1/N of a revolution.

Pulse Width (P): The number of electrical degrees that an output is high during 1 cycle. This value is nominally 180°e or 1/2 cycle.

Pulse Width Error (ΔP): The deviation, in electrical degrees, of the pulse width from its ideal value of 180°e.

State Width (S): The number of electrical degrees between a transition in the output of channel A and the neighboring transition in the output of channel B. There are 4 states per cycle, each nominally 90°e.

State Width Error (ΔS): The deviation, in electrical degrees, of each state width from its ideal value of 90°e.

Phase (ϕ): The number of electrical degrees between the center of the high state of channel A and the center of the high state of channel B. This value is nominally 90°e for quadrature output.

Phase Error ($\Delta\phi$): The deviation of the phase from its ideal value of 90°e.

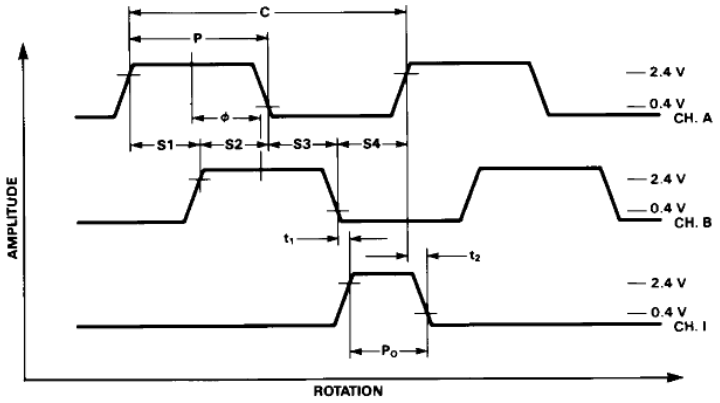
Absolute Maximum Ratings

Parameter	HEDS-55XX/56XX	HEDM-550X/560X
Storage Temperature, T_S	-40°C to 100°C	-40°C to +70°C
Operating Temperature, T_A	-40°C to 100°C	-40°C to +70°C
Supply Voltage, V_{CC}	-0.5 V to 7 V	-0.5 V to 7 V
Output Voltage, V_O	-0.5 V to V_{CC}	-0.5 V to V_{CC}
Output Current per Channel, I_{OUT}	-1.0 mA to 5 mA	-1.0 mA to 5 mA
Vibration	20 g, 5 to 1000 Hz	20 g, 5 to 1000 Hz
Shaft Axial Play	± 0.25 mm (± 0.010 in.)	± 0.175 mm (± 0.007 in.)
Shaft Eccentricity Plus Radial Play	0.1 mm (0.004 in.) TIR	0.04 mm (0.0015 in.) TIR
Velocity	30,000 RPM	30,000 RPM
Acceleration	250,000 rad/sec^2	250,000 rad/sec^2

Direction of Rotation: When the codewheel rotates in the counter-clockwise direction (as viewed from the encoder end of the motor), channel A will lead channel B. If the codewheel rotates in the clockwise direction, channel B will lead channel A.

Index Pulse Width (P_O): The number of electrical degrees that an index output is high during one full shaft rotation. This value is nominally 90°e or 1/4 cycle.

Output Waveforms



Recommended Operating Conditions

Parameter	Symbol	Min.	Typ.	Max.	Units	Notes
Temperature HEDS Series	T_A	-40		100	℃	
Temperature HEDM Series	T_A	-40		70	℃	non-condensing atmosphere
Supply Voltage	V_{CC}	4.5	5.0	5.5	Volts	Ripple < 100 mV_{p-p}
Load Capacitance	C_L			100	pF	2.7 kΩ pull-up
Count Frequency	f			100	kHz	Velocity (rpm) x N/60
Shaft Perpendicularity Plus Axial Play (HEDS Series)				± 0.25 (± 0.010)	mm (in.)	6.9 mm (0.27 in.) from mounting surface
Shaft Eccentricity Plus Radial Play (HEDS Series)				0.04 (0.0015)	mm (in.) TIR	6.9 mm (0.27 in.) from mounting surface
Shaft Perpendicularity Plus Axial Play (HEDM Series)				± 0.175 (± 0.007)	mm (in.)	6.9 mm (0.27 in.) from mounting surface
Shaft Eccentricity Plus Radial Play(HEDM Series)				0.04 (0.0015)	mm (in.) TIR	6.9 mm (0.27 in.) from mounting surface

Note: The module performance is guaranteed to 100 kHz but can operate at higher frequencies. 2.7 kΩ pull-up resistors required for HEDS-5540 and 5640.

Encoding Characteristics

Encoding Characteristics over Recommended Operating Range and Recommended Mounting Tolerances unless otherwise specified. Values are for the worst error over the full rotation.

Part No.	Description		Sym.	Min.	Typ.*	Max.	Units
HEDS-5500 HEDS-5600 (Two Channel)	Pulse Width Error		ΔP		7	45	°e
	Logic State Width Error		ΔS		5	45	°e
	Phase Error		Δϕ		2	20	°e
	Position Error		ΔΘ		10	40	min. of arc
	Cycle Error		ΔC		3	5.5	°e
HEDM-5500 HEDM-5600 (Two Channel)	Pulse Width Error		ΔP		10	45	°e
	Logic State Width Error		ΔS		10	45	°e
	Phase Error		Δϕ		2	15	°e
	Position Error		ΔΘ		10	40	min. of arc
	Cycle Error		ΔC		3	7.5	°e
HEDS-5540 HEDS-5640 (Three Channel)	Pulse Width Error		ΔP		5	35	°e
	Logic State Width Error		ΔS		5	35	°e
	Phase Error		Δϕ		2	15	°e
	Position Error		ΔΘ		10	40	min. of arc
	Cycle Error		ΔC		3	5.5	°e
	Index Pulse Width		P_O	55	90	125	°e
	CH. I rise after CH. A or CH. B fall	-40℃ to +100℃	t_2	-300	100	250	ns
	CH. I fall after CH. B or CH. A rise	-40℃ to +100℃	t_2	70	150	1000	ns

Note: See Mechanical Characteristics for mounting tolerances.
*Typical values specified at V_{CC} = 5.0 V and 25°C.

Electrical Characteristics

Electrical Characteristics over Recommended Operating Range.

Part No.	Parameter	Sym.	Min.	Typ.*	Max.	Units	Notes
HEDS-5500 HEDS-5600	Supply Current High Level Output Voltage Low Level Output Voltage	I_{CC} V_{OH} V_{OL}	 2.4 	17 	40 0.4	mA V V	 I_{OH} = -40 µA max. I_{OL} = 3.2 mA
	Rise Time Fall Time	t_r t_f	 	200 50	 	ns ns	C_L = 25 pF R_L = 11 kΩ pull-up
HEDS-5540 HEDS-5640 HEDM-5500 HEDM-5600	Supply Current High Level Output Voltage Low Level Output Voltage	I_{CC} V_{OH} V_{OL}	30 2.4 	57 	85 0.4	mA V V	 I_{OH} = -200 µA max. I_{OL} = 3.86 mA
	Rise Time Fall Time	t_r t_f	 	180 40	 	ns ns	C_L = 25 pF R_L = 2.7 kΩ pull-up
HEDM-5500 HEDM-5600	Supply Current High Level Output Voltage Low Level Output Voltage	I_{CC} V_{OH} V_{OL}	30 2.4 	57 	85 0.4	mA V V	 I_{OH} = -40 µA max. I_{OL} = 3.86 mA
	Rise Time Fall Time	t_r t_f	 	180 40	 	ns ns	C_L = 25 pF R_L = 3.2 kΩ pull-up

*Typical values specified at V_{CC} = 5.0 V and 25°C.

Mechanical Characteristics

Parameter	Symbol	Dimension	Tolerance[1]	Units
Codewheel Fits These Standard Shaft Diameters		2 3 4 5 6 8	+0.000 -0.015	mm
		5/32 1/8 3/16 1/4	+0.0000 -0.0007	in
Moment of Inertia	J	0.6 (8.0 x 10^{-6})		g-cm^2 (oz-in-s^2)
Required Shaft Length[2]		14.0 (0.55)	± 0.5 (± 0.02)	mm (in.)
Bolt Circle[3]	2 screw mounting	19.05 (0.750)	± 0.13 (± 0.005)	mm (in.)
	3 screw mounting	20.90 (0.823)	± 0.13 (± 0.005)	mm (in.)
	external mounting ears	46.0 (1.811)	± 0.13 (± 0.005)	mm (in.)
Mounting Screw Size[4]	2 screw mounting	M 2.5 or (2-56)		mm (in.)
	3 screw mounting	M 1.6 or (0-80)		mm (in.)
	external mounting ears	M 2.5 or (2-56)		mm (in.)
Encoder Base Plate Thickness		0.33 (0.130)		mm (in.)
Hub Set Screw		(2-56)		(in.)

Notes:
1. These are tolerances required of the user.
2. The HEDS-55X5 and 56X5, HEDM-5505, 5605 provide an 8.9 mm (0.35 inch) diameter hole through the housing for longer motor shafts. See Ordering Information.
3. The HEDS-5540 and 5640 must be aligned using the aligning pins as specified in Figure 3, or using the alignment tool as shown in "Encoder Mounting and Assembly". See also "Mounting Considerations."
4. The recommended mounting screw torque for 2 screw and external ear mounting is 1.0 kg-cm (0.88 in-lbs). The recommended mounting screw torque for 3 screw mounting is 0.50 kg-cm (0.43 in-lbs).

Electrical Interface

To insure reliable encoding performance, the HEDS-5540 and 5640 three channel encoders require 2.7 kΩ (± 10%) pull-up resistors on output pins 2, 3, and 5 (Channels I, A, and B) as shown in Figure 1. These pull-up resistors should be located as close to the encoder as possible (within 4 feet). Each of the three encoder outputs can drive a single TTL load in this configuration.

The HEDS-5500, 5600, and HEDM-5500, 5600 two channel encoders do not normally require pull-up resistors. However, 3.2 kΩ pull-up resistors on output pins 3 and 5 (Channels A and B) are recommended to improve rise times, especially when operating above 100 kHz frequencies.

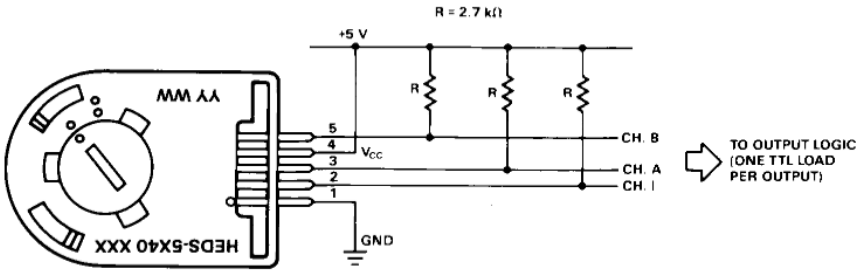


Figure 1. Pull-up Resistors on HEDS-5X40 Encoder Outputs.

Mounting Considerations

The HEDS-5540 and 5640 three channel encoders and the HEDM Series high resolution encoders must be aligned using the aligning pins as specified in Figure 3, or using the HEDS-8910 Alignment Tool as shown in Encoder Mounting and Assembly.

The use of aligning pins or alignment tool is recommended but not required to mount the HEDS-5500 and 5600. If these two channel encoders are attached to a motor with the screw sizes and mounting tolerances specified in the mechanical characteristics section without any additional mounting bosses, the encoder output errors will be within the maximums specified in the encoding characteristics section.

The HEDS-5500 and 5540 can be mounted to a motor using either the two screw or three screw mounting option as shown in Figure 2. The optional aligning pins shown in Figure 3 can be used with either mounting option.

The HEDS-5600, 5640, and HEDM-5600 have external mounting ears which may be used for mounting to larger motor base plates. Figure 4 shows the necessary mounting holes with optional aligning pins and motor boss.

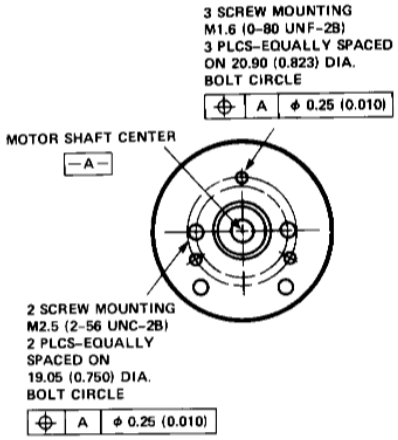


Figure 2. Mounting Holes.

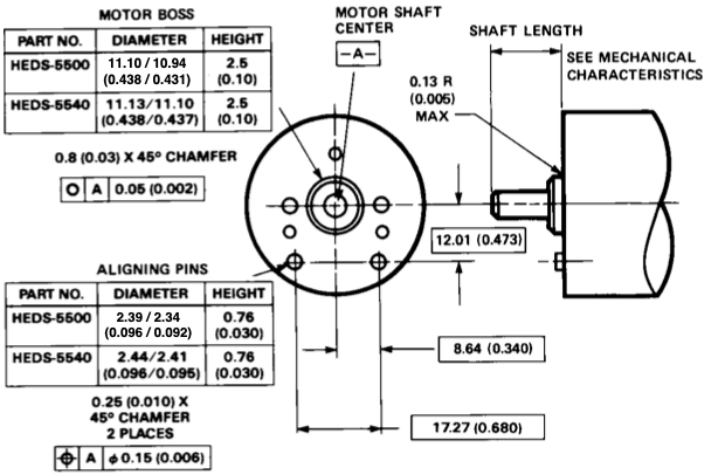


Figure 3. Optional Mounting Aids.

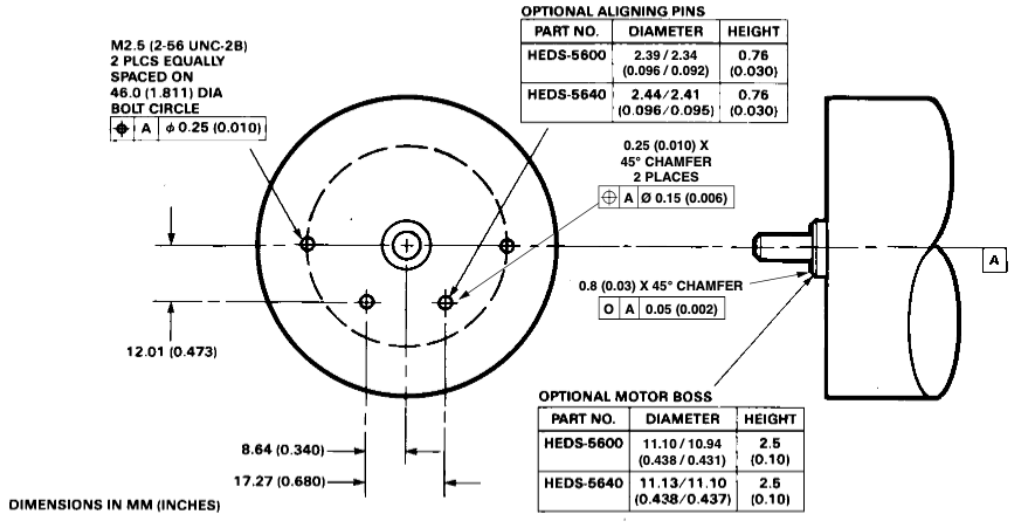
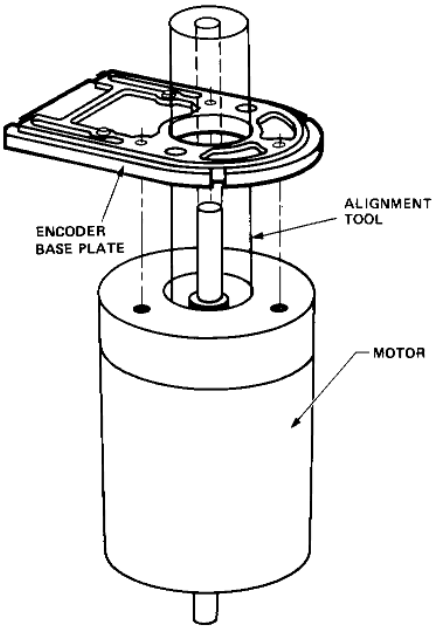


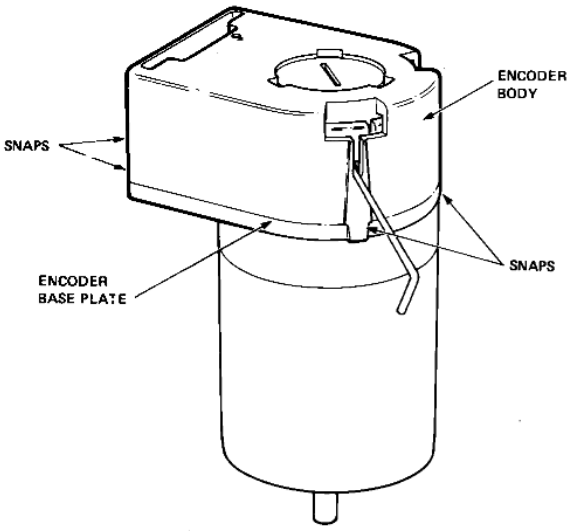
Figure 4. Mounting with External Ears.

Encoder Mounting and Assembly

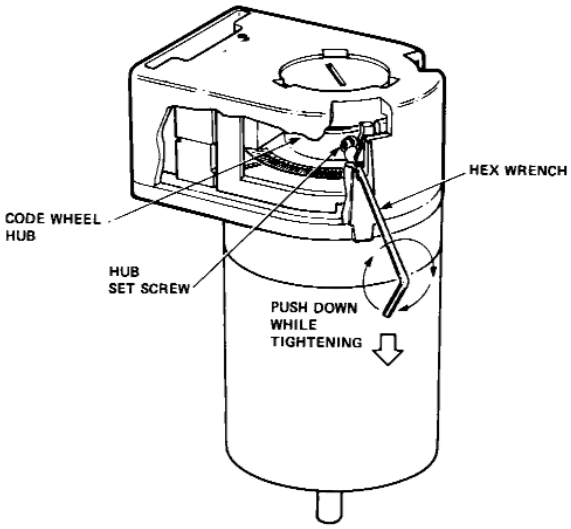


1. For HEDS-5500 and 5600: Mount encoder base plate onto motor. Tighten screws. Go on to step 2.

1a. For HEDS-5540, 5640 and HEDM-5500, 5600: Slip alignment tool onto motor shaft. With alignment tool in place, mount encoder baseplate onto motor as shown above. Tighten screws. Remove alignment tool.



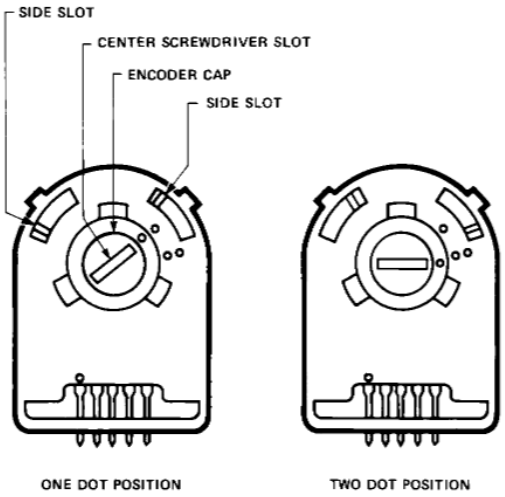
2. Snap encoder body onto base plate locking all 4 snaps.



3a. Push the hex wrench into the body of the encoder to ensure that it is properly seated into the code wheel hub set screws. Then apply a downward force on the end of the hex wrench. This sets the code wheel gap by levering the code wheel hub to its upper position.

3b. While continuing to apply a downward force, rotate the hex wrench in the clockwise direction until the hub set screw is tight against the motor shaft. The hub set screw attaches the code wheel to the motor's shaft.

3c. Remove the hex wrench by pulling it straight out of the encoder body.

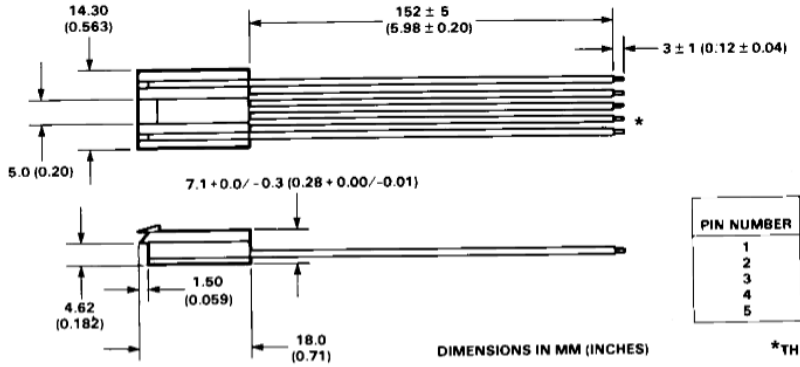


4. Use the center screwdriver slot, or either of the two side slots, to rotate the encoder cap dot clockwise from the one dot position to the two dot position. Do not rotate the encoder cap counterclockwise beyond the one dot position.

The encoder is ready for use!

Connectors

Manufacturer	Part Number
AMP	103686-4 640442-5
Dupont/Berg	65039-032 with 4825X-000 term.
Agilent (designed to mechanically lock into the HEDS-5XXX, HEDM-5X0X Series)	HEDS-8902 (2 ch.) with 4-wire leads
	HEDS-8903 (3 ch.) with 5-wire leads
Molex	2695 series with 2759 series term.

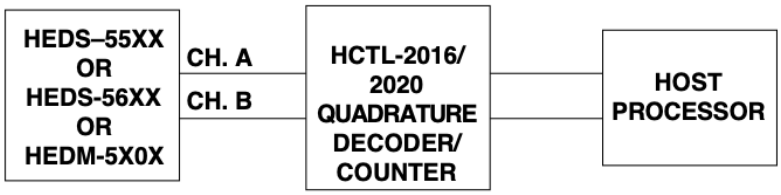


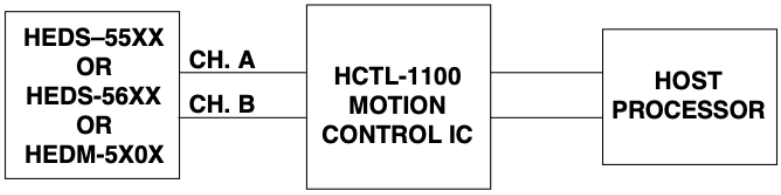
PIN NUMBER	PARAMETER	HEDS-8902 COLORS	HEDS-8903 COLORS
1	GROUND	BLACK	BLACK
2	CH. I	N/A*	BLUE
3	CH. A	WHITE	WHITE
4	V_{CC}	RED	RED
5	CH. B	BROWN	BROWN

*THIS WIRE IS NOT ON HEDS-8902.

Figure 5. HEDS-8902 and 8903 Connectors.

Typical Interfaces





Ordering Information

Encoders with Film Codewheels

HEDM-5 □ 0 □ Option □ □

Mounting Type	Outputs	Through Hole	Resolution (Cycles/Rev)	Shaft Diameter
5 - Standard 6 - External Mounting Ears	0 - 2 Channel	0 - None 5 - 8.9 mm (0.35 in.)	B - 1000 CPR J - 1024 CPR	01 - 2 mm 02 - 3 mm 03 - 1/8 in. 04 - 5/32 in. 05 - 3/16 in. 06 - 1/4 in. 11 - 4 mm 14 - 5 mm 12 - 6 mm 13 - 8 mm

HEDS-8910 0 □ □ Alignment Tool

(Included with each order of HEDM-550X/560X two channel encoders)

Encoders with Metal Codewheels

HEDS-5 □ □ □ Option □ □ □

Mounting Type	Outputs	Through Hole	Resolution (Cycles/Rev)	Shaft Diameter
5 - Standard 6 - External Mounting Ears	0 - 2 Channel 4 - 3 Channel	0 - None 5 - 8.9 mm (0.35 in.)	(HEDS-550X, 560X 2 Channel) S - 50 CPR K - 96 CPR C - 100 CPR D - 192 CPR E - 200 CPR F - 256 CPR G - 360 CPR H - 400 CPR A - 500 CPR I - 512 CPR (HEDS-554X, 564X 3 Channel) S - 50 CPR K - 96 CPR C - 100 CPR E - 200 CPR F - 256 CPR G - 360 CPR H - 400 CPR A - 500 CPR I - 512 CPR	01 - 2 mm 02 - 3 mm 03 - 1/8 in. 04 - 5/32 in. 05 - 3/16 in. 06 - 1/4 in. 11 - 4 mm 14 - 5 mm 12 - 6 mm 13 - 8 mm

HEDS-8910 0 □ □ Alignment Tool

(Included with each order of HEDS-554X/564X three channel encoders)

		01	02	03	04	05	06	11	12	13	14
HEDM-5500	**B** **J**	* 	* *				* *	*	* *	* *	* *
HEDM-5505	**B** **J**	 	 	 *	*	 	 *	 	 	 *	
HEDM-5600	**B** **J**	 	 	 	 	 	* *	 	 	* 	
HEDM-5605	**B** **J**	 	 	 	 	 	* *	 	 	 	
HEDS-5500	**A** **C** **E** **F** **G** **H** **I** **K** **S**	* * * *	* * * * * *	* * *	* * * * * *	* * * * * * * * *	* * * * * * * * *	* * * * * *	* * * * * * * *	* * * *	* * * * * * * *
HEDS-5505	**A** **C** **E** **F** **G** **H** **I** **K**	 	 	 	* * * * * * *	 	* * * * * * * *	 	 * 	* *	* * * * *
HEDS-5540	**A** **C** **E** **F** **G** **H** **I**	* * * *	* * *	*	*	*	* * * * * *	* * * * *	* * *	* * *	* * * * *
HEDS-5545	**A** **C** **H** **I**	 	 	 	 	 	 * *	 	* * 	 	* *
HEDS-5600	**A** **C** **E** **G** **H** **I**	 *	 	 	 	 	* * * * * 	 *	* * * 	* * 	* *

		01	02	03	40	05	06	11	12	13	14
HEDS-5605	**A** **C** **E** **F** **G** **H** **I**						* * * * * *			* * 	 *
HEDS-5640	**A** **E** **F** **H**						* * * *		*	*	
HEDS-5645	**A** **C** **E** **G** **H** **I**						* * 		* * 	* * * * *	 *

Agilent Technologies

www.agilent.com/semiconductors

For product information and a complete list of distributors, please go to our web site.

For technical assistance call:

Americas/Canada: +1 (800) 235-0312 or (408) 654-8675

Europe: +49 (0) 6441 92460

China: 10800 650 0017

Hong Kong: (+65) 271 2451

India, Australia, New Zealand: (+65) 271 2394

Japan: (+81 3) 3335-8152(Domestic/International), or 0120-61-1280(Domestic Only)

Korea: (+65) 271 2194

Malaysia, Singapore: (+65) 271 2054

Taiwan: (+65) 271 2654

Data subject to change.

Obsoletes 5988-2579EN
January 17, 2002
5988-3996EN

International
IOR Rectifier

Data Sheet No. PD-6.028C

IR2111

HALF-BRIDGE DRIVER

Features

- Floating channel designed for bootstrap operation
  Fully operational to +600V
  Tolerant to negative transient voltage
  dV/dt immune
- Gate drive supply range from 10 to 20V
- Undervoltage lockout for both channels
- CMOS Schmitt-triggered inputs with pull-down
- Matched propagation delay for both channels
- Internally set deadtime
- High side output in phase with input

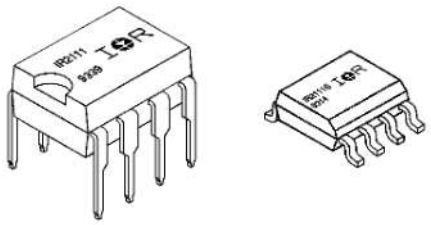
Product Summary

V_{OFFSET}	600V max.
I_O+/-	200 mA / 420 mA
V_{OUT}	10 - 20V
$t_{on/off}$ (typ.)	850 & 150 ns
Deadtime (typ.)	700 ns

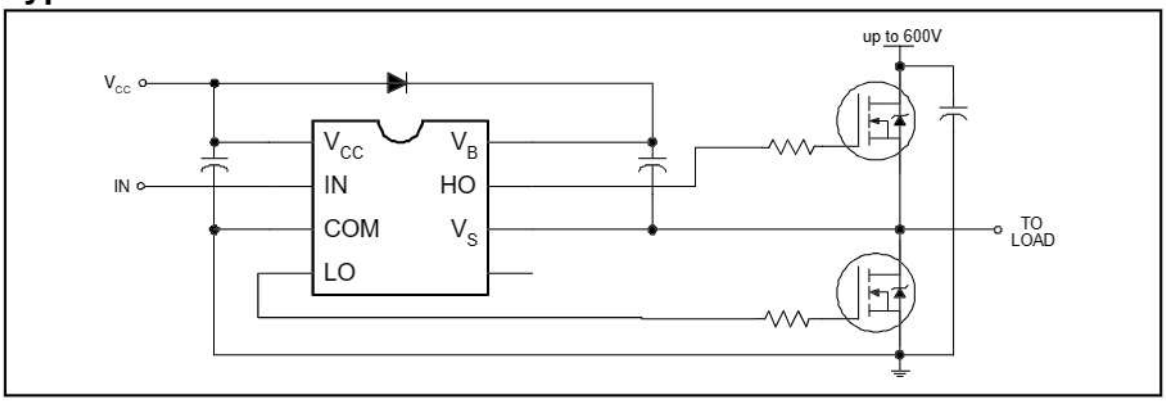
Description

The IR2111 is a high voltage, high speed power MOSFET and IGBT driver with dependent high and low side referenced output channels designed for half-bridge applications. Proprietary HVIC and latch immune CMOS technologies enable ruggedized monolithic construction. Logic input is compatible with standard CMOS outputs. The output drivers feature a high pulse current buffer stage designed for minimum driver cross-conduction. Internal deadtime is provided to avoid shoot-through in the output half-bridge. The floating channel can be used to drive an N-channel power MOSFET or IGBT in the high side configuration which operates up to 600 volts.

Packages



Typical Connection



Absolute Maximum Ratings

Absolute Maximum Ratings indicate sustained limits beyond which damage to the device may occur. All voltage parameters are absolute voltages referenced to COM. The Thermal Resistance and Power Dissipation ratings are measured under board mounted and still air conditions. Additional information is shown in Figures 7 through 10.

Parameter		Value		
Symbol	Definition	Min.	Max.	Units
V_B	High Side Floating Supply Voltage	-0.3	625	V
V_S	High Side Floating Supply Offset Voltage	V_B - 25	V_B + 0.3	
V_{HO}	High Side Floating Output Voltage	V_S - 0.3	V_B + 0.3	
V_{CC}	Low Side and Logic Fixed Supply Voltage	-0.3	25	
V_{LO}	Low Side Output Voltage	-0.3	V_{CC} + 0.3	
V_{IN}	Logic Input Voltage	-0.3	V_{CC} + 0.3	
dV_S/dt	Allowable Offset Supply Voltage Transient (Figure 2)	—	50	V/ns
P_D	Package Power Dissipation @ $T_A \leq +25°C$ (8 Lead DIP)	—	1.0	W
	(8 Lead SOIC)	—	0.625	
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient (8 Lead DIP)	—	125	°C/W
	(8 Lead SOIC)	—	200	
T_J	Junction Temperature	—	150	°C
T_S	Storage Temperature	-55	150	
T_L	Lead Temperature (Soldering, 10 seconds)	—	300	

Recommended Operating Conditions

The Input/Output logic timing diagram is shown in Figure 1. For proper operation the device should be used within the recommended conditions. The V_S offset rating is tested with all supplies biased at 15V differential.

Parameter		Value		
Symbol	Definition	Min.	Max.	Units
V_B	High Side Floating Supply Absolute Voltage	V_S + 10	V_S + 20	V
V_S	High Side Floating Supply Offset Voltage	Note 1	600	
V_{HO}	High Side Floating Output Voltage	V_S	V_B	
V_{CC}	Low Side and Logic Fixed Supply Voltage	10	20	
V_{LO}	Low Side Output Voltage	0	V_{CC}	
V_{IN}	Logic Input Voltage	0	V_{CC}	
T_A	Ambient Temperature	-40	125	°C

Note 1: Logic operational for V_S of -5 to +600V. Logic state held for V_S of -5V to $-V_{BS}$.

Dynamic Electrical Characteristics

V_{BIAS} (V_{CC}, V_{BS}) = 15V, C_L = 1000 pF and T_A = 25°C unless otherwise specified. The dynamic electrical characteristics are measured using the test circuit shown in Figure 3.

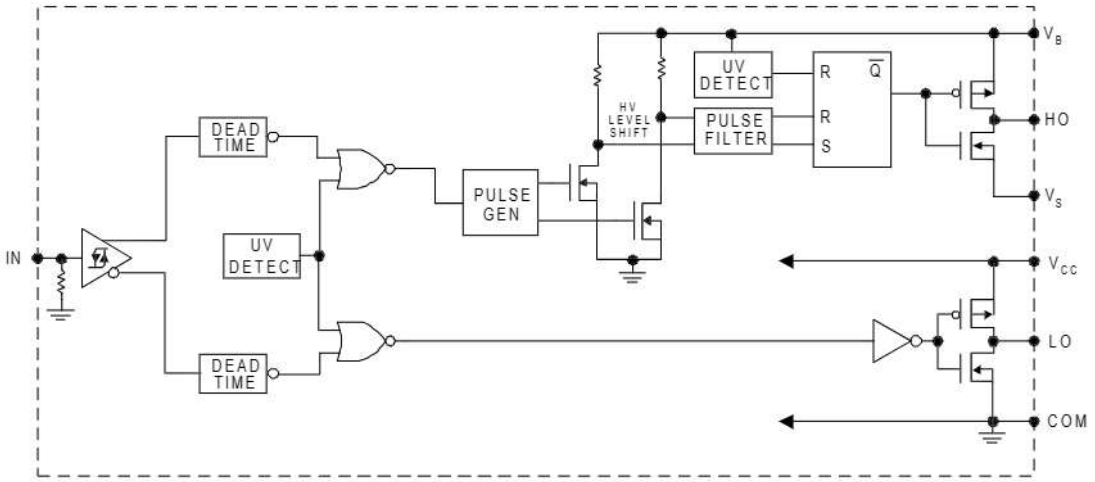
Symbol	Parameter Definition	Value Min.	Typ.	Max.	Units	Test Conditions
t_{on}	Turn-On Propagation Delay	—	850	1,000	ns	V_S = 0V
t_{off}	Turn-Off Propagation Delay	—	150	180		V_S = 600V
t_r	Turn-On Rise Time	—	80	130		
t_f	Turn-Off Fall Time	—	40	65		
DT	Deadtime, LS Turn-Off to HS Turn-On & HS Turn-Off to LS Turn-On	—	700	900		
MT	Delay Matching, HS & LS Turn-On/Off	—	30	—		

Static Electrical Characteristics

V_{BIAS} (V_{CC}, V_{BS}) = 15V and T_A = 25°C unless otherwise specified. The V_{IN}, V_{TH} and I_{IN} parameters are referenced to COM. The V_O and I_O parameters are referenced to COM and are applicable to the respective output leads: HO or LO.

Symbol	Parameter Definition	Value Min.	Typ.	Max.	Units	Test Conditions
V_{IH}	Logic "1" Input Voltage for HO & Logic "0" for LO	6.4	—	—	V	V_{CC} = 10V
		9.5	—	—		V_{CC} = 15V
		12.6	—	—		V_{CC} = 20V
V_{IL}	Logic "0" Input Voltage for HO & Logic "1" for LO	—	—	3.8		V_{CC} = 10V
		—	—	6.0		V_{CC} = 15V
		—	—	8.3		V_{CC} = 20V
V_{OH}	High Level Output Voltage, V_{BIAS} - V_O	—	—	100	mV	I_O = 0A
V_{OL}	Low Level Output Voltage, V_O	—	—	100		I_O = 0A
I_{LK}	Offset Supply Leakage Current	—	—	50	µA	V_B = V_S = 600V
I_{QBS}	Quiescent V_{BS} Supply Current	—	50	100		V_{IN} = 0V or V_{CC}
I_{QCC}	Quiescent V_{CC} Supply Current	—	70	180		V_{IN} = 0V or V_{CC}
I_{IN+}	Logic "1" Input Bias Current	—	20	40		V_{IN} = V_{CC}
I_{IN-}	Logic "0" Input Bias Current	—	—	1.0		V_{IN} = 0V
V_{BSUV+}	V_{BS} Supply Undervoltage Positive Going Threshold	7.3	8.4	9.5	V	
V_{BSUV-}	V_{BS} Supply Undervoltage Negative Going Threshold	7.0	8.1	9.2		
V_{CCUV+}	V_{CC} Supply Undervoltage Positive Going Threshold	7.6	8.6	9.6		
V_{CCUV-}	V_{CC} Supply Undervoltage Negative Going Threshold	7.2	8.2	9.2		
I_{O+}	Output High Short Circuit Pulsed Current	200	250	—	mA	V_O = 0V, V_{IN} = V_{CC} PW ≤ 10 µs
I_{O-}	Output Low Short Circuit Pulsed Current	420	500	—		V_O = 15V, V_{IN} = 0V PW ≤ 10 µs

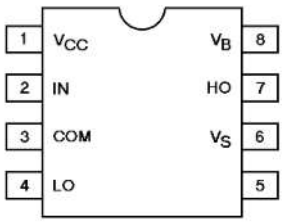
Functional Block Diagram



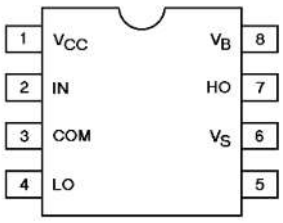
Lead Definitions

Lead Symbol	Description
IN	Logic input for high side and low side gate driver outputs (HO & LO), in phase with HO
V_B	High side floating supply
HO	High side gate drive output
V_S	High side floating supply return
V_{CC}	Low side and logic fixed supply
LO	Low side gate drive output
COM	Low side return

Lead Assignments



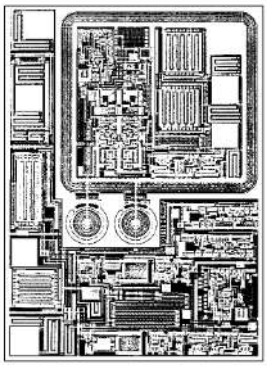
8 Lead DIP



SO-8

IR2111	IR2111S
Part Number	

Device Information

Process & Design Rule		HVDCMOS 4.0 µm
Transistor Count		164
Die Size		70 X 96 X 26 (mil)
Die Outline		
Thickness of Gate Oxide		800Å
Connections	Material	Poly Silicon
First Layer	Width	4 µm
	Spacing	6 µm
	Thickness	5000Å
Second Layer	Material	Al - Si (Si: 1.0% ±0.1%)
	Width	6 µm
	Spacing	9 µm
	Thickness	20,000Å
Contact Hole Dimension		8 µm X 8 µm
Insulation Layer	Material	PSG (SiO_2)
	Thickness	1.5 µm
Passivation	Material	PSG (SiO_2)
	Thickness	1.5 µm
Method of Saw		Full Cut
Method of Die Bond		Ablebond 84 - 1
Wire Bond	Method	Thermo Sonic
	Material	Au (1.0 mil / 1.3 mil)
Leadframe	Material	Cu
	Die Area	Ag
	Lead Plating	Pb : Sn (37 : 63)
Package	Types	8 Lead PDIP / SO-8
	Materials	EME6300 / MP150 / MP190
Remarks:		

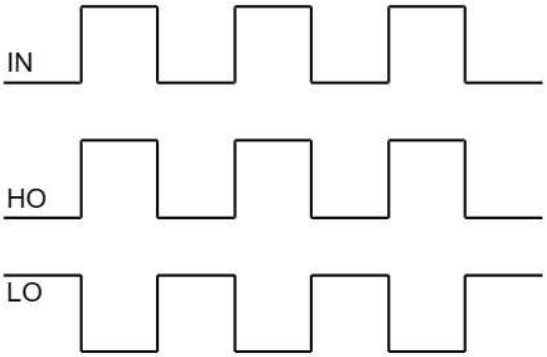


Figure 1. Input/Output Timing Diagram

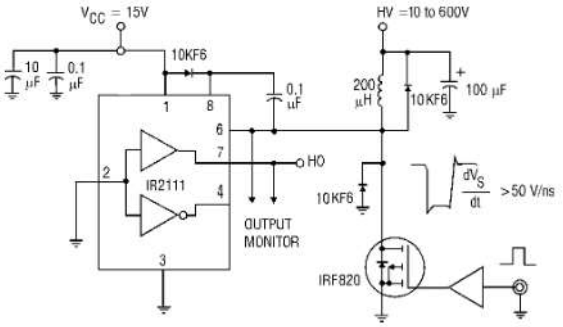


Figure 2. Floating Supply Voltage Transient Test Circuit

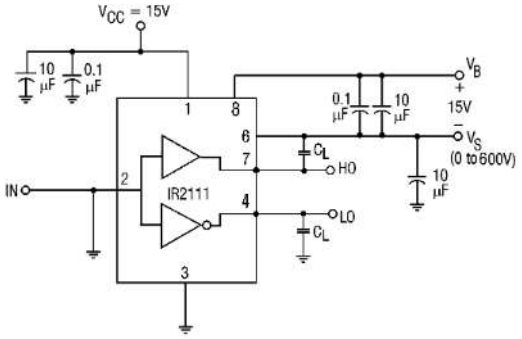


Figure 3. Switching Time Test Circuit

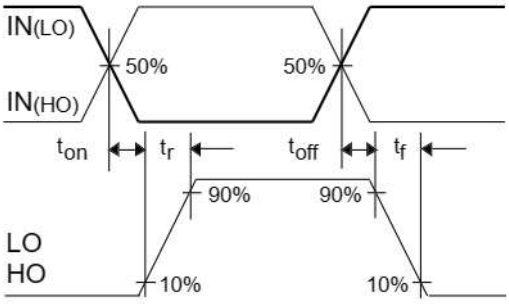


Figure 4. Switching Time Waveform Definition

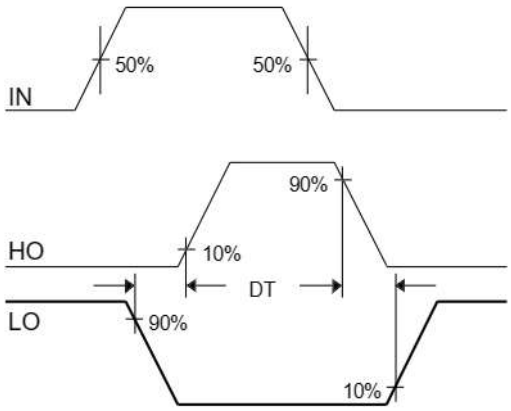


Figure 5. Deadtime Waveform Definitions

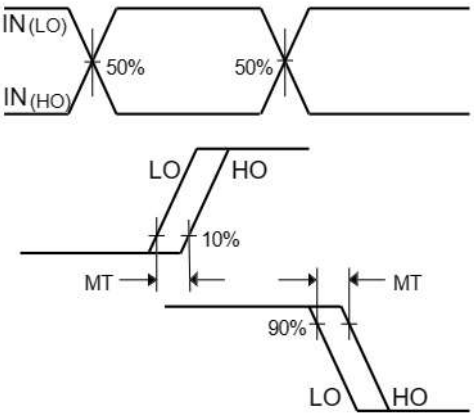


Figure 6. Delay Matching Waveform Definitions

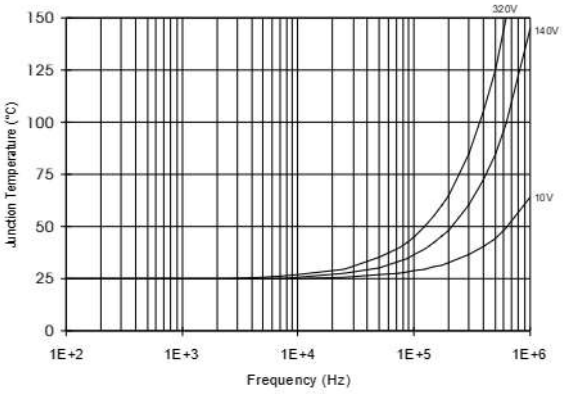


Figure 7. IR2111 T_J vs. Frequency (IRFBC20)
R_{GATE} = 33Ω, V_{CC} = 15V

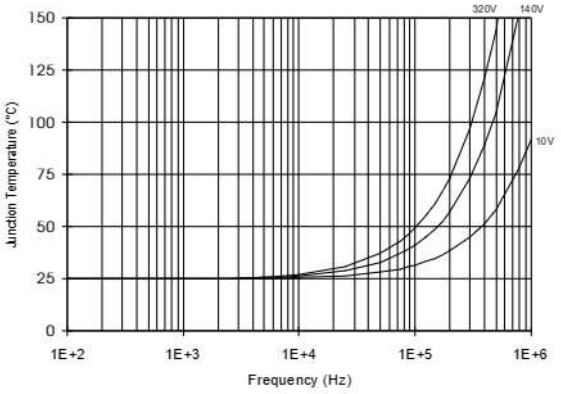


Figure 8. IR2111 T_J vs. Frequency (IRFBC30)
R_{GATE} = 22Ω, V_{CC} = 15V

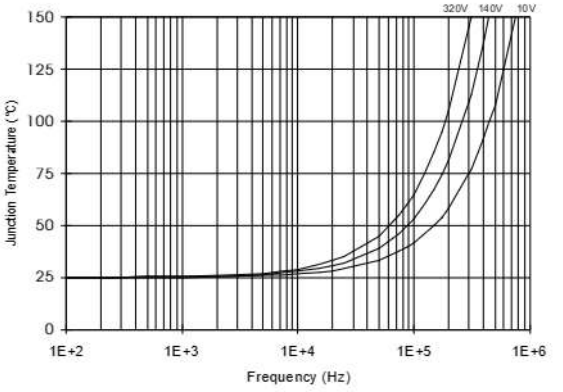


Figure 9. IR2111 T_J vs. Frequency (IRFBC40)
R_{GATE} = 15Ω, V_{CC} = 15V

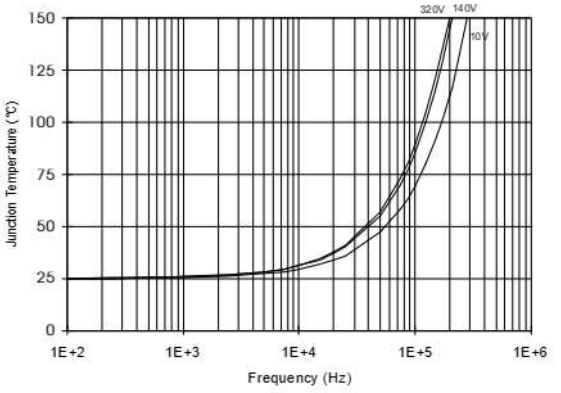
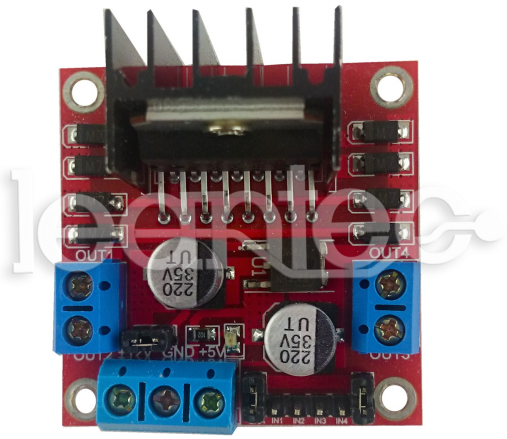


Figure 10. IR2111 T_J vs. Frequency (IRFPE50)
R_{GATE} = 10Ω, V_{CC} = 15V

Controlador de Motores L298N (Rojo)



Descripción:

La base de este módulo es el circuito integrado L298N, el cual es un doble puente H.
Este es capaz de manejar niveles altos voltaje y de corriente, además de estar diseñado para soportar cargas inductivas tales como relés, solenoides, motores de corriente continua y motores paso a paso. Este tipo de cargas las soporta gracias a unos diodos, los cuales absorben las corrientes inversas que producen estas cargas.

Dispone de dos puentes para habilitar o deshabilitar las salidas independientemente de las señales de entrada. También incorpora un interruptor para la conexión y desconexión de toda la placa.

Otra de las cosas muy útiles de la que dispone, es un regulador 7805, el cual, estabiliza la tensión de entrada de la placa a 5V y la entrega por una salida.

Descripción de las partes del Driver:

-1: Conector para la salida 3 y 4.

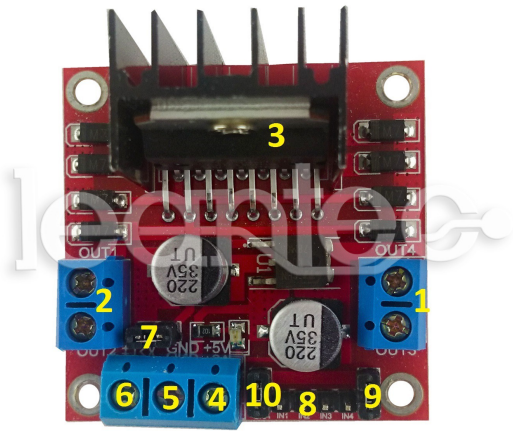
-2: Conector para la salida 1 y 2.

-3: Driver L298N.

-4: Salida de 5V.

-5: GND

-6: VCC



-7: Jumper para la activación.

-8: Pines para el control de giro de los motores.

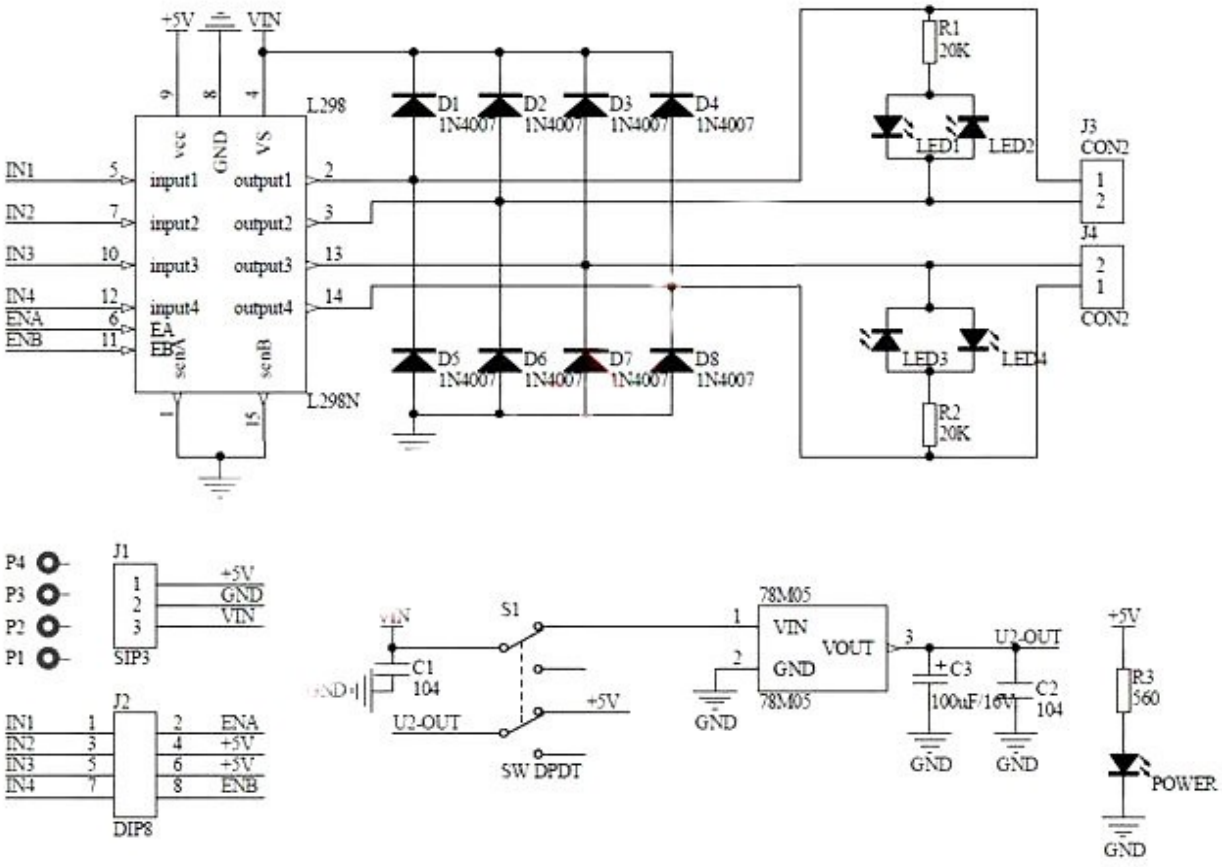
-9: Pin para el control PWM del motor 2.

-10: Pin para el control PWM del motor 1.

Características electrónicas:

-Driver: L298N.
-Tensión de alimentación del driver: 6-48V
-Intensidad máxima de cada canal del driver: 2A.
-Salida lógica de 5V.
-Potencia máxima 25W, a una temperatura de 75ºC.
-Temperatura de trabajo: -25ºC a 130ºC.
-Peso aproximado de módulo: 48g.

Esquema electrónico:



Ejemplo de conexión:

-Vcc: Tensión a la que van a funcionar los motores.

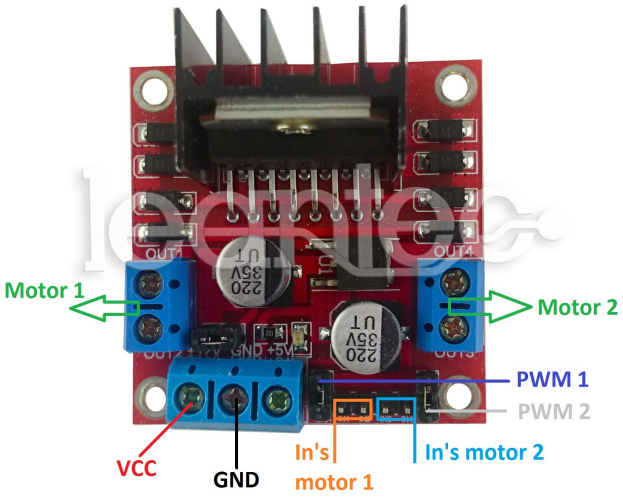
-Gnd.

- In´s motor 1: Pines para el control de giro del motor 1.

- In´s motor 2: Pines para el control de giro del motor 2.

-PWM 1: Control de velocidad del motor 1.

-PWM 2: Control de velocidad del motor 2.



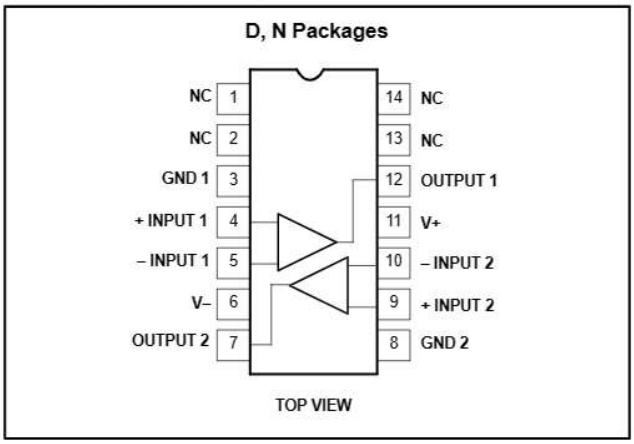
Dual voltage comparator | LM219/319

DESCRIPTION

The LM319 series are precision high-speed dual comparators fabricated on a single monolithic chip. They are designed to operate over a wide range of supply voltages down to a single 5V logic supply and ground. Further, they have higher gain and lower input currents than devices like the ∝A710. The uncommitted collector of the output stage makes the LM319 compatible with RTL, DTL, and TTL as well as capable of driving lamps and relays at currents up to 25mA.

Although designed primarily for applications requiring operation from digital logic supplies, the LM319 series are fully specified for power supplies up to ±15V. It features faster response than the LM111 at the expense of higher power dissipation. However, the high-speed, wide operating voltage range and low package count make the LM319 much more versatile than older devices like the ∝A711.

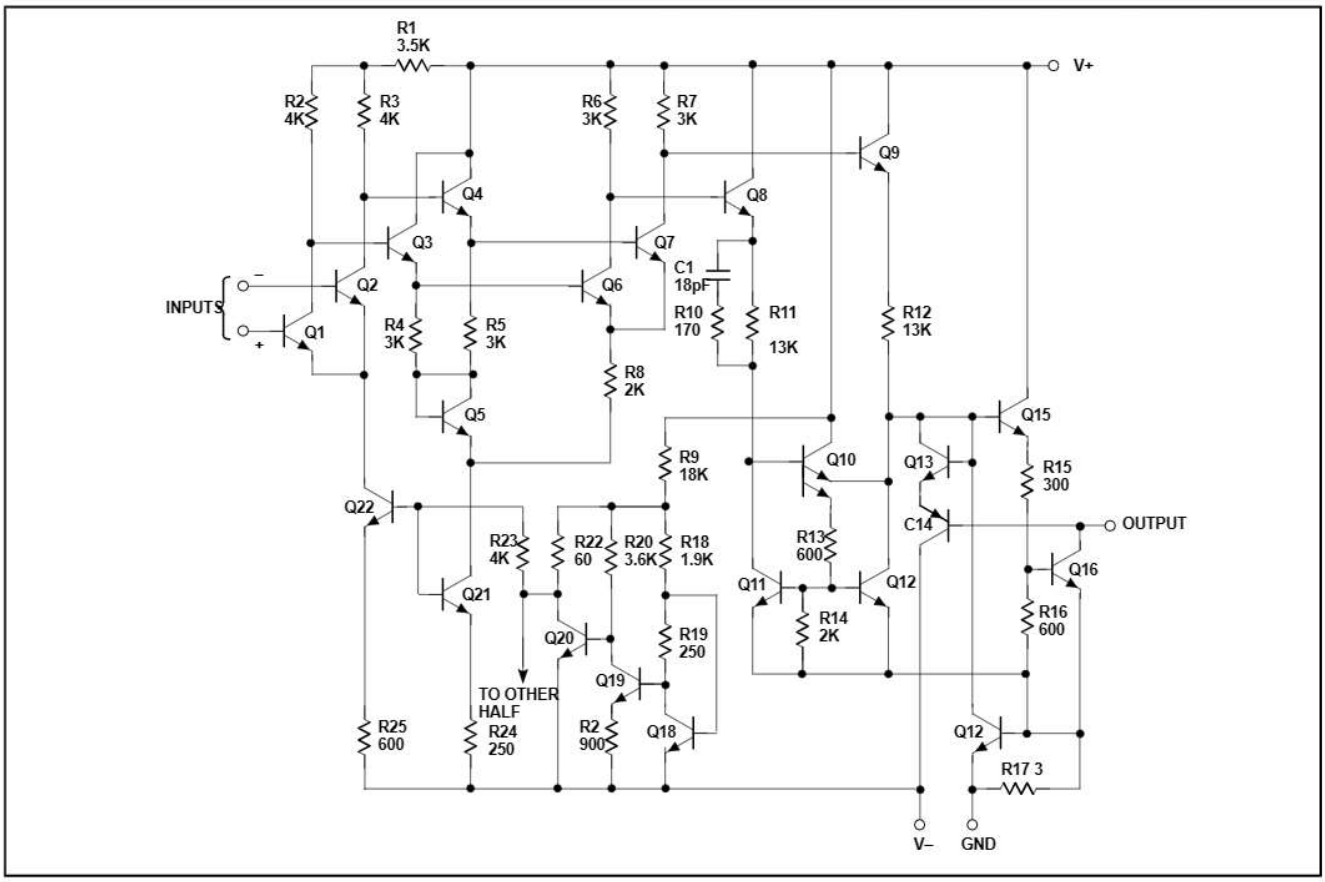
PIN CONFIGURATION



FEATURES

- Two independent comparators
- Operates from a single 5V supply
- Typically 80ns response time at ±15V
- Minimum fanout of 3 (each side)
- Maximum input current of 1∝A over temperature
- Inputs and outputs can be isolated from system ground
- High common-mode slew rate

EQUIVALENT SCHEMATIC



ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
14-Pin Plastic Small Outline (SO) Package	-25 to +85°C	LM219D	0175D
14-Pin Plastic Small Outline (SO) Package	0 to +70°C	LM319D	0175D
14-Pin Plastic Dual In-Line Package (DIP)	0 to +70°C	LM319N	0405B

ABSOLUTE MAXIMUM RATINGS

SYMBOL	PARAMETER	RATING	UNIT
V_S	Total supply voltage	36	V
	Output to negative supply voltage	36	V
	Ground to negative supply voltage	25	V
	Ground to positive supply voltage	18	V
	Differential input voltage	±5	V
V_{IN}	Input voltage[1]	±15	V
	Maximum power dissipation, T_A=25°C (still-air)[2] N package D package	 1420 1040	 mW mW
	Output short-circuit duration	10	s
T_A	Operating temperature range LM219 LM319	-25 to +85 0 to +70	°C
T_{STG}	Storage temperature range	-65 to +150	°C
T_{SOLD}	Lead soldering temperature (10sec max)	300	°C

NOTES:
1. For supply voltages less than ±15V, the absolute maximum rating is equal to the supply voltage.
2. Derate above 25°C, at the following rates:
   N package at 11.4mW/°C
   D package at 8.3mW/°C

DC ELECTRICAL CHARACTERISTICS

V_S=±15V, -25°C ≤ T_A ≤ 85°C for LM219, 0°C ≤ T_A ≤ 70°C for LM319, unless otherwise specified.

SYMBOL	PARAMETER	TEST CONDITIONS	LM219			LM319			UNIT
			Min	Typ	Max	Min	Typ	Max	
V_{OS}	Input offset voltage[1, 2]	R_S≤5kΩ, T_A=25°C Over temp.		0.7	4.0 7		2.0	8.0 10	mV
I_{OS}	Input offset current[1, 2]	T_A=25°C Over temp.		30	75 100		80	200 300	nA
I_B	Input bias current[1]	T_A=25°C Over temp.		150	500 1000		250	1000 1200	nA nA
A_V	Voltage gain	T_A=25°C	8	40		8	40		V/mV
V_{OL}	Saturation voltage	V_{IN}≤-10mV, I_{OUT}=25mA, T_A=25°C, V+≥4.5V, V-=0 V_{IN}≤-10mV, I_{OUT}=3.2mA		0.75 0.3	1.5 0.6		0.75 0.3	1.5 0.4	V
I_{OH}	Output leakage current	V-=0V, V_{IN}≥10mV V_{OUT}=35V, T_A=25°C		0.2	10		0.2	10	∝A
V_{IN}	Input voltage range	V_S=±15V V+=5V, V-=0V	1	±13	3	1	±13	3	V
V_{ID}	Differential input voltage				±5			±5	V
I+	Positive supply current	V+=5V, V-=0V, T_A=25°C		4.3			4.3		mA
I+	Positive supply current	V_S=±15V, T_A=25°C		8.0	12.5		8.0	12.5	mA
I-	Negative supply current	V_S=±15V, T_A=25°C		3.0	5.0		3.0	5.0	mA

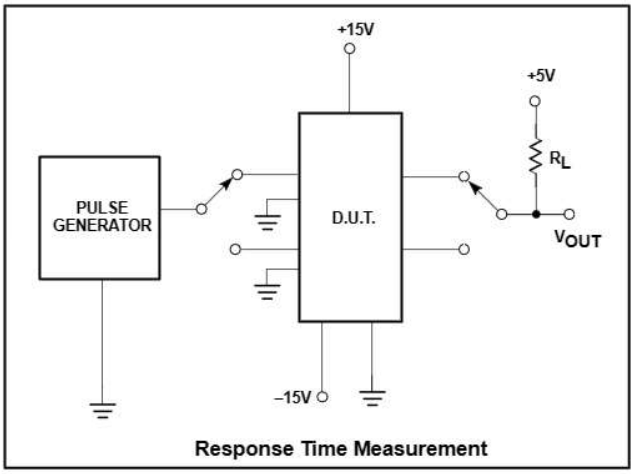
NOTES:
1. V_{OS}, I_{OS} and I_B specifications apply for a supply voltage range of V_S=±15V down to a single 5V supply.
2. The offset voltages and offset currents given are the maximum values required to drive the output to within 1V of either supply with a 1mA load. Thus these parameters define an error band and take into account the worst case effects of voltage gain and input impedance.

AC ELECTRICAL CHARACTERISTICS

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
t_R	Response time[1]	V_S=±15V, T_A=25°C R_L=500Ω (see test figure)		80		ns

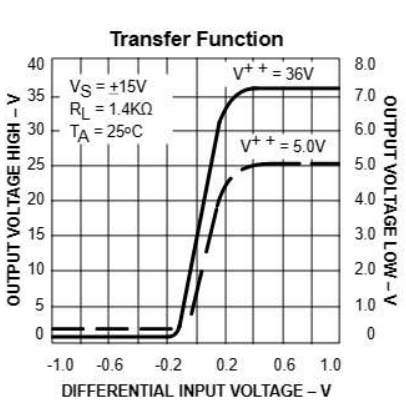
NOTES:
1. The response time specified is for a 100mV step with 5mV overdrive.

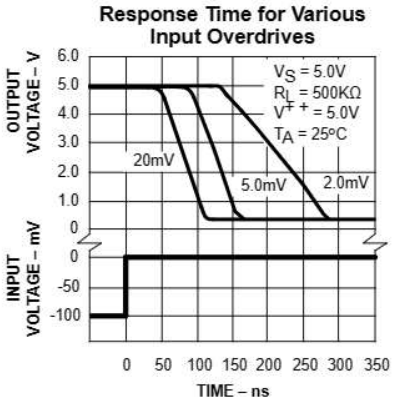
TEST CIRCUIT

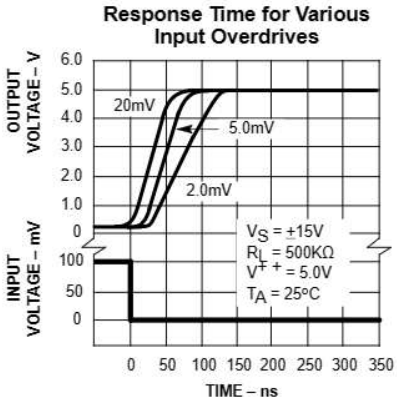


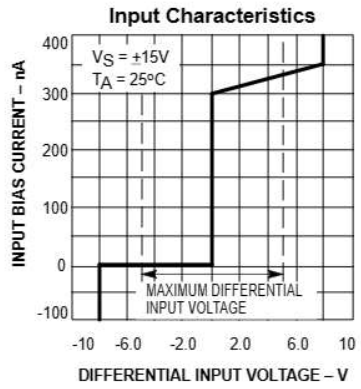
Response Time Measurement

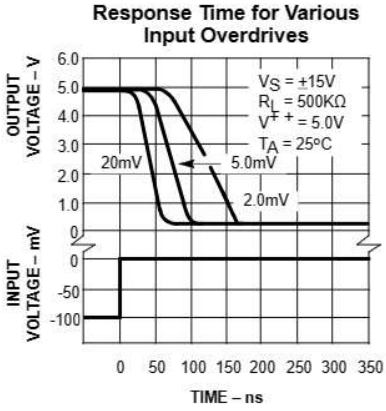
TYPICAL PERFORMANCE CHARACTERISTICS

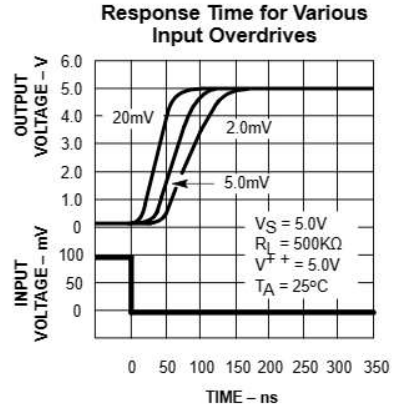


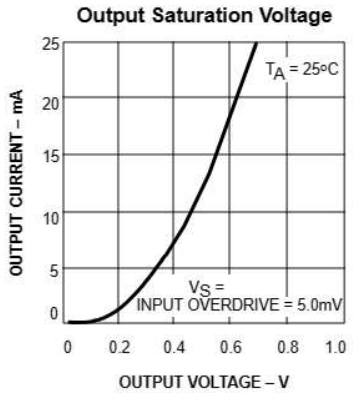


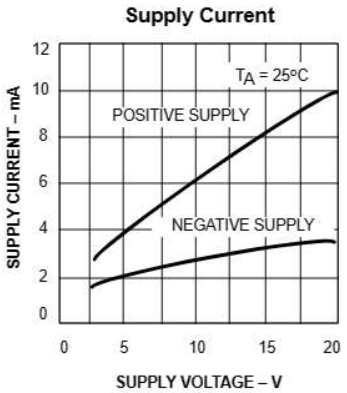


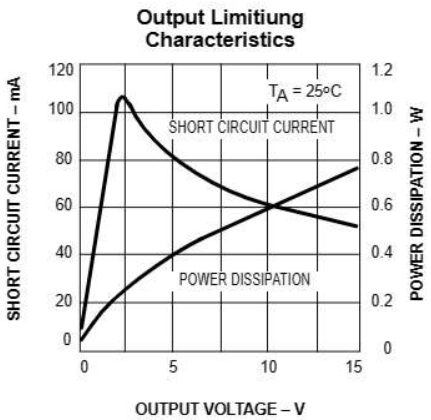




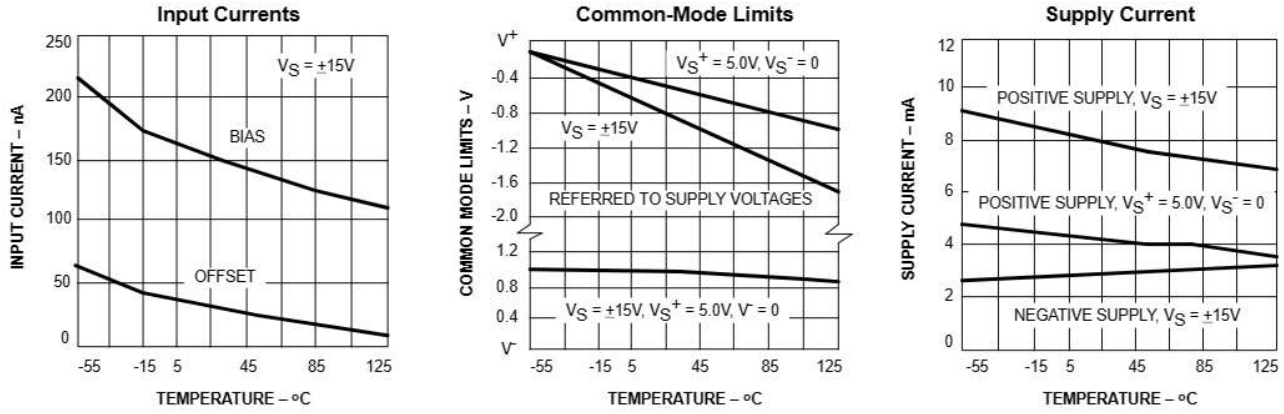




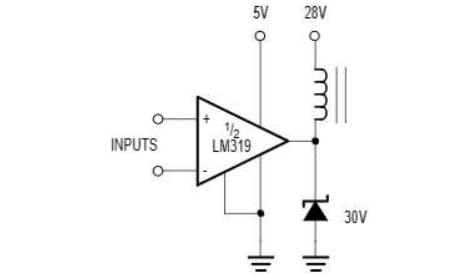




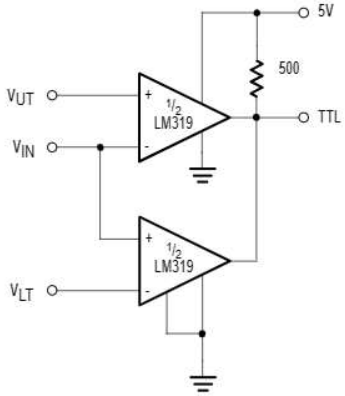
TYPICAL PERFORMANCE CHARACTERISTICS (Continued)



TYPICAL APPLICATIONS

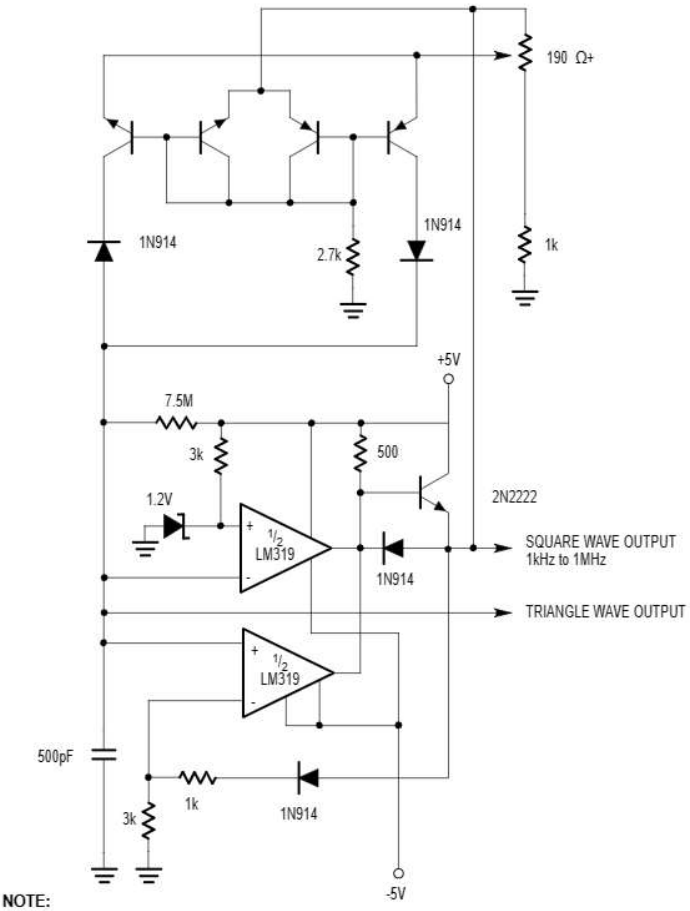


Relay Driver



Window Detector

NOTES:
V_{OUT} = 5V for $V_{LT} < V_{IN} < V_{UT}$
V_{OUT} = 0V for $V_{IN} < V_{LT}$ or $V_{IN} > V_{UT}$



NOTE:
Frequency adjust must be buffered for $R_L \leq 10\Omega$

Wide Range Variable Oscillator