ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática

Ingeniería de Sistemas y Automática

TRABAJO FIN DE GRADO

Diseño basado en microcontrolador para el control de un péndulo invertido

Grado en

Ingeniería Electrónica, Robótica y Mecatrónica

Autor: **Diego Camacho Montero**

Tutor: **Alfonso Jesús García Cerezo**

MÁLAGA, enero de 2023

Resumen: En este proyecto se ha abordado el diseño de un prototipo de péndulo invertido de bajo coste que sirva como banco de pruebas para futuras estrategias de control. Se ha diseñado además un sistema de control lineal del desplazamiento angular del péndulo con el objetivo de lograr su estabilización en torno a un punto de equilibrio. Este sistema ha sido implementado por medio del microcontrolador Arduino Mega 2560 y Arduino IDE.

Abstract: In this project, the design of a low-cost inverted pendulum prototype that serves as a test bench for future control strategies has been addressed. In addition, a linear control system for the angular displacement of the pendulum has been designed in order to achieve its reach around an equilibrium point. This system has been implemented through the Arduino Mega 2560 microcontroller and Arduino IDE.

Palabras clave: Péndulo invertido, prototipo, control, PD, todo-nada, microcontrolador, Arduino, Matlab, Simulink.

Índice

1. Introducción 9
2. Modelo matemático del péndulo invertido 11
2.1. Ecuaciones dinámicas 11
2.2. Función de transferencia 16
2.3. Espacio de estados 17
3. Diseño del prototipo del péndulo 19
3.1. Planteamiento del diseño 19
3.2. Estudio inicial de los componentes 24
3.2.1. Driver de potencia L298N 24
3.2.2. Placa de desarrollo Arduino Mega 2560 30
3.2.3. Motor DC con escobillas modelo IG220019X00015R, de *Digilent*: 35
3.2.4. Potenciómetro de precisión Serie 157 R5K, de Vishay 37
3.2.5. Interruptor final de carrera modelo 10T85 µ, de *CQC* 40
4. Diseño del sistema de control 42
4.1. Pasos previos para el diseño del sistema de control 42
4.1.1. Obtención de la función de transferencia y parámetros del motor cc 43
4.1.2. Fenómenos no lineales 48
4.1.3. Caracterización de la zona muerta del motor 49
4.1.4. Truncado 52
4.1.5. Calibrado del potenciómetro 54
4.2. Diseño del algoritmo de control 55
4.2.1. Modelo de control en bucle cerrado 57
4.2.2. Implementación del sistema de control. Código Arduino 57
5. Resultados 69
6. Conclusiones 76
7. Referencias 77
8. Anexos 79

Índice de figuras

Figura 1: Transbordador espacial Discovery [5] 10
Figura 2: Representación del problema físico del sistema péndulo invertido. (Extraído de [6]) 11
Figura 3: Diagrama de los cuerpos rígidos de carro y péndulo. (Adaptado de [6]).....13
Figura 4: Maqueta provisional.....19
Figura 5: Modelo 3D del soporte del potenciómetro 21
Figura 6: Alzado de la maqueta definitiva 22
Figura 7: Planta de la maqueta definitiva.....23
Figura 8: Aspecto del módulo L298N y terminales de conexión [12], [13]. 25
Figura 9: Esquemático del módulo L298N.....26
Figura 10: Modos de funcionamiento de un puente H de transistores 26
Figura 11: Esquemático del módulo L298 [14] 27
Figura 12: Modos de funcionamiento de un puente de transistores con control lógico. Figura adaptada de [15].....28
Figura 13: Placa Arduino Mega 2560 [23] 31
Figura 14: Esquemático de Arduino Mega 2560.....33
Figura 15: Aspecto de un modelo de motor similar al utilizado [25].....35
Figura 16: Aspecto del modelo de potenciómetro utilizado [27] 37
Figura 17: Esquemático de un potenciómetro [29].....38
Figura 18: Esquemático del potenciómetro 39
Figura 19: Aspecto de un modelo de final de carrera similar al utilizado [30].....40
Figura 20: Esquemático interno simplificado de un motor cc (Adaptado de [6]) 43
Figura 21: Representación simplificada de la respuesta de velocidad en función de la tensión suministrada en un motor cc.....50
Figura 22: Representación de truncado en una señal PWM.....52
Figura 23: Distribución angular en el potenciómetro tras el proceso de calibrado.....54
Figura 24: Curva de histéresis [32] 55
Figura 25: Vista general del modelo de control en bucle cerrado 57
Figura 26: Péndulo en equilibrio 69
Figura 27: Vista general del modelo de lectura de motor 88
Figura 28: Vista general de la sección de entrada del modelo de lectura de motor.....89
Figura 29: Vista general de la sección de salida del modelo de lectura de motor 89
Figura 30: Vista general del bloque motor 91

Índice de gráficas

Gráfica 1 : Respuesta en velocidad del motor ante una entrada escalón de tensión.....47
Gráfica 2: Respuesta en velocidad del motor ante una entrada rampa de tensión 51
Gráfica 3: Posición angular en el punto de equilibrio (rad) 70
Gráfica 4: Velocidad en el punto de equilibrio (rad/s) 70
Gráfica 5: Valores de tensión codificada comandada en el punto de equilibrio 71
Gráfica 6: Posición angular en el punto de equilibrio, tras un choque con un final de carrera y posterior recuperación, y tras un choque sin recuperación (rad).....71
Gráfica 7: Velocidad en el punto de equilibrio y tras el choque con un final de carrera y posterior recuperación (rad/s).....72
Gráfica 8: Valores de tensión comandada en el punto de equilibrio y tras el choque con un final de carrera sin recuperación 72
Gráfica 9: Posición angular en torno al punto de reposo (rad) , con choque con final de carrera 73
Gráfica 10: Velocidad en torno al punto de reposo (rad/s), con choque con final de carrera....74
Gráfica 11: Valores de tensión codificada comandada en torno al punto de reposo, con choque con final de carrera 74

Índice de códigos

Código 1: Programa Arduino 61
Código 2: Función lectura_encoders 98

1. Introducción

Históricamente, el péndulo invertido ha venido suscitando un importante interés para la ingeniería de control no lineal, tanto por el reto teórico que supone el diseño de un sistema de control para su modelo no lineal como por sus aplicaciones prácticas, siendo uno de sus problemas básicos y fundamentales; además, es usado habitualmente para probar el funcionamiento de estrategias de control experimentales [1]. El modelo más simple y extendido, que es el desarrollado en este trabajo, consiste en un péndulo que oscila sobre un eje situado en uno de sus extremos e instalado sobre una plataforma o carro que se desplaza de forma lineal gracias a la actuación de la fuerza de control. Con esta actuación se logra controlar la posición del cuerpo del péndulo, normalmente con el objetivo de mantener al mismo en posición invertida tras dejarlo caer manualmente desde dicha posición.

Profundizando un poco más en las aplicaciones prácticas del péndulo invertido, se puede destacar su utilidad en múltiples campos, como por ejemplo en la arquitectura, donde construcciones de grandes dimensiones como edificios son modeladas como péndulos invertidos. Sin embargo, en el campo que nos ocupa, la robótica, el péndulo invertido no se queda atrás: es usado para implementar sistemas de balanceo, en sistemas de apoyo para pacientes con movilidad reducida, en el transporte de objetos mediante drones, en robots bípedos, brazos robóticos o, su uso comercial más famoso, en el medio de transporte unipersonal *Segway* [1], [3]. También es útil en industria aeroespacial, ya que el control de un péndulo invertido es análogo al de un cohete tras su lanzamiento; en él, se pretende mantener el eje vertical balanceándose en torno a un punto de equilibrio inestable. [4] De igual forma, satélites, aviones de despegue vertical, proyectiles y otros dispositivos de la ingeniería aeroespacial poseen sistemas de control derivados del del péndulo invertido [3].

Figura 1: Transbordador espacial Discovery [5]

El objetivo de este proyecto es el diseño de una maqueta de péndulo invertido sobre plataforma móvil, de bajo coste, que sirva como equipo de prueba para la experimentación de diferentes estrategias de control.

2. Modelo matemático del péndulo invertido

En este capítulo se deducirá el sistema matemático que modela el comportamiento físico del péndulo invertido *(este capítulo ha sido extraído y adaptado de [6]).*

2.1. Ecuaciones dinámicas

El objetivo del siguiente apartado es obtener las ecuaciones que modelan el comportamiento no lineal del sistema del péndulo invertido. El péndulo invertido es un sistema inestable porque puede girar en cualquier momento y en cualquier dirección mientras no se le aplique una fuerza de control. Se considerará que la masa del péndulo está distribuida uniformemente a lo largo de la varilla, por lo que su centro de gravedad se encontrará en el centro de ésta. Además, no se considerará fricción.

Sea la representación del sistema:

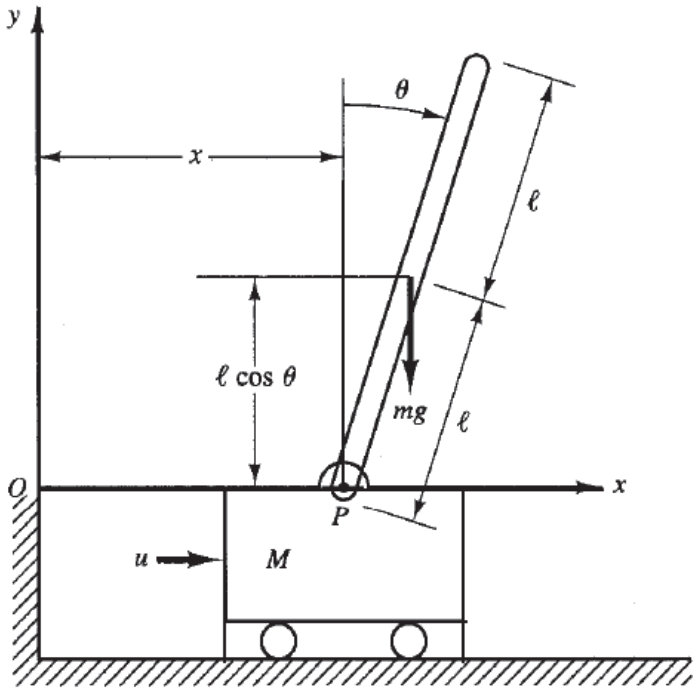


Figura 2: Representación del problema físico del sistema péndulo invertido. (Extraído de [6])

Donde las variables representadas son:

M: masa del carro o plataforma (157g)

m: masa de la varilla (22g)

l: longitud al centro de masa de la varilla desde su origen. (12,5 cm)

u: fuerza de actuación aplicada al carro o plataforma

x: posición horizontal del carro

ɵ: ángulo del péndulo respecto al eje vertical

g: aceleración de la gravedad

P: punto de pivote

El péndulo invertido se puede analizar como un cuerpo rígido con un movimiento limitado a dos dimensiones. Las ecuaciones fundamentales de movimiento plano de un cuerpo rígido son:

$$\sum F_i = ma_i \quad (1)$$

$$\sum F_j = ma_j \quad (2)$$

$$\sum F_G = I\alpha_g \quad (3)$$

Las dos primeras ecuaciones son la aplicación de la Segunda Ley de Newton para las componentes horizontales *i* y para las verticales *j*. La tercera ecuación proviene también de la Segunda Ley de Newton y es la suma de momentos producidos por las fuerzas actuantes sobre un punto G cualquiera, que es igual al producto del momento de inercia I por la aceleración angular α producida sobre el cuerpo rígido.

Partiendo de este sistema, es fácil deducir una expresión de las coordenadas del centro de gravedad (x_a, y_c) para el sistema de referencia representado:

$$x_G = x + l\,sen\,\theta \tag{4}$$

$$y_G = l\,cos\,\theta \tag{5}$$

Para simplificar el análisis del sistema, a partir de ahora se considerarán el sistema del carro y del propio péndulo como dos cuerpos rígidos independientes. De esta separación surgen dos nuevas componentes: la componente horizontal *H* y la vertical *V*, que actúan sobre el punto de pivote *P*:

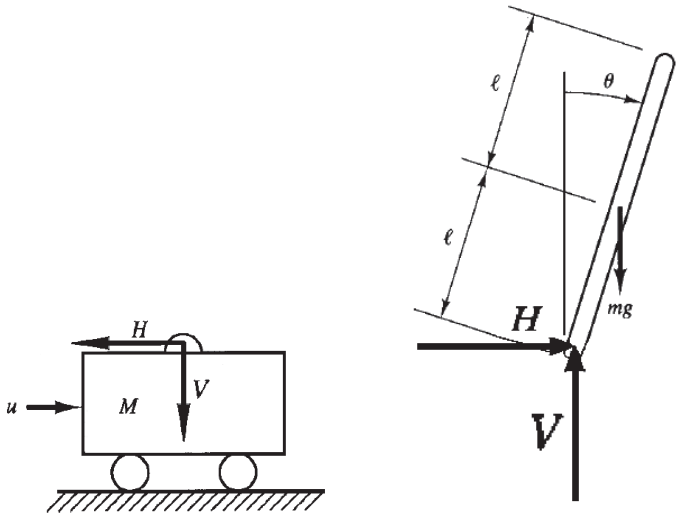


Figura 3: Diagrama de los cuerpos rígidos de carro y péndulo. (Adaptado de [6])

El movimiento rotacional del péndulo alrededor de su centro de gravedad se describe mediante:

$$I\ddot{\theta} = Vl\,sen\,\theta - Hl\,cos\,\theta \tag{6}$$

Donde I es el momento de inercia del péndulo alrededor de su centro de gravedad.

Si se aplican las ecuaciones 1.1 y 1.2, Segunda Ley de Newton para las componentes de fuerza horizontales y verticales que actúan sobre el péndulo, se obtiene:

$$\sum F_i = ma_i = m\ddot{x} = m\frac{d^2}{dt^2}(x + l\,sen\,\theta) = H \tag{7}$$

$$\sum F_j = ma_j = m\ddot{y} = m\frac{d^2}{dt^2}(l\,cos\,\theta) = V - mg \tag{8}$$

Del mismo modo, para las componentes que actúan sobre el carro:

$$\sum F_i = M\frac{d^2x}{dt^2} = u - H \tag{9}$$

$$\sum F_j = Mg = V \tag{10}$$

Estas cuatro ecuaciones, que describen el movimiento del sistema del péndulo invertido en el carro, no son lineales, puesto que contienen $sen\,\theta$ y $cos\,\theta$. Para linealizarlas, se supondrá un ángulo θ muy pequeño, de tal forma que se puede aproximar $\cos\theta \approx 1$ y $\text{sen}\,\theta \approx \theta$. Por lo tanto, las ecuaciones anteriores linealizadas serían:

$$I\ddot{\theta} = Vl\,\theta - Hl \tag{11}$$

$$m\left(\ddot{x} + l\ddot{\theta}\right) = H \tag{12}$$

$$0 = V - mg \tag{13}$$

$$M\ddot{x} = u - H \tag{14}$$

$$Mg = V \tag{15}$$

A partir de 12 y 14, se obtiene:

$$(M + m)\ddot{x} + ml\ddot{\theta} = u \tag{16}$$

A partir de 11 y 13 se obtiene:

$$I\ddot{\theta} = mgl\,\theta - Hl = mgl\,\theta - l(m\ddot{x} + ml\ddot{\theta}) \tag{17}$$

El momento de inercia del péndulo con respecto a su centro de gravedad es:

$$I = \frac{1}{12}m(2l)^2 = \frac{1}{3}ml^2 \quad (18)$$

La ecuación (17) puede expresarse de la forma:

$$(I + ml^2)\ddot{\theta} + ml\ddot{x} = mgl\theta \quad (19)$$

O bien, si se tiene en cuenta la ecuación (19), expresión de la inercia:

$$\frac{4}{3}ml^2\ddot{\theta} + ml\ddot{x} = mgl\theta \quad (20)$$

Las ecuaciones (16) y (20) representan el modelo matemático del sistema péndulo invertido. Estas ecuaciones pueden ser modificadas a:

$$\left(\frac{4}{3}M + \frac{1}{3}m\right)\ddot{\theta}l = (M + m)g\theta - u \quad (21)$$

$$(4M + m)\ddot{x} = 4u - 3mg\theta \quad (22)$$

La ecuación (21) se ha obtenido eliminando $\ddot{x}$ de las ecuaciones (16) y (20) y la ecuación (22) eliminando $\ddot{\theta}$ de las mismas ecuaciones.

2.2. Función de transferencia

La función de transferencia es un modelo matemático que describe el comportamiento de un sistema, analizando cómo es su salida ante diferentes entradas o excitaciones. [7]. Se representa como el cociente de la transformada de Laplace de la salida del sistema entre la transformada de Laplace de la entrada. Una función de transferencia presenta la forma:

$$H(s) = \frac{Y(s)}{U(s)} \tag{23}$$

Siendo $H(s)$ la función de transferencia, $Y(s)$ la transformada de Laplace de la salida y $U(s)$ la transformada de Laplace de la entrada.

Buscaremos a continuación una expresión de la función de transferencia del sistema del péndulo invertido a partir de su modelo matemático. Aplicando la transformada de Laplace a la ecuación (21), en la que aparecen tanto la salida θ como la entrada u del sistema, se obtiene:

$$\left(\frac{4}{3}M + \frac{1}{3}m\right) ls^2\theta(s) = (M+m)g\theta(s) - U(s) \tag{24}$$

Despejando, la función de transferencia de la planta es:

$$\frac{\theta(s)}{-U(s)} = \frac{1}{\left(\frac{4}{3}M+\frac{1}{3}m\right)ls^2-(M+m)g} \tag{25}$$

Expresando el denominador como producto de raíces, se obtiene:

$$\frac{\theta(s)}{-U(s)} = \frac{1}{\left(s-\frac{\sqrt{3}\sqrt{g(m+M)}}{\sqrt{l(m+4M)}}\right)\left(s+\frac{\sqrt{3}\sqrt{g(m+M)}}{\sqrt{l(m+4M)}}\right)} \tag{26}$$

Siendo los polos de esta planta:

$$\left(-\frac{\sqrt{3}\sqrt{g(m+M)}}{\sqrt{l(m+4M)}},\ \frac{\sqrt{3}\sqrt{g(m+M)}}{\sqrt{l(m+4M)}}\right) \tag{27}$$

Como presenta un polo en el eje real positivo y otro en el eje real negativo, la planta es inestable en lazo abierto.

2.3. Espacio de estados

El espacio de estados es un modelo de representación de un sistema a través de matrices y vectores que relaciona entre sí formando un espacio n-dimensional. Las variables de estado son aquellas que informan sobre el estado de un sistema, esto es, la información compactada de la actividad pasada que ha tenido el mismo, para así determinar futuras actuaciones [8].

Sean las variables de estado:

$$x_1 = \theta \tag{28}$$

$$x_2 = \dot{\theta} \tag{29}$$

$$x_3 = x \tag{30}$$

$$x_4 = \dot{x} \tag{31}$$

Donde θ indica el ángulo de rotación del péndulo con respecto al punto de pivote P y x representa la ubicación del carro, siendo consideradas las salidas del sistema:

$$x = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \theta \\ x \end{bmatrix} = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \tag{32}$$

A partir de la definición de variables de estado y de (21) y (22), se obtiene:

$$\dot{x}_1 = x_2 \tag{33}$$

$$\dot{x}_2 = \frac{M+m}{\left(\frac{4}{3}M+\frac{1}{3}m\right)l} g x_1 - \frac{1}{\left(\frac{4}{3}M+\frac{1}{3}m\right)l} u \tag{34}$$

$$\dot{x}_3 = x_4 \tag{35}$$

$$\dot{x}_4 = \frac{-3m}{4M+m} g x_1 + \frac{4}{4M+m} u \tag{36}$$

En términos de las ecuaciones vectoriales, tenemos:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M+m}{\left(\frac{4}{3}M+\frac{1}{3}m\right)l} g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-3m}{4M+m} g & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{\left(\frac{4}{3}M+\frac{1}{3}m\right)l} \\ 0 \\ \frac{4}{4M+m} \end{bmatrix} u \tag{37}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{38}$$

Estas ecuaciones describen el espacio de estados del péndulo invertido.

3. Diseño del prototipo del péndulo

Este apartado se dedicará a comentar el diseño y montaje del prototipo propuesto y a la descripción y análisis de cada uno de los dispositivos utilizados.

3.1. Planteamiento del diseño

Sin duda ha supuesto una de la partes más complejas y extensas de este proyecto. Se han realizado dos maquetas, una provisional para la realización de pruebas y una definitiva.

- Primer diseño y maqueta provisional

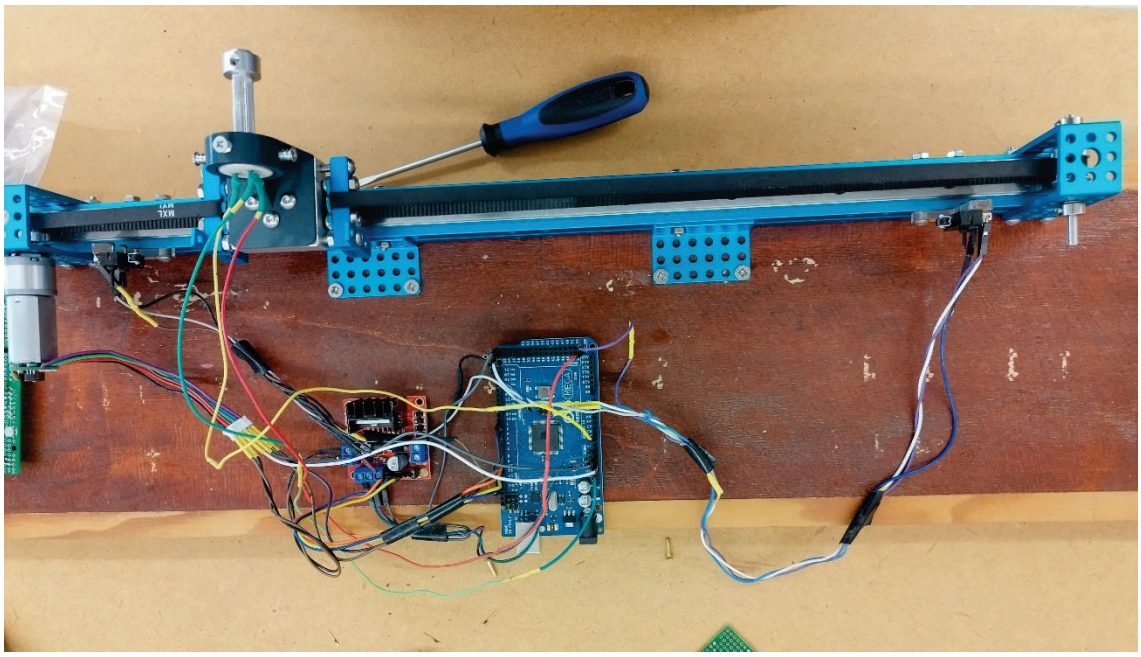


Figura 4: Maqueta provisional

La primera idea fue utilizar la tarjeta *MakeBlock Orion*, de *MakeBlock*, empresa dedicada a la divulgación sencilla de la robótica con fines educativos y programarla a través del IDE de Arduino. Sin embargo, esta placa trabaja mediante comunicación I2C con los elementos periféricos, por lo que la única posibilidad era trabajar con funciones predefinidas de la librería específica de *MakeBlock* para Arduino, de muy alto nivel y que

no resultaron ejercer un buen desempeño a la frecuencia de muestreo deseada. Por esta razón, finalmente se optó por trabajar con la tarjeta Arduino Mega 2560.

La utilización del modelo de controladora de motor L298N para la etapa de potencia estuvo clara desde un principio, ya que se había trabajado previamente con él en otros proyectos. Este modelo se analizará más adelante.

En cuanto al motor, se probaron distintos modelos, hasta que se dio con uno capaz de entregar un par capaz de ejercer una aceleración suficiente en el carro; también se estudiaron distintos potenciómetros hasta elegir el modelo definitivo, con un coeficiente de rozamiento mínimo y una zona muerta aceptable.

Sobre el montaje, el punto de partida fue una estructura de *MakeBlock*. Se trata de un perfil cuadrado de aluminio de 50.4 cm de longitud a la que se atornilló un riel de acero sobre el que se dispone la plataforma o carro. A ello se añadieron sendos soportes para motores en sus extremos, aguantando uno el eje del motor y el otro un eje pasivo, cada uno con un piñón solidario con una cinta unida a la plataforma. Se atornillaron los finales de carrera a la estructura y dos soportes de aluminio tipo P1 a cada lado de la plataforma, para provocar la pulsación de los finales de carrera al llegar la plataforma al límite de su recorrido. Para las conexiones, se utilizaron cables Dupont que se soldaron al potenciómetro y a los finales de carrera, manteniendo los terminales de conexión a las placas sin soldar.

La base una tabla de madera de 62x17.5 cm a la que se fijó la estructura con dos pestañas de aluminio que se atornillaron directamente a base y a la estructura. Se atornillaron además las placas Arduino y L298N a la base de madera.

Uno de los principales problemas a solucionar fue encontrar una pieza capaz de fijar firmemente el potenciómetro a la plataforma. Se probaron varias ideas, pero se llegó a la conclusión de que la mejor manera de conseguirlo era diseñar una pieza en tres dimensiones con las medidas oportunas e imprimirla vía impresora 3d de plástico por inyección.

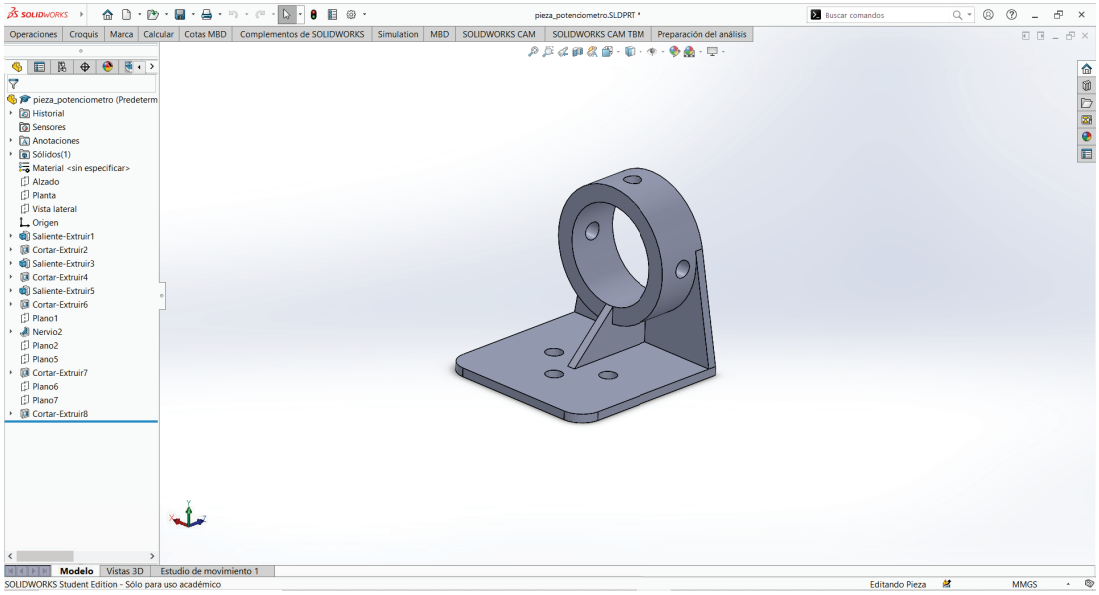


Figura 5: Modelo 3D del soporte del potenciómetro

Las vistas de la pieza se encuentran en *[Anexo I]*.

- Segundo diseño y maqueta definitiva

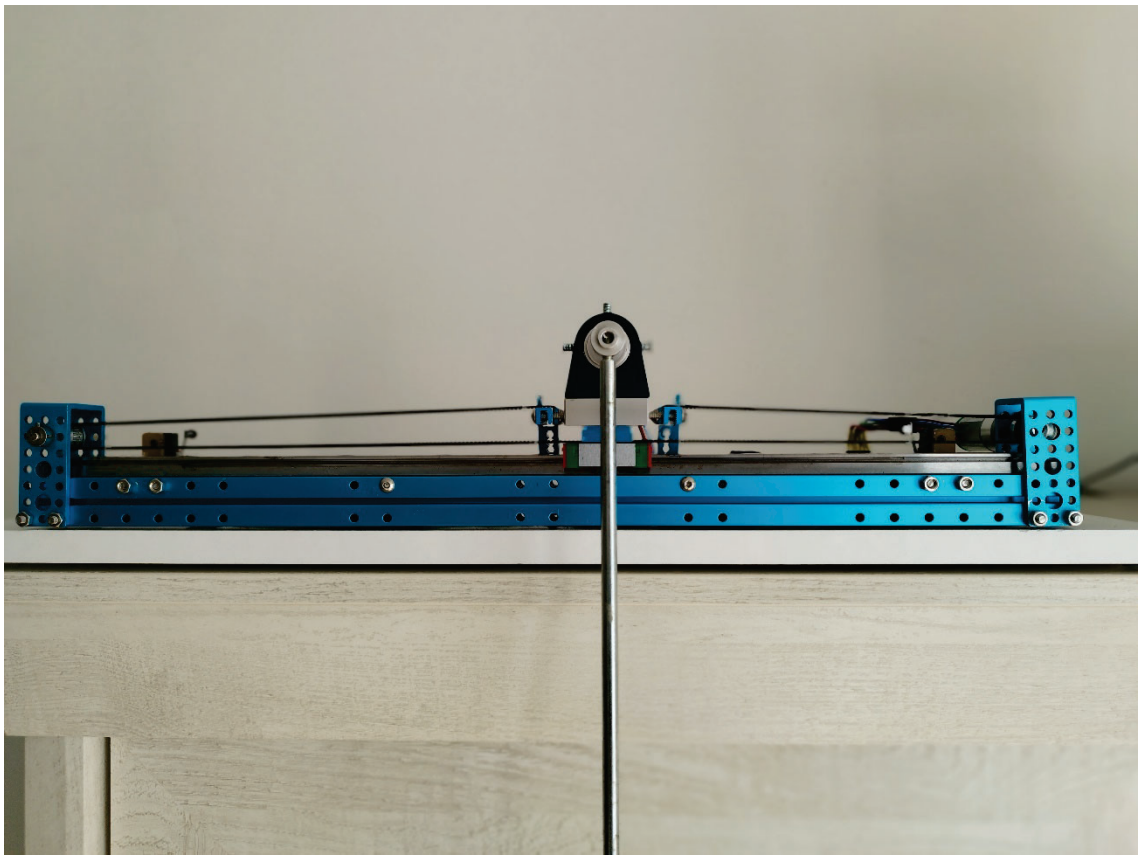


Figura 6: Alzado de la maqueta definitiva

Para una mejor presentación, se traspasó la plataforma a una tabla de aglomerado revestida de láminas de melamina blanca, de 60x29.5 cm.

Buscando una mayor firmeza en las conexiones, se decidió adquirir una placa perforada Arduino Uno para poder conectar los dispositivos de forma permanente mediante soldadura. La placa consta de una serie de perforaciones, llamadas pistas, revestidas con láminas de un metal conductor a las que se pueden soldar terminales de cables. Además, este tipo de extensiones traen consigo líneas de terminales macho que han de ser soldados a mano y que permiten un perfecto encaje con los terminales hembra del modelo de placa para la que están diseñadas, en este caso, Arduino Mega 2560.

Cada pin de conexión de la placa del microcontrolador tiene su correspondencia con una pista de la placa perforada, todas situadas de forma perimetral. En algunos casos, como en el de las alimentaciones o las referencias a tierra, fue necesario conectar varias pistas

de una misma fila entre sí para poder disponer de más conexiones de igual potencial; esto se consiguió trenzando un hilo de cobre entre las pistas.

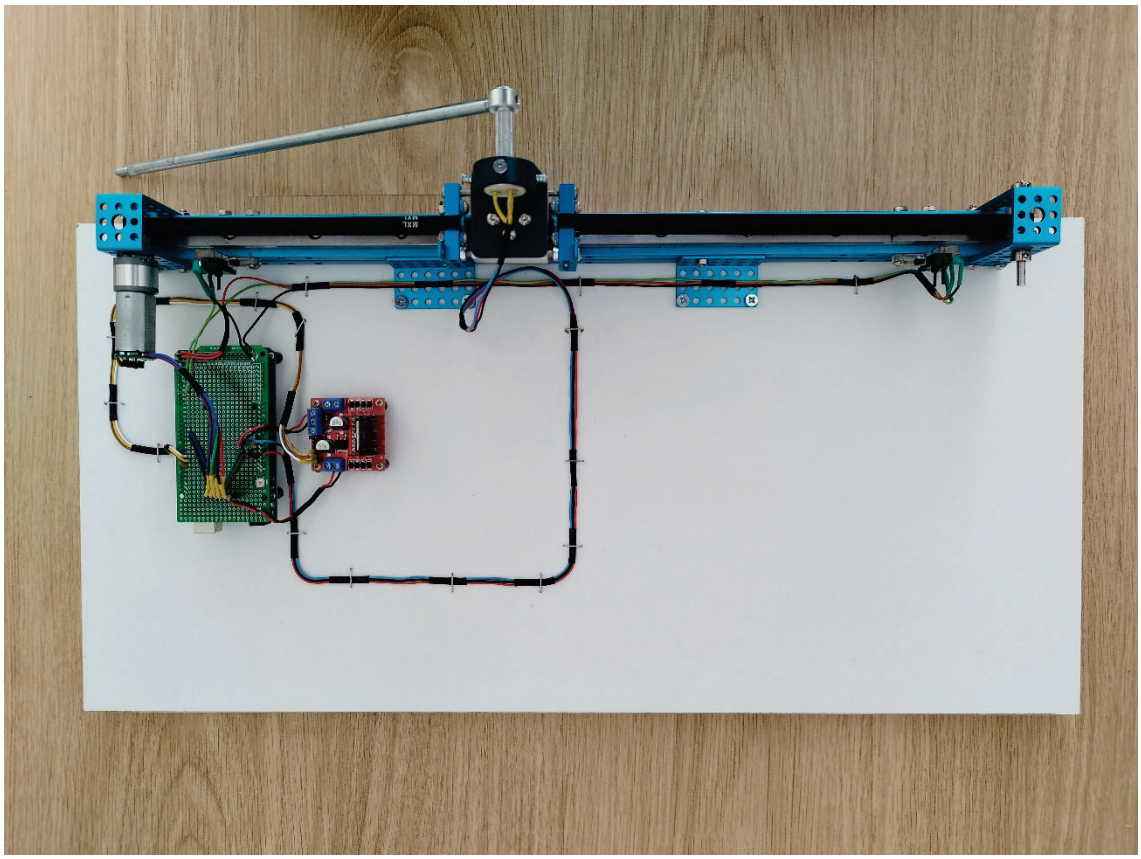


Figura 7: Planta de la maqueta definitiva

Mientras que el cableado del modelo provisional no respondía a ningún código, para este modelo se respetó un código de colores, que se puede consultar en *[Anexo II]*. Además, se evitó realizar empalmes entre cables, por lo que se sustituyeron los cables Dupont, de menor longitud, por cables de teléfono y se graparon a la base de madera.

Las placas, al igual que en modelo anterior, también han sido atornilladas a la base, pero en este caso se han usado separadores de plástico entre ellas y la base. Esto favorece la circulación de aire y evita el sobrecalentamiento de los dispositivos.

El alzado y la planta de la maqueta se encuentran en *[Anexo III], [Anexo IV]*.

3.2. Estudio inicial de los componentes

A continuación se analizarán con profundidad los módulos utilizados en el prototipo, realizando un análisis electrónico cuando se ha considerado oportuno. Estos análisis electrónicos se estructuran en base a unos diseños esquemáticos que han sido realizados en el programa *Orcad Capture CIS* y cuyo interconexionado puede consultarse en *[Anexo V]*.

3.2.1. Driver de potencia L298N

Se trata del controlador del motor, su función es transformar señales de control de baja intensidad de corriente y transformarlas en señales de alta intensidad que puedan alimentar un motor; este dispositivo es capaz de controlar dos motores DC o un motor paso a paso. El circuito propiamente dicho, el L298N, consiste en dos puentes H de transistores, para la salida A y para la salida B. A derecha e izquierda consta de un par de clemas con las salidas OUT1 y OUT2 para el motor A y OUT3 y OUT4 para el motor B, suministrando a cada motor A o B los comandos de velocidad necesarios. Una vez conectadas las salidas al motor, dependiendo de si el comando de velocidad llega desde la primera o segunda salida se puede controlar además el sentido de giro.

En la parte inferior se sitúa una clema con tres terminales: a la izquierda está la entrada de tensión de alimentación del motor, siendo admisible una tensión comprendida entre 3 y 35V, situándose a su derecha la referencia a tierra. Existe un tercer terminal conectado al circuito, *Vlógico*, que es controlado por un regulador de tensión, activo mediante un jumper. Si la tensión de alimentación es menor que 5V o mayor que 15V, es necesario retirar el jumper y suministrar al terminal 5V de tensión. En caso contrario, se puede utilizar dicho terminal para alimentar otros dispositivos a 5V.

Junto a la clema inferior se sitúan los pines de entrada *ENA,* de activación de la salida A y *ENB*, de activación de la salida B; estas entradas pueden ser activadas o desactivadas mediante jumper. Por otro lado, están *IN1* e *IN2*, que controlan el sentido de giro y

velocidad del motor A, suministrada al mismo por la salida A, e *IN3* e *IN4*, ídem para el motor B. [9], [10], [11].

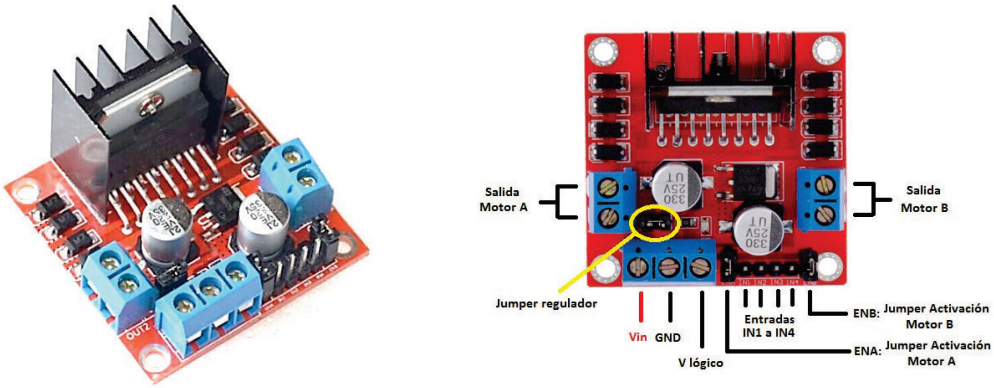


Figura 8: Aspecto del módulo L298N y terminales de conexión [12], [13].

La hoja de datos del módulo se encuentra en *[Anexo VI]*.

- **Análisis electrónico del módulo driver de potencia L289N**

El módulo driver de potencia L298N está compuesto por un doble puente H de transistores NPN, que en Orcad Capture CIS se encuentran dentro del bloque L298/MUTIV, de la librería *Design Cache* y que ha sido modificado para añadir la referencia a tierra.

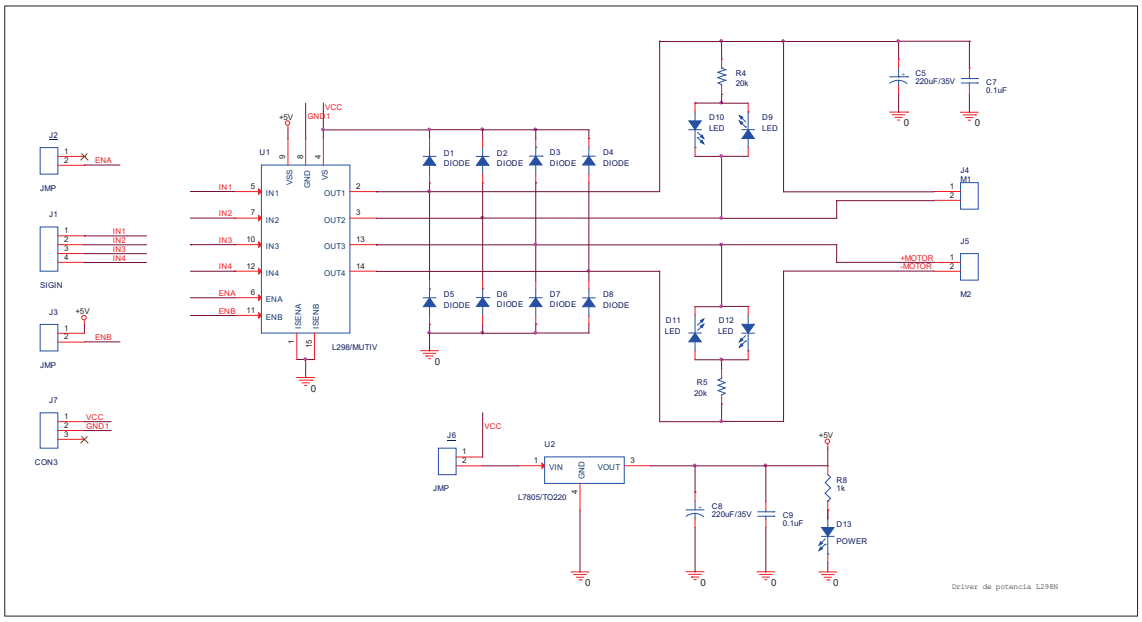


Figura 9: Esquemático del módulo L298N

Un puente H de transistores es un circuito usado de manera habitual en el control de motores de corriente continua. Consiste en dos pares de transistores en paralelo funcionando como interruptores que se activan o desactivan simultáneamente de manera diagonal dos a dos. Un esquema simplificado se puede observar en la siguiente figura:

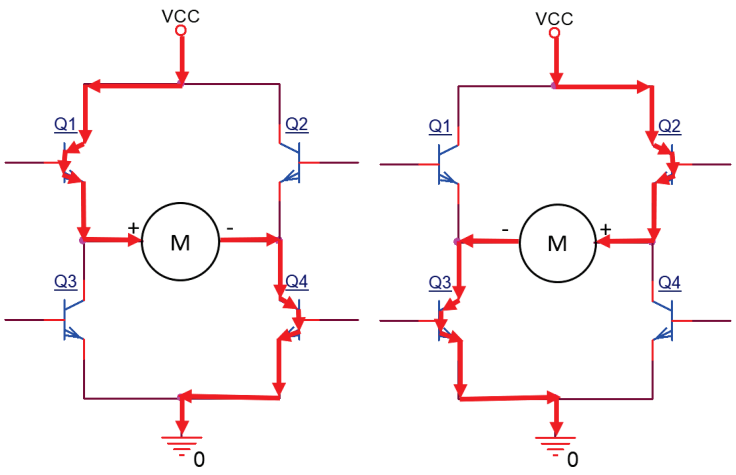


Figura 10: Modos de funcionamiento de un puente H de transistores

La figura representa los dos modos de funcionamiento que posee el diseño. En el primer modo, el par de transistores Q1 y Q4 se encuentran activos y Q2 y Q3 inactivos, por lo

que el motor queda polarizado en un sentido y la corriente circula por el camino del circuito asociada a estos dos transistores. Cuando la pareja de transistores activa cambia a Q2 y Q3, el motor queda polarizado en sentido inverso al del primer modo de funcionamiento y la corriente circula por el otro camino del circuito. Con esto se consigue el control del sentido de giro del motor de corriente continua.

Uno de los problemas a evitar en los puentes H es la activación simultánea de dos transistores pertenecientes a una misma rama; esto provocaría el paso de la corriente sin existir resistencia alguna más allá de la de los propios transistores, lo que causaría daños en el circuito. Para ello, los puentes H comerciales incorporan una lógica de control a base de puertas lógicas que garantiza la no activación simultánea de transistores consecutivos, tal y como se aprecia en el propio bloque L298:

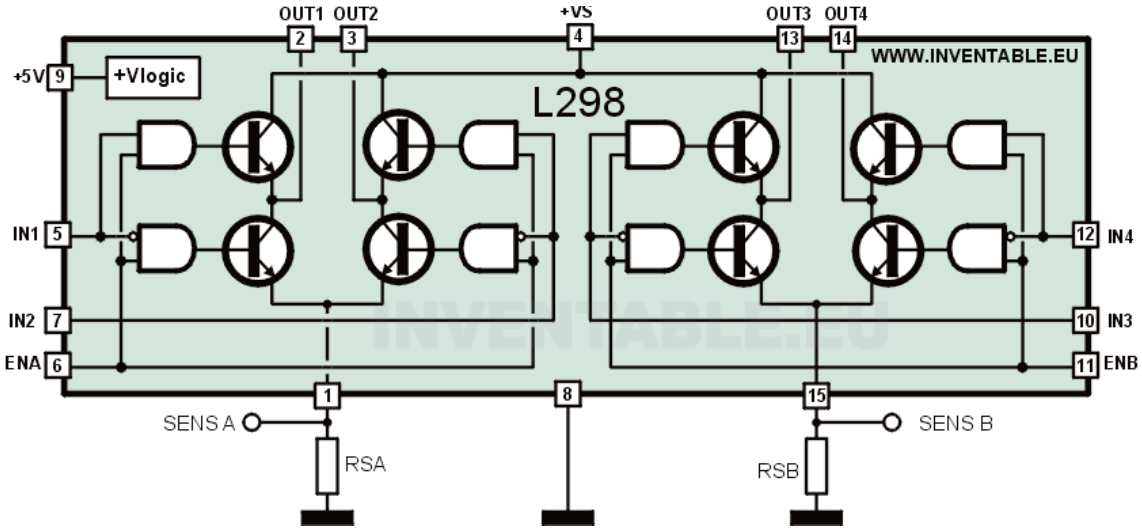


Figura 11: Esquemático del módulo L298 [14]

En el esquemático de este bloque aparecen ambos puentes H asociados a cada motor. Para explicar su funcionamiento primero es necesario definir la naturaleza de las señales PWM y el objetivo de su uso en este módulo.

Una señal PWM (Pulse Width Modulation) no es más que un tren de pulsos cuyo ancho de pulso puede ser modificado con el fin de controlar el Duty Cycle o tiempo de trabajo

de la señal; esto es, la relación que existe entre el tiempo en el que la señal se encuentra en estado activo y el periodo de esta. Se trata de una forma de simular señales analógicas a partir de señales digitales en automatismos como Arduino, que no son capaces de generar señales analógicas propiamente dichas, sino que trabajan con dos únicos valores de tensión Vcc+ o Vcc- a los que asignan el estado activo o inactivo de una señal digital. De la modulación en ancho de pulso se extrae un valor de tensión medio entre Vcc+ y Vcc- dependiendo del Duty Cycle a través de la siguiente relación:

$$V_{medio} = (V_{cc+} - V_{cc-}) \frac{DutyCycle}{100}$$

Expresado el Duty Cycle en porcentaje. Esto permite trabajar de forma analógica con valores de tensión comprendidos en el rango [Vcc-, Vcc+].

Es necesario recordar que cada motor tiene asociadas dos señales PWM de entrada que como ya se ha explicado, controlando cada una de ellas la velocidad del motor en un sentido de giro concreto. Por ello, no se puede producir la activación simultánea de ambas. Sabiendo esto, es fácil interpretar el funcionamiento del control lógico de un puente H:

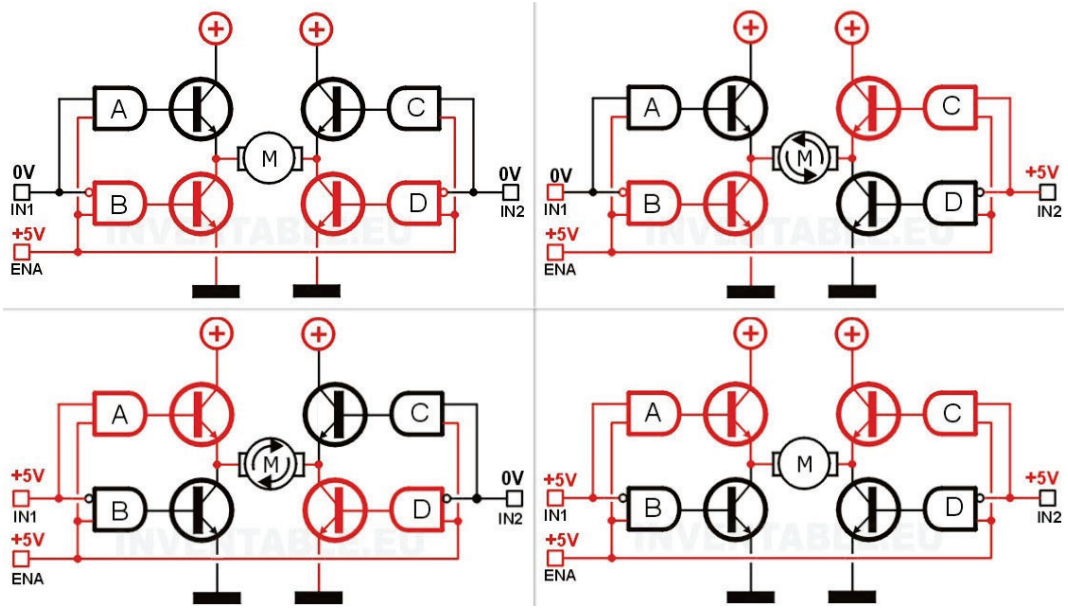


Figura 12: Modos de funcionamiento de un puente de transistores con control lógico. Figura adaptada de [15]

Las entradas al puente son la señal PWM de control de velocidad sentido horario IN1, la señal PWM de control de velocidad sentido antihorario IN2 y la señal de enable o habilitación del motor ENA, que siempre va a estar en estado activo.

Como se ve, cada transistor tiene conectada a su base una puerta AND; cada una de ellas tiene a su vez como entradas la señal de habilitación ENA y, dependiendo del semipuente, las señales de control de velocidad IN1 o IN2. Las entradas asociadas a IN1 e IN2 de las puertas inferiores del puente están negadas.

En el primer caso, la señal de habilitación ENA toma un valor de tensión activo, pero las señales de control de velocidad toman un valor de tensión medio de 0V. Las puertas AND inferiores están activas al estar sus entradas activas (ENA en estado activo e IN1 e IN2 con valor lógico inactivo pero negadas). Los dos transistores inferiores están activos cerrando el circuito, pero la intensidad no circula al no haber conexión con la alimentación y el motor no se mueve.

En el segundo caso, con ENA en estado activo e IN2 con un valor de tensión medio igual o superior a 5V (la entrada a Vlógico, VSS es +5V), lo que se traduce en un valor lógico activo. La puerta AND con entradas IN2 y ENA (superior derecha) está en estado activo, activando su transistor. Sin embargo, el transistor de la misma rama permanece inactivo, ya que se puerta AND con entradas ENA e IN2 negada toma el valor lógico inactivo. Por el contrario, la puerta AND con entrada negada asociada a la entrada IN1 (inferior izquierda) está activada y por ello también su transistor, puesto que IN1 toma el valor lógico inactivo y ENA está en activo. La puerta AND asociada a IN1 y ENA sin la entrada IN1 negada permanece inactiva. De forma análoga se deduce el funcionamiento en el tercer caso.

En el cuarto caso la entrada ENA continúa en estado activo y también lo están las señales de control de velocidad IN1 e IN2. Las puertas AND con entradas IN1 e IN2 no negadas son las únicas que permanecen activas (puertas superiores). La corriente no fluye ya que no hay conexión con la referencia a tierra y el motor no se mueve.

Para terminar la explicación de este bloque, cabe mencionar la existencia de dos terminales, ISENA e ISENB que permiten medir la corriente circulante en cada uno de los

puentes. Para ello, sería necesario añadir sendas resistencias de muy bajo valor, llamadas *resistencias de Shunt*, entre la conexión a tierra y los terminales. En este caso, los terminales están en desuso y han sido conectados directamente a tierra [16].

Continuando con el módulo L298N, para resolver el problema de la generación de tensiones inducidas en las entradas a los motores y por consiguiente un posible daño de los puentes de transistores, existe un sistema de seguridad formado por diodos rectificadores, cuatro por motor. En un modo de funcionamiento normal, la tensión en el ánodo es menor que la del cátodo, el diodo está polarizado en forma inversa y no conduce. Si aparece un elevado potencial en los terminales de entrada al motor, éste puede superar el valor de tensión de la alimentación o ser de valor negativo, polarizando el diodo de forma directa de tal forma que la corriente se disipe bien a la alimentación o bien a tierra, respectivamente.

Cada línea de entrada a los motores tiene asociado un diodo led, cuyo encendido informa del sentido de giro de cada uno. Además, para cada par de entradas a cada motor existe un filtro de condensadores para evitar picos de tensión producidos por el inducido [17].

Por último, el diseño consta de un regulador de tensión L7805 que provee a los jumpers de habilitación de los motores de una alimentación de +5V.

3.2.2. Placa de desarrollo Arduino Mega 2560

Se trata de una placa de desarrollo de la compañía Arduino, desarrolladora y comercializadora de dispositivos de hardware libre y suministradora de software libre para programación de sistemas electrónicos. Arduino es responsable de la creación de una comunidad internacional de desarrolladores que utilizan y modifican tanto su lenguaje de programación como las propias placas de desarrollo, suponiendo el germen de miles de proyectos, tanto educativos como científicos. Arduino ofrece un *IDE* o entorno de desarrollo que permite la programación en su propio código abierto de los

distintos microcontroladores, todo mediante comunicación mediante bus serie USB con el ordenador. [18], [19], [20]. El software de Arduino está compuesto por este entorno de desarrollo y por el cargador de arranque o *bootloader*, un pequeño programa que se ejecuta en el microcontrolador cada vez que se inicializa y que permite borrar y escribir o programar datos en la memoria del microcontrolador. [21]

Una de estas placas de desarrollo es *Arduino Mega 2560*, basada en el microcontrolador *ATmega2560*. Consta de 54 entradas/salidas digitales (15 destinadas a salidas PWM), 16 entradas analógicas y pines de alimentación *GND* (tierra), *5V*, *3V3* (3,3V) y *Vin*, cuya tensión es la suministrada por el conector Jack de la conexión serie USB; cuenta además con 4 puertos serie y puerto de comunicación mediante protocolo SPI y 2 puertos, SDA y SCL, para comunicación I2C o TWI [22].

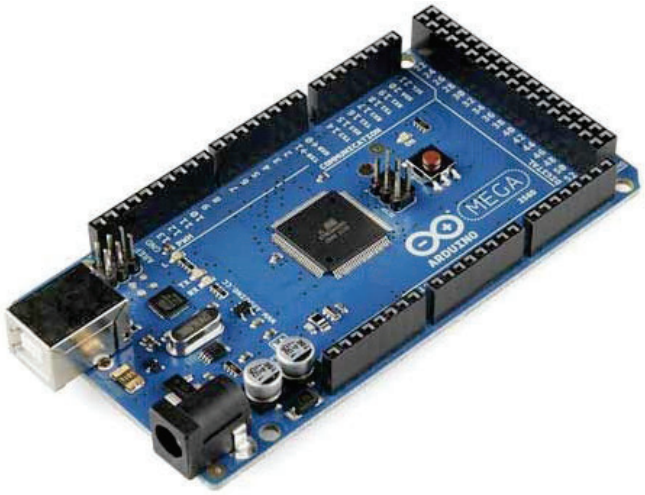


Figura 13: Placa Arduino Mega 2560 [23]

La placa de desarrollo Arduino Mega 2560 puede considerarse sucesora de la tradicional Arduino UNO, presentando diversas ventajas que la convierten en una alternativa muy interesante:

- Contiene 4 puertos de comunicación serie frente a un único puerto del modelo Arduino UNO.
- Memoria flash más potente (256 KB frente a 32KB). La memoria flash se encarga del almacenamiento del código generado en el IDE de Arduino, además de

contener el *bootloader*, que se ejecuta en el microcontrolador al finalizar el proceso de arranque de la placa.

- Memoria SRAM más potente (8KB frente a 2KB). Equivalente a la memoria RAM de un ordenador, la memoria SRAM se encarga de crear las variables declaradas en el programa.
- Memoria EEPROM más potente (4KB frente a 1KB). Memoria destinada al almacenaje de información que ha de conservarse, aunque la placa no esté conectada a la alimentación.
- Mayor número de entradas y salidas. Cabe destacar el significativo aumento de las entradas y salidas digitales (54 frente a 14) [24].

La hoja de datos del módulo se encuentra en *[Anexo VII]*.

- **Conexionado del módulo Arduino Mega 2560**

El análisis electrónico del microcontrolador Arduino Mega 2560 se encuentra al margen del alcance de este proyecto. Sin embargo, es importante indicar cuál es el conexionado de los componentes con la placa Arduino, que queda aclarado por su figura en el esquemático general:

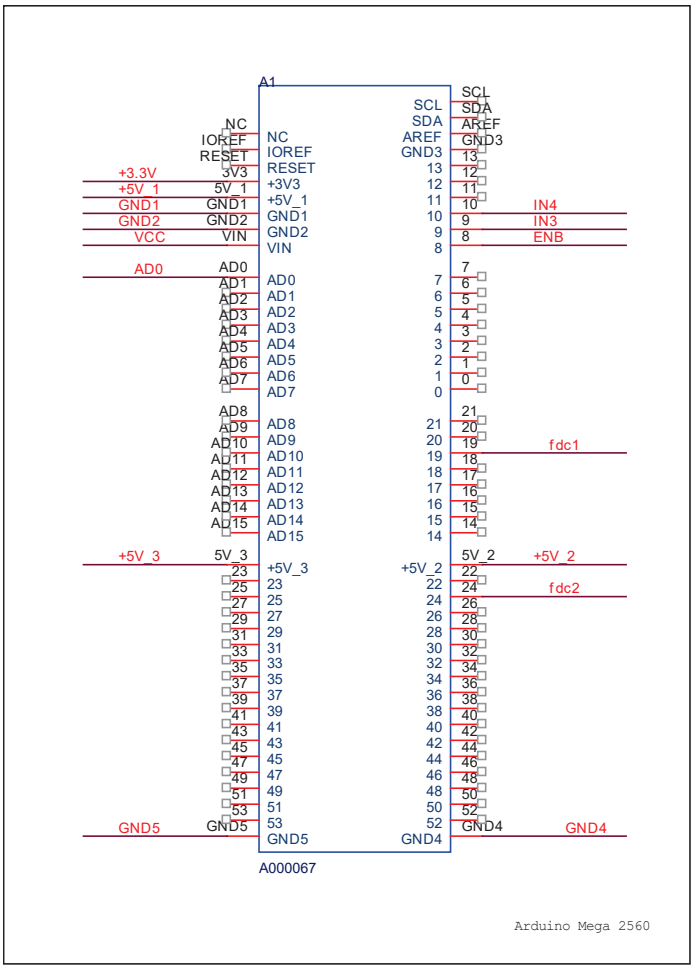


Figura 14: Esquemático de Arduino Mega 2560

Las entradas y salidas utilizadas son:

- Terminales de alimentación.
  - +3V3. Terminal de alimentación de +3.3V. Conexión con el terminal CW del potenciómetro, para establecimiento del sentido horario como positivo.
  - +5V_1. Primer terminal de alimentación de +5V. Conexión con el terminal de alimentación del encoder de efecto Hall.
  - +5V_2. Segundo terminal de alimentación de +5V. Conexión con el terminal NO (normalmente abierto) del final de carrera 1 (izquierdo).
  - +5V_3. Tercer terminal de alimentación de +5V. Conexión con el terminal NO (normalmente abierto) del final de carrera 2 (derecho).

  - VIN. Terminal de tensión el valor de alimentación de la placa. Conexión con VS en el módulo L298, valor de tensión de referencia de los puentes de transistores
  - GND1. Primer terminal de conexión a tierra. Conexión con la referencia a tierra del encoder de efecto Hall, con el terminal CCW del potenciómetro y con la referencia a tierra del módulo L298 del driver de potencia.
  - GND2. Segundo terminal de conexión a tierra. Conexión con borne de la resistencia de 1M de eliminación de valores espurios del montaje del potenciómetro.
  - GND4. Cuarto terminal de conexión a tierra. Conexión con el terminal NC (normalmente cerrado) del final de carrera 2 (derecho).
  - GND5. Quinto terminal de conexión a tierra. Conexión con el terminal NC (normalmente cerrado) del final de carrera 1 (izquierdo).
- Entradas analógicas.
  - AD0. Conexión con terminal *S* (salida) del potenciómetro. Entrada a Arduino del valor de tensión de salida del potenciómetro.
- Entradas y salidas digitales.
  - 2. Entrada digital. Conexión con el terminal *HALL_SENSOR_A_Vout* del encoder de efecto Hall. Entrada a Arduino del tren de pulsos generado por el sensor de efecto Hall A.
  - 3. Entrada digital. Conexión con el terminal *HALL_SENSOR_B_Vout* del encoder de efecto Hall. Entrada a Arduino del tren de pulsos generado por el sensor de efecto Hall B.
  - 8. Salida digital para señal de habilitación. Conexión con el terminal *ENB* de habilitación del motor B del driver de potencia.
  - 9. Salida digital en forma de señal PWM. Conexión con la entrada IN3 del driver de potencia para la señal de control de velocidad sentido horario del motor B.
  - 10. Salida digital en forma de señal PWM. Conexión con la entrada IN4 del driver de potencia para la señal de control de velocidad sentido antihorario del motor B.

- 19. Entrada digital. Conexión con COM (común) del final de carrera 1 (izquierdo). Entrada del valor lógico de salida del final de carrera 1.
- 24. Entrada digital. Conexión con COM (común) del final de carrera 2 (izquierdo). Entrada del valor lógico de salida del final de carrera 2.

3.2.3. Motor DC con escobillas modelo IG220019X00015R, de *Digilent*:

Se trata de un motor de corriente continua con escobillas de la compañía *Digilent*. Acepta una tensión de alimentación máxima de 12V, con velocidad en vacío de 789 rpm 15% y una velocidad nominal de 639 rpm 15%. El par nominal es de 0,3 Kgf x cm y el cociente de reducción es 1:19,225. La constante de par es 0,91 Kgf x cm/A y la constante de tensión es 8,68 mV/r/min.

Figura 15: Aspecto de un modelo de motor similar al utilizado [25]

El motor posee un encoder de dos canales con conector JST PHR-6 con 6 terminales. Los encoder de dos canales no solo permiten conocer la velocidad de giro del eje del motor, sino también su sentido, ya sea horario o antihorario, mediante la superposición de las señales PWM de salida de cada canal con cierto desfase. En este caso, el encoder posee dos sensores de efecto Hall para este propósito con una separación de 90º. Unido a su

eje, existe un imán de ferrita que al girar, produce variaciones en el campo magnético que atraviesa los sensores de efecto Hall. Así, las señales PWM o trenes de pulsos emitidos por cada sensor de efecto Hall son causados por un proceso de inducción magnética, que provoca una tensión de salida cuando se producen estas variaciones [26].

Los terminales de conexión son:

- *-MOTOR*: entrada del comando de velocidad del motor para giro en sentido horario.
- *+MOTOR*: entrada del comando de velocidad del motor para giro en sentido antihorario.
- *HALL SENSOR Vcc*: Terminal de alimentación de los sensores; la salida 3V3 de la Arduino Mega 2560, de 3,3V, es suficiente.
- *HALL SENSOR GND*: Terminal de conexión a tierra.
- *HALL SENSOR A out*: Terminal de salida de la señal PWM producida por el sensor A.
- *HALL SENSOR B out*: Terminal de salida de la señal PWM producida por el sensor B.

La hoja de datos del módulo se encuentra en *[Anexo VIII]*.

- **Codificación del motor**

Como se ha comentado previamente, el driver de potencia L298N acepta señales PWM como entradas que determinan el valor de tensión suministrado al motor, comprendido entre 0 V y +Vin V. Cada motor consta de dos entradas para señales PWM al driver de potencia, que controlan el sentido de giro del motor: para ello, solo una de las entradas recibirá una señal PWM de tensión media no nula en cada periodo; dependiendo de cuál de ellas sea, la tensión suministrada tomará una u otra polaridad y el Duty Cycle de la señal determinará su valor.

Las señales PWM se pueden codificar en binario con una palabra de 8 bits (codificación de los números naturales del 0 al 255), por lo que bin(0) da lugar a un Duty Cycle del 0% y bin(255) a uno del 100%. Teniendo en cuenta que la tensión suministrada al driver de potencia es de 12V, la conversión de valor codificado a tensión de alimentación es:

$$e = \frac{12}{255}u$$

Siendo *e* la tensión de alimentación al motor en voltios y *u* la palabra de codificación de Duty Cycle en formato decimal.

Para determinar la dirección de entrada de la señal PWM, se añadirá a la palabra de 8 bits un bit más que determinará su signo. De esta manera, el signo de la palabra establecerá la entrada a la que va dirigida la señal PWM y por tanto el sentido de giro del motor.

Se ha diseñado un modelo en Simulink para la lectura del motor para la realización de pruebas y puede consultarse en *[Anexo IX], [Anexo X]* y *[Anexo XI]*.

3.2.4. Potenciómetro de precisión Serie 157 R5K, de Vishay

Figura 16: Aspecto del modelo de potenciómetro utilizado [27]

Sensor potenciómetro de una única vuelta con resistencia variable de 5K (tolerancia del 20%) y cociente de linealidad 0,5%, esto es, la relación que existe entre la tensión de

salida del sensor y la rotación mecánica del eje [28]. Su potencia nominal es de 1W. Se trata de un potenciómetro con un coeficiente de rozamiento mínimo, pensado para uso en la industria aeronáutica. Posee además un recorrido angular útil de 340±4º, por lo que su zona muerta es de 20±4º.

Un potenciómetro está constituido por una resistencia variable, cuyo valor es modificable manualmente a través de un eje de giro. La salida del potenciómetro será el valor de tensión de caída en la resistencia variable y dependerá de la posición angular y número de vueltas del eje de giro.

Dependiendo de qué sentido toma la caída de tensión en la resistencia variable, el valor de la resistencia a la salida del potenciómetro aumentará para el giro horario o antihorario. Una alimentación de 3,3 V ha demostrado ser suficiente.

Posee tres terminales:

- Terminal 1 ó CW: *clockwise*, o sentido horario. Si se conecta este terminal a Vcc y CCW a tierra, el valor de tensión a la salida del potenciómetro aumentará para el giro horario.
- Terminal 2 ó S: terminal de salida del potenciómetro. A través de él se obtiene la señal analógica de salida del potenciómetro con el valor de tensión de caída.
- Terminal 3 ó CCW: *counterclockwise*, o sentido antihorario. Si se conecta este terminal a Vcc y CW a tierra, el valor de tensión a la salida del potenciómetro aumentará para el giro antihorario.

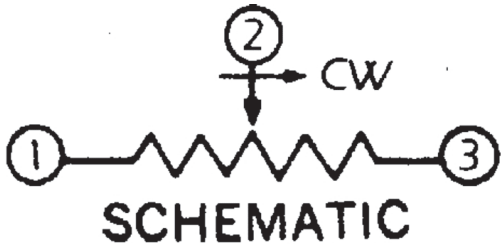


Figura 17: Esquemático de un potenciómetro [29]

El recorrido angular útil de este modelo es 340º, por lo que existe un ángulo de 20º en el que el potenciómetro marcará a su salida valores de tensión espurios (zona muerta). La solución a esto se comentará más adelante.

Su objetivo en este proyecto será medir la posición angular del péndulo invertido.

La hoja de datos del módulo se encuentra en *[Anexo XII]*.

- **Análisis electrónico del potenciómetro**

El potenciómetro se ha modelado como dos resistencias variables de cuyo valor depende el valor de tensión a la salida. Se ha conectado a la salida una resistencia de 1M para forzar la salida en la zona muerta a 0V; de esta forma, se evita la aparición de valores de tensión espurios que puedan producir la aparición de velocidades no deseadas en el motor.

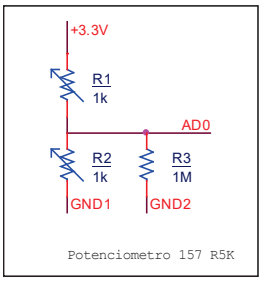


Figura 18: Esquemático del potenciómetro

3.2.5. Interruptor final de carrera modelo 10T85 μ, de *CQC*

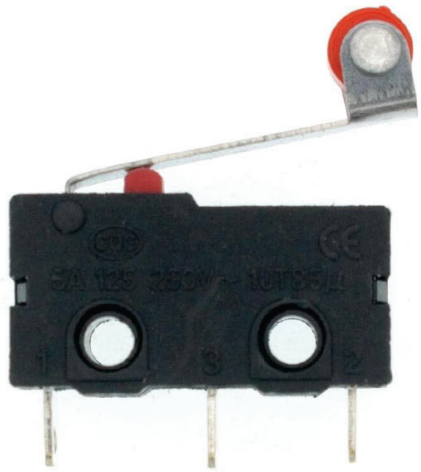


Figura 19: Aspecto de un modelo de final de carrera similar al utilizado [30]

Interruptor final de carrera de intensidad máxima de 5A, tensión máxima de 250V.

Un interruptor final de carrera es un conmutador que permite dos estados: un estado activo y un estado inactivo, según se pulse o no el interruptor. Dependiendo del estado, el terminal *COM* tomará el valor de tensión de uno de los dos terminales restantes.

Los tres terminales son:

- *COM* (*common*): Terminal común para ambos estados, adquiere el valor de tensión de *NC* o *NO*.
- *NC* (*Normally closed*): La conexión entre *NC* y *COM* se encuentra cerrada en estado inactivo, por lo que cuando el interruptor no se encuentra pulsado *COM* adquiere el valor de tensión de *NC*. Al pulsarse el interruptor, la conexión se abre.
- *NO* (*Normally opened*): La conexión entre *NO* y *COM* se encuentra abierta en estado inactivo; cuando el interruptor se pulsa se produce la conexión entre ambos terminales y *COM* adquiere el valor de tensión de *NO*. Si se deja de pulsar el interruptor, la conexión se abre.

Para este proyecto se han utilizado dos interruptores, cuyo objetivo será detener la actuación del motor para el control del péndulo llegada la plataforma a los límites del recorrido; se trata pues, de un mecanismo de seguridad.

La hoja de datos del módulo se encuentra en *[Anexo XIII]*.

4. Diseño del sistema de control

Existen varios subproblemas, cada uno objeto de un control independiente, asociados al péndulo invertido. La conducción del péndulo desde una posición cualquiera hasta una posición de equilibrio en posición invertida requiere de cierto aporte de energía, que es objeto de control; el equilibrio del péndulo una vez llegado a ese punto (objetivo de este proyecto) también requiere un control independiente. El control de posición angular del péndulo no acota el desplazamiento de la plataforma, razón por la cual se han incorporado los finales de carrera; si se quisiera restringir este desplazamiento, sería necesario incorporar, de nuevo, un sistema de control para la posición de ésta o utilizar otro diseño como el Péndulo de Furuta [2].

Todo esto deja claro que existen múltiples subproblemas susceptibles de ser controlados para perfeccionar el comportamiento de este sistema; sin embargo, el objetivo de este proyecto es únicamente el control de posición angular del péndulo.

La variable controlada es la tensión aportada al motor, ya que es la única variable que se puede controlar directamente. Se llevó a cabo una metodología heurística para determinar el tipo de control y los parámetros adecuados.

4.1. Pasos previos para el diseño del sistema de control

Antes del diseño del sistema de control, es necesario tener en cuenta la existencia de fenómenos no lineales que en ocasiones pueden ser solucionados y que implican la necesidad de calibración de algunos instrumentos.

4.1.1. Obtención de la función de transferencia y parámetros del motor cc

(Este capítulo ha sido parcialmente extraído y adaptado de [6]).

A continuación se presenta un esquema simplificado del circuito asociado a un motor de corriente continua:

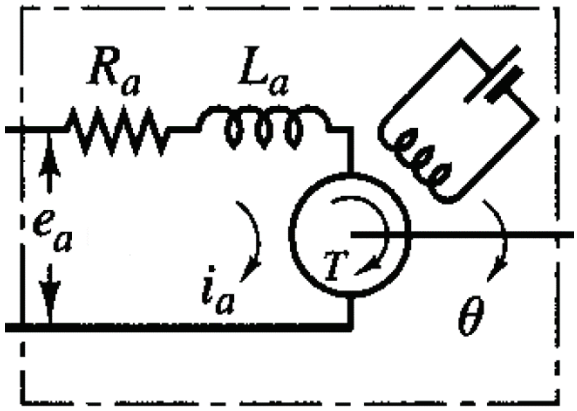


Figura 20: Esquemático interno simplificado de un motor cc (Adaptado de [6])

Donde e_a es la tensión de armadura suministrada al motor, R_a es la resistencia de armadura, L_a la inductancia de armadura, i_a la corriente de armadura, *T* el par ejercido en el eje del motor y θ el desplazamiento angular en el eje del motor.

Para una corriente de campo constante, el par que desarrolla el motor es proporcional a la corriente de armadura:

$$T = K_m i_a \tag{39}$$

Donde K_m es la constante de par e i_a dicha corriente de armadura.

Cuando la armadura gira, se induce en ella una tensión proporcional al producto del flujo magnético y la velocidad angular. Si el flujo es constante, la tensión inducida es directamente proporcional a la velocidad angular, de la forma:

$$e_b = K_b \frac{d\theta}{dt} \tag{40}$$

Donde e_b es la fuerza contraelectromotriz y K_b la constante de fuerza contraelectromotriz.

Aplicando la segunda Ley de Kirchoff al circuito de armadura, se obtiene su ecuación diferencial:

$$L_a \frac{di_a}{dt} + R_a i_a + e_b = e_a \tag{41}$$

O bien:

$$L_a \frac{di_a}{dt} + R_a i_a + K_b \frac{d\theta}{dt} = e_a \tag{42}$$

La ecuación para el equilibrio de pares es:

$$J_0 \frac{d^2\theta}{dt^2} + b_0 \frac{d\theta}{dt} = T = K_m i_a \tag{43}$$

Donde J_0 es la inercia del motor y b_0 es el coeficiente de fricción viscosa, ambos referidos a la flecha del motor.

La función de transferencia del motor que relaciona desplazamiento angular y la tensión de entrada se obtiene a partir de las ecuaciones (42) y (43):

$$\frac{\theta(s)}{E_a(s)} = \frac{K_m}{s(L_a s + R_a)(J_0 s + b_0) + K_m K_b s} \tag{44}$$

Teniendo en cuenta el cociente de reducción del motor, la flecha de salida gira *n* veces por cada revolución de la flecha del motor:

$$C(s) = n\theta(s) \tag{45}$$

Donde C(s) es la transformada de Laplace del desplazamiento angular en la flecha de salida.

Relacionando las ecuaciones (44) y (45), se obtiene la función de transferencia del motor de corriente continua que relaciona la tensión de entrada y el desplazamiento angular de la flecha de salida:

$$G_\theta(s) = \frac{C(s)}{\theta(s)}\frac{\theta(s)}{E_a(s)} = \frac{K_m n}{s[(L_a s+R_a)(J_0 s+b_0)+K_m K_b]} \quad (46)$$

Como L_a suele ser en general muy pequeña se puede despreciar, por lo que la función (46) quedaría:

$$G_\theta(s) = \frac{K_m n}{s[R_a(J_0 s+b_0)+K_m K_b]} = \frac{K_m n/R_a}{J_0 s^2+\left(b_0+\frac{K_m K_b}{R_a}\right)s} \quad (47)$$

Cuando los términos J_0 y $b_0 + \frac{K_m K_b}{R_a}$ se multiplican por $\frac{1}{n^2}$, la inercia y el coeficiente de fricción viscosa quedan expresados con respecto a la flecha de salida. Si denominamos a estos términos con unos nuevos parámetros:

$$J = \frac{J_0}{n^2}\,,\; B = \frac{\left(b_0+\frac{K_m K_b}{R_a}\right)}{n^2},\; K = \frac{K_m}{nR_a} \;\; ;$$

La función (47) queda simplificada como:

$$G_\theta(s) = \frac{K}{Js^2+Bs} \;\; ;$$

o bien:

$$G_\theta(s) = \frac{K_{mm}}{s(T_{mm}s+1)} \quad (48)$$

donde:

$$K_{mm} = \frac{K}{\mathrm{B}},\;\; T_{mm} = \frac{J}{\mathrm{B}} = \frac{R_a J_0}{R_a b_0+K_m K_b} \;\; ;$$

Siendo T_{mm} la constante de tiempo.

Para el modelo de simulación, se necesita una expresión de función de transferencia

del motor de corriente continua que relacione la tensión de entrada y el par en la flecha de salida. Para ello, se obtiene en primer lugar la relación entre desplazamiento angular y par de salida referidos a la flecha del motor en el dominio de Laplace:

$$(J_0 s^2 + b_0 s)\theta(s) = T(s) \tag{49}$$

Con (10) y (11), se obtiene la expresión de la función de transferencia del motor que relaciona par ejercido a la salida y tensión de entrada:

$$G_\tau(s) = \frac{T(s)}{E_a(s)} = \frac{K_{mm} s(J_0 s + b_0)}{s(T_{mm} s + 1)} \tag{50}$$

que, desarrollando y simplificando, se convierte en:

$$G_\tau(s) = \frac{K_{mm}(J_0 s + b_0)}{(T_{mm} s + 1)} = \frac{\frac{K}{B}(J_0 s + b_0)}{\left(\frac{J}{B} s + 1\right)} = \frac{\frac{\frac{K_m}{nR_a}}{\frac{\left(b_0 + \frac{K_m K_b}{R_a}\right)}{n^2}}(J_0 s + b_0)}{\left(\frac{\frac{J_0}{n^2}}{\frac{\left(b_0 + \frac{K_m K_b}{R_a}\right)}{n^2}} s + 1\right)} = \frac{\frac{\frac{K_m}{nR_a}}{\frac{\left(\frac{R_a b_0 + K_m K_b}{R_a}\right)}{n^2}}(J_0 s + b_0)}{\left(\frac{R_a J_0}{R_a b_0 + K_m K_b} s + 1\right)} =$$

$$\frac{\frac{nK_m}{R_a b_0 + K_m K_b}(J_0 s + b_0)}{\left(\frac{R_a J_0}{R_a b_0 + K_m K_b} s + 1\right)} = \frac{nK_m(J_0 s + b_0)}{R_a J_0 s + R_a b_0 + K_m K_b} \tag{51}$$

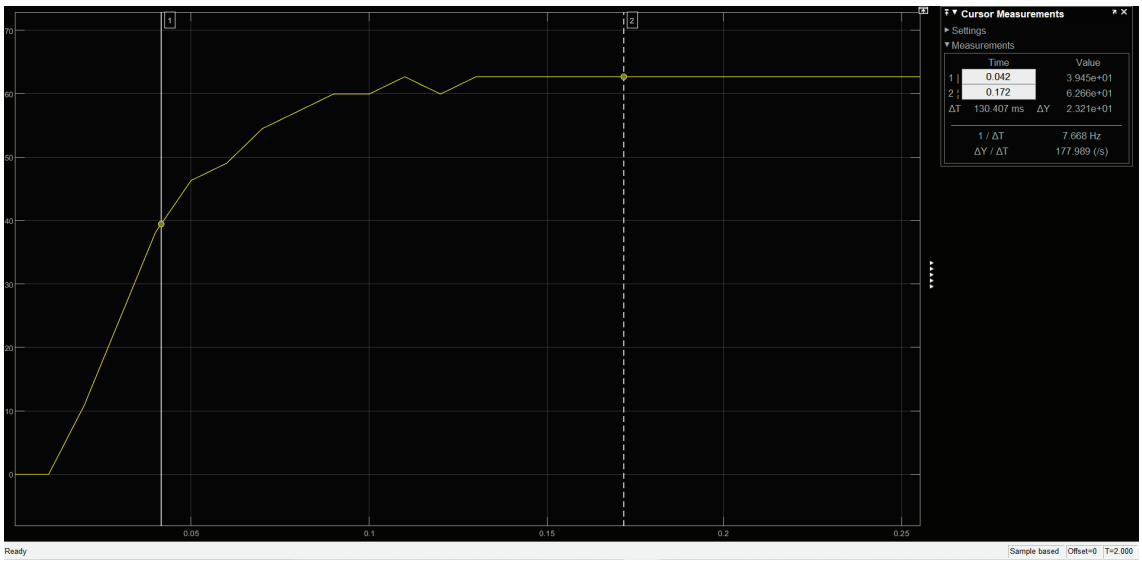
Para pasar de la salida en forma de par a fuerza lineal, basta con multiplicar por el radio del piñón unido al eje de salida del motor, que produce el movimiento de la correa:

$$G_f(s) = \frac{rnK_m(J_0 s + b_0)}{R_a J_0 s + R_a b_0 + K_m K_b} \tag{52}$$

La función de transferencia $G_\tau(s)$ puede expresarse también de la forma:

$$G_\tau(s) = \frac{K_{mm}(J_0 s + b_0)}{(T_{mm} s + 1)} = \frac{K_{mm}(\frac{J_0}{b_0} s + 1)}{(T_{mm} s + 1)} = \frac{K_{mm}(\tau_1 s + 1)}{(\tau_2 s + 1)} \quad (53)$$

La constante de tiempo del motor se define como el tiempo que transcurre hasta que la respuesta del motor alcanza el 63% de su valor final menos el tiempo debido al retardo puro. Para hallarla, se introduce una entra escalón de tensión máxima y se observa la zona transitoria de la salida de velocidad:



Gráfica 1 : Respuesta en velocidad del motor ante una entrada escalón de tensión

El valor final de la velocidad en estado estacionario es 62.66 rad/s y el 63% de ese valor, 39,47 rad/s, se alcanza a los 0.042s. Midiendo en la gráfica, como el retardo puro es de 0.01s, la constante de tiempo es $T_{mm} = 0.042 - 0.01 = 0.032s$.

La aplicación directa de (51) o (52) en el modelo de simulación no resulta posible, puesto que se desconocen los valores de la inercia del motor J_0 y del coeficiente de fricción viscosa b_0, que no son especificados en la hoja de datos del motor. Sin embargo, se puede realizar una aproximación. Tratándose una función de trasnferencia con un polo y un cero, su diagrama de Bode posee dos tramos de invariabilidad de la ganancia: uno para bajas frecuencias y otro para altas frecuencias, con valores de ganancia distintos. Como la frecuencia interna del sistema se puede considerar baja, la función de transferencia se puede aproximar a una ganacia constante:

$$|G_\tau(j\omega)|_{100Hz} \approx K$$

Ganancia que se ha hallado de manera experimental, colgando un peso de la plataforma y determinando la fuerza ejercida por el motor a tensión máxima. Ésta ha resultado ser de 17.66 N, siendo la ganancia $K = \frac{17.66}{12}$, expresada en N/V.

4.1.2. Fenómenos no lineales

El modelo linealizado del péndulo solamente es válido de manera local, para una horquilla de ángulo muy reducida en torno al punto de equilibrio (el modelo ha sido linealizado suponiendo que $sen\,\theta \approx \theta$ y $cos\,\theta \approx 1$) y su control asumible. Fuera de esta horquilla, el problema de control es difícil de atajar, al tratarse de un sistema no lineal e inestable en bucle abierto [2]. Y es que el modelo del péndulo invertido es eminentemente no lineal, más allá de que algunos de sus procesos físicos puedan ser linealizados localmente.

Incluso, a pesar de trabajar con un sistema linealizado, existen multitud de elementos que intervienen en el proceso de control de la posición angular que tienen asociados

fenómenos no lineales, y suponen un problema a la hora de lograr un control que cumpla con los requisitos exigidos. Algunos de estos fenómenos son:

- La zona muerta del motor.
- La zona muerta asociada a la salida del potenciómetro.
- El huelgo en la reductora del motor.
- La elasticidad de la cinta, que provoca que la plataforma no se mueva inmediatamente al aplicar el motor un par.
- El truncado de señales.
- La propia no linealidad del sistema.
- No linealidades en el desempeño de los dispositivos electrónicos.

Muchos de estos fenómenos son difíciles de atajar, requieren complejos sistemas de control y están fuera de los objetivos del presente proyecto; sin embargo, existen algunos comportamientos de no linealidad que sí pueden ser solucionados de forma simple.

4.1.3. Caracterización de la zona muerta del motor

Es común la aparición de zonas de no linealidad en el comportamiento de un motor real, siendo una de ellas la zona muerta.

La zona muerta es un efecto de no linealidad en el que ante una entrada de tensión no nula la salida del motor es nula, por lo que, hasta no superar cierto umbral en la entrada de tensión, no se aprecia una respuesta en el motor. El aspecto de esta zona se puede apreciar en la siguiente gráfica:

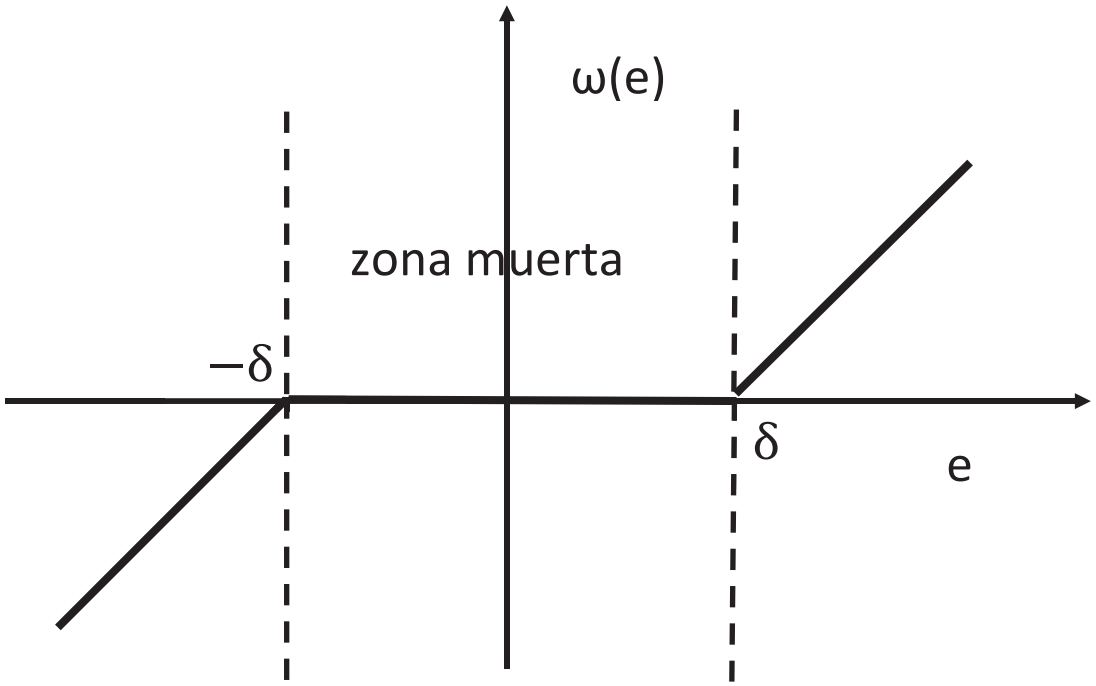
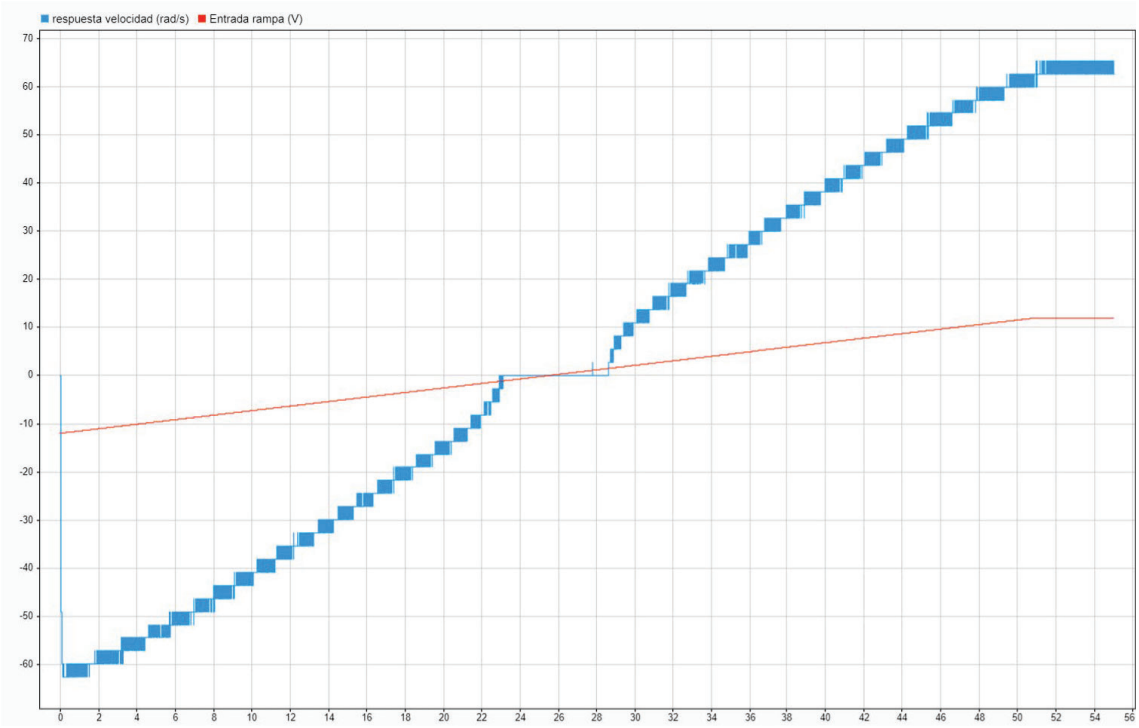


Figura 21: Representación simplificada de la respuesta de velocidad en función de la tensión suministrada en un motor cc

La gráfica muestra de forma esquemática la evolución de la velocidad angular a la salida en función de la entrada de tensión en un motor de corriente continua. La zona muerta está delimitada por dos valores, -δ y δ, que son los umbrales de tensión de entrada partir de los cuales comienza la respuesta lineal del motor, para tensiones negativas y positivas, respectivamente. Estos valores no tienen por qué coincidir en valor absoluto.

Es necesario atajar este problema para poder garantizar la respuesta adecuada del motor en todas las situaciones y antes de diseñar el sistema de control. El objetivo será hallar los umbrales δ y δ para realizar una traslación en el eje horizontal de la función, de $e_t = e - \delta$ para el semiplano positivo y de $e_t = e + (-\delta)$ en el negativo, logrando así salvar la no linealidad.

Para obtener estos valores experimentalmente, se introduce como entrada de tensión una rampa unitaria saturada en 12V y se realizan varias pruebas. El aspecto de la salida de velocidad es:



Gráfica 2: Respuesta en velocidad del motor ante una entrada rampa de tensión

Midiendo los valores de tensión a los que comienza la respuesta de velocidad no nula para cada prueba, en este caso 5, se realiza un promedio, redondeando el resultado por mayoración:

$$\delta_{(0,255]} = \frac{1.430 + 1.614 + 1.628 + 1.632 + 1.750}{5} = 1.611 \approx 1.7\ V$$

$$-\delta_{[-255,0)} = \frac{-1.025 - 1.073 - 1.078 - 1.092 - 1.115}{5} = -1.077 \approx -1.1\ V$$

Para trabajar con valores de tensión codificados, basta con aplicar el factor de conversión:

$$u_{\delta(0,255]} = 1.7\frac{255}{12} \approx 37$$

$$u_{-\delta[-255,0)} = -1.1\frac{255}{12} \approx -24$$

Que son los valores que se aplicarán en la traslación de la función.

4.1.4. Truncado

Como se ha explicado en apartados previos, la entrada de tensión al motor es controlada por señales PWM que son suministradas al driver de potencia. Estas señales poseen una frecuencia y amplitud fijas, siendo el Duty Cycle el único parámetro que varía en cada periodo de muestreo del programa. Es evidente que la frecuencia de las señales PWM debe ser siempre mayor que la frecuencia de muestreo.

Es importante tener en cuenta la aparición de inexactitudes debido a una mal sincronización de las frecuencias de las señales PWM y de muestreo, que pueden afectar al correcto desempeño del motor.

Una de estas inexactitudes es el posible solapamiento entre el periodo de muestreo y el periodo de la señal PWM. Ocurre cuando el final del periodo de muestreo coincide con uno de los pulsos en los que la señal se encuentra en estado activo, es decir, la señal PWM generada finaliza en un momento de su periodo en el que está activa y por tanto uno de sus pulsos queda “cortado”. Este fenómeno se conoce como truncado y se ilustra en el siguiente esquema:

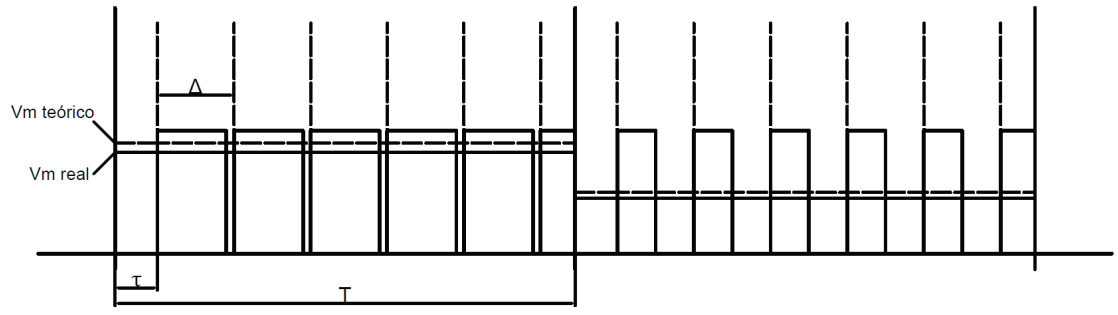


Figura 22: Representación de truncado en una señal PWM

Donde τ es el retardo en la señal PWM producido por los procesos computacionales del programa, Δ el periodo de la señal PWM y T el periodo de muestreo. El retardo τ en la

generación de la señal PWM está compuesto a su vez de distintos subretardos, principalmente, el retardo inherente al sistema físico y el retardo debido a los tiempos de cálculo, que es variable dependiendo del coste computacional.

La consecuencia de este suceso es un valor de tensión medio incorrecto e inferior al valor teórico estipulado en el Duty Cycle de la señal PWM. Los valores de tensión media real y teórica se puede calcular con las siguientes expresiones:

$$V_{m\,teórico} = \frac{N(\,V_{cc+} - \,V_{cc-})\,\frac{DutyCycle}{100}}{N}$$

$$V_{m\,real} = \frac{(N-1)(\,V_{cc+} - \,V_{cc-})\,\frac{DutyCycle}{100} + (\,V_{cc+} - \,V_{cc-})\,\frac{t}{\Delta}}{N}$$

Siendo *N* el número de ciclos de la señal PWM, completos o no, de los que consta una iteración del periodo de muestreo, *Δ* el periodo de la señal PWM y *t* la duración del nivel alto de la señal en el último ciclo.

Una forma de solucionar este problema sería modelar el retardo τ y modificar la frecuencia PWM para que el periodo de muestreo fuese múltiplo de la suma del periodo PWM y del retardo. Como esto no es algo fácil de medir, se puede optar por minimizar todo lo posible la influencia del truncado; esto se consigue aumentando lo máximo posible la frecuencia PWM para que el impacto del último ciclo truncado sobre el valor medio de tensión de la señal sea mínimo. Las salidas digitales para señales PWM utilizadas 9 y 10, tienen asociadas por defecto una frecuencia de 490 Hz; sin embargo, esta frecuencia es modificable, como se verá en el apartado destinado al control.

4.1.5. Calibrado del potenciómetro

A pesar de que la existencia de un ángulo no hábil en el potenciómetro no es evitable, sí que se puede decidir su ubicación para que afecte lo menos posible al comportamiento del sistema. Esto se consigue situando esta área alejada de la zona próxima al punto de equilibrio, donde el sistema físico tiene un comportamiento fuertemente no lineal y fuera del efecto del control.

Para alejar lo máximo posible la zona muerta del área se orientó el potenciómetro sobre la plataforma procurando que la bisectriz de su ángulo no hábil de 20º fuese perpendicular a la horizontal. De esta manera, el ángulo hábil queda dividido en dos sectores de 170º cada uno.

A continuación, es necesario situar el desplazamiento nulo (0º) en la posición de equilibrio buscada. Éste es el proceso de calibrado. Para ello, se midió la lectura del potenciómetro cuando el péndulo se encuentra en esta posición, obteniéndose un valor de lectura de 345 en el CAD (Conversor Analógico Digital). Basta restar este valor a la lectura del potenciómetro para ubicar el origen del desplazamiento en la posición buscada y asignar lecturas de igual valor absoluto y distinto signo para ángulos de desplazamiento negativos y positivos a partir del origen. Finalmente, leyendo el valor de salida máximo de lectura en el CAD (675) y sabiendo el recorrido angular útil (340º), se puede obtener la conversión de lectura en el CAD a grados. El valor 675 viene dado por el conversor analógico digital de Arduino Mega 2560, con una resolución de 10 bits para una tensión máxima de 5V. En este prototipo, el valor de alimentación de tensión del potenciómetro es 3.3V, por lo que su resolución va de 0 a 675 [31].

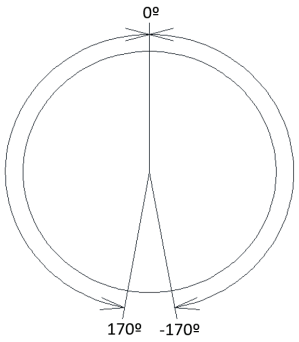


Figura 23: Distribución angular en el potenciómetro tras el proceso de calibrado

Por tanto, la ecuación para el calibrado del potenciómetro es:

$$\theta_{(grados)} = \frac{\theta_{(uds\_angulares)} \times 340}{675}$$

4.2. Diseño del algoritmo de control

En primer lugar, se probó el control todo-nada con histéresis. La idea era utilizar la tensión máxima del motor ±255 y en función del signo de la derivada del error de ángulo (ésta expresada en valor absoluto), es decir, de si el error aumentaba o disminuía, establecer la tensión a +255 o a -255 en la zona próxima al punto de equilibrio, prolongando así la tendencia del valor de tensión más allá de dicho punto. Con esto se conseguirían dos curvas distintas en función de la derivada del error, tal y como se observa en una curva de histéresis clásica.

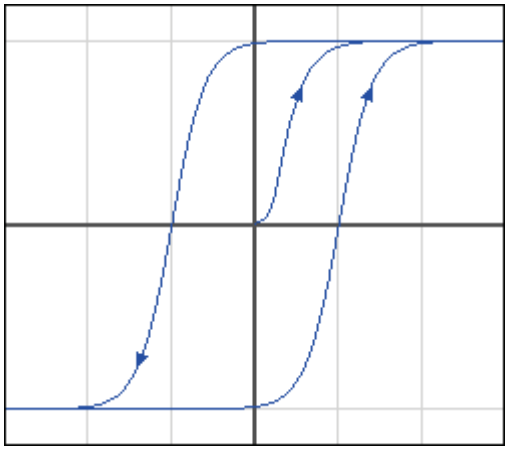


Figura 24: Curva de histéresis [32]

Sin embargo, este diseño no logró cumplir los objetivos, por lo que comenzó a diseñarse un control PID.

El método heurístico seguido para el diseño del PID se asemeja al segundo método de Zieger-Nichols, aunque éste no es aplicable al tratarse de un sistema inestable.

En primer lugar, se comenzó por aumentar la ganancia proporcional procurando evitar la saturación de la tensión para valores de error de posición bajos (si la ganancia fuese muy grande, los valores de tensión comandados se saturarían a 255 y -255 para cualquier error por pequeño que fuese, funcionando el control en la práctica como un todo-nada). Ajustando el valor, se consiguió la oscilación en torno al punto de equilibrio en estado estacionario para una constante proporcional $K_p = 100$.

La constante derivativa se anticipa al comportamiento del sistema, lo que permite la estabilización en régimen permanente. El parámetro a ajustar es la constante de tiempo derivativa T_d, siendo la constante derivativa $K_d = K_p T_d$, y se va aumentando paulatinamente hasta que se logre la estabilización deseada. El valor elegido fue $T_d = 0.1$, es decir, una constante derivativa $K_d = 10$.

El añadido de factor integral no solo no es necesario sino que podría ser contraproducente, ya que un aumento en la velocidad de la respuesta podría conducir al péndulo a la zona de no linealidad, es decir, fuera de la zona de control. Por lo tanto, el controlador elegido es un PD de constante proporcional $K_p = 100$ y constante derivativa $K_d = 10$.

Además, se realizó un segundo control PD con una ganancia proporcional K_p= 300 y una constante derivativa K_d=30 que consiguió alzar el péndulo desde la posición de reposo. Se comprobó que debido a la saturación, y a la velocidad de subida, este control funciona en la práctica como un control no lineal de tipo todo-nada. Sin embargo resulta útil, ya que tiene capacidad para operar fuera de la zona de linealidad, pudiendo ser útil en algoritmos que contemplen el levantamiento del péndulo, garantizando en todo caso la estabilidad.

4.2.1. Modelo de control en bucle cerrado

El aspecto del sistema de control en bucle cerrado es el siguiente:

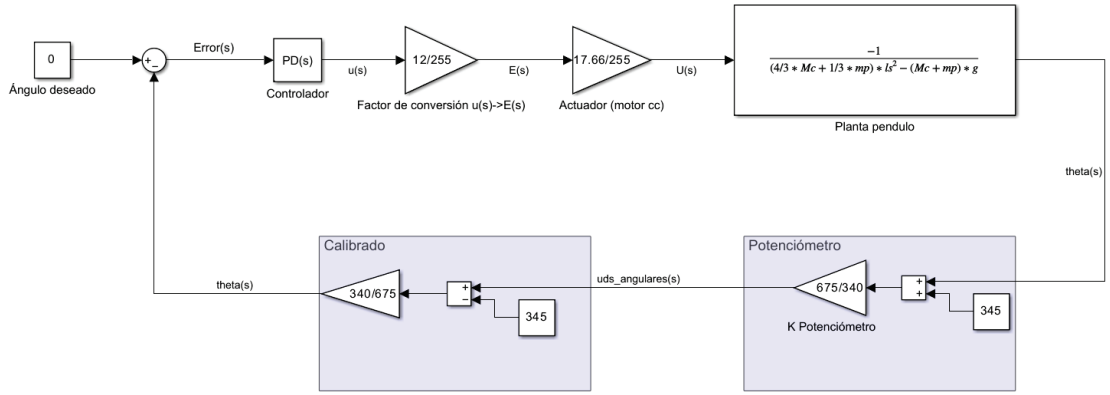


Figura 25: Vista general del modelo de control en bucle cerrado

Consta de la referencia de ángulo, un controlador PD (más adelante se justificará su uso), la aproximación en bajas frecuencias a la función de transferencia del motor, la planta del péndulo y en el brazo de realimentación, la función de transferencia del potenciómetro (una ganancia). La simulación del sistema no se ha considerado oportuna, ya que, como se vio en el subapartado referente a los fenómenos no lineales, el comportamiento obtenido en este modelo linealizado discreparía mucho del comportamiento real para valores de posición angular lejanos al punto de equilibrio.

4.2.2. Implementación del sistema de control. Código Arduino

Para implementar el sistema de control, se diseñó un programa en el entorno de desarrollo de Arduino. Originalmente se realizó el diseño mediante Simulink, pero este entorno no resultó lo suficientemente fiable en los tiempos de ejecución.

Para una mayor fiabilidad en la ejecución periódica de los procesos, se ha diseñado el programa en base a una interrupción temporizada al periodo de muestreo y cuya

activación depende de uno de los relojes o temporizadores internos del procesador *atmega2560* de Arduino Mega 2560. Esto evita el impacto de retrasos propios del procesamiento de las funciones del programa.

El código completo es el siguiente:

```
#include <GyverTimers.h>


const int fdc1 = 19;
const int fdc2 = 24;
const int enable_motor = 8;
const int IN3 = 9;
const int IN4 = 10;
unsigned int contador = 0;

int valor_pot = 0;
float valor_calibrado = 0;
float valor_calibrado_ant = 0;

float error_act = 0;
float error_ant = 0;
float derror = 0;

int U_comandada = 0;

float velocidad = 0;


volatile int estado1 = 0; //Para fdc1
volatile int estado2 = 0; //Para fdc2
volatile int biestable = 0;


8. Diseño del sistema de control
//Funcion calibrado

float calibrado(int lectura_CAD){

  float valor_calibrado=0;


    valor_calibrado = (float(lectura_CAD)*340)/675; //*340/675 340º de recorrido angular,
                                    //675 uds angulares


  return valor_calibrado;

}


//Funcion derivada
```

```
void derivada (float valor_calibrado){


  error_act = -valor_calibrado; //La distancia a cero


  derror = (abs(error_act)-abs(error_ant))/(0.01); //El incremento de tiempo entre una
                                        //interrupción y la siguiente es de 0.01 s

  velocidad = (valor_calibrado-valor_calibrado_ant)/(0.01);

  error_ant = error_act;
  valor_calibrado_ant = valor_calibrado;

}


//Funcion controlador

int control(float derror)
{
  int U = 0;
  const int kp=100;
  const int Td=0.1;

  U= round(-kp*(error_act + Td*derror));

  if (U>=255){

    U=255;
  }

  else if (U<=-255){

    U=-255;
  }

  else if (U>-255 && U<0){

    U=U-24;
  }

  else if (U>0 && U<255){

    U=U+37;
  }


  return U;

}

//Interrupcion del temporizador

ISR(TIMER1_A)
{
```

```
 estado1=digitalRead(fdc1);
 estado2=digitalRead(fdc2);
 valor_pot = analogRead(A0)-345;


  if(estado1==1){
    biestable=-1;
    contador=50;

  }

  else if(estado2==1){
    biestable=1;
    contador=50;
  }



  if (contador>0){
     if(biestable==-1){
       analogWrite(IN4,255); //-255.
       analogWrite(IN3,0);

     }

     else if(biestable==1){
       analogWrite(IN3,255); //+255
       analogWrite(IN4,0);
     }
    contador--;
  }

  else{

    biestable=0;

    valor_calibrado = calibrado(valor_pot);
    derivada(valor_calibrado);
    U_comandada = control(derror);

    Serial.println(velocidad);

       if (U_comandada >=0){
         analogWrite(IN3,U_comandada);
         analogWrite(IN4,0);
       }
       else{
         analogWrite(IN4,-U_comandada);
         analogWrite(IN3,0);
       }

  }

}
```

```
void setup() {


   Serial.begin(115200);
   pinMode(fdc1, INPUT_PULLUP);
   pinMode(fdc2, INPUT_PULLUP);
   pinMode(enable_motor, OUTPUT);
   pinMode(IN3, OUTPUT);
   pinMode(IN4, OUTPUT);

   TCCR2B = TCCR2B & B11111000 | B00000001;  // Para una frecuencia de PWM de
31372.55 Hz

   Timer1.setFrequency(100);
   Timer1.enableISR(CHANNEL_A);

   digitalWrite(enable_motor, HIGH);

}

void loop() {


}
```

Código 1: Programa Arduino

A continuación, se abordarán con más detalle las partes que forman el programa.

- **Librerías**

```
#include <GyverTimers.h>
```

Permite la configuración de los temporizadores del procesador atmega2560.

- **Cabecera. Declaración de variables globales**

```
const int fdc1 = 19
const int fdc2 = 24
```

Números de pin de las entradas digitales asociadas a las salidas de los finales de carrera. El final de carrera 1 es el situado a la izquierda de la plataforma y el 2 el situado a la derecha, mirando el alzado de la maqueta.

```
const int enable_motor = 8
const int IN3 = 9
const int IN4 = 10
```

Números de pin de la salida digital para señal de habilitación del motor B y de las salidas para señales PWM.

```
unsigned int contador = 0
```

Declaración de un contador. Este contador será utilizado para temporizar el desplazamiento de la plataforma una vez pulsado alguno de los finales de carrera.

```
int valor_pot = 0
float valor_calibrado = 0
float valor_calibrado_ant = 0
```

Variables asociadas al desplazamiento angular. *valor_pot* representa la salida del potenciómetro menos 345, es decir, el valor de salida del potenciómetro con el cero centrado en el punto de equilibrio. *valor_calibrado* es el desplazamiento angular en grados tras el calibrado y *valor_calibrado_ant* es el valor calibrado en la iteración anterior.

```
float error_act = 0
float error_ant = 0
float derror = 0
```

Variables asociadas al error de ángulo: error actual, error en la iteración anterior y derivada del error.

```
int U_comandada = 0
```

Valor de tensión codificada para suministrar al motor.

```
float velocidad = 0
```

Velocidad angular.

```
volatile int estado1 = 0                    //Para fdc1
volatile int estado2 = 0                    //Para fdc2
volatile int biestable = 0
```

Variables asocidadas a los finales de carrera. Estado lógico del final de carrera 1, estado lógico del final de carrera 2 y una variable que actúa como biestable, tomando dos valores distintos en función del final de carrera activado.

- **Función de Setup**

```
Serial.begin(115200)
pinMode(fdc1, INPUT_PULLUP)
pinMode(fdc2, INPUT_PULLUP)
pinMode(enable_motor, OUTPUT)
pinMode(IN3, OUTPUT)
pinMode(IN4, OUTPUT)
```

En primer lugar, se establecen los baudios de comunicación con el monitor serie. A continuación, con la función *pinmode* se configuran las entradas y salidas utilizadas: el estado de la entrada fdc1 toma un valor lógico alto tras la detección de una pulsación por las resistencias pull-up internas de Arduino Mega, al igual que fdc2. enable_motor, IN3 e IN4 se configuran como salidas.

```
TCCR2B = TCCR2B & B11111000 | B00000001    // Para una frecuencia de PWM de 31372.55 Hz
```

Esta línea modifica la frecuencia de las señales PWM emitidas por las salidas digitales 9 y 10 (IN2 e IN3), concretamente a 31372.55 Hz, frente a los 490 Hz por defecto.

```
Timer1.setFrequency(100)
Timer1.enableISR(CHANNEL_A)
```

Se configura el temporizador 1, que posee una franja de frecuencias configurables que contiene a la frecuencia de muestreo buscada de 100Hz. Existen seis temporizadores en ATmega2560, cada uno de ellos con dos o tres canales asociados a un pin [33]. En este caso, se ha escogido el canal A, asociado al pin 11 de Arduino Mega, que se encuentra libre, para iniciar la interrupción.

```
digitalWrite(enable_motor, HIGH)
```

Esta línea establece un valor lógico alto a la salida de habilitación del motor B.

- **Bucle**

El bucle principal está vacío; las iteraciones las va a llevar a cabo la interrupción principal.

- **Función calibrado**

```
//Funcion calibrado

float calibrado(int lectura_CAD){

 float valor_calibrado=0;


   valor_calibrado = (float(lectura_CAD)*340)/675; //*340/675 340º de recorrido angular,
                                    //675 uds angulares


 return valor_calibrado;

}
```

Tiene como entrada un valor de salida del potenciómetro ya tarado, es decir, al que se ha restado el valor de salida en el origen de equilibrio 345. Esta función se encarga de devolver en formato *float* el valor calibrado de salida del potenciómetro, es decir, la posición angular, en grados. Para ello, realiza un *casting* de *int* a *float* de la entrada *lectura_CAD* para aplicarle el factor de conversión de valores leídos en el CAD a grados y que el resultado obtenido sea de tipo *float*.

- **Función derivada**

```
//Funcion derivada
void derivada (float valor_calibrado){


 error_act = -valor_calibrado; //La distancia a cero


 derror = (abs(error_act)-abs(error_ant))/(0.01); //El incremento de tiempo entre una
                                    //interrupción y la siguiente es de 0.01 s

 velocidad = (valor_calibrado-valor_calibrado_ant)/(0.01);

 error_ant = error_act;
```

```
  valor_calibrado_ant = valor_calibrado;

}
```

Función de tipo *void* que tiene como entrada un valor calibrado de posiciónangular en grados. Se encarga de calcular la derivada del error y la velocidad angular; para ello, en primer lugar calcula el error restando a 0 el valor de posición angular. Después, calcula la derivada del error con el valor actual del error y el valor que tenía en la anterior iteración, cada uno en valor absoluto, y lo divide entre el tiempo entre iteraciones (el periodo de muestreo). Esta es una aproximación lineal a la derivada. Los errores se toman en valor absoluto para que el signo de la derivada indique si el error aumenta o disminuye en magnitud. De igual manera, se calcula la velocidad angular como la derivada de la posición angular, resultando de signo negativo si el desplazamiento se produce en sentido horario y positivo para el antihorario.

- **Función control**

```
int control(float derror)
{
  int U = 0;
  const int kp=100;
  const int Td=0.1;
```

5. Diseño del sistema de control

Esta función toma como entrada la derivada del error y devuelve como salida la tensión codificada a suministrar al motor *U*. Se encarga pues de implementar el controlador PD. En primer lugar, se declaran como variables locales U, la constante proporcional Kp y la constante de tiempo derivativa Td.

```
  U= round(kp*(error_act + Td*derror));
```

La siguiente línea implementa la acción de control: el valor de U queda determinado por la suma de la acción proporcional kp*(error_act) y de la acción derivativa kp*Td*(d_error). Como *U* solo toma valores enteros en el intervalo [-255, 255], hay que redondear el resultado.

```
if (U>=255){

   U=255;
 }

 else if (U<=-255){

   U=-255;
}
```

En estas líneas se satura la entrada de tensión codificada si su valor se encuentra fuera del intervalo [-255, 255].

```
else if (U>-255 && U<0){

   U=U-24;
 }

 else if (U>0 && U<255){

   U=U+37;
 }


 return U;

}
```

Por último, se implementa la compensación de la zona muerta.

- **Interrupción temporizada**

Es la parte principal del programa. La interrupción es controlada por la activación del temporizador hardware 1, que lo hace con una frecuencia de 100 Hz, la frecuencia de muestreo.

```
//Interrupcion del temporizador

ISR(TIMER1_A)
{

estado1=digitalRead(fdc1);
estado2=digitalRead(fdc2);
valor_pot = analogRead(A0)-345;
```

Primero se guardan los estados de los finales de carrera leyendo sus entradas asociadas. Se lee el valor marcado por el potenciómetro y se tara.

```
if(estado1==1){
  biestable=-1;
  contador=50;

 }
```

```
 else if(estado2==1){
  biestable=1;
  contador=50;
 }
```

A continuación, si se ha detectado una activación de alguno de los finales de carrera, se asigna un valor a la variable *biestable*, distinto para cada final de carrera: si se ha activado el final de carrera 1, se asigna el valor -1 y si lo ha hecho el 2, el 1. En ambos casos, se establece un valor al contador de 50.

```
 if (contador>0){
    if(biestable==-1){
      analogWrite(IN4,255); //-255.
      analogWrite(IN3,0);

    }

    else if(biestable==1){
      analogWrite(IN3,255); //+255
      analogWrite(IN4,0);
    }
  contador--;
 }
```

Estas líneas se encargan del desplazamiento de la plataforma al centro del recorrido en caso de detectarse la pulsación de algún final de carrera. El condicional principal funciona como un bucle dentro de la interrupción, ya que será lo único que ejecutará junto a las líneas anteriores hasta que el contador no tome el valor 0. Mientras sea mayor que 0 y dependiendo del final de carrera pulsado (biestable=-1 para fdc1 (izquiedo) o biestable=1 para fdc2 (derecho)), genera una señal PWM de Duty Cicle 100% (255), es decir, provee al motor de una entrada de tensión máxima. La salida que genera dicha señal PWM se escoge en función de si se requiere el giro antihorario del motor para un desplazamiento a la izquierda (activación del final de carrera 2) o un

desplazamiento a la derecha (activación del final de carrera 1). La otra salida emitirá una señal nula.

En cada iteración se resta una unidad al contador. Si el periodo de muestreo es T=0.01s, el contador no tomará el valor 0 hasta pasados 500 ms (50 iteraciones), tiempo suficiente para que la plataforma se desplace hasta el centro del recorrido.

```
  else{

   biestable=0;

   valor_calibrado = calibrado(valor_pot);
   derivada(valor_calibrado);
   U_comandada = control(derror);
```

Si el contador es igual a 0, se procede a realizar el control. Se llama secuencialmente a las funciones de calibrado, de derivada y de control, obteniéndose finalmente el valor de tensión codificada a comandar *U_comandada* a partir de *valor_pot*, el valor de salida del potenciómetro tarado.

```
    Serial.println(velocidad)
```

La función *Serial.println()* permite visualizar el valor del parámetro de entrada en el monitor serie. También permite elaborar gráficas usando la herramienta *Serial Plotter*.

```
      if (U_comandada >=0){
        analogWrite(IN3,U_comandada);
        analogWrite(IN4,0);
      }
      else{
        analogWrite(IN4,-U_comandada);
        analogWrite(IN3,0);
      }

  }

}
```

Por último, dependiendo del signo del valor de tensión codificada a comandar *U_comandada*, que determina el sentido de giro del motor, se genera la señal PWM en la salida adecuada con un Duty Cicle determinado por el valor absoluto de dicha *U_comandada.*

5. Resultados

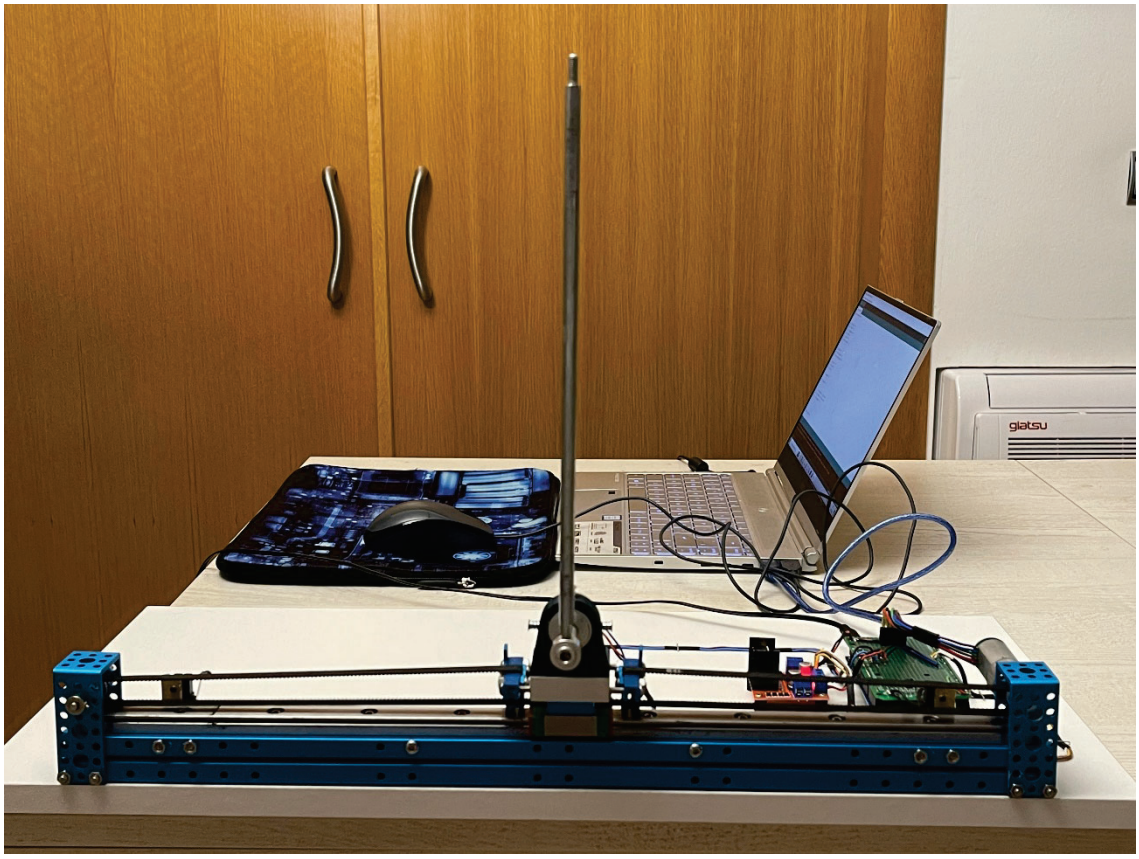
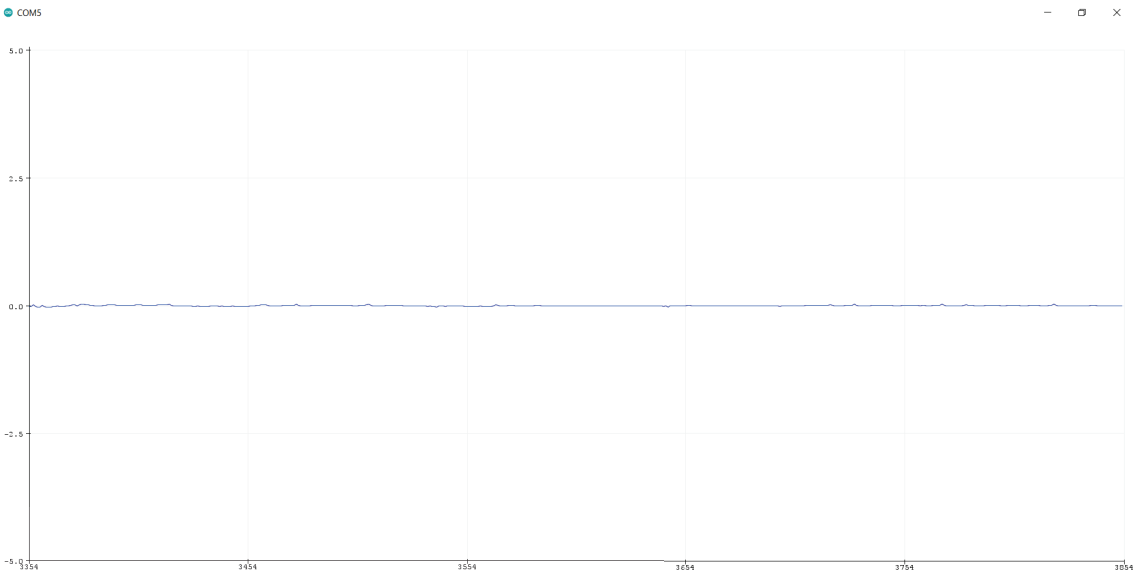


Figura 26: Péndulo en equilibrio

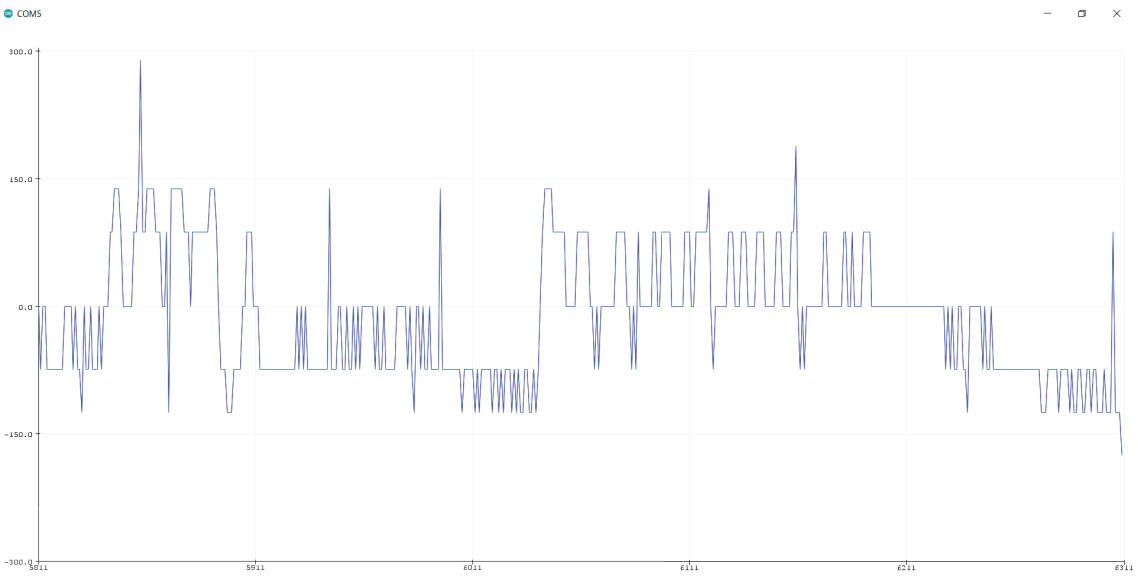
A continuación se presentan los resultados obtenidos para la ejecución del sistema de control. Se ha sometido al péndulo a tres situaciones distintas para estudiar su respuesta en posición angular, velocidad y tensión de alimentación al motor; estas tres situaciones son: el posicionamiento manual en el punto de equilibrio, el choque con los finales de carrera desde una situación de equilibrio y el posicionamiento en el punto de reposo. Las gráficas han sido obtenidas desde la herramienta *Serial Plotter* del entrono de desarrollo de Arduino. Las gráficas presentadas en cada situación corresponden a distintos experimentos.

- Respuesta en el punto de equilibrio



Gráfica 3: Posición angular en el punto de equilibrio (rad)

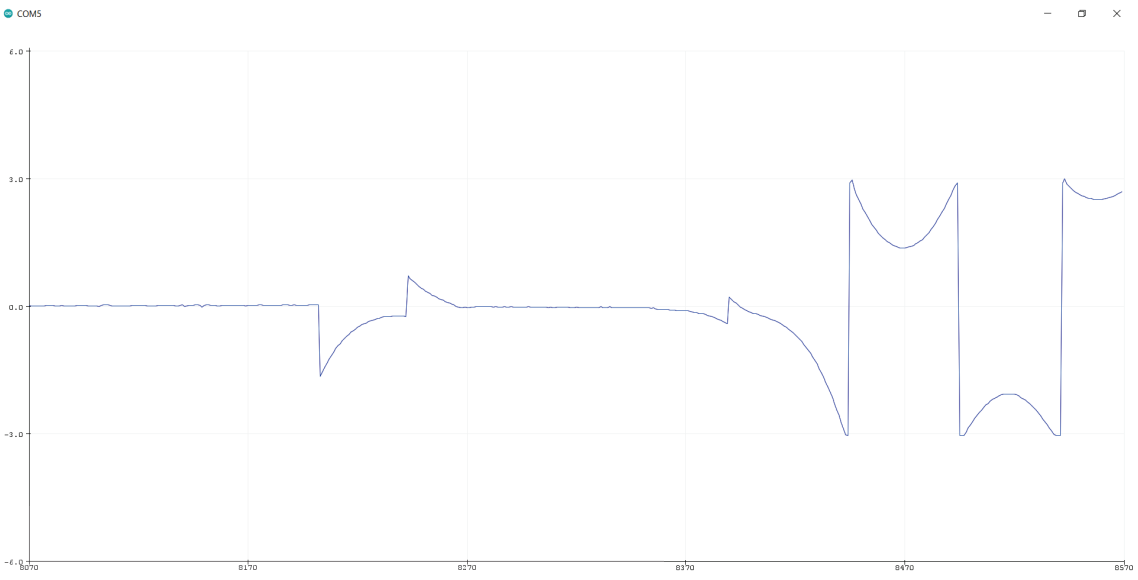
Gráfica 4: Velocidad en el punto de equilibrio (rad/s)



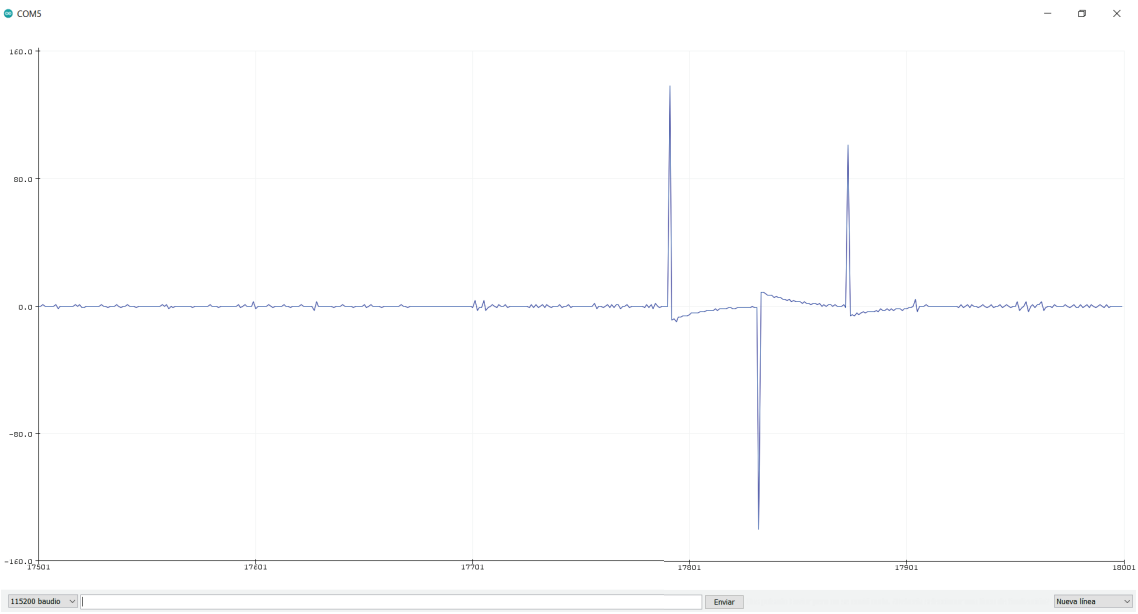
Gráfica 5: Valores de tensión codificada comandada en el punto de equilibrio

Se observa cómo se consigue el objetivo de estabildad en torno al punto de equilibrio con un desplazamiento mínimo. El sistema de control envía constantemente picos de tensión que se traducen en picos de velocidad para intentar mantener un desplazamiento angular de 0º. Esto muestra la validez del diseño.

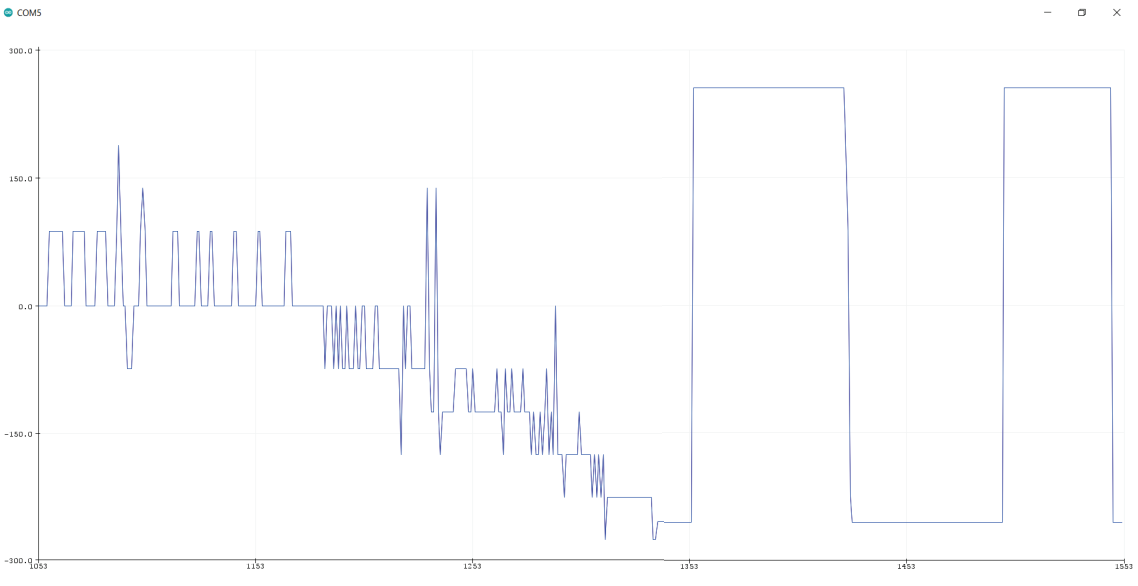
- Choque desde el equilibrio



Gráfica 6: Posición angular en el punto de equilibrio, tras un choque con un final de carrera y posterior recuperación, y tras un choque sin recuperación (rad)

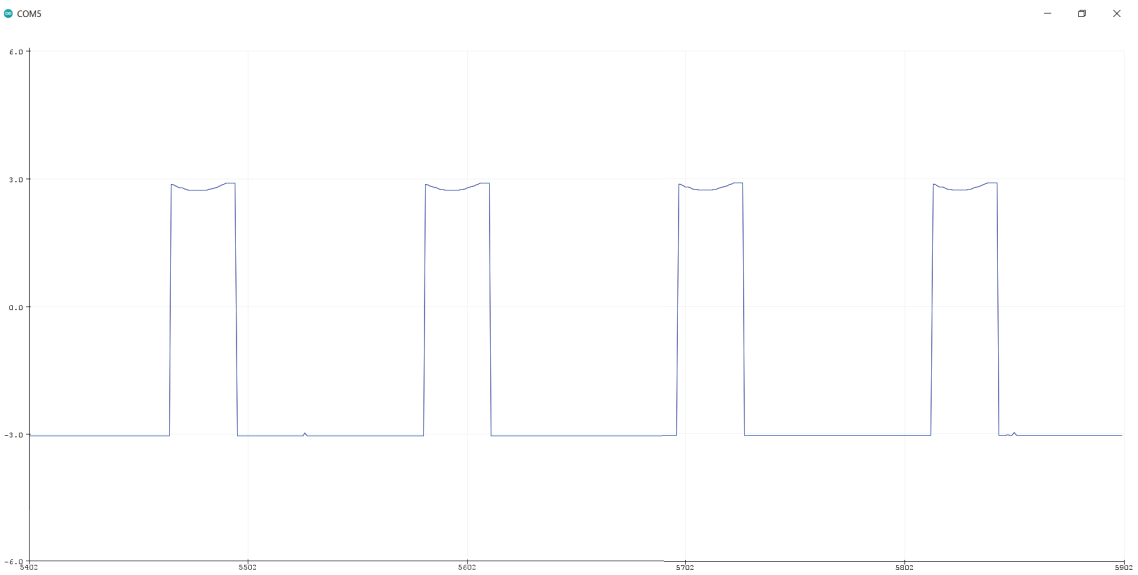


Gráfica 7: Velocidad en el punto de equilibrio y tras el choque con un final de carrera y posterior recuperación (rad/s)

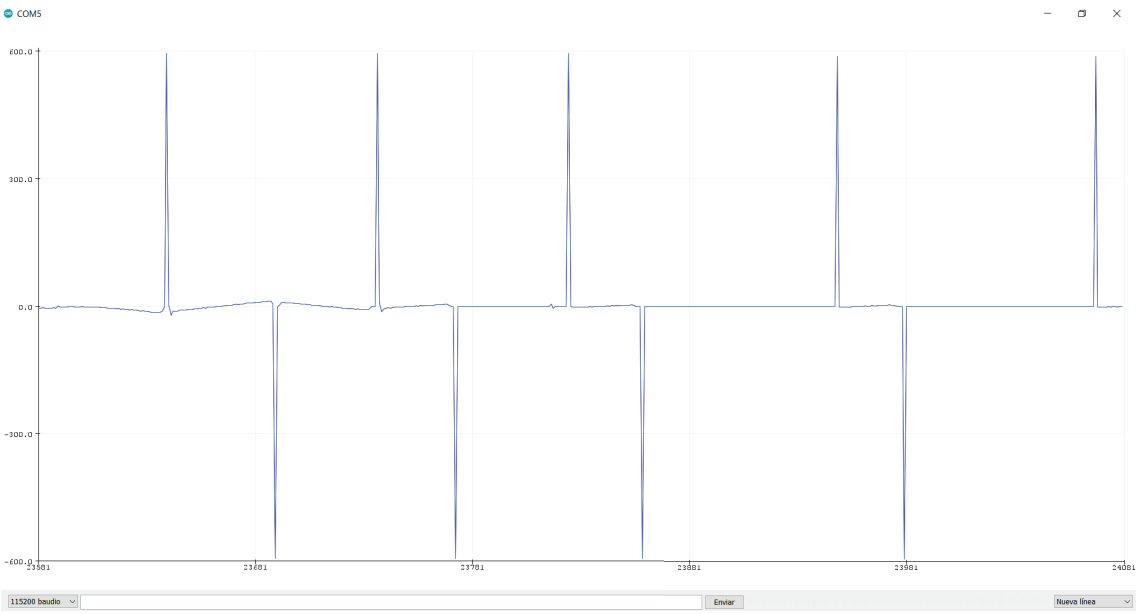


Gráfica 8: Valores de tensión comandada en el punto de equilibrio y tras el choque con un final de carrera sin recuperación

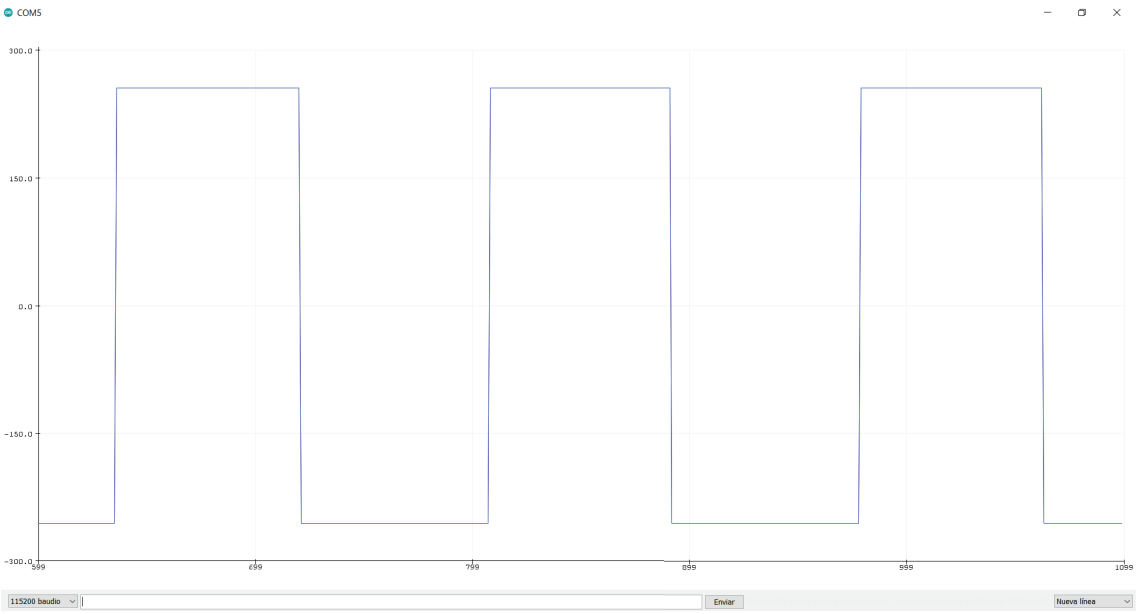
Desde el equilibrio, el sistema de control envía órdenes al motor para desplazar la plataforma y mantener el péndulo en la posición deseada. Cuando la plataforma choca con un final de carrera, se omiten los comandos de tensión del controlador y entra en juego el sistema de posicionamiento de la plataforma en el centro del recorrido. En ocasiones, puede recuperarse el equilibrio tras el choque si el desplazamiento angular no ha sido de gran magnitud en el proceso de centrado de la plataforma.



Gráfica 9: Posición angular en torno al punto de reposo (rad) , con choque con final de carrera



Gráfica 10: Velocidad en torno al punto de reposo (rad/s), con choque con final de carrera



Gráfica 11: Valores de tensión codificada comandada en torno al punto de reposo, con choque con final de carrera

En torno al punto de reposo el sistema de control no es capaz de elevar el péndulo hasta el punto de equilibrio. El envío de comandos de tensión máxima para intentar levantar el péndulo provoca el desplazamiento de la plataforma hasta los límites, chocando con

los finales de carrera. En la gráfica de velocidad se aprecia cómo el cruce con la zona muerta del potenciómetro, en el que la lectura del CAD es 0, provoca que no se envíe ningún comando de tensión al motor.

En la gráfica de tensión, los tramos de valor +255 corresponden a la tensión comandada por el sistema de control, mientras que aquellos de valor -255 corresponden al sistema de centrado de la plataforma.

6. Conclusiones

El sistema físico del péndulo invertido no supone un reto fácil en cuanto a control se refiere por su naturaleza fuertemente no lineal. Por ello los equipos de prueba como en el desarrollado en este proyecto, se consideran fundamentales para la evaluación del comportamiento de distintas estrategias de control. En este proyecto se ha desarrollado con éxito un prototipo de péndulo invertido sobre plataforma móvil utilizando componentes de bajo coste. La base de la plataforma es un sistema de posicionamiento de un grado de libertad de *MakeBlock*. EL péndulo invertido está sensorizado mediante un potenciómetro de giro continuo de calidad aeronáutica. Como equipo de control, se emplea un Arduino tipo Mega.

Se ha demostrado que el sistema funciona adecuadamente, alcanzándose la estabilización tanto con controles lineales como en saturación (todo o nada). Asimismo, se ha comprobado que el sistema tiene también capacidad para producir la levantada del péndulo desde la posición del equilibrio estable. No obstante, la realización de controladores de este tipo no entra en el objetivo del proyecto.

Para el futuro, se proponen mejoras del prototipo, como es la sustitución del motor con reductora por uno de tracción directa (lo que eliminaría no linealidades relacionadas con el juego o huelgo de la reductora), y la utilización de un sensor mejorado de posición para el péndulo. También sería deseable la utilización de un encoder para el motor de mayor resolución, para abordar problemas de control de posición conjunto de la plataforma y del péndulo.

7. Referencias

[1] Kafetzis, I., y Moysis, L. (Marzo de 2017) *Inverted Pendulum: A system with innumerable applications*. Tesalónica: School of Mathematical Sciences, Aristotle University of Thessaloniki. https://ikee.lib.auth.gr/record/288541/files/Inverted%20pendulum%20-%20A%20system%20with%20innumerable%20applications.pdf

[2] Aracil, J. y Gordillo, F. (2010) El Péndulo Invertido: un Desafío para el Control No Lineal. *Revista Iberoamericana de Automática e Informática industrial*, 2(2), 8–19. https://polipapers.upv.es/index.php/RIAI/article/view/8056

[3] Arévalo Parra, P. A., Vergara Esparza, R., Flores Alvarado, M., Guevara Carrillo, V., Serrano Martínez, A. A., y Zaleta Amador, C. A. (2017). Sistema de control de un péndulo invertido rotacional basado en control moderno. *Jóvenes en la ciencia, revista de divulgación científica,* 3(2), 417. https://www.jovenesenlaciencia.ugto.mx/index.php/jovenesenlaciencia/article/view/2294/1786

[4] Orchard, M., Sáez, D., Duarte, M., Jofré, E., Guajardo, F., Cartagena, O., y Barriga, R. (15 de junio de 2017) *Control en variables de estado para un péndulo invertido*. Chile: Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile. https://www.u-cursos.cl/ingenieria/2017/1/EL5205/1/material_docente/bajar?id_material=1823699

[5] NASA. (25 de julio de 2005). *Shuttle Discovery July 25 pre-launch-crop*. [Fotografía]. https://upload.wikimedia.org/wikipedia/commons/4/4a/Shuttle_Discovery_July_25_pre-launch-crop.jpg

[6] Ogata, K. (1998). *Ingeniería de control moderna*, Pearson.

[7] Función de transferencia (s.f.). Wikipedia. Recuperado el 11 de agosto de 2022 de https://es.wikipedia.org/wiki/Funci%C3%B3n_de_transferencia

[8] Valera Fernández, Á. (2002). *Modelado y control en el espacio de estados*. Valencia: Universitat Politècnica de València.

[9] Crespo, E. (4 de julio de 2016) *Motores.* Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/2016/07/04/motores/#:~:text=Un%20motor%20driver%20es%20un,que%20pueda%20alimentar%20el%20motor.

[10] Llamas Binaburo, L. (26 de mayo de 2016) *Controlar motores de corriente continua con Arduino y L298N.* Luis Llamas: Ingeniería, informática y diseño. (https://www.luisllamas.es/arduino-motor-corriente-continua-l298n/

[11] Uriarte, I. (s.f.) *El módulo controlador de motores L298N: Controlar la velocidad y el sentido de giro de dos motores de corriente continua*. Prometec. https://www.prometec.net/l298n/

[12] Uriarte, I. (s.f.). *Conexiones-L298N*. [Fotografía]. https://www.prometec.net/wp-content/uploads/2016/11/Conexiones-L298N.png

[13] Uriarte, I. (s.f.), (s.t.). [Fotografía]. https://www.prometec.net/wp-content/uploads/2016/10/HTB1m34rIpXXXXc9XXXXq6xXFXXXN.jpg

[14] L298_internal. (s.f.). [Fotografía]. https://www.inventable.eu/wp-content/uploads/2017/05/L298_internal.png

[15] motor_dc_puente_con_L298_BD, motor_dc_puente_con_L298_AD, motor_dc_puente_con_L298_BC, motor_dc_puente_con_L298_AC. (s.f.). [Imágenes] https://www.inventable.eu/2017/05/26/funciona-puente-motores-corriente-continua/

[16] Inventable. (26 de mayo de 2017). *Cómo funciona un puente para motores de corriente continua*. https://www.inventable.eu/2017/05/26/funciona-puente-motores-corriente-continua/

[17] Aranzábal Olea, A. (2001). *Filtro por condensador*. Electrónica básica: Curso de Electrónica Básica en Internet. Departamento de Electrónica y Telecomunicaciones de la Escuela Universitaria de Ingeniería Técnica Industrial de Éibar. http://www.sc.ehu.es/sbweb/electronica/elec_basica/tema4/Paginas/Pagina6.htm

[18] Arduino. (s.f.). Wikipedia. Recuperado el 12 de agosto de 2022 de https://es.wikipedia.org/wiki/Arduino

[19] Getting Started: What is Arduino? (5 de febrero de 2018). Arduino. Recuperado el 20 de agosto de 2022 de https://www.arduino.cc/en/Guide/Introduction

[20] Fernández, Y. (3 de agosto de 2020) *Qué es Arduino, cómo funciona y qué puedes hacer con uno*. Xataka. https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

[21] Guadrón Gutiérrez, R., y Guevara Vásquez, J. J. (2014) Implementación de Bootloaders en Microcontroladores PIC16 y PIC18 de Microchip Inc.
Revista Tecnológica ITCA – FEPADE, 7(1), 59-60. http://www.redicces.org.sv/jspui/bitstream/10972/2545/1/CAP%2010.pdf

[22] Arduino Mega 2560. (s.f.) Arduino. https://arduino.cl/arduino-mega-2560/

[23] Electrónica Embajadores. (s.f.), (s.t.). [Fotografía] https://www.electronicaembajadores.com/es/Productos/Detalle/LCA1010/modulos-electronicos/arduino/arduino-2560-rv03-arduino-mega-2560-rev-3-original-a000067/

[24] Guerrera Carmenate, J. (s.f.) Arduino Mega 2560 el hermano mayor de Arduino UNO. *Programarfacil*. https://programarfacil.com/blog/arduino-blog/arduino-mega-2560/

[25] Mouser. (s.f.), (s.t.). [Fotografía] https://www.mouser.es/ProductDetail/Digilent/290-008?qs=s%2FdyVPQMB4zvWqNo5wm3QA%3D%3D

[26] Principios de funcionamiento del encoder cuadratura de efecto Hall para medición de velocidad en motores de corriente directa. (s.f.). Punto Flotante S.A. https://www.puntoflotante.net/FUNCIONAMIENTO-ENCODER-CUADRATURA-EFECTO-HALL.htm#:~:text=Los%20sensores%20de%20efecto%20Hall%20son%20transductores%20que%20generan%20un,en%20motores%20de%20corriente%20directa.

[27] Vishay. (s.f.), (s.t.). [Fotografía] https://es.farnell.com/vishay/157-11502/conductive-plastic-potentiometer/dp/1986806

[28] Potentiometers Linearity. Technical Note (s.f.). Bourns, Inc. https://www.bourns.com/docs/technical-documents/technical-library/sensors-controls/technical-notes/Bourns_pot_linearity_technote.pdf

[29] RS. (s.f.), (s.t.). [Fotografía]. https://res.cloudinary.com/rsc/image/upload/b_rgb:FFFFFF,c_pad,dpr_1.0,f_auto,q_auto,w_700/c_pad,w_700/LF162631-11

[30] CQC. (s.f.), (s.t.). [Fotografía]. https://es.aliexpress.com/i/1005004307227166.html

[31] Crespo, E. (22 de octubre de 2017) *Entradas y Salidas Analógicas Arduino. PWM.* Aprendiendo Arduino.
https://aprendiendoarduino.wordpress.com/tag/conversor-analogico-digital/

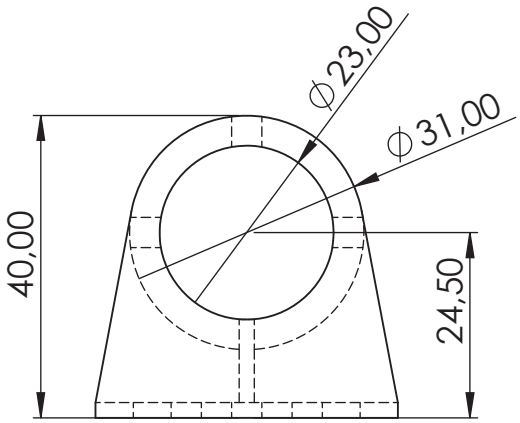
[32] Hysteresiscurve. (19 de marzo de 2004). [Imagen].
https://upload.wikimedia.org/wikipedia/commons/2/2a/Hysteresiscurve.png

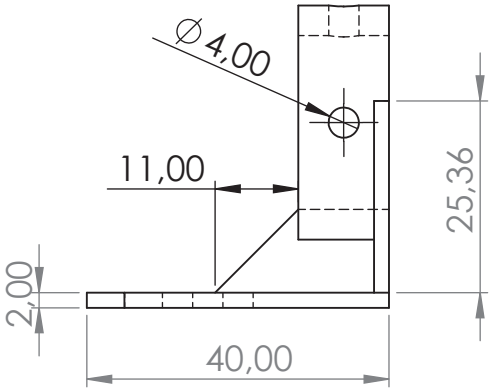
[33] El temporizador de interrupciones en Arduino (s.f.). ElectroDaddy. https://www.electrodaddy.com/el-temporizador-de-interrupciones-en-arduino/

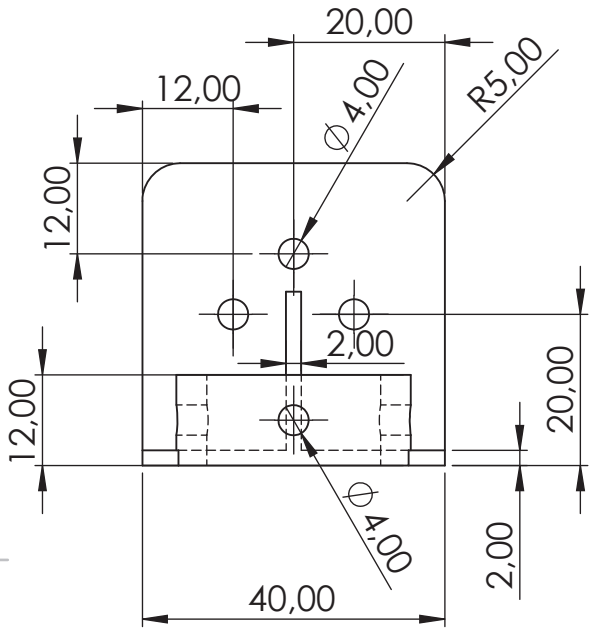
[34] Camacho Montero, D., (4 de noviembre de 2021). Función de lectura encoders. *Campus Virtual UMA: Portafolio Laboratorio de Robótica*.

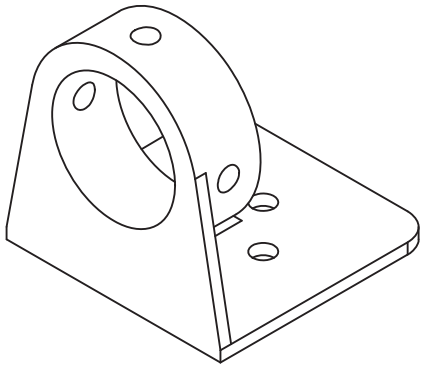
8. Anexos

ANEXO I:
Vistas del soporte del potenciómetro









SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.		CAMACHO MONTERO DIEGO - 45111997G Firmado digitalmente por CAMACHO MONTERO DIEGO - 45111997G Fecha: 2023.01.13 23:37:54 +01'00'				pieza_potenciometro	
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
				PESO:		ESCALA:1:1	HOJA 1 DE 1

Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

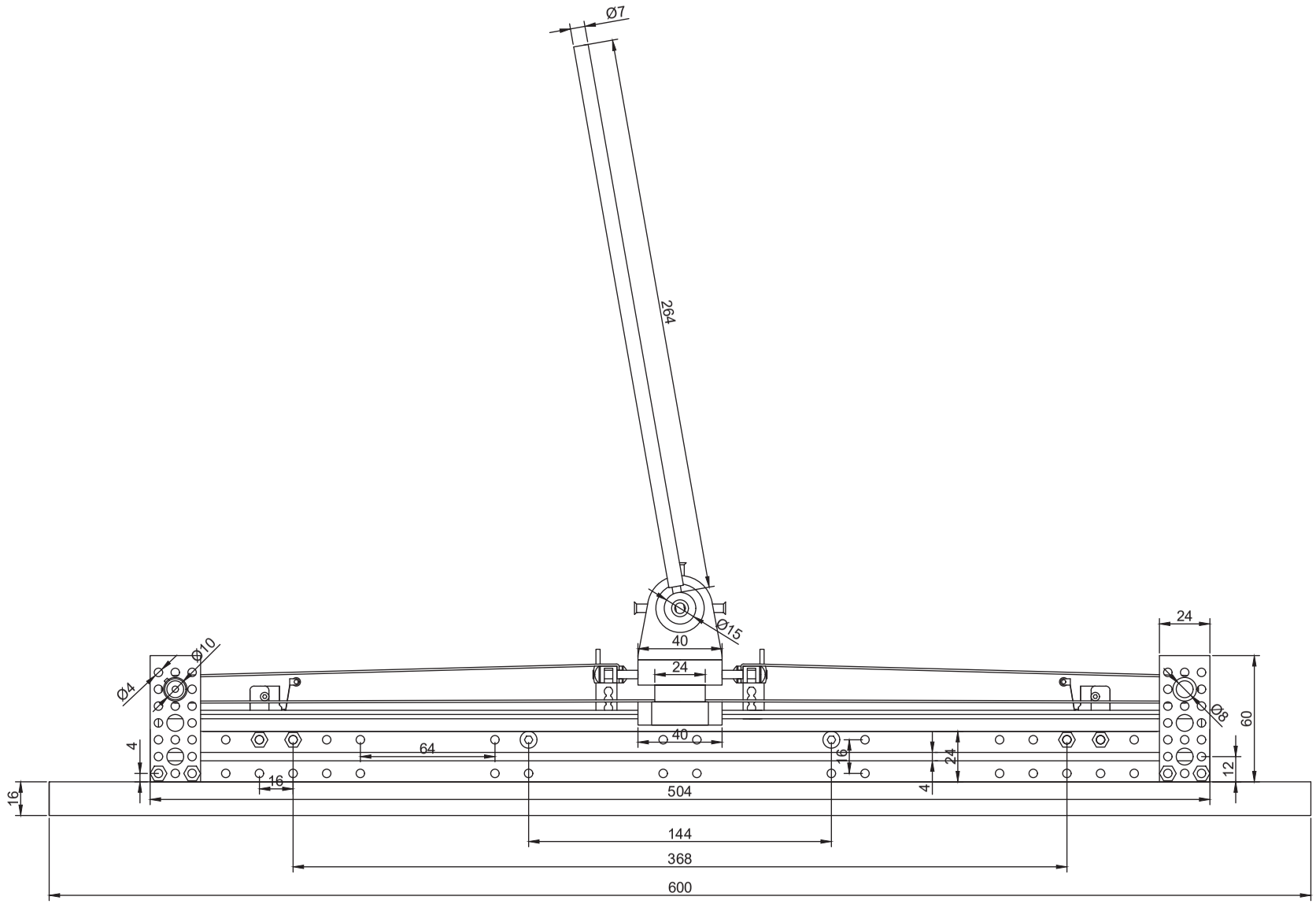
ANEXO II:
Código de colores del cableado

El código de colores del cableado es:

- Conexiones de alimentación y MOTOR+: rojo.
- Conexiones de tierra y MOTOR-: negro.
- Conexión salida potenciómetro-AD0: azul.
- Conexiones salidas finales de carrera-entradas digitales 19 y 24: verde.
- Conexión salida 8-ENB driver: blanco.
- Conexión salida 9-IN3 driver: amarillo.
- Conexión salida 10-IN4 driver: marrón.
- Conexión HALL_SENSOR_VCC: marrón.
- Conexión HALL_SENSOR_GND: verde.
- Conexión HALL_SENSOR_A_Vout-entrada digital 2: azul.
- Conexión HALL_SENSOR_B_Vout-entrada digital 3: morado.

Se han respetado los colores preestablecidos para los terminales de motor y encoder.

ANEXO III:
Alzado de la maqueta del péndulo



UNIVERSIDAD DE MÁLAGA

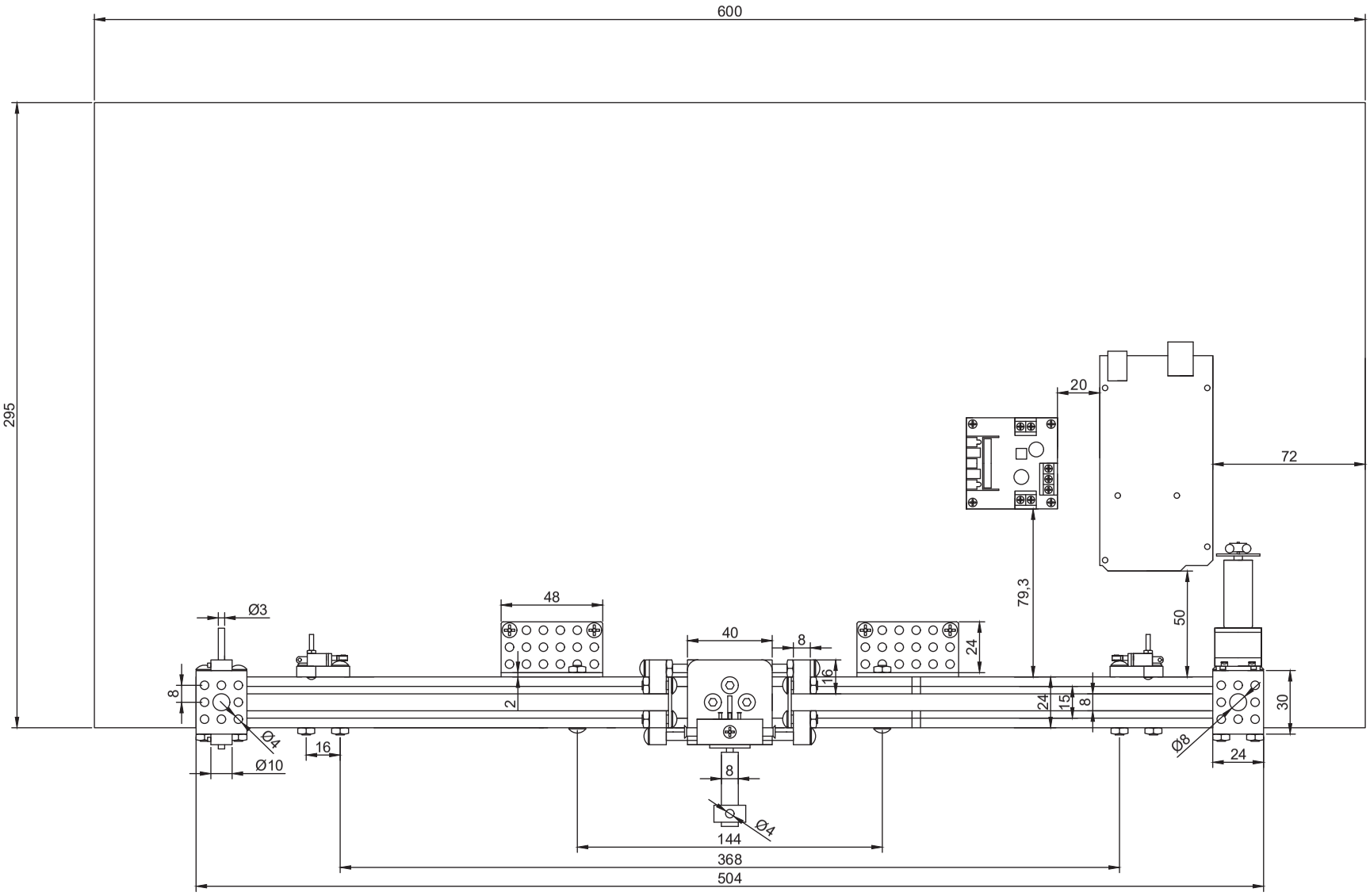
TÍTULO:

Vista del alzado de la maqueta del péndulo invertido

PLANO Nº:	
1	

AUTOR:	REVISOR:	ESCALA:	COTAS EN:	FIRMA:
Diego Camacho Montero	Alfonso Jesús García Cerezo	1:2	mm	CAMACHO MONTERO DIEGO - 45111997G Firmado digitalmente por CAMACHO MONTERO DIEGO - 45111997G Fecha: 2023.01.13 23:38:26 +01'00'
TITULACIÓN: Grado en Ingeniería Electrónica, Robótica y Mecatrónica		FECHA: Nov. 2022		Diego Camacho Montero

ANEXO IV:
Planta de la maqueta del péndulo



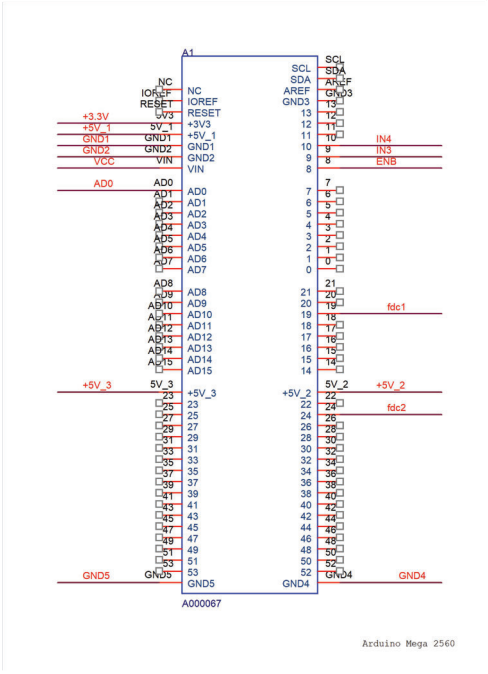
UNIVERSIDAD DE MÁLAGA

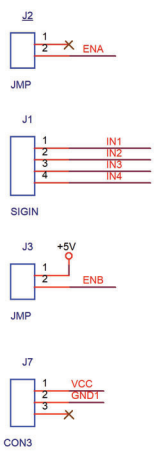
TÍTULO:

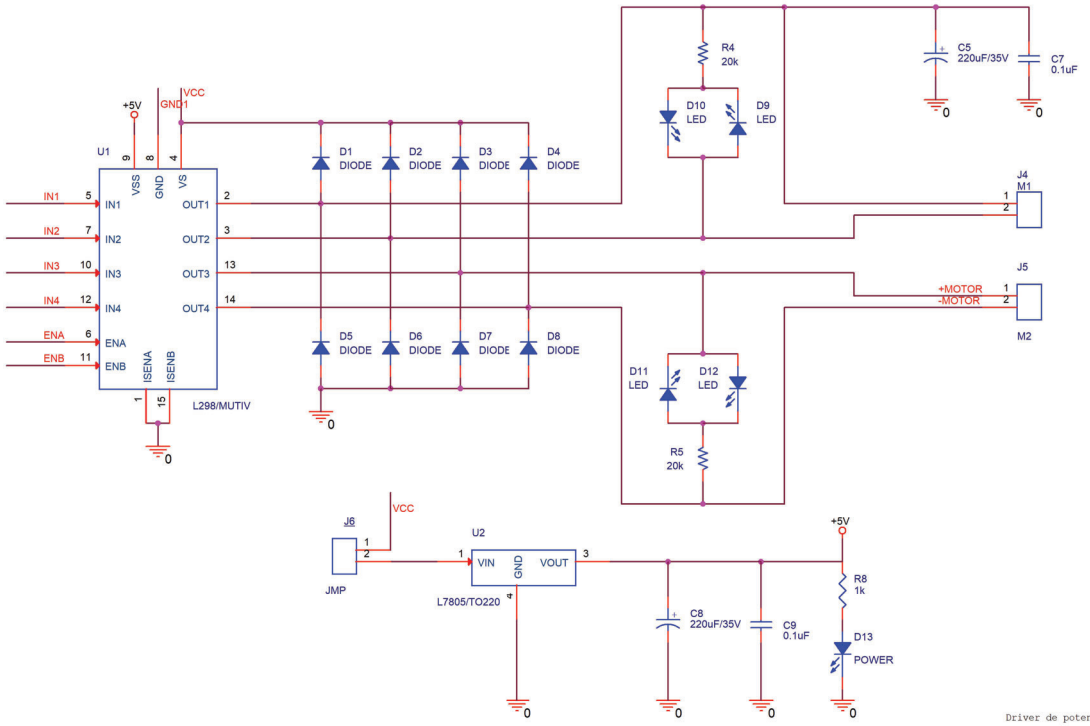
Vista de la planta de la maqueta del péndulo invertido

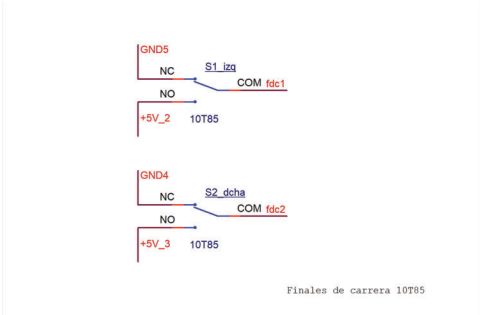
PLANO Nº: 2				
AUTOR: Diego Camacho Montero	REVISOR: Alfonso Jesús García Cerezo	ESCALA: 1:2	COTAS EN: mm	FIRMA: CAMACHO MONTERO DIEGO - 45111997G. Firmado digitalmente por CAMACHO MONTERO DIEGO - 45111997G. Fecha: 2023.01.13 23:38:41 +01'00'
TITULACIÓN: Grado en Ingeniería Electrónica, Robótica y Mecatrónica		FECHA: Nov. 2022		Diego Camacho Montero

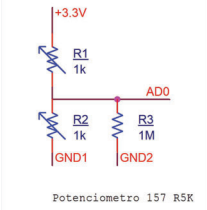
ANEXO V:
Esquemático general del circuito

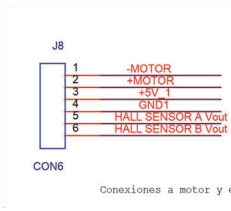


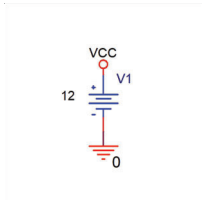












UNIVERSIDAD DE MÁLAGA

TÍTULO:

Esquemático del circuito

PLANO Nº:	
3	

AUTOR:	REVISOR:	FECHA:	FIRMA:
Diego Camacho Montero	Alfonso Jesús García Cerezo	Dic. 2022	CAMACHO MONTERO DIEGO - 45111997G. Firmado digitalmente por CAMACHO MONTERO DIEGO - 45111997G. Fecha: 2023.01.13 23:38:57 +01'00'. Diego Camacho Montero

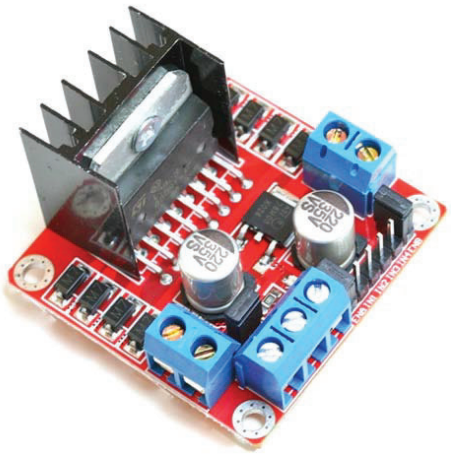
TITULACIÓN:
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

ANEXO VI:
Hoja de datos del módulo L298N

User Guide

L298N Dual H-Bridge Motor Driver

This dual bidirectional motor driver, is based on the very popular L298 Dual H-Bridge Motor Driver Integrated Circuit. The circuit will allow you to easily and independently control two motors of up to 2A each in both directions.It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. It can also be interfaced with simple manual switches, TTL logic gates, relays, etc. This board equipped with power LED indicators, on-board +5V regulator and protection diodes.

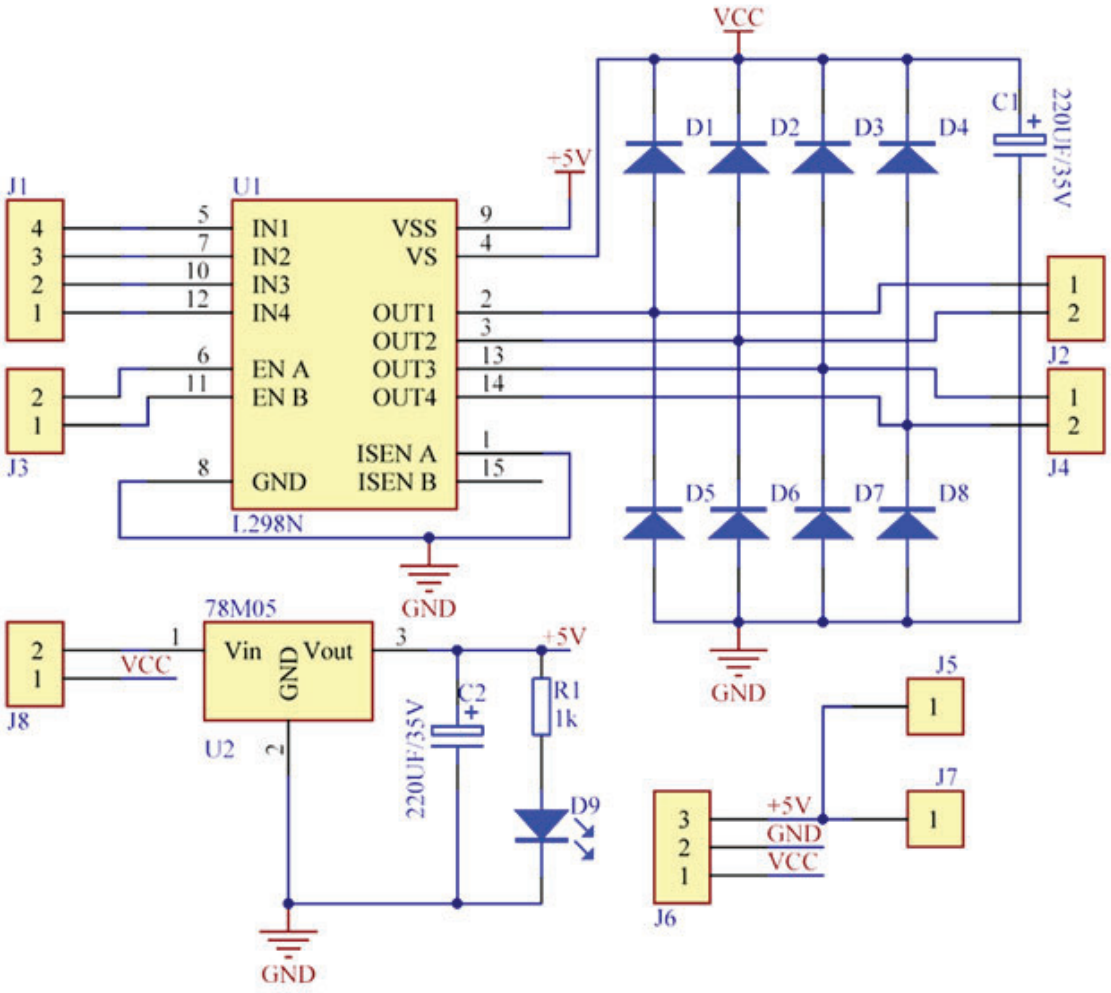


SKU: MDU-1049

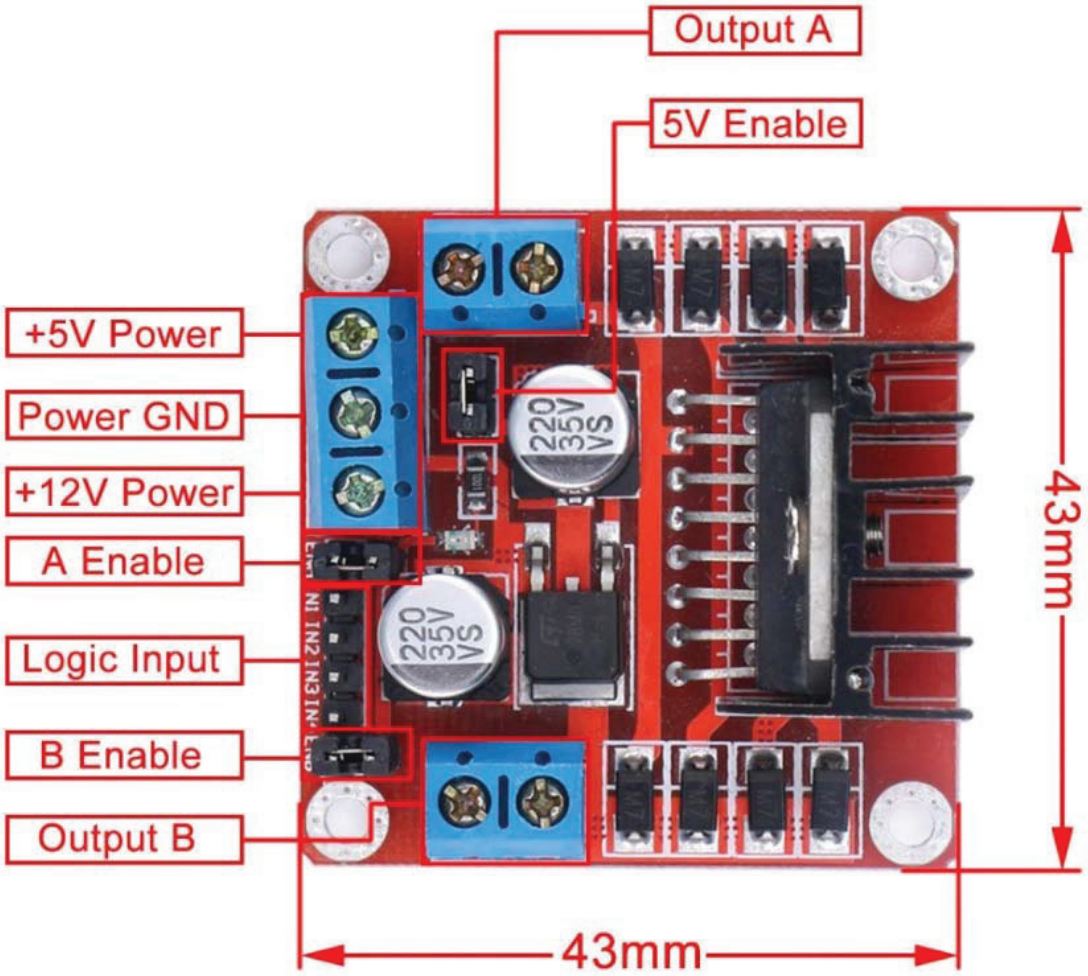
Brief Data:

- Input Voltage: 3.2V~40Vdc.
- Driver: L298N Dual H Bridge DC Motor Driver
- Power Supply: DC 5 V - 35 V
- Peak current: 2 Amp
- Operating current range: 0 ~ 36mA
- Control signal input voltage range :
- Low: -0.3V ⩽ Vin ⩽ 1.5V.
- High: 2.3V ⩽ Vin ⩽ Vss.
- Enable signal input voltage range :
    - Low: -0.3 ⩽ Vin ⩽ 1.5V (control signal is invalid).
    - High: 2.3V ⩽ Vin ⩽ Vss (control signal active).
- Maximum power consumption: 20W (when the temperature T = 75 ℃).
- Storage temperature: -25 ℃ ~ +130 ℃.
- On-board +5V regulated Output supply (supply to controller board i.e. Arduino).
- Size: 3.4cm x 4.3cm x 2.7cm

Schematic Diagram:



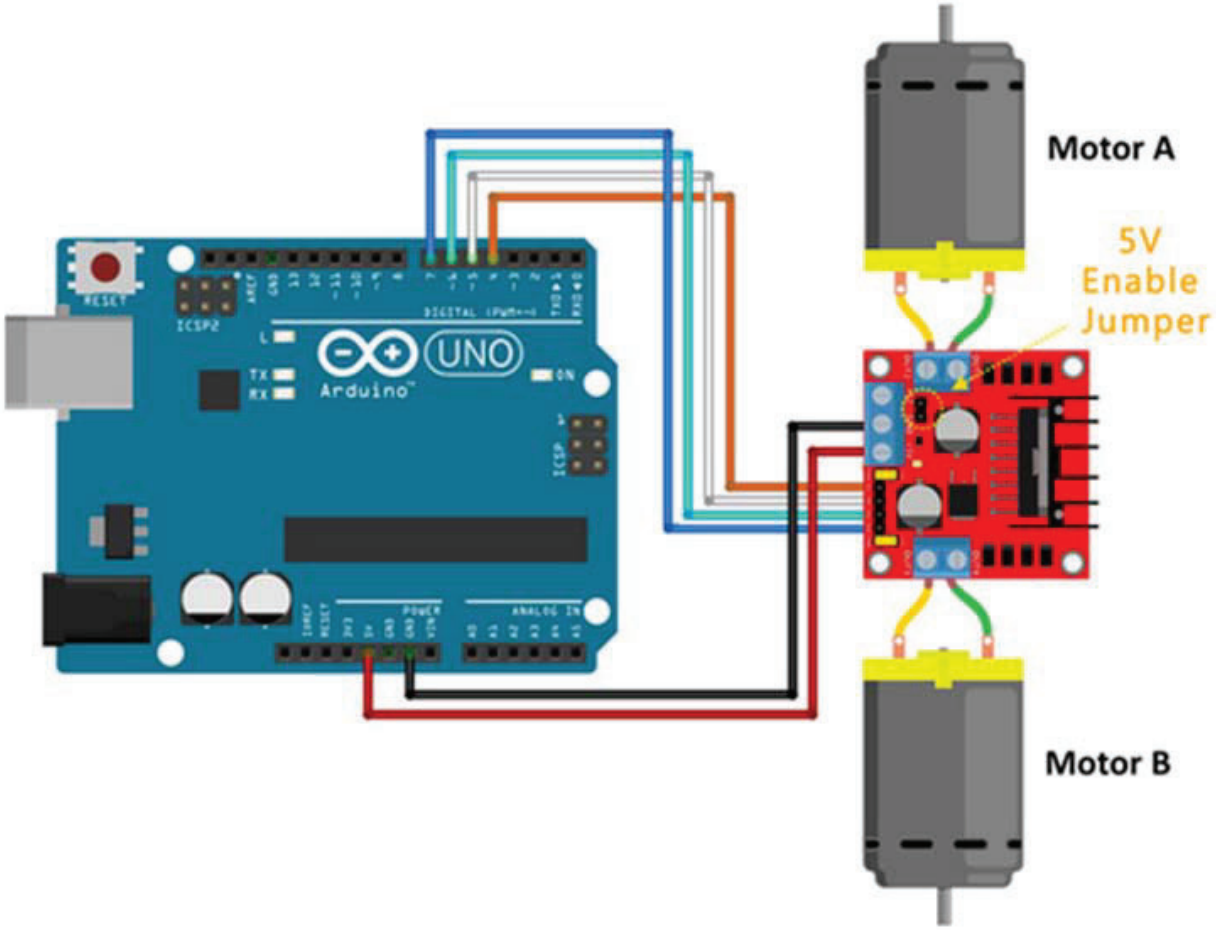
Board Dimension & Pins Function:



Connection Examples:

Controlling 2-DC Motor with +5V Arduino onboard Power Supply:

Below is the circuit connection use the on-board +5V power supply from Arduino board, and should be done without the 5V Enable Jumper on (Active 5V). This connection can drive two 5V DC motors simultaneously.



Sketch Listing:

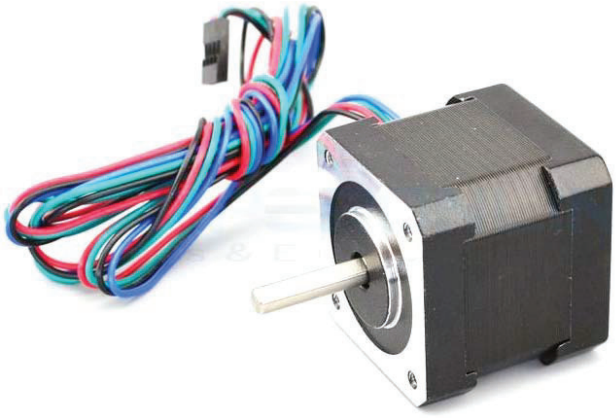
Copy and paste the sketch below to Arduino IDE and upload to Arduino Uno/Mega board.

```
/*=====================================================================
//  Author      : Handson Technology
//  Project     : Arduino Uno
//  Description : L298N Motor Driver
//  Source-Code : L298N_Motor.ino
//  Program: Control 2 DC motors using L298N H Bridge Driver
//=====================================================================
*/

// Definitions Arduino pins connected to input H Bridge
int IN1 = 4;
int IN2 = 5;
int IN3 = 6;
int IN4 = 7;

void setup()
{
 // Set the output pins
```

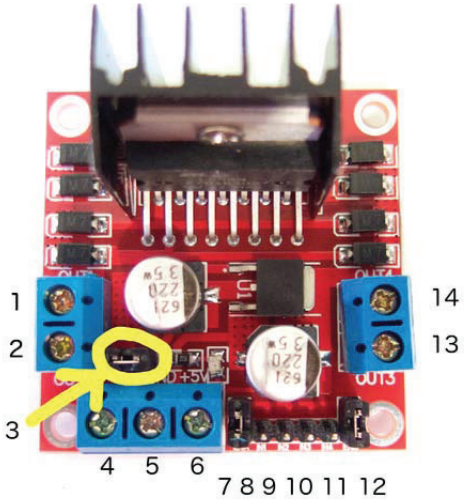
```
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);
}

void loop()
{
 // Rotate the Motor A clockwise
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
 delay(2000);
 // Motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);

 // Rotate the Motor B clockwise
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);
 delay(2000);
 // Motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);

 // Rotates the Motor A counter-clockwise
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
 delay(2000);
 // Motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);

 // Rotates the Motor B counter-clockwise
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, HIGH);
 delay(2000);
 // Motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);
}
```

Controlling Stepper Motor

In this example we have a typical NEMA-17 stepper motor with four wires:



The key to successful stepper motor control is identifying the wires - that is which one is which. You will need to determine the A+, A-, B+ and B- wires. With our example motor these are red, green, yellow and blue. Now let's get the wiring done.



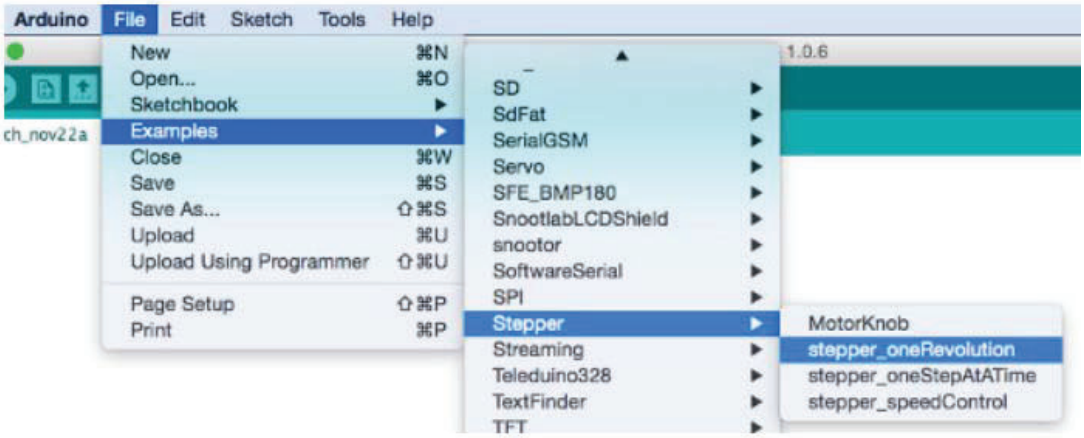
Connect the A+, A-, B+ and B- wires from the stepper motor to the module connections 1, 2, 13 and 14 respectively. Place the jumpers included with the L298N module over the pairs at module points 7 and 12. Then connect the power supply as required to points 4 (positive) and 5 (negative/GND).

Once again if your stepper motor's power supply is less than 12V, fit the jumper to the module at point 3 which gives you a neat 5V power supply for your Arduino.

Next, connect L298N module pins IN1, IN2, IN3 and IN4 to Arduino digital pins D8, D9, D10 and D11 respectively. Finally, connect Arduino GND to point 5 on the module, and Arduino 5V to point 6 if sourcing 5V from the module.

Controlling the stepper motor from your sketches is very simple, thanks to the *Stepper* Arduino library included with the Arduino IDE as standard.

To demonstrate your motor, simply load the "*stepper_oneRevolution*" sketch that is included with the *Stepper* library, for example:



Finally, check the value for

```
const int stepsPerRevolution = 200;
```

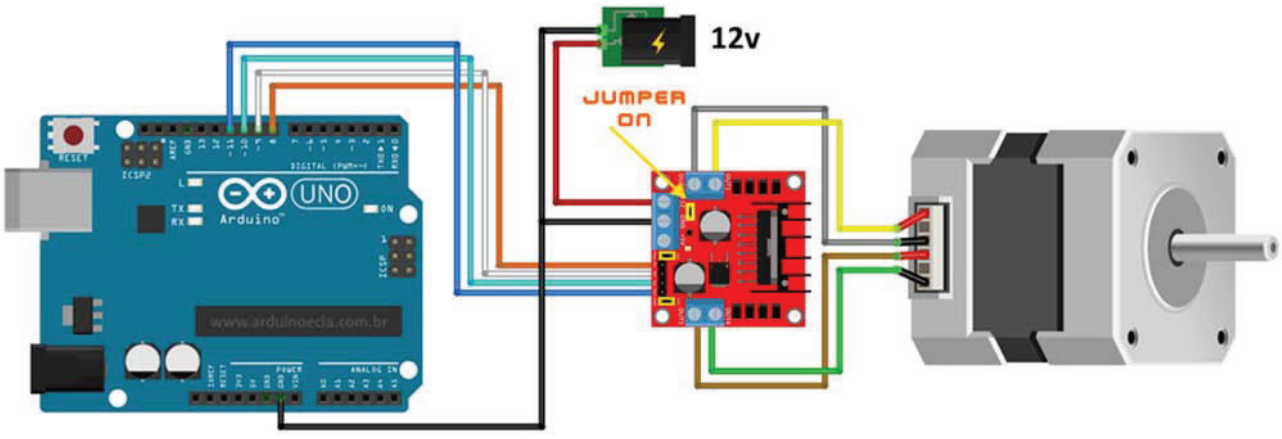
in the sketch and change the 200 to the number of steps per revolution for your stepper motor, and also the speed which is preset to 60 RPM in the following line:

```
myStepper.setSpeed(60);
```

Now you can save and upload the sketch, which will send your stepper motor around one revolution, then back again. This is achieved with the function

```
myStepper.step(stepsPerRevolution); // for clockwise
myStepper.step(-stepsPerRevolution); // for anti-clockwise
```

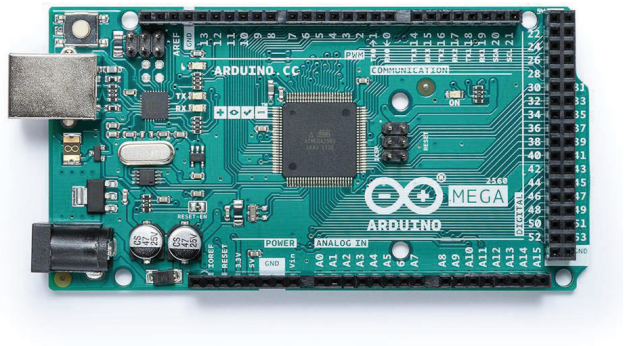
Connection for the sketch "*stepper_oneRevolution*":



Web Resources:

ANEXO VII:
Hoja de datos del microcontrolador Arduino Mega 2560

Product Reference Manual
SKU: A000067



Description

Arduino® Mega 2560 is an exemplary development board dedicated for building extensive applications as compared to other maker boards by Arduino. The board accommodates the ATmega2560 microcontroller, which operates at a frequency of 16 MHz. The board contains 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a USB connection, a power jack, an ICSP header, and a reset button.

Target Areas

3D Printing, Robotics, Maker

Features

- **ATmega2560 Processor**
  - Up to 16 MIPS Throughput at 16MHz
  - 256k bytes (of which 8k is used for the bootloader)
  - 4k bytes EEPROM
  - 8k bytes Internal SRAM
  - 32 × 8 General Purpose Working Registers
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Four Programmable Serial USART
  - Controller/Peripheral SPI Serial Interface
- **ATmega16U2**
  - Up to 16 MIPS Throughput at 16 MHz
  - 16k bytes ISP Flash Memory
  - 512 bytes EEPROM
  - 512 bytes SRAM
  - USART with SPI master only mode and hardware flow control (RTS/CTS)
  - Master/Slave SPI Serial Interface
- **Sleep Modes**
  - Idle
  - ADC Noise Reduction
  - Power-save
  - Power-down
  - Standby
  - Extended Standby
- **Power**
  - USB Connection
  - External AC/DC Adapter
- **I/O**
  - 54 Digital
  - 16 Analog
  - 15 PWM Output

Contents

1 The Board **4**
1.1 Application Examples 4
1.2 Accessories 4
1.3 Related Products 4
2 Ratings **5**
2.1 Recommended Operating Conditions 5
2.2 Power Consumption 5
3 Functional Overview **5**
3.1 Block Diagram 5
3.2 Board Topology 6
3.3 Processor 7
3.4 Power Tree 7
4 Board Operation **8**
4.1 Getting Started - IDE 8
4.2 Getting Started - Arduino Web Editor 8
4.3 Sample Sketches 8
4.4 Online Resources 8
5 Connector Pinouts **8**
5.1 Analog 10
5.2 Digital 10
5.3 ATMEGA16U2 JP5 12
5.4 ATMEGA16U2 ICSP1 12
5.5 Digital Pins D22 - D53 LHS 12
5.6 Digital Pins D22 - D53 RHS 13
6 Mechanical Information **13**
6.1 Board Outline 13
6.2 Board Mount Holes 14
7 Declaration of Conformity CE DoC (EU) **14**
8 Declaration of Conformity to EU RoHS & REACH 211 01/19/2021 **3**
9 Conflict Minerals Declaration **16**
10 FCC Caution **16**
11 Company Information **17**
12 Reference Documentation **17**
13 Revision History **17**

1 The Board

Arduino® Mega 2560 is a successor board of Arduino Mega, it is dedicated to applications and projects that require large number of input output pins and the use cases which need high processing power. The Arduino® Mega 2560 comes with a much larger set of IOs when we compare it with traditional Uno board considering the form factor of both the boards.

1.1 Application Examples

- **Robotics**: Featuring the high processing capacitity, the Arduino Mega 2560 can handle the extensive robotic applications. It is compatible with the motor controller shield that enables it to control multiple motors at an instance, thus making it perfect of robotic applications. The large number of I/O pins can accommodate many robotic sensors as well.
- **3D Printing**: Algorithms play a significant role in implementation of 3D printers. Arduino Mega 2560 has the power to process these complex algorithms required for 3D printing. Additionally, the slight changes to the code is easily possible with the Arduino IDE and thus 3D printing programs can be customized according to user requirements.
- **Wi-Fi**: Integrating wireless functionality enhances the utility of the applications. Arduino Mega 2560 is compatible with WiFi shields hence allowing the wireless features for the applications in 3D printing and Robotics.

1.2 Accessories

1.3 Related Products

- Arduino® Uno Rev 3
- Arduino® Nano
- Arduino® DUE without headers

2 Ratings

2.1 Recommended Operating Conditions

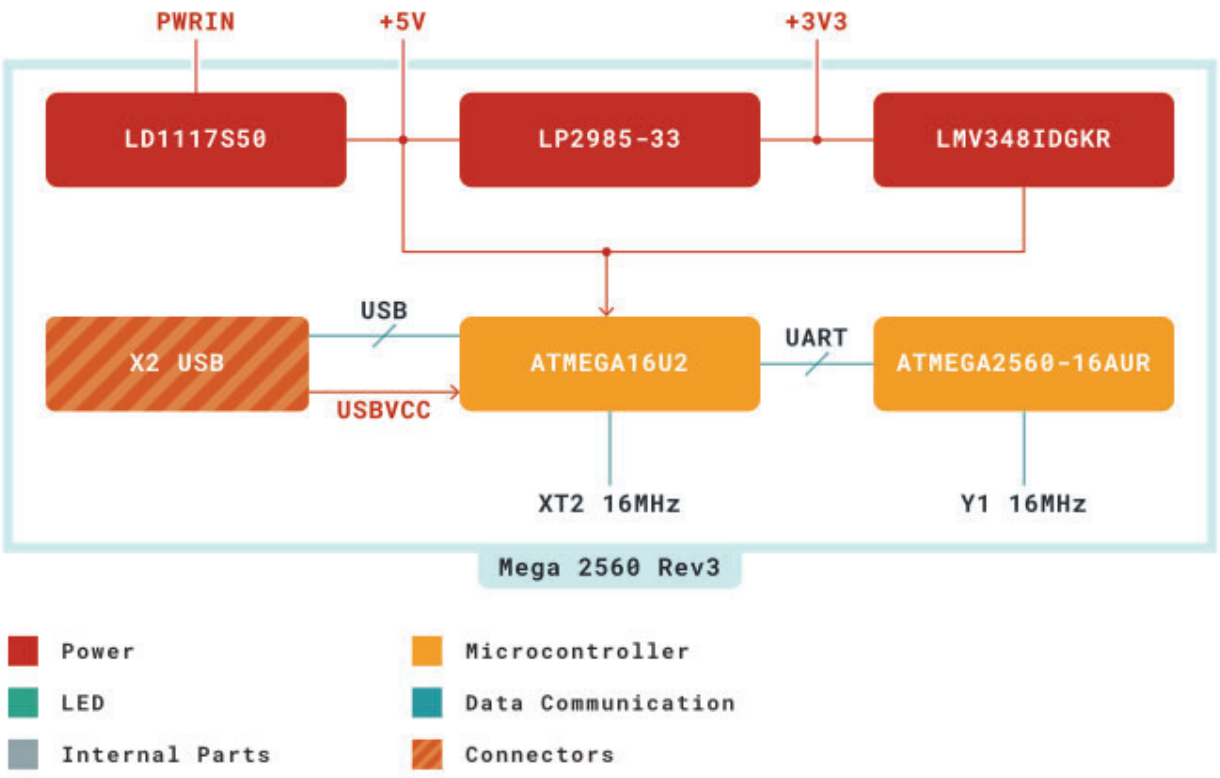
Symbol	Description	Min	Max
TOP	Operating temperature:	-40 °C	85 °C

2.2 Power Consumption

Symbol	Description	Min	Typ	Max	Unit
PWRIN	Input supply from power jack		TBC		mW
USB VCC	Input supply from USB		TBC		mW
VIN	Input from VIN pad		TBC		mW

3 Functional Overview

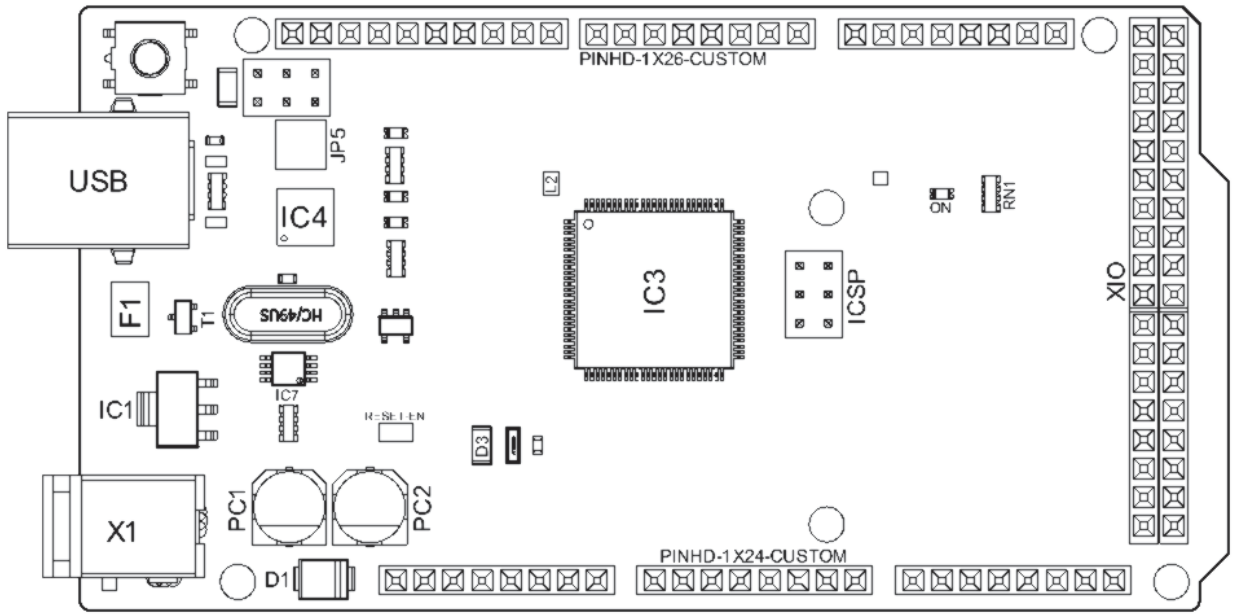
3.1 Block Diagram



Arduino MEGA Block Diagram

3.2 Board Topology

Front View



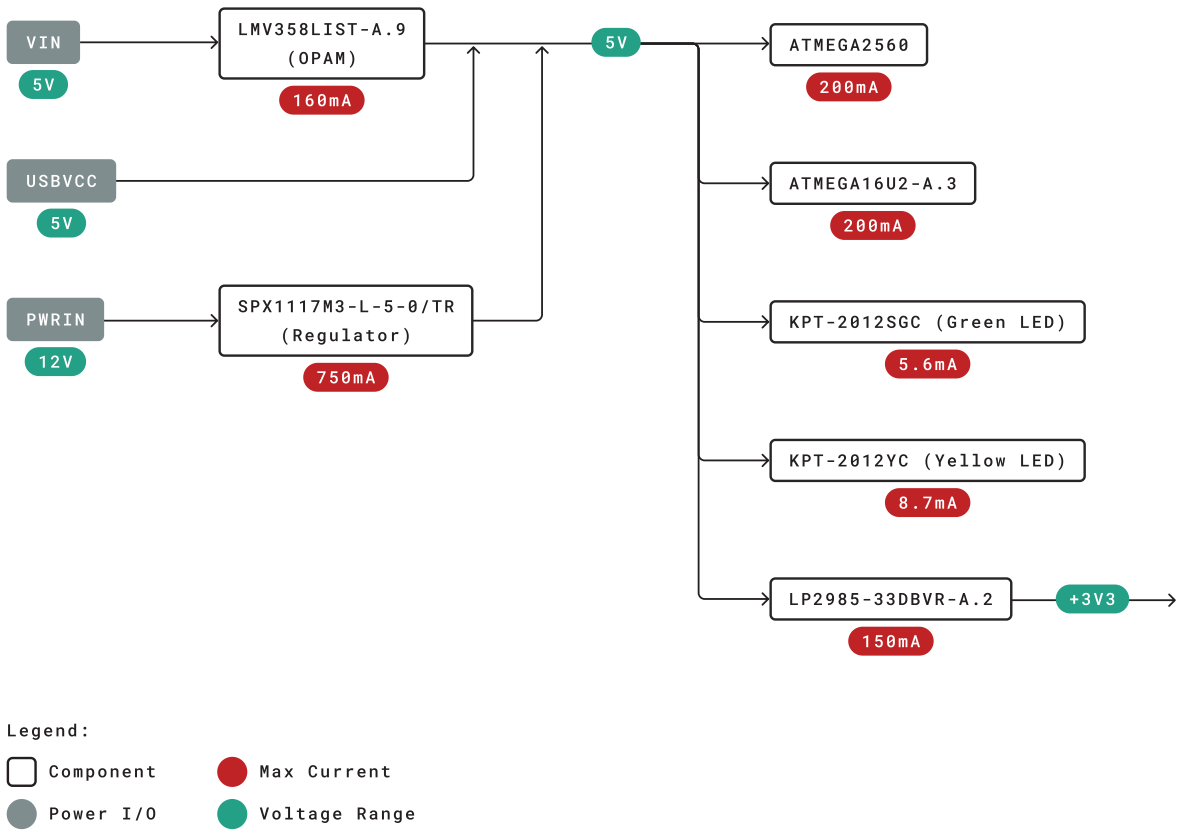
Arduino MEGA Top View

Ref.	Description	Ref.	Description
USB	USB B Connector	F1	Chip Capacitor
IC1	5V Linear Regulator	X1	Power Jack Connector
JP5	Plated Holes	IC4	ATmega16U2 chip
PC1	Electrolytic Alumninum Capacitor	PC2	Electrolytic Alumninum Capacitor
D1	General Purpose Rectifier	D3	General Purpose Diode
L2	Fixed Inductor	IC3	ATmega2560 chip
ICSP	Connector Header	ON	Green LED
RN1	Resistor Array	XIO	Connector

3.3 Processor

Primary processor of Arduino Mega 2560 Rev3 board is ATmega2560 chip which operates at a frequency of 16 MHz. It accommodates a large number of input and output lines which gives the provision of interfacing many external devices. At the same time the operations and processing is not slowed due to its significantly larger RAM than the other processors. The board also features a USB serial processor ATmega16U2 which acts an interface between the USB input signals and the main processor. This increases the flexibility of interfacing and connecting peripherals to the Arduino Mega 2560 Rev 3 board.

3.4 Power Tree



Power Tree

4 Board Operation

4.1 Getting Started - IDE

If you want to program your Arduino® MEGA 2560 while offline you need to install the Arduino® Desktop IDE **[1]** To connect the Arduino® MEGA 2560 to your computer, you'll need a Type-B USB cable. This also provides power to the board, as indicated by the LED.

4.2 Getting Started - Arduino Web Editor

All Arduino® boards, including this one, work out-of-the-box on the Arduino® Web Editor **[2]**, by just installing a simple plugin.

The Arduino® Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow **[3]** to start coding on the browser and upload your sketches onto your board.

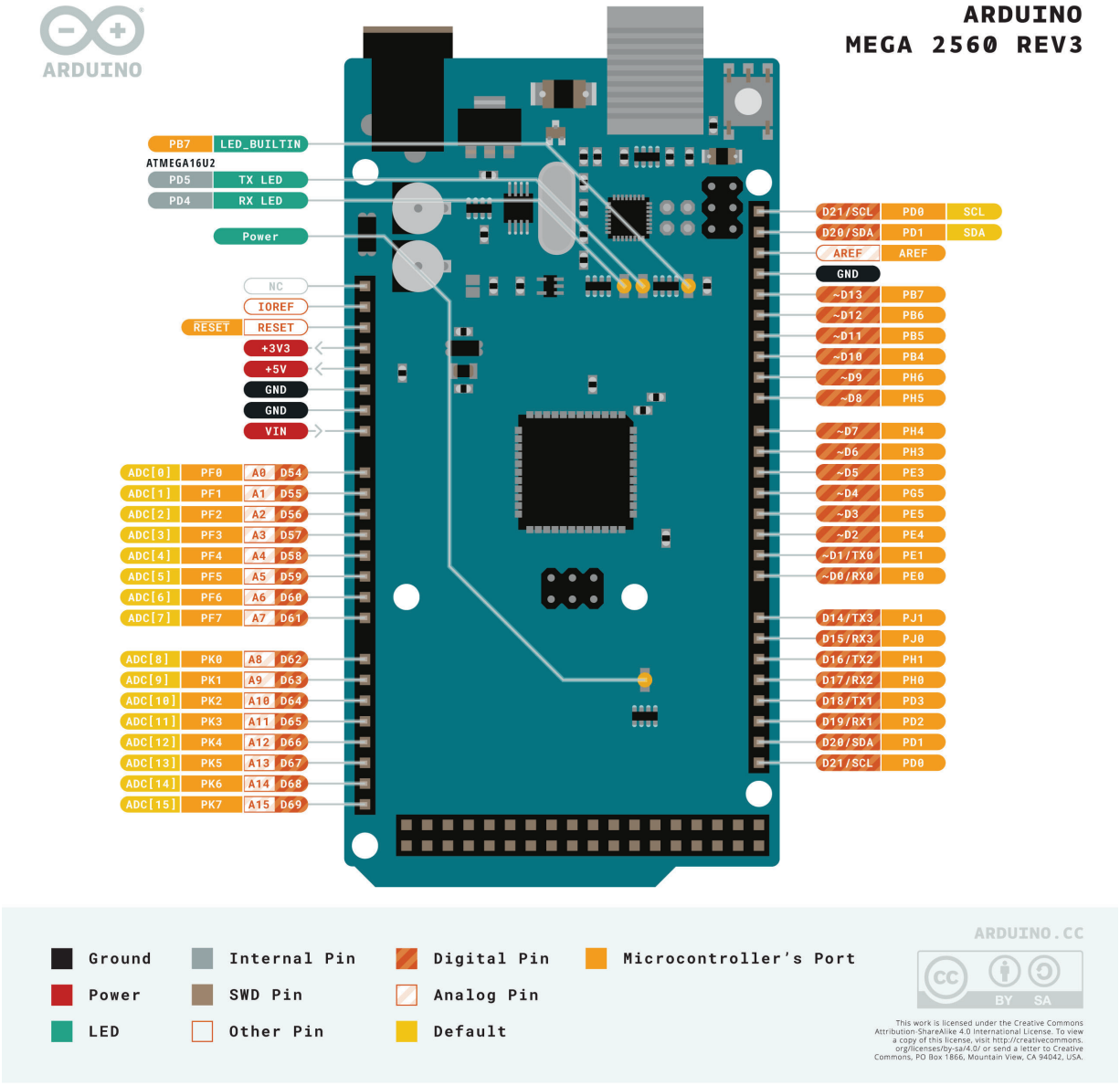
4.3 Sample Sketches

Sample sketches for the Arduino® MEGA 2560 can be found either in the "Examples" menu in the Arduino® IDE

4.4 Online Resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on ProjectHub **[5]**, the Arduino® Library Reference **[6]** and the online store **[7]** where you will be able to complement your board with sensors, actuators and more.

5 Connector Pinouts



Arduino Mega Pinout

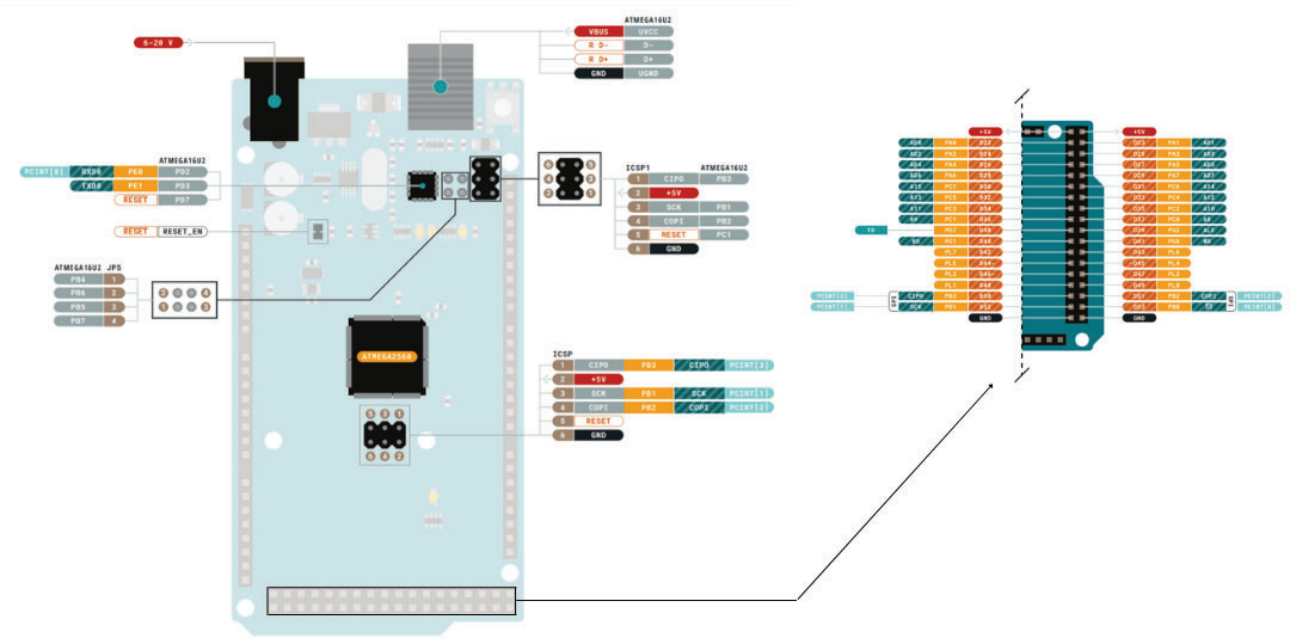
5.1 Analog

Pin	Function	Type	Description
1	NC	NC	Not Connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog	Analog input 0 /GPIO
10	A1	Analog	Analog input 1 /GPIO
11	A2	Analog	Analog input 2 /GPIO
12	A3	Analog	Analog input 3 /GPIO
13	A4	Analog	Analog input 4 /GPIO
14	A5	Analog	Analog input 5 /GPIO
15	A6	Analog	Analog input 6 /GPIO
16	A7	Analog	Analog input 7 /GPIO
17	A8	Analog	Analog input 8 /GPIO
18	A9	Analog	Analog input 9 /GPIO
19	A10	Analog	Analog input 10 /GPIO
20	A11	Analog	Analog input 11 /GPIO
21	A12	Analog	Analog input 12 /GPIO
22	A13	Analog	Analog input 13 /GPIO
23	A14	Analog	Analog input 14 /GPIO
24	A15	Analog	Analog input 15 /GPIO

5.2 Digital

Pin	Function	Type	Description
1	D21/SCL	Digital Input/I2C	Digital input 21/I2C Dataline
2	D20/SDA	Digital Input/I2C	Digital input 20/I2C Dataline
3	AREF	Digital	Analog Reference Voltage
4	GND	Power	Ground
5	D13	Digital/GPIO	Digital input 13/GPIO
6	D12	Digital/GPIO	Digital input 12/GPIO
7	D11	Digital/GPIO	Digital input 11/GPIO
8	D10	Digital/GPIO	Digital input 10/GPIO
9	D9	Digital/GPIO	Digital input 9/GPIO
10	D8	Digital/GPIO	Digital input 8/GPIO
11	D7	Digital/GPIO	Digital input 7/GPIO
12	D6	Digital/GPIO	Digital input 6/GPIO
13	D5	Digital/GPIO	Digital input 5/GPIO
14	D4	Digital/GPIO	Digital input 4/GPIO

Pin	Function	Type	Description
15	D3	Digital/GPIO	Digital input 3/GPIO
16	D2	Digital/GPIO	Digital input 2/GPIO
17	D1/TX0	Digital/GPIO	Digital input 1 /GPIO
18	D0/Tx1	Digital/GPIO	Digital input 0 /GPIO
19	D14	Digital/GPIO	Digital input 14 /GPIO
20	D15	Digital/GPIO	Digital input 15 /GPIO
21	D16	Digital/GPIO	Digital input 16 /GPIO
22	D17	Digital/GPIO	Digital input 17 /GPIO
23	D18	Digital/GPIO	Digital input 18 /GPIO
24	D19	Digital/GPIO	Digital input 19 /GPIO
25	D20	Digital/GPIO	Digital input 20 /GPIO
26	D21	Digital/GPIO	Digital input 21 /GPIO



Arduino Mega Pinout

5.3 ATMEGA16U2 JP5

Pin	Function	Type	Description
1	PB4	Internal	Serial Wire Debug
2	PB6	Internal	Serial Wire Debug
3	PB5	Internal	Serial Wire Debug
4	PB7	Internal	Serial Wire Debug

5.4 ATMEGA16U2 ICSP1

Pin	Function	Type	Description
1	CIPO	Internal	Controller In Peripheral Out
2	+5V	Internal	Power Supply of 5V
3	SCK	Internal	Serial Clock
4	COPI	Internal	Controller Out Peripheral In
5	RESET	Internal	Reset
6	GND	Internal	Ground

5.5 Digital Pins D22 - D53 LHS

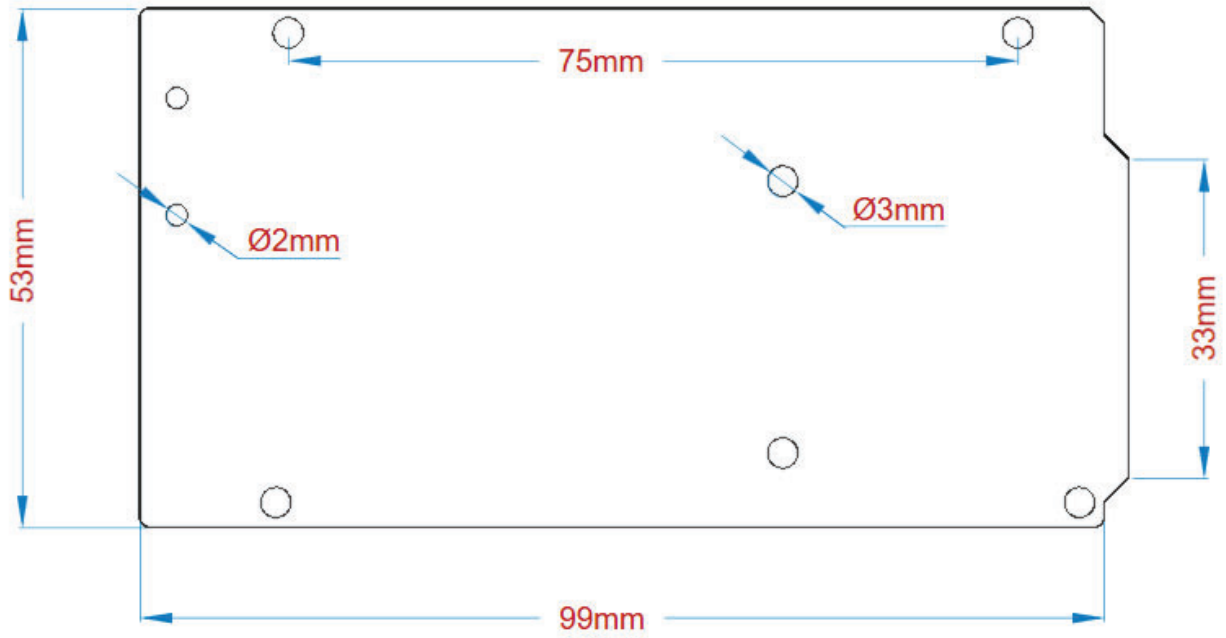
Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D22	Digital	Digital input 22/GPIO
3	D24	Digital	Digital input 24/GPIO
4	D26	Digital	Digital input 26/GPIO
5	D28	Digital	Digital input 28/GPIO
6	D30	Digital	Digital input 30/GPIO
7	D32	Digital	Digital input 32/GPIO
8	D34	Digital	Digital input 34/GPIO
9	D36	Digital	Digital input 36/GPIO
10	D38	Digital	Digital input 38/GPIO
11	D40	Digital	Digital input 40/GPIO
12	D42	Digital	Digital input 42/GPIO
13	D44	Digital	Digital input 44/GPIO
14	D46	Digital	Digital input 46/GPIO
15	D48	Digital	Digital input 48/GPIO
16	D50	Digital	Digital input 50/GPIO
17	D52	Digital	Digital input 52/GPIO
18	GND	Power	Ground

5.6 Digital Pins D22 - D53 RHS

Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D23	Digital	Digital input 23/GPIO
3	D25	Digital	Digital input 25/GPIO
4	D27	Digital	Digital input 27/GPIO
5	D29	Digital	Digital input 29/GPIO
6	D31	Digital	Digital input 31/GPIO
7	D33	Digital	Digital input 33/GPIO
8	D35	Digital	Digital input 35/GPIO
9	D37	Digital	Digital input 37/GPIO
10	D39	Digital	Digital input 39/GPIO
11	D41	Digital	Digital input 41/GPIO
12	D43	Digital	Digital input 43/GPIO
13	D45	Digital	Digital input 45/GPIO
14	D47	Digital	Digital input 47/GPIO
15	D49	Digital	Digital input 49/GPIO
16	D51	Digital	Digital input 51/GPIO
17	D53	Digital	Digital input 53/GPIO
18	GND	Power	Ground

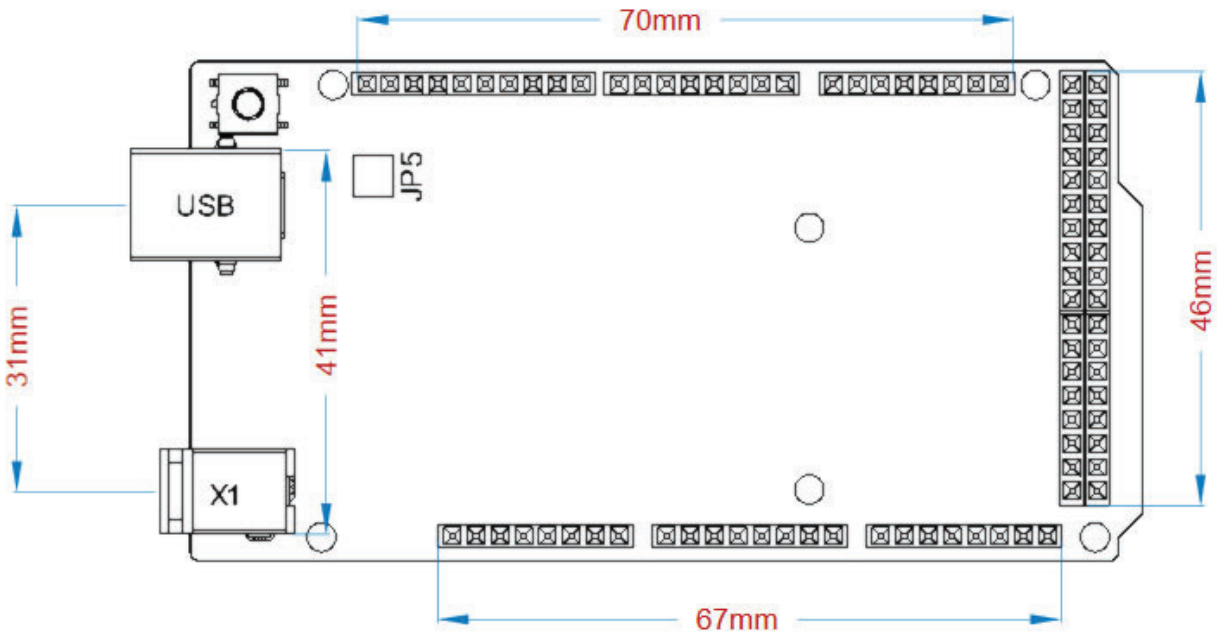
6 Mechanical Information

6.1 Board Outline



Arduino Mega Outline

6.2 Board Mount Holes



Arduino Mega Mount Holes

Certifications

7 Declaration of Conformity CE DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

8 Declaration of Conformity to EU RoHS & REACH 211 01/19/2021

Arduino boards are in compliance with RoHS 2 Directive 2011/65/EU of the European Parliament and RoHS 3 Directive 2015/863/EU of the Council of 4 June 2015 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Substance	Maximum Limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000
Bis(2-Ethylhexyl} phthalate (DEHP)	1000
Benzyl butyl phthalate (BBP)	1000
Dibutyl phthalate (DBP)	1000
Diisobutyl phthalate (DIBP)	1000

Exemptions : No exemptions are claimed.

Arduino Boards are fully compliant with the related requirements of European Union Regulation (EC) 1907 /2006 concerning the Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). We declare none of the SVHCs (https://echa.europa.eu/web/guest/candidate-list-table), the Candidate List of Substances of Very High Concern for authorization currently released by ECHA, is present in all products (and also package) in quantities totaling in a concentration equal or above 0.1%. To the best of our knowledge, we also declare that our products do not contain any of the substances listed on the "Authorization List" (Annex XIV of the REACH regulations) and Substances of Very High Concern (SVHC) in any significant amounts as specified by the Annex XVII of Candidate list published by ECHA (European Chemical Agency) 1907 /2006/EC.

9 Conflict Minerals Declaration

As a global supplier of electronic and electrical components, Arduino is aware of our obligations with regards to laws and regulations regarding Conflict Minerals, specifically the Dodd-Frank Wall Street Reform and Consumer Protection Act, Section 1502. Arduino does not directly source or process conflict minerals such as Tin, Tantalum, Tungsten, or Gold. Conflict minerals are contained in our products in the form of solder, or as a component in metal alloys. As part of our reasonable due diligence Arduino has contacted component suppliers within our supply chain to verify their continued compliance with the regulations. Based on the information received thus far we declare that our products contain Conflict Minerals sourced from conflict-free areas.

10 FCC Caution

Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference

(2) this device must accept any interference received, including interference that may cause undesired operation.

FCC RF Radiation Exposure Statement:

1. This Transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.
2. This equipment complies with RF radiation exposure limits set forth for an uncontrolled environment.
3. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

English: User manuals for licence-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both. This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions:

(1) this device may not cause interference

(2) this device must accept any interference, including interference that may cause undesired operation of the device.

French: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes :

(1) l' appareil nedoit pas produire de brouillage

(2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

IC SAR Warning:

English This equipment should be installed and operated with minimum distance 20 cm between the radiator and your body.

French: Lors de l' installation et de l' exploitation de ce dispositif, la distance entre le radiateur et le corps est d 'au moins 20 cm.

Important: The operating temperature of the EUT can't exceed 85°C and shouldn't be lower than -40°C.

Hereby, Arduino S.r.l. declares that this product is in compliance with essential requirements and other relevant provisions of Directive 201453/EU. This product is allowed to be used in all EU member states.

11 Company Information

Company name	Arduino S.r.l.
Company Address	Arduino SRL, Via Andrea Appiani 25, 20900 Monza MB, Italy

12 Reference Documentation

Ref	Link
Arduino IDE (Desktop)	https://www.arduino.cc/en/Main/Software
Arduino IDE (Cloud)	https://create.arduino.cc/editor
Cloud IDE Getting Started	https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-4b3e4a
Arduino Pro Website	https://www.arduino.cc/pro
Project Hub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
Library Reference	https://www.arduino.cc/reference/en/libraries/
Online Store	https://store.arduino.cc/

13 Revision History

Date	Revision	Changes
29/09/2020	1	First Release

ANEXO VIII:
Hoja de datos del motor cc

No.131211

承　　認　　書

(SPECIFICATION FOR APPROVAL)

CUSTOMER： 肯創

MODEL： IG220019X00015R

DATE： Dec 13, 2013

SUPPLIER：

研發 2013.12.13 張慶隆

CUSTOMER：

肯創科技 許江海

祥儀企業股份有限公司
SHAYANG YE INDUSTRIAL CO.,LTD.

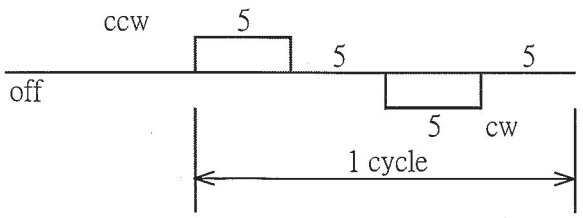
SHA YANG YE

桃園縣桃園市桃鶯路 461 號
TEL:(03)3623452
FAX:(03)3628404
http://www.shayye.com.tw

NO.461,Taoying Rd.,Taoyuan City,
Taoyuan County,Taiwan R.O.C.

E-MAIL:sales@shayye.com.tw

Technical Data

Part Number	IG220019X00015R	
Customer P/N		
ITEM	**Specifications**	**Note**
1. Operation Status		
1.1 Rated Voltage	12V D.C.	Stable power source
1.2 Rated torque	0.3 kgf.cm	6mm from shaft end
1.3 Radial load	8N (0.8kg-f)	
1.4 Axial load	6N (0.6kg-f)	
1.5 Turning direction	Shaft horizontal	
1.6 Reverse direction	CW.CCW	
1.7 Using environment	Temperature -10~60 ℃ Humidity 20~90% RH	
1.8 Preserve environment	Temperature -20~70 ℃ Humidity 20~90% RH	
1.9 Using voltage range	12V (D.C.) ±10%	
2.Electrical Characteristics		
2.1 No Load current	200 mA max.	
2.2 No Load speed	789 rpm ±15%	
2.3 Rated current	620 mA	
2.4 Rated speed	639 rpm ±15%	
2.5 Stall current	2.4 A	
2.6 Stall torque	2.2 kgf.cm	
2.7 Insulation	D.C. 100V meg. 1.0 MΩ min	Motor terminal shell
2.8 Durable voltage	100V (A.C.)， 1 minute min	Motor terminal shell
2.9 Coil resistance	5Ω	Reference
2.10 Torque constant	0.91 kgf.cm/A	Reference
2.11 Voltage constant	8.68 mV/r/min	Reference
3.Mechanical characteristic		
3.1 Reduction ratio	1/19.225	
3.2 Thrust play of shaft	0.2 mm max.	
3.3 Radial play of shaft	0.05 mm max.	
3.4 Back lash	3° max.	
3.5 Outside Appearance	No scratch defective....	By visual judgment
2.Life Cycle	72000 cycles min.  	After the rated life cycle test current @ rated load must stay within ±30% of the initial value and r.p.m. @ rated load must stay within ±20% of the initial value. However change of mechanical noise level was not considered as part of the testing

ASSEMBLY, MAINTENANCE, OPERATION

1. Install: To avoid internal geared motor touched by overlong screws and caused defective. Please check screw size and length on external dimension drawing when installing geared motor into construction.

2. Reprocess: Heavy impact and vibration during reprocessing output shaft may cause loose screws and lead to unbalance gear operation. Please avoid reprocessing output shaft.
Must to prevent overheat when weld wires into terminal and cause breakdown due to burnt internal geared motor parts.
Please do not overload the radial load limitation of output shaft when using belt pulley or chain pulley as power transmission. Please do not overload the axial load limitation of output shaft when pressing parts upon it as well.

3. Environment: The parts of geared motors or itself may corroded or damaged easier when using or maintaining in out of range environment. Must to pay close attention that gears may corroded even under an allowed environment in long term.

4. Impact: Must prevent geared motor from falling and impact, or the parts will get damaged, the screws will be loosed, and the gear operation will unbalance etc…

5. Locked out: Please well prepared current transmitting protection in case of burnt motor coil easy and damaged gear from locked out geared motor.

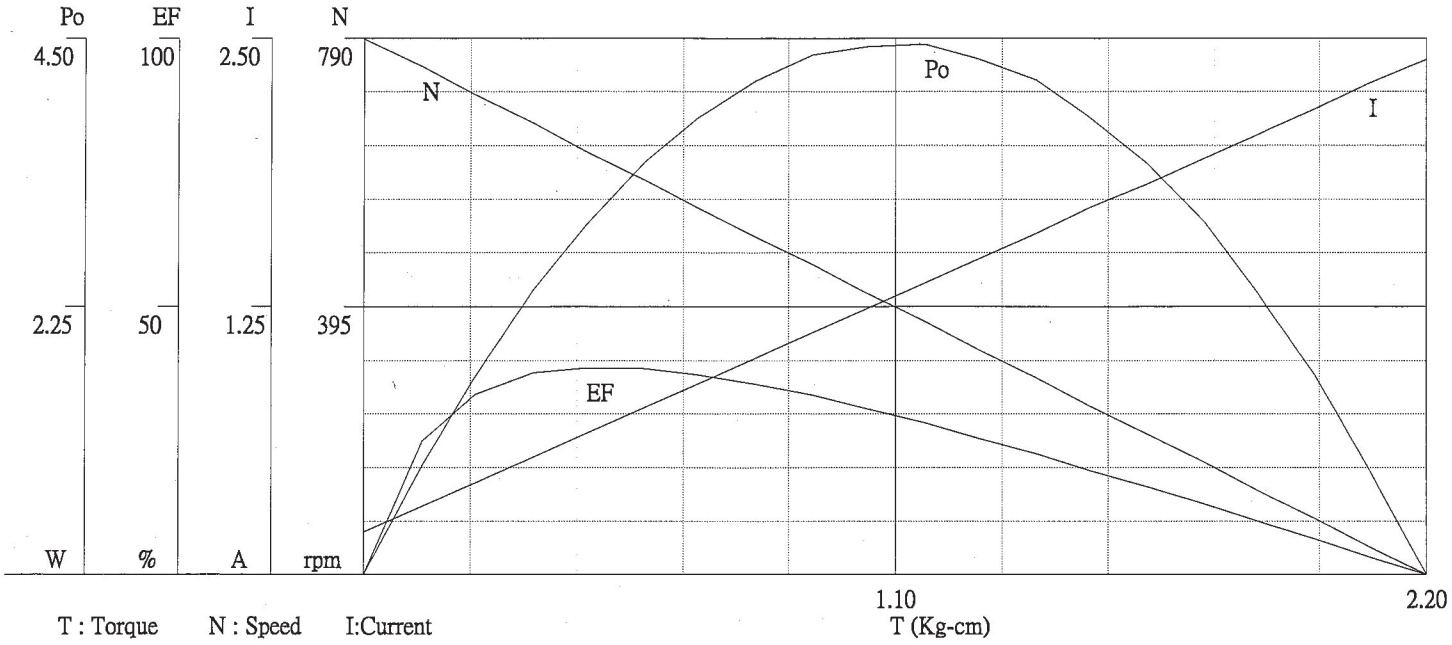
6. Output shaft Turning:
Please note that it is easier to damage gear when directly turning output shaft.

7. PWM controlling:
The graphite brush of motor will be abnormally wore out or the commutator interval will be blocked by carbon powder when using in the condition of D/T under 60%.
Moreover, please pay attention to the motor with capacitor due to there is ineffectual capacity cycle scope.

8. Momentary reverse:
The graphite brush will be abnormally wore out or coil getting aggravated when geared motor is reversed momentary.
Also, the commutator interval will be stuck if switch frequently.

SHAYANG YE INDUSTRIAL CO.,LTD.

Mode: IG220019X00015R S/N: 001 Voltage : 12 V Date: 20131213



T : Torque N : Speed I:Current

Po : Output Power EF : Efficieney

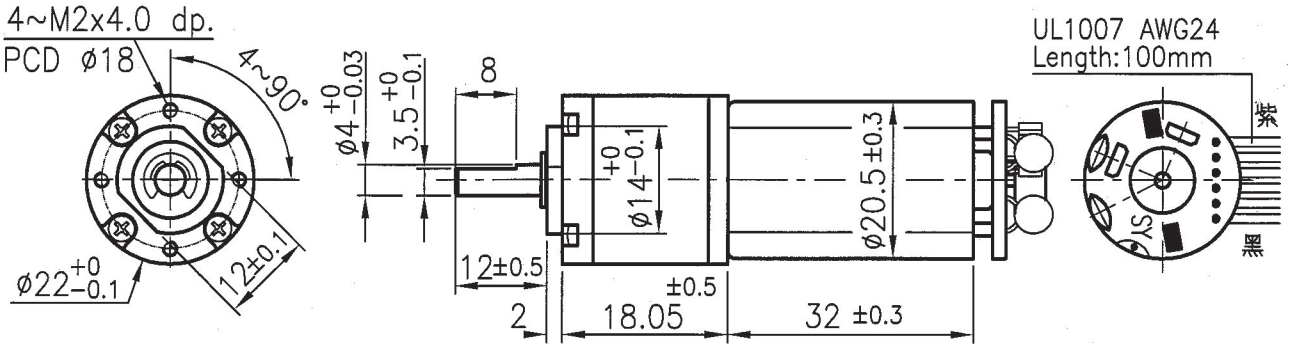
	T(N/m)	[Kg/cm]	N(rpm)	I(A)	Po(W)	EF(%)
No Load	0	0.00	789.00	.20	0	0
	.01	.12	748.21	.32	.92	25
	.03	.35	664.99	.55	2.39	37.73
	.06	.58	581.51	.78	3.46	38.49
	.07	.69	540.29	.89	3.83	37.32
	.08	.81	498.15	1.01	4.14	35.59
	.09	.93	457.23	1.13	4.36	33.55
	.1	1.04	415.47	1.24	4.43	31.05
	.12	1.27	332.34	1.47	4.33	25.56
	.14	1.39	290.84	1.59	4.15	22.65
	.15	1.50	249.35	1.71	3.84	19.5
	.16	1.62	207.75	1.82	3.45	16.48
	.17	1.74	165.96	1.94	2.96	13.25
	.18	1.85	124.67	2.05	2.37	10.03
	.2	2.08	41.56	2.29	.89	3.36
Stall	.22	2.20	0.00	2.40	0	0
Po...(max)	.11	1.16	373.66	1.36	4.45	28.37
EF...(max)	.05	.46	622.40	.66	2.94	38.62

參考 REF.	
估價 EVALUATE	
試作 INITAL	
製作 MASS	

Two Channel Encoder Connections :

JST PHR-6
P=2.0-6P

: -MOTOR
2. Red : +MOTOR
3. Brown : HALL SENSOR Vcc
4. Green : HALL SENSOR GND
5. Blue : HALL SENSOR A Vout
6. Purple : HALL SENSOR B Vout



REDUCTION RATIO: 1/19.22
OUTPUT SHAFT: STEEL
MOTOR SPECIFICATION: 12V 15000RPM
OUTPUT-789RPM/ ≦ 200mA WITH NO LOAD

核准 APPROVED BY	一般公差表	軸徑 -0/-1 孔徑 +1/0 長度，角度 ± 一							
		一 指定等級區分之數值公差							
		等級 \ 區分	1~6	6~18	18~50	50~120	120~250	250~500	500~1000	角度
		1級	0.1	0.2	0.3	0.4	0.5	0.6	0.8	1°
		2級	0.2	0.4	0.6	0.8	1.2	1.6	2.0	2°
		未標示孔徑距公差	±0.1				±0.2			

1	新出圖		
設變單號 ECN.		設變單號 ECN.	

核對 CHECKED BY	熱處理 HEAT TREATMENT		件名 PART NAME	GEARMOTOR	單位 UNITS mm
	表面處理 SURFACE		圖號 DWG.	IG220019X00015R	比例 SCALE 9/10
	材質 MATERIAL				
繪圖 DRAWIN BY	重量 WEIGHT	克(g)	頁次 SHEET	1/1	

祥儀企業股份有限公司
SHA YANG YE INDUSTRIAL CO., LTD.

IG22

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Digilent:
290-006

ANEXO IX:
Modelo de lectura del motor

Se ha diseñado un modelo en Simulink específico para lectura de la respuesta del motor. El aspecto general de este modelo es el siguiente:

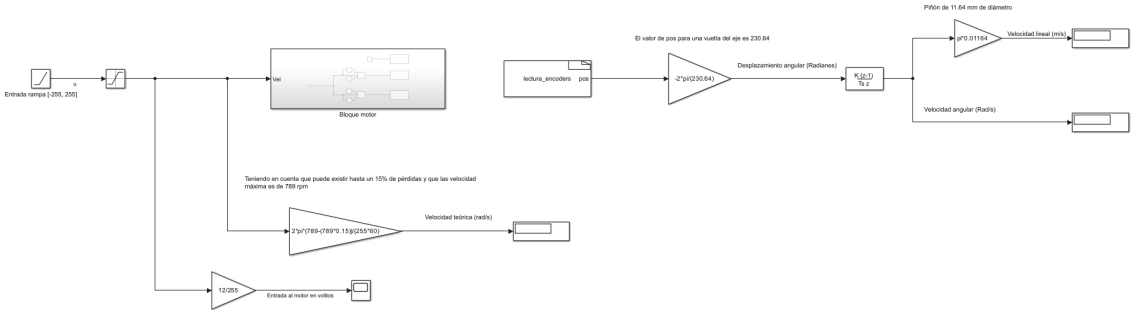


Figura 27: Vista general del modelo de lectura de motor

El propósito de este modelo es realizar pruebas con el motor variando la tensión de entrada. Se puede dividir en dos partes: una parte destinada al envío de los comandos de velocidad al motor y otra parte donde se obtiene la respuesta en posición y velocidad gracias al encoder.

Consta de dos subsistemas en el diseño: el bloque motor [Anexo] y la función *lectura_encoders* [Anexo].

- Sección de entrada

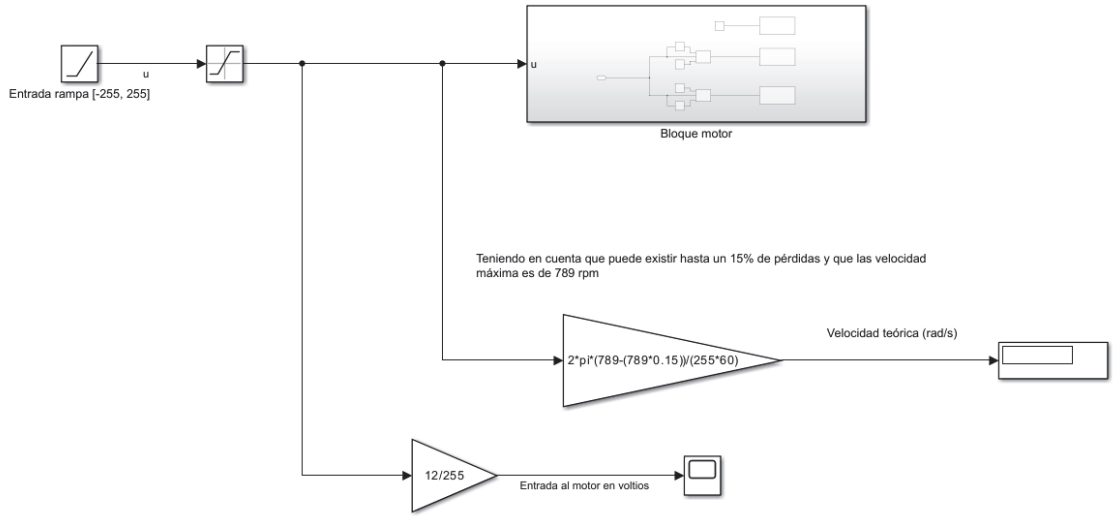


Figura 28: Vista general de la sección de entrada del modelo de lectura de motor

La sección de entrada aparece configurada para una entrada rampa de tensión codificada con saturación. Para obtener el valor de entrada suministrado en voltios, basta aplicar el factor de conversión voltios/tensión codificada 12/255. Sabiendo que existe una velocidad máxima teórica de 789 rpm, que esta se consigue con una entrada de tensión máxima (12V) y que pueden existir hasta un 15% de pérdidas, es posible obtener la velocidad teórica en radianes por segundo que debería alcanzar el motor en función de la tensión aportada y teniendo en cuenta un comportamiento lineal perfecto. Esto es lo que calcula la segunda ganancia.

- Sección de salida

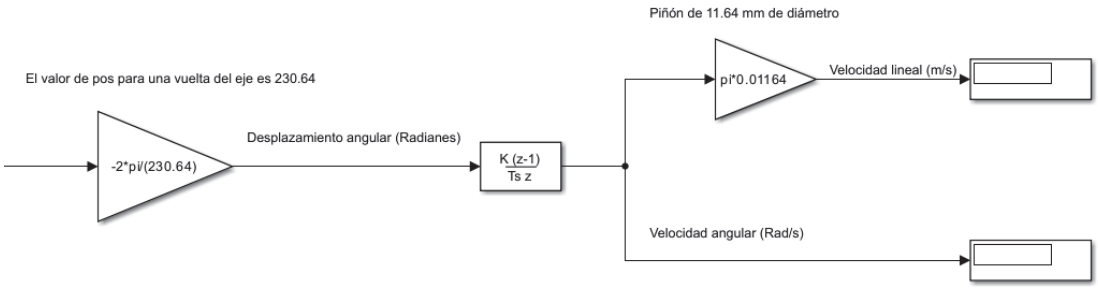


Figura 29: Vista general de la sección de salida del modelo de lectura de motor

Para obtener el desplazamiento angular en revoluciones del eje, basta con girar manualmente el imán del encoder una vuelta y multiplicar el valor de *pos* obtenido por el cociente de reducción (19.22/1). Para convertir el resultado en radianes, se multiplica por 2π. Esto es lo que realiza la primera ganancia a la salida de la función *Lectura_ encoders.*

La velocidad angular en radianes por segundo se obtiene aplicando la derivada discreta al desplazamiento angular en radianes; para ello se utiliza el bloque *discrete_derivative* de Simulink. Para convertir la velocidad angular en lineal, se multiplica por el perímetro del piñón que conecta el eje del motor con la cinta de la plataforma, de 11.64 mm de diámetro. De ello se encarga la segunda ganancia.

ANEXO X:
Bloque motor

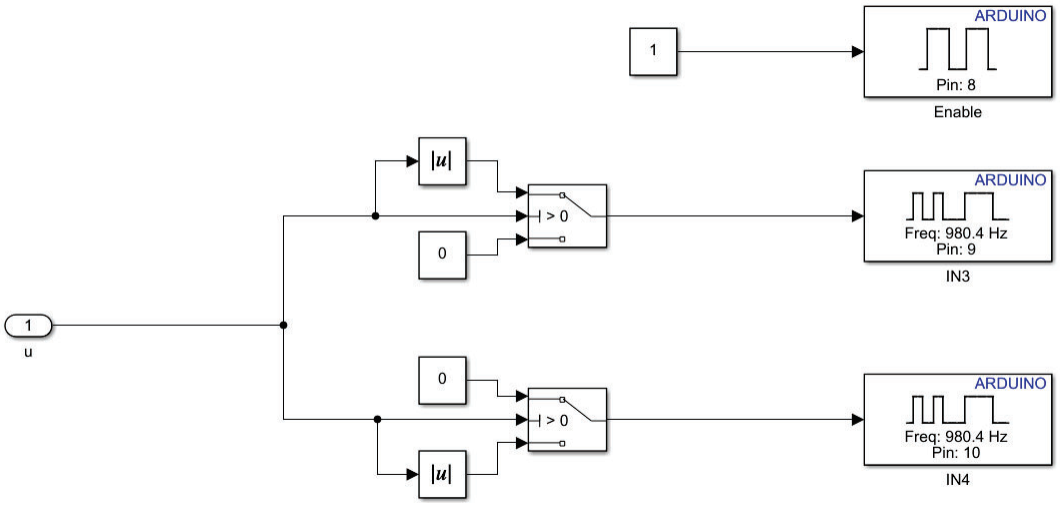


Figura 30: Vista general del bloque motor

Consta de una entrada de tensión codificada u de 9 bits (8 bits para y 1 bit de signo) suministrada en formato decimal. Las salidas son dos valores enteros entre 0 y 255 que controlan el Duty Cycle de dos señales PWM emitidas por dos salidas distintas de la placa Arduino: las salidas digitales 9 y 10. Para ello se han usado sendos bloques de generación de señales PWM asociados a cada salida, utilizando para ello la librería *Simulink Support Package for Arduino Hardware*. Las salidas 9 y 10 se conectan respectivamente a las entradas del driver de potencia IN3 y IN4 y dependiendo de la señal activa en cada momento y de su Duty Cycle, se controla el sentido de giro y la velocidad del motor B. La activación de estas señales nunca es simultánea, como ya se ha explicado. Para decidir qué salida emite la señal PWM de valor medio no nulo en cada momento, se ha ideado un sistema de switches. Si la entrada u es de signo positivo (se busca el giro antihorario del eje del motor, luego la señal PWM de valor medio no nulo ha de emitirse por la salida 9), los switches detectan que es mayor que 0. El switch asociado a la salida 9 proporciona el valor absoluto de u al bloque de generación PWM para la salida 9, mientras que el switch asociado a la salida 10 proporciona valor nulo

(Duty Cycle 0%, señal PWM de valor medio nulo). De forma análoga se consigue la generación de la señal PWM para la salida 10.

Por último, la salida digital 8 se encuentra siempre establecida a un valor lógico alto, ya que se encuentra conectada a ENB (habilitador del motor B) del driver de potencia.

ANEXO XI:
Función *lectura_encoders*

Se trata de una función predefinida proporcionada por MatLab que ha sido modificada para adaptarla al encoder y a las entradas y salidas utilizadas.

```
/* Includes_BEGIN */
#include <math.h>
/* Includes_END */

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


/*
 * 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 */
# ifndef MATLAB_MEX_FILE

# include <Arduino.h>

typedef struct { int pinA; int pinB; long pos; } Encoder;
volatile Encoder Enc[1] = {{2,3,0}};

/* auxiliary function to handle encoder attachment         */
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;
```

```
47    default:
48      return -1;
49    }
50  }
51
52
53  /* Interrupt Service Routine: change on pin A for Encoder 0  */
54  void irsPinAEn0(){
55
56    /* read pin B right away                          */
57    int drB = digitalRead(Enc[0].pinB);
58
59    /* possibly wait before reading pin A, then read it       */
60    int drA = digitalRead(Enc[0].pinA);
61
62    /* this updates the counter                         */
63    if (drA == HIGH) {   /* low->high on A? */
64
65      if (drB == LOW) {  /* check pin B */
66                    Enc[0].pos++;  /* going clockwise: increment       */
67      } else {
68                    Enc[0].pos--;  /* going counterclockwise: decrement  */
69      }
70
71    } else {                  /* must be high to low on A */
72
73      if (drB == HIGH) { /* check pin B */
74                    Enc[0].pos++;  /* going clockwise: increment       */
75      } else {
76                    Enc[0].pos--;  /* going counterclockwise: decrement  */
77      }
78
79    } /* end counter update                           */
80
81  } /* end ISR pin A Encoder 0                          */
82
83
84
85  /* Interrupt Service Routine: change on pin B for Encoder 0  */
86  void isrPinBEn0(){
87
88    /* read pin A right away                          */
89    int drA = digitalRead(Enc[0].pinA);
90
91    /* possibly wait before reading pin B, then read it       */
92    int drB = digitalRead(Enc[0].pinB);
93
94    /* this updates the counter                         */
95    if (drB == HIGH) {   /* low->high on B? */
96
97      if (drA == HIGH) { /* check pin A */
98                    Enc[0].pos++;  /* going clockwise: increment       */
99      } else {
100                   Enc[0].pos--;  /* going counterclockwise: decrement  */
101     }
102
```

```
103   } else {                  /* must be high to low on B */
104
105     if (drA == LOW) {  /* check pin A */
106                   Enc[0].pos++;  /* going clockwise: increment        */
107     } else {
108                   Enc[0].pos--;  /* going counterclockwise: decrement  */
109     }
110
111   } /* end counter update */
112
113 } /* end ISR pin B Encoder 0  */
114
115
116
117
118 # endif
119 /* Externs_END */
120
121 void lectura_encoders_Start_wrapper(real_T *xD)
122 {
123 /* Start_BEGIN */
124 /*
125  * Custom Start code goes here.
126  /* %%%-SFUNWIZ_wrapper_includes_Changes_END --- EDIT HERE TO _BEGIN */
127 #define y_width 1
128
129 /*
130  * Create external references here.
131  *
132  */
133 /* %%%-SFUNWIZ_wrapper_externs_Changes_BEGIN --- EDIT HERE TO _END */
134 /* extern double func(double a); */
135 /* %%%-SFUNWIZ_wrapper_externs_Changes_END --- EDIT HERE TO _BEGIN */
136
137 /*
138  * Output function
139  *
140  */
141 void sfcn_encoder_Outputs_wrapper(int16_T *pos,
142                                   const real_T *xD,
143                                   //const uint8_T *enc, const int_T p_width0,
144                                   const uint8_T *pinA, const int_T p_width1,
145                                   const uint8_T *pinB, const int_T p_width2)
146 {
147 /* %%%-SFUNWIZ_wrapper_Outputs_Changes_BEGIN --- EDIT HERE TO _END */
148 /* wait until after initialization is done */
149 if (xD[0]==1) {
150
151   /* don't do anything for mex file generation */
152   # ifndef MATLAB_MEX_FILE
153
154     /* get encoder position and set is as output */
155     pos[0]=Enc[0].pos;
156
157   # endif
158
```

```
}
/* %%%-SFUNWIZ_wrapper_Outputs_Changes_END --- EDIT HERE TO _BEGIN */
}

/*
 * Updates function
 *
 */
void sfcn_encoder_Update_wrapper(int16_T *pos,
                                 real_T *xD,
                                 //const uint8_T *enc, const int_T p_width0,
                                 const uint8_T *pinA, const int_T p_width1,
                                 const uint8_T *pinB, const int_T p_width2)
{
/* %%%-SFUNWIZ_wrapper_Update_Changes_BEGIN --- EDIT HERE TO _END */
if (xD[0]!=1) {

  /* don't do anything for MEX-file generation */
  # ifndef MATLAB_MEX_FILE

    /* enc[0] is the encoder number and it can be 0,1 or 2  */
    /* if other encoder blocks are present in the model     */
    /* up to a maximum of 3, they need to refer to a        */
    /* different encoder number                             */

    /* store pinA and pinB in global encoder structure Enc  */
    /* they will be needed later by the interrupt routine   */
    /* that will not be able to access s-function parameters */


    /* set encoder pins as inputs     izq                   */
    pinMode(Enc[0].pinA, INPUT);
    pinMode(Enc[0].pinB, INPUT);

    /* turn on pullup resistors                             */
    digitalWrite(Enc[0].pinA, HIGH);
    digitalWrite(Enc[0].pinB, HIGH);


    /* attach interrupts                                    */

      attachInterrupt(getIntNum(Enc[0].pinA), irsPinAEn0, CHANGE);
      attachInterrupt(getIntNum(Enc[0].pinB), isrPinBEn0, CHANGE);


    # endif

  /* initialization done */
  xD[0]=1;
}
/* %%%-SFUNWIZ_wrapper_Update_Changes_END --- EDIT HERE TO _BEGIN */
}
/* Start_END */
}

void lectura_encoders_Outputs_wrapper(int32_T *pos,
```

```
                              const real_T *xD)
{
/* Output_BEGIN */
/* wait until after initialization is done */
if (xD[0]==1) {

  /* don't do anything for mex file generation */
  # ifndef MATLAB_MEX_FILE

    /* get encoder position and set is as output */
    pos[0]=Enc[0].pos;

  # endif

}
/* Output_END */
}

void lectura_encoders_Update_wrapper(int32_T *pos,
                              real_T *xD)
{
/* Update_BEGIN */
if (xD[0]!=1) {

  /* don't do anything for MEX-file generation */
  # ifndef MATLAB_MEX_FILE

    /* enc[0] is the encoder number and it can be 0,1 or 2   */
    /* if other encoder blocks are present in the model      */
    /* up to a maximum of 3, they need to refer to a         */
    /* different encoder number                              */

    /* store pinA and pinB in global encoder structure Enc   */
    /* they will be needed later by the interrupt routine    */
    /* that will not be able to access s-function parameters */


    /* set encoder pins as inputs     izq                    */
    pinMode(Enc[0].pinA, INPUT);
    pinMode(Enc[0].pinB, INPUT);

    /* turn on pullup resistors                              */
    digitalWrite(Enc[0].pinA, HIGH);
    digitalWrite(Enc[0].pinB, HIGH);


    /* attach interrupts                                     */

      attachInterrupt(getIntNum(Enc[0].pinA), irsPinAEn0, CHANGE);
      attachInterrupt(getIntNum(Enc[0].pinB), isrPinBEn0, CHANGE);


    # endif

  /* initialization done */
  xD[0]=1;
```

```
}
/* Update_END */
}

void lectura_encoders_Terminate_wrapper(real_T *xD)
{
/* Terminate_BEGIN */
/*
 * Custom Terminate code goes here.
 */
/* Terminate_END */
}
```

Código 2: Función lectura_encoders

Analizando esta función, en primer lugar se define la estructura *Encoder* como el conjunto de los enteros *pinA*, *pinB* y *pos* (posición), por lo que las entradas a esta función son los terminales de la placa o pines que sirven como entrada digital a la placa de los trenes de pulsos generados por el encoder (en este caso, un único encoder que utiliza las entradas digitales 2 y 3). La salida es *pos*, una magnitud que mide el desplazamiento angular en el eje del motor. Hay que tener en cuenta que la salida *pos* no posee unidades, es una magnitud que se actualiza cada vez que se detecta un flanco de subida en alguno de los dos trenes de pulsos. A continuación, existe una función de manejo de interrupciones que a cada terminal le asigna un número entero de interrupción.

Seguidamente, existen dos pares de funciones *irsPinXEnX* de tipo *void*, cada una de ellas referida a un pin concreto del encoder. Las señales emitidas permiten saber tanto la velocidad del motor como su sentido de giro, gracias a la detección de variaciones en el campo magnético que perciben cada uno de los sensores de efecto Hall en este tipo de encoders. La frecuencia de las señales determinará la velocidad y el desplazamiento de una señal respecto a la otra, siempre de 90º, permitirá adivinar el sentido de giro: si la primera señal A está adelantada con respecto a la segunda señal B, el sentido es antihorario. De esto es de lo que se encargan estas funciones, incrementando o decrementando la posición en función de si se trata de un desplazamiento antihorario u horario [34].

Se han respetado los colores preestablecidos para los terminales de motor y encoder.

ANEXO XII:
Hoja de datos del potenciómetro

7/8" (22.2 mm) Precision Industrial Potentiometer, Bushing And Servo Mount Versions, Conductive Plastic

FEATURES

- High quality
- Rugged one piece metal housing
- Long rotational life
- Wide operating temperature range
- Linearities down to ± 0.25 % special
- Optional sealed construction (bushing mount only)
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

RoHS COMPLIANT

QUICK REFERENCE DATA	
Sensor type	ROTATIONAL, conductive plastic
Output type	Output by turrets
Market appliance	Industrial
Dimensions	7/8" (22.2 mm)

ELECTRICAL SPECIFICATIONS	
PARAMETER	**MIL-PRF-39023 TEST PROCEDURES APPLY**
Resistance	1 kΩ to 100 kΩ
Resistance Tolerance Special to	± 20 % ± 10 %
Linearity Special to	± 2.0 % ± 0.25 %
Temperature Coefficient of Resistance	± 600 ppm/°C
Power Rating Derated to	1.0 W at 40 °C ambient 0 W at 125 °C
Electrical Angle	340° ± 4°
End Voltage	0.5 % maximum
Dielectric Withstanding	1000 V_{RMS}, 60 Hz
Insulation Resistance	100 MΩ minimum, 500 V_{DC}
Output Smoothness	0.1 %

MECHANICAL SPECIFICATIONS		
PARAMETER		
Weight	0.5 oz. maximum (14 g)	
Rotation	360° (continuous)	
Mount Bearing Type	**BUSHING** Sleeve bearing	**SERVO** Ball bearing
Operating Torque Starting Running	 0.30 oz. - in (21.6 g - cm) 0.25 oz. - in (18 g - cm)	 0.25 oz. - in (18 g - cm) 0.15 oz. - in (10.8 g - cm)
Mechanical Tolerance (in/mm) (maximum)		
Shaft Runout (TIR)	0.002" (0.05 mm)	0.002" (0.05 mm)
Pilot Dia Runout (TIR)	-	0.002" (0.05 mm)
Lateral Runout (TIR)	0.005" (0.13 mm)	0.002" (0.05 mm)
Shaft End Play	0.006" (0.15 mm)	0.005" (0.13 mm)
Shaft Radial Play	0.003" (0.08 mm)	0.002" (0.05 mm)

For technical questions, contact: sferprecisionpot@vishay.com

THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE. THE PRODUCTS DESCRIBED HEREIN AND THIS DOCUMENT ARE SUBJECT TO SPECIFIC DISCLAIMERS, SET FORTH AT www.vishay.com/doc?91000

ORDERING INFORMATION

1	5	7	B	5	0	3	M	C				

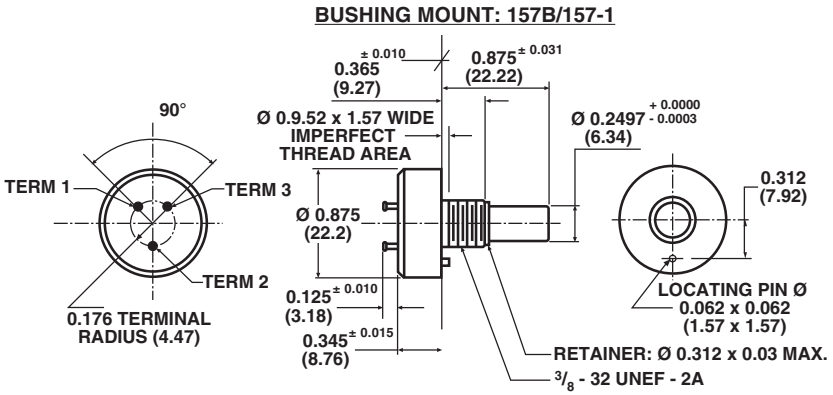
MODEL	STYLE	OHMIC VALUE	TOLERANCE ON OHMIC VALUE	LINEARITY	SPECIAL REQUEST
157	**B**: Bushing **S**: servo	**470** = 47 Ω **222** = 2.200 Ω **103** = 10 kΩ For ohmic value range see electrical specification	**M** = ± 20 % **K** = ± 10 %	**X** = ± 2 % **B** = ± 0.50 % **C** = ± 0.25 %	**xxxx**

PART NUMBER DESCRIPTION (for information only)

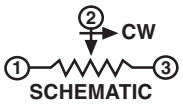
157-	2	503	xxxx
MODEL	STYLE	OHMIC VALUE	SPECIAL
	B: 1 **S**: 2		

DIMENSIONS in inches (millimeters)

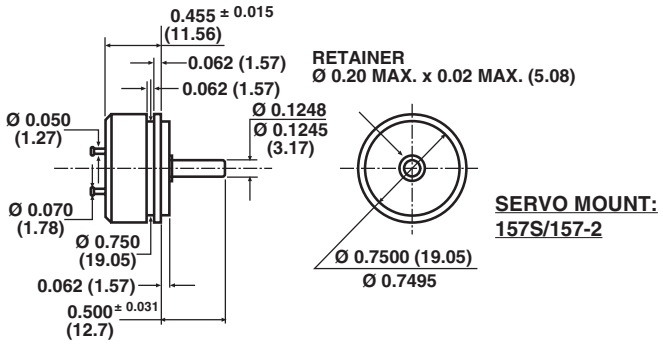
BUSHING MOUNT: 157B/157-1



Example of a marking for a standard part:
157-11103



TOLERANCES: UNLESS OTHERWISE NOTED.
DECIMALS ± 0.005 ANGLES ± 2°



SERVO MOUNT: 157S/157-2

For technical questions, contact: sferprecisionpot@vishay.com
THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE. THE PRODUCTS DESCRIBED HEREIN AND THIS DOCUMENT ARE SUBJECT TO SPECIFIC DISCLAIMERS, SET FORTH AT www.vishay.com/doc?91000

MATERIAL SPECIFICATIONS	
Housing/Bushing	Aluminum, anodized
Rear Lid	Ceramic
Shaft	Stainless steel
Terminals	Solderable
Bushing Mount Hardware	Lockwasher, internal tooth steel, nickel plated
Panel Nut	Brass, nickel plated

ENVIRONMENTAL SPECIFICATIONS		
Temperature	-55 °C to +125 °C	
Rotational Life	**BUSHING** 5 million shaft revolutions	**SERVO** 10 million shaft revolutions
Moisture Resistant	Yes	
Vibration	15 *g* 10 to 2000 Hz	
Shock	50 *g*	
Salt Spray	96 h	
Load Life	900 h	

Note

- Nothing stated herein shall be construed as a guarantee of quality or durability.

For technical questions, contact: sferprecisionpot@vishay.com

THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE. THE PRODUCTS DESCRIBED HEREIN AND THIS DOCUMENT ARE SUBJECT TO SPECIFIC DISCLAIMERS, SET FORTH AT www.vishay.com/doc?91000

Disclaimer

ALL PRODUCT, PRODUCT SPECIFICATIONS AND DATA ARE SUBJECT TO CHANGE WITHOUT NOTICE TO IMPROVE RELIABILITY, FUNCTION OR DESIGN OR OTHERWISE.

Vishay Intertechnology, Inc., its affiliates, agents, and employees, and all persons acting on its or their behalf (collectively, “Vishay”), disclaim any and all liability for any errors, inaccuracies or incompleteness contained in any datasheet or in any other disclosure relating to any product.

Vishay makes no warranty, representation or guarantee regarding the suitability of the products for any particular purpose or the continuing production of any product. To the maximum extent permitted by applicable law, Vishay disclaims (i) any and all liability arising out of the application or use of any product, (ii) any and all liability, including without limitation special, consequential or incidental damages, and (iii) any and all implied warranties, including warranties of fitness for particular purpose, non-infringement and merchantability.

Statements regarding the suitability of products for certain types of applications are based on Vishay’s knowledge of typical requirements that are often placed on Vishay products in generic applications. Such statements are not binding statements about the suitability of products for a particular application. It is the customer’s responsibility to validate that a particular product with the properties described in the product specification is suitable for use in a particular application. Parameters provided in datasheets and / or specifications may vary in different applications and performance may vary over time. All operating parameters, including typical parameters, must be validated for each customer application by the customer’s technical experts. Product specifications do not expand or otherwise modify Vishay’s terms and conditions of purchase, including but not limited to the warranty expressed therein.

Except as expressly indicated in writing, Vishay products are not designed for use in medical, life-saving, or life-sustaining applications or for any other application in which the failure of the Vishay product could result in personal injury or death. Customers using or selling Vishay products not expressly indicated for use in such applications do so at their own risk. Please contact authorized Vishay personnel to obtain written terms and conditions regarding products designed for such applications.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document or by any conduct of Vishay. Product names and markings noted herein may be trademarks of their respective owners.

© 2017 VISHAY INTERTECHNOLOGY, INC. ALL RIGHTS RESERVED

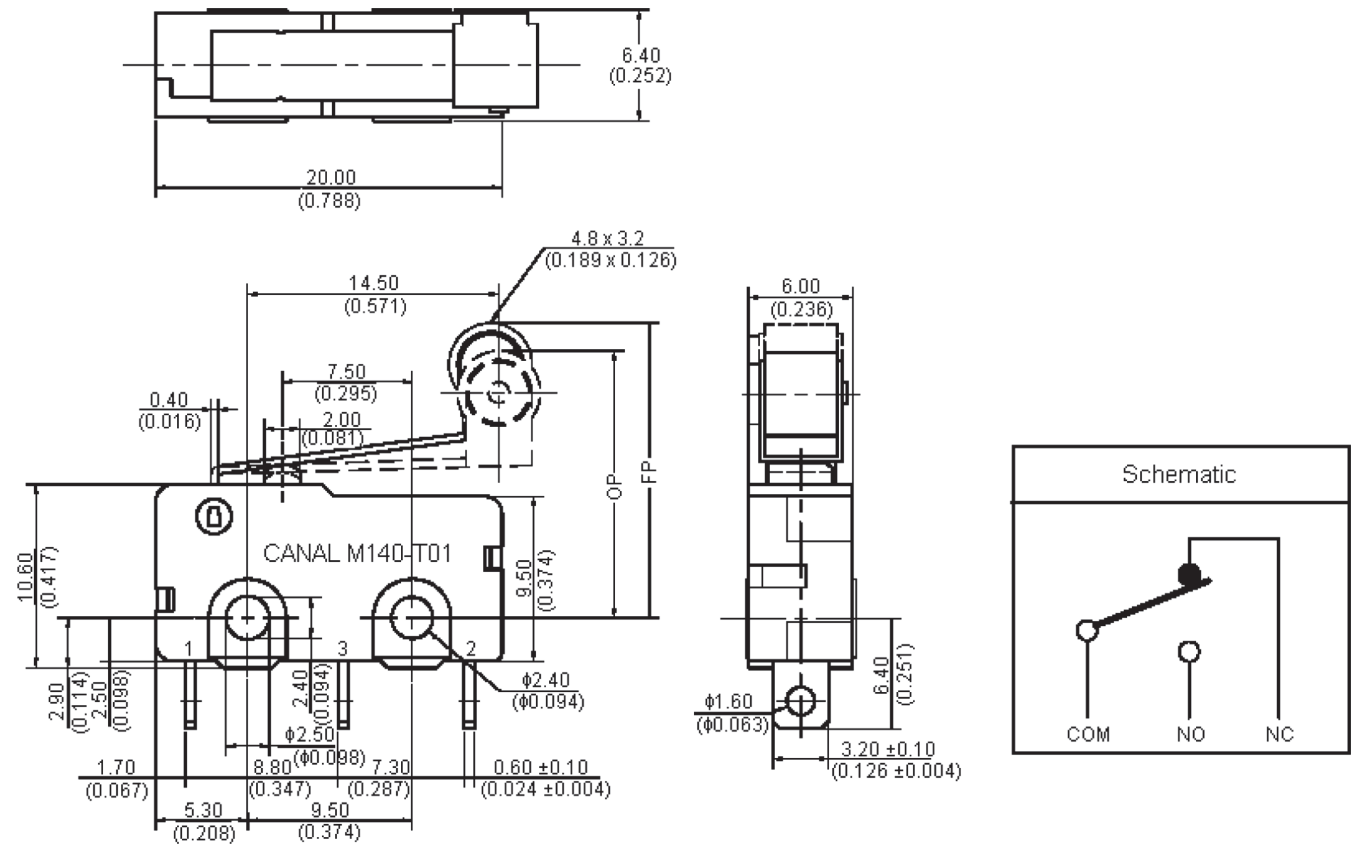
ANEXO XIII:
Hoja de datos del final de carrera

Microswitch
SPDT, Roller

Specifications:

Rating	: 5A, 125/250V AC
	: 5(2)A, 125/250V AC (NEMKO/ENEC/CQC)
	: 40T125 µ5E4
Maximum Contact Resistance	: 500MΩ
Insulation Resistance	: >100MΩ 500V DC
Withstand Voltage	: 1,000V AC for 1 minute
Operating Frequency	: Mechanical 60 operations/minute
	: Electrical 6 to 12 operation/minute
Ambient Temperature	: -25°C to +85°C
Ambient Humidity	: ≤ 85%RH
Mechanical Life	: 1,000,000 cycles
Electrical Life (at rate load)	: 50,000 cycles
Contact Configuration	: SPDT
Microswitch Type	: Snap Action
Switch Operation	: Off-On
Switch Terminals	: Solder
Contact Resistance	: 50mΩ
External Depth	: 6.4mm
External Width	: 19.8mm
Height	: 10.6mm
Operating Force	: 0.59N
Pre-travel	: 4mm
Series	: M140T01
Switch Case	: PBT (Polybutylene Terephthalate)

Microswitch
SPDT, Roller



Note: Tolerances unless otherwise specified: ±0.016/0.4 inches/mm

Dimensions: Millimetres

Part Number Table

Description	Operating force (gf)					OP (mm)	Max. FP (mm)	Part number
	A	B	C	D	E			
Microswitch, SPDT, 5A, Roller	20	30	40	50	55	15.5 ±1.5	18.9	M140T01-AE0505A

Important Notice : This data sheet and its contents (the "Information") belong to the members of the AVNET group of companies (the "Group") or are licensed to it. No licence is granted for the use of it other than for information purposes in connection with the products to which it relates. No licence of any intellectual property rights is granted. The Information is subject to change without notice and replaces all data sheets previously supplied. The Information supplied is believed to be accurate but the Group assumes no responsibility for its accuracy or completeness, any error in or omission from it or for any use made of it. Users of this data sheet should check for themselves the Information and the suitability of the products for their purpose and not make any assumptions based on information included or omitted. Liability for loss or damage resulting from any reliance on the Information or use of it (including liability resulting from negligence or where the Group was aware of the possibility of such loss or damage arising) is excluded. This will not operate to limit or restrict the Group's liability for death or personal injury resulting from its negligence. Multicomp Pro is the registered trademark of Premier Farnell Limited 2019.