ACTUADOR ELECTROMECANICO DE MOVIMIENTO LINEAL ACCIONADO POR UN MOTOR DE PASO PROGRAMADO CON ARDUINO UNO

P. Cabezas Amaza[1], N. Maureira Carsalade[2], Frank Sanhueza Espinoza[3]

RESUMEN

Esta investigación presenta el desarrollo y marcha blanca de un prototipo de actuador electro-mecánico capaz de aplicar a un sistema estructural un forzante controlado por movimiento en un solo eje. En este caso se utilizó un motor Stepper Nema 34, el cual entrega movimientos precisos, es de fácil control y sencilla instalación. El motor a utilizar es de 200 pasos por revolución, lo que quiere decir que cada paso permite un giro horario (+) o antihorario (-) de 1.8 grados de circunferencia. Es accionado por un microcontrolador Arduino–UNO, el cual permite el control desde un PC. Se tomó como desafío realizar dos movimientos preestablecidos: "Movimiento armónico" (función seno) y "Diente de Sierra". En este documento se explica cómo se desarrolla la programación de Arduino–Uno, las conexiones del motor, se exponen el montaje del prototipo y su accionamiento. Se obtuvo como resultado un prototipo exitoso pero que presenta limitaciones de capacidad de carga.

Palabras Claves: Motor Stepper Nema 34, Programación Arduino–Uno, Movimiento armónico, Movimiento Diente de Sierra.

ABSTRACT

This research presents the development and trial run of an electro-mechanical actuator prototype capable of applying a single-axis motion-controlled forcing force to a structural system. In this case, a Nema 34 Stepper motor was used, which delivers precise movements, is easy to control and simple to install. The motor to be used has 200 steps per revolution, which means that each step allows a clockwise (+) or counterclockwise (-) turn of 1.8 degrees of circumference. It is driven by an Arduino-UNO microcontroller, which allows control from a PC. Two pre-established movements were taken as a challenge: "Harmonic movement" (sine function) and "Sawtooth". This document explains how the Arduino-Uno programming is developed, the connections of the motor, the assembly of the prototype and its drive are exposed. A successful prototype was obtained as a result, but it has load capacity limitations.

Keywords: Nema 34 Stepper Motor, Arduino–Uno Programming, Harmonic Motion, Sawtooth Motion.

1. INTRODUCCIÓN

En el análisis de sistemas o mecanismos estructurales se utilizan los ensayos experimentales para validar o calibrar modelos numéricos, con el propósito de realizar diseños más fidedignos y prevenir riesgos de falla o colapso. Se han desarrollado modelos a escala reducida, utilizando los mismos materiales estructurales que serían usados en el prototipo real. Estos modelos son sometidos a forzantes o combinación de movimientos, permitiendo predecir con gran confiabilidad el comportamiento de dispositivos y sistemas estructurales [1].

En la UCSC, se ha implementado un laboratorio experimental para ingeniería, en el que se dispone de un marco de ensayo pequeño con potencial de acoplamiento de un forzante controlado por movimiento uni-axial, que aún no cuenta con un actuador electromecánico. Para este uso específico, se debe contar con la fuerza suficiente, un control preciso de movimientos en términos de dirección, cantidad de giros, y velocidades programables del motor según requerimientos del investigador.

[1] Estudiante, Carrera de Ingeniería Civil, Universidad Católica de la Santísima Concepción, CHILE, pacabezas@ing.ucsc.cl
[2] Profesor guía, Departamento de Ingeniería Civil, Universidad Católica de la Santísima Concepción, CHILE, nmaureira@ucsc.cl
3 Profesor informante, Dpto. de Ingeniería Civil, Universidad Católica de la Santísima Concepción, CHILE, franksanhueza@ucsc.cl

El profesor guía de esta tesis planteó, como un desafío aplicado, el desarrollar un prototipo de actuador electro-mecánico capaz de aplicar a un sistema estructural, un forzante controlado por movimiento uni-axial. Para tal efecto, se adquirieron componentes disponibles en el mercado, incluyendo el dispositivo electrónico Arduino-UNO, que es un micro controlador programable que, entre otras funciones, tiene el potencial de controlar motores de paso con software de libre uso. Anteriormente se había realizado, con esos mismos elementos, una experiencia práctica de pruebas básicas de conexiones entre estos dispositivos, que logró generar giros horario y anti horario con un motor stepper. Esto sirvió de motivación para continuar con el proyecto y proceder a desarrollar la programación de los movimientos requeridos por un investigador, ejecutar un armado del prototipo a nivel marcha blanca y hacer pruebas de testeo experimental de los movimientos generados con esta alternativa de prototipo de actuador electro mecánico.

En el presente informe se da cuenta de la experiencia de ingeniería aplicada, realizada como parte de la habilitación profesional de un alumno de Ingeniería Civil de la UCSC, de asumir ese desafío ingenieril. El proyecto tiene como restricción "usar lo disponible", la necesidad de integrar conocimientos y competencias de las disciplinas complementarias requeridas a bajo costo, para finalmente, según los resultados del avance en etapas, y de ser factible, se llegase a ejecutar pruebas del prototipo a nivel de marcha blanca.

Para efectos de mejor comprensión, se presenta la experiencia dividida en etapas sucesivas, describiendo las principales actividades, las adaptaciones prácticas a las restricciones de los elementos disponibles, y otras. El período de ejecución de esta experiencia fue en su totalidad durante la pandemia COVID 19, lo que en vez de ser un disuasivo para abortarla, puso un nuevo desafío que fue enfrentado como ingenieros, es decir con ingenio y creatividad.

2. OBJETIVOS

Objetivo general:

- Adaptar y ensayar un motor de paso micro-controlado a un actuador de movimiento lineal para lograr reproducir secuencias de movimiento controlados por computadora.

Objetivos Específicos:

- Mover el actuador lineal con un motor stepper Nema-34 controlado por Arduin-UNO programado por computadora.
- Ensayar movimientos pre establecido de tipo "diente de sierra" y "armónico" de amplitud y frecuencia previamente definidas.
- Adaptar componentes disponibles en el mercado para realizar ensayo experimental y desarrollar un programa de control compatible con los componentes utilizados.
- Ejecutar una actividad experimental midiendo movimientos generados por el actuador micro-controlado y la fuerza impuesta por este a una estructura simple.

3. MARCO TEORICO

Los ensayos experimentales permiten registrar, analizar, y probar el comportamiento de sistemas estructurales para validar o calibrar modelos numéricos y así poder realizar diseños más fidedignos para prevenir futuros riesgos de falla o colapso estructural. Los modelos son sometidos a forzantes o combinación de movimientos, permitiendo predecir con gran confiabilidad el comportamiento de dispositivos y sistemas estructurales [1].

El estudio del comportamiento de las estructuras se torna complejo, ya que no solo interfieren las cargas permanentes y las cargas temporales que van a tener que soportar, sino que también se debe incluir la influencia de las fuerzas externas naturales, como lo son el viento y lo sismos entre otros.

Estas interacciones externas son impredecibles, por lo que no se puede verificar con certeza como se comportara la estructura a lo largo de la vida útil, pero se puede lograr dimensionar mediante ensayos experimentales [2].

Contar con un actuador controlado por desplazamiento para efectuar ensayos experimentales en un laboratorio, como lo es tener una mesa con actuador de movimientos controlados y a un costo accesible, aporta a la docencia en disciplinas de la ingeniería que lo requieran.

Lo esencial es reproducir los movimientos requeridos en el marco de ensayos, esto es, recorrido, dirección y fuerza en instantes determinados, para lograr las aceleraciones y desaceleraciones que sean pre definidas por el operador.

El actuador está compuesto por un mecanismo que cuenta con un husillo de bolas que transforma la rotación y torque de un motor stepper en desplazamiento y fuerza.

En el laboratorio de Ingeniería de la Universidad Católica de la Santísima Concepción, sólo se disponía del husillo de bolas y elementos para formar el marco de pruebas. Faltando el elemento propulsor que controladamente y sin riesgos para el operador, entregue la cantidad de giros, fracciones de giro y su dirección con las fuerzas requerida a través de su eje, en el periodo de tiempo resultante del cálculo para este dispositivo de conversión del movimiento giratorio a movimiento lineal.

Se asumió el desafío de programar un motor de paso disponible en el departamento de ingeniería, del tipo stepper Nema 34 con un driver y fuente de poder para el control del equipo. Se planteó la alternativa de utilizar un micro controlador del tipo Arduino–UNO. Esto basado en una experiencia previa, realizada por otro alumno de ingeniería, en la cual se avanzó en posibles conexiones electrónicas con pulsos generados desde el Arduino–UNO hacia un driver de un motor stepper Nema 42, experiencia en la cual se logró, un movimiento del motor en una sola dirección.

Al tratarse de un micro-controlador, posible de programar a nivel binario, utilizando una herramienta de lenguaje interpretativo de código abierto, y con una inter fase disponible para operar desde un computador, se generó un programa de control con los parámetros calculados para cada movimiento. Se dividió el movimiento en 20 sub periodos de tiempos para cada ciclo, además de control sobre la cantidad de repeticiones de cada ciclo. Para el logro de esta fase, se requirió de apoyo de un programador, que tradujera los requerimientos de los movimientos al lenguaje de Arduino–UNO.

Para una mejor comprensión de lo recién mencionado, a continuación se describirá los elementos utilizados y sus características:

3.1 Elementos eléctricos y electrónicos que forman parte de este proyecto

a) Arduino-UNO

Es una placa electrónica construida alrededor de un micro controlador ATmega328, de 8 bits operando con un reloj de 16 Mhz, 131 instrucciones realizables en un ciclo en su mayoría (excepto división y multiplicación), diseñada y fabricada por Arduino. Su código es abierto y posee un *lenguaje* de programación similar al C++ y un IDE (Entorno de Desarrollo Integrado) que funciona mediante conexión USB a un PC o MAC que lo contenga. Además, este IDE y la electrónica interna de la placa permiten programar tareas y dejar al Arduino programado indefinidamente, hasta nuevo programa, si se desea [3].

Figura N° 1: Componentes Placa Arduino [Fuente: G. Aguasca2021]

La placa Arduino-UNO (Figura 1) posee, entre otras prestaciones: 14 salidas digitales, un regulador de tensión para alimentarlo con voltaje entre 9 y 20 V (se recomienda usar 5V regulados, que es la tensión nativa del ATmega 328). Usualmente la USB proporciona esta tensión desde el PC anfitrión.

Como la placa Arduino UNO usa un *lenguaje* de alto nivel (C++) y los programas del usuario son escritos en él, antes de cargar el programa en la *Memoria de programa* del ATmega328 el IDE realiza una compilación, es decir una transformación desde instrucciones de alto nivel a las que realiza la máquina. Esto que parece trivial en muchas aplicaciones, para este caso *establece limitaciones a la velocidad de giro*, ya que se debe tener en cuenta que una aparentemente simple instrucción de lenguaje de Arduino puede dar origen a cientos de instrucciones de la máquina para lograr su ejecución, con el tiempo que eso involucra. Esto es particularmente notorio si la aritmética del problema requiere operaciones en 16 o 32 bits, las que aún en lenguaje de máquina requieren mucho algoritmo debido a que la Unidad de Cálculos interna es de 8 bits.

b) Motor Stepper Nema 34

El principio de funcionamiento está basado en un estator construido por varios arrollamientos independientes devanados sobre un material ferromagnético y un rotor que puede girar libremente en el interior, rodeado por el estator.

El motor de pasos utilizado es de tecnología bipolar (Figura 2). Esto significa que las bobinas deben recibir una tensión (60V en este caso) el que es aplicada con *Polaridad* a los extremos de ésas. Se caracteriza porque, a diferencia de los otros motores eléctricos, *no gira* continuamente mientras tenga el poder eléctrico aplicado. Por el contrario, su estructura y construcción sólo le permiten recibir voltajes de valor fijo que activan sus bobinas de estator en una cierta secuencia y cuando esto ocurre el eje del motor *avanza* un paso, realiza un giro en una u otra dirección [4].

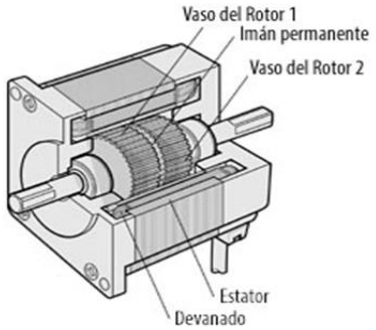

Figura N° 2: Motor Stepper Nema 34[Fuente: Oriental Motor]

Ocurrido el giro, aunque las bobinas sigan energizadas, el eje del motor *sigue estático* en la posición en que quedó, hasta que se reciba una nueva y adecuada secuencia de voltajes en las bobinas. Por esto, es más un *motor de posición* controlada y su uso más común es para hacer *control de posición* sin requerir de un sistema de Control de Lazo Cerrado, ya que, definiendo su posición inicial es posible en todo instante saber exactamente en cuantos grados se encuentra el eje.

Además, es necesario tener en cuenta que este motor se usa para *posicionar* el eje en forma muy precisa y controlada, en aplicaciones en las que el *tiempo* para posicionarse no tiene mayor importancia. La presente aplicación saca al motor de pasos de su uso fundamental y *lo obligará a girar*, (velocidad) lo que hará, sin duda, pero habrá limitaciones por fabricación y como se verá, también en *el tiempo* requerido para la generación del control electrónico, el cual requiere de muchos cálculos para el procesador.

Otra consideración a tener en cuenta en este proyecto es que *el aumento de velocidad (RPM) debilita el torque del motor*, lo que incluso puede llevar a que si la exigencia del comando es muy rápida el motor vibrará y no será capaz de salir a la próxima posición. *Esto va a limitar tanto la amplitud como la frecuencia de la señal a obtener.* En este aspecto habrá que experimentar con diferentes velocidades para conocer el límite que sin duda lo tendrá.

Sin embargo, si se envía una secuencia adecuada de voltajes de activación a sus bobinas, es posible hacerlo avanzar de paso en paso, (de allí su nombre) y dependiendo de la velocidad con que se le entregue la secuencia, se podrá lograr que el eje gire mientras se esté recibiendo la secuencia. En cuanto esta desaparezca, el eje se quedará estático en la última posición a la espera de nuevo comando.

El motor a utilizar es de 200 pasos por revolución, lo que quiere decir que cada paso permite un giro horario (+) o antihorario (-) de 1.8 grados de circunferencia.

Si bien es cierto, el *Driver* puede generar secuencias de 400, 800 o más pasos mediante combinación digital y recibiendo la cantidad y velocidad de pulsos requerida en cada caso, el motor funciona, pero el posicionamiento en esos medios o cuartos de paso es débil. Además, la acción misma pierde considerablemente el *Torque* (fuerza), factor muy importante en esta aplicación, lo que hace ***recomendable que el motor trabaje con los pasos de diseño físico, esto es 200 pasos.***

El límite de RPM de un motor de pasos no está bien definido, pero en general, mientras más rápido se le apliquen los pasos (pulsos a las bobinas) el torque va disminuyendo progresivamente hasta que puede llegar a no ser capaz de dar el paso y no hay giro, Hay que tener en cuenta esto pues afectará la *amplitud* del movimiento que se desee obtener.

c) Driver DM860A

Un driver para motores paso a paso es un circuito que permite controlar los motores de corriente continua de una forma simple (Figura 3). Estos controladores permiten manejar los voltajes e intensidades a los que se está suministrando al motor para así controlar la velocidad de giro, mediante

el uso de una fuente de poder.

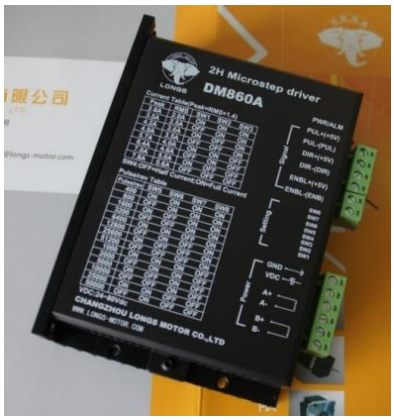

Figura N° 3: Driver DM860A [Fuente: Longs- Motor]

Entre sus características está la de manejar motores de paso desde 24 V a 80V y la de permitir la programación de los pasos, la que se logra por combinaciones digitales y aumento de la frecuencia de los trenes de pulso de entrada. Posee dos entradas una para los pulsos (PUL (+/-) y otra para la dirección del giro DIR(+/-) [5].

Este elemento es un complejo circuito digital que tiene por misión convertir las señales (trenes de pulsos) digitales que genera Arduino u otro microcontrolador que son de 5V en los voltajes del nivel (60 V para el NEMA 34), polaridad (sentido de aplicación a bobinas del motor) y momento en que se activan estas bobinas con la tensión ya indicada. Aunque excede a los alcances de este trabajo, es necesario mencionar que el driver permite preseleccionar ciertos interruptores que permiten simular un motor de más pasos que los que realmente está construido para dar. Ya se mencionó que esto se puede hacer, pero a costa de pérdida de torque y si se trata de movimiento de giro, puede haber pérdida o saltos de posición. No es recomendable usar esta característica en este trabajo. Debe ajustarse a 200 pasos.

En breves palabras, para funcionar el *Driver* DM860A requiere recibir un tren de *pulsos eléctricos, 4 en total*, en el PIN PUL(-) y otro similar en DIR(-) para lograr la activación adecuada de las bobinas y lograr que el motor *gire un paso*, en el sentido horario (+) o anti horario(-), según sea la fase relativa al PUL(-) de la señal recibida en el PIN DIR(-).

Tabla N° 1: Descripción de señales [Fuente: Elaboración propia]

PIN	**Función**
PUL +	**Señal de pulso:** En el modo de pulso único (pulso/dirección), esta entrada representa la señal de pulso, cada flanco ascendente o descendente activo; 4.5-5V en PUL(+) y 0-0.5V en PUL(-).
PUL -	
DIR +	**Señal DIR:** En modo de pulso único, esta señal tiene niveles de voltaje bajo/alto, lo que representa dos direcciones de rotación del motor;4.5-5V en DIR(+), y 0-0.5V en DIR(-).
DIR-	
ENA+	**Señal de habilitación:** esta señal se utiliza para habilitar/deshabilitar el controlador. Nivel alto para habilitar el controlador y nivel bajo para deshabilitar el controlador. Por lo general, se deja DESCONECTADO.
ENA-	

d) Fuente de poder 60 VDC 3.3 A

La electrónica del Driver y el motor necesitan 60 VDC (corriente continua), proporcionados por una fuente de poder (Figura 4). La alimentación primaria es 220 V alternos, 50 Hz, por lo que este elemento posee la electrónica necesaria para convertir la corriente alterna en continua y bajar de 220V a 60 V. El lado que se conecta a la tensión AC es un enchufe común con Neutro, Línea y Tierra al centro, por lo que no hay problemas para su conexión [6].

Figura N° 4: Fuente de poder 60 VD [Fuente: Longs-motor]

En el lado de la corriente continua, entrega 60 V 3.3 A entre los terminales +60 y -60. Se debe conectar directamente al *Driver*.

Por lo anterior y debido a que existe una relación entre los giros del eje del actuador (del motor en este caso) y el avance axial de 5 mm/vuelta, el presente problema se reduce a hacer girar el motor de pasos en la cantidad giros necesarios en un tiempo determinado, lo que para este motor se logrará enviando una serie de pulsos a una frecuencia de repetición de acorde a la velocidad del movimiento final requerido.

Recordando que entre paso y paso el motor se detiene hasta recibir el siguiente, se hace evidente que su giro *no es continuo* ni suave, sino que avanzará 200 "saltos" de 1.8 grados para lo que requerirá recibir 800 pulsos de control para *dar un giro* y avanzar 5 mm. Además, para seguir la función de la Figura N°2, será necesario una variación de la velocidad del giro, lo que se traducirá finalmente en una variación de la frecuencia de repetición de los pulsos requeridos, según el sector de la función seno en que se encuentre el movimiento.

Entonces, no sólo se deben enviar una cierta cantidad de pulsos para avanzar, sino que su frecuencia también deberá variar para ajustarse a la función seno o diente de sierra requerida.

Para establecer el marco teórico base del presente trabajo será necesario enfocarse en los siguientes aspectos:

1. Movimiento armónico en un motor de pasos
2. Matemática detrás del software
3. Filosofía del software para el actuador controlado por Arduino -UNO

3.2 Movimiento armónico en un motor de pasos

Establecido el modo de avance del motor de pasos se hace necesario analizar el comportamiento que deberá presentar para generar el movimiento más fundamental de la plataforma: Un movimiento armónico. (Figura N°1)

La ecuación (1) define el movimiento a estudiar y producir (lo que dará las bases para cualquier otro tipo de movimiento) como:

$$u(t) = A \cdot sen(\omega \cdot t) \qquad (1)$$

Donde $\boldsymbol{A}$ es la Amplitud del desplazamiento, $\boldsymbol{\omega}$ es la frecuencia angular y $\boldsymbol{t}$ es el tiempo.

Cuya gráfica de Amplitud en función del tiempo está dada por la Figura 5:

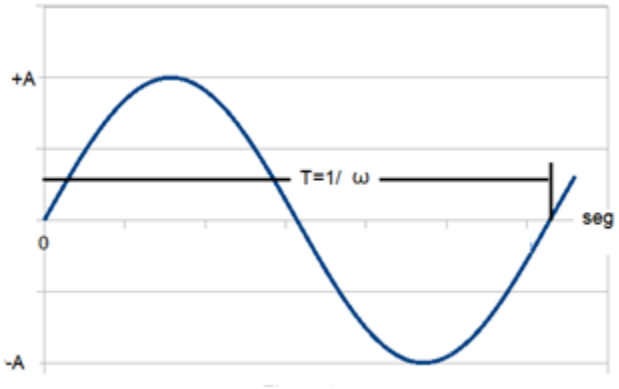

Figura N° 5: Amplitud en función del tiempo [Fuente: Ingeniero Civil Electrónico Gustavo Sanhueza G.]

Si se aplica la función anterior a la plataforma que se desea mover axialmente, esta función deberá anotarse como desplazamiento en cm en función del tiempo. Entonces, considerando una amplitud máxima conocida, por ejemplo 5 cm y con un período de movimiento de 2 segundos (ecuación 2).

$$D(t) = 5 \cdot sen\left(2 \cdot \pi \cdot \frac{t}{2}\right)\ cm/seg \qquad (2)$$

Y el movimiento queda expresado por la gráfica de la Figura 6:

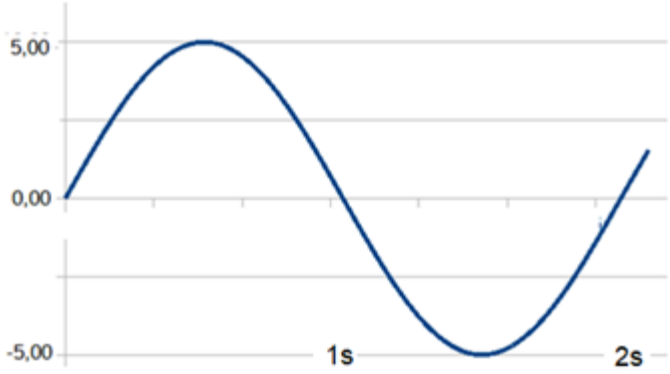

Figura N° 6: Movimiento Armónico [Fuente: Ingeniero Civil Electrónico Gustavo Sanhueza G.]

Aunque se muestra sólo un ciclo, esta forma de onda es periódica y el ciclo se podrá repetir las veces que se requiera en alguna aplicación.

3.3. Matemática detrás del software

El Controlador Arduino-UNO fue utilizado para generar un tren de pulsos de 5 V para la entrada PUL(-) y otro para la entrada DIR(-) del *Driver* DM860A, los que deben ser generados por el Arduino-UNO utilizando adecuadamente las instrucciones:

- ✓ ***myStepper.setSpeed (RPM)*** para establecer la *velocidad de giro* en RPM, relacionado con la frecuencia a la que serán emitidos los pulsos en la instrucción siguiente [3].
- ✓ ***myStepper.step (xpulsos)*** para que *físicamente* aparezcan los pulsos en la entrada PUL(-) y DIR (-) del DRIVER y éste genere los pasos necesarios de dar a esa velocidad, sentido de giro incluido [3].

3.4 Análisis de la señal de movimiento impuesto

Se aprecia que el movimiento sinusoidal impuesto requiere de una variación de la velocidad y cantidad de pulsos de control producidos por el Arduino-UNO. Sin embargo, debido a que es necesario programar las dos *instrucciones* anteriormente mencionadas para que exista un movimiento determinado, se hace necesario *dividir* el período en una cantidad de sub-periodos que mediante el uso de grafico por trazos (Figura N° 3) se aprecia que se necesitan más de 10 sub-periodos. Aquí se ve que 20 trazos o sub-periodos en que se divida el período de la sinusoide muestran una suavidad y aproximación que cuesta diferenciar de la sinusoide de trazo continúo. Experimentalmente se comprobó la suavidad y aproximación del movimiento, al usar 20 sectores o instantes de cambio de datos y salida de nueva instrucción. Es evidente que una división mayor, por ejemplo 30 o 50 aproximaría aún más la forma de onda al ser más pequeños los trazos rectos. Sin embargo, se estima que esto no sería posible por la limitación del tiempo de cálculos y proceso del Arduino, lo que produciría retardos y alargue del periodo a generar, sin considerar que el motor puede no responder a la frecuencia que le llegarían los pulsos. *Se sugiere usar 20 sub-periodos en el desarrollo de esta aplicación.*

Finalmente, al observar la Figura 7 pueden verse los puntos de origen y término de los trazos rectos. De alguna manera estos puntos indican que allí hay un tiempo de procesamiento y preparación para ejecutar las instrucciones del lenguaje de alto nivel que se va a utilizar.

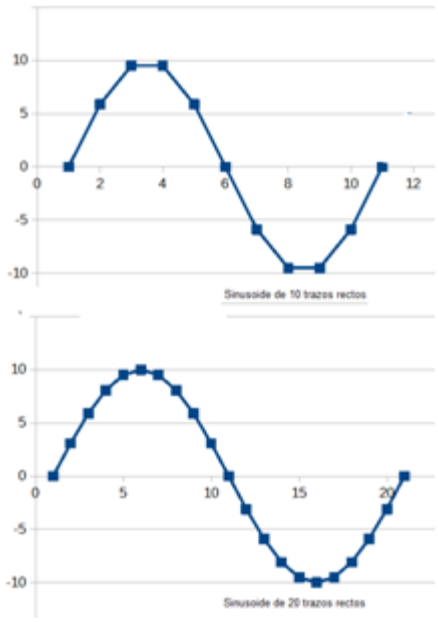

Figura N° 7: Movimiento Armónico de 20 puntos de control [Fuente: Ingeniero Civil Electrónico Gustavo Sanhueza G.]

Para obtener un ciclo completo el *software* deberá proveer 20 instantes de control, cada uno de ellos con una cantidad de pulsos y una *"velocidad"* adecuada para que todos salgan en el tiempo del sub-período.

El análisis de la filosofía del *software* se hará con un sector de la sinusoide, con el primer cuarto de un periodo. Es decir, con los 5 primeros tramos y para explicar el procedimiento se recurrirá a un ejemplo numérico: Si la amplitud máxima del movimiento fuera de 5 cm y el período fuera de 2 segundos, los sub períodos serán de 0.1 segundo cada uno. La Figura 8 muestra esta situación para los 5 primeros sub periodos de la sinusoide a generar.

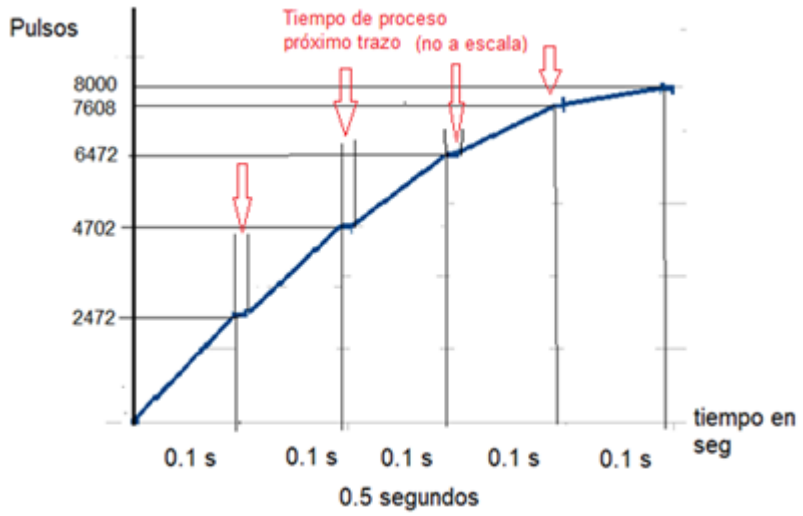

Figura N° 8: Grafica de 1/4 de recorrido [Fuente: Ingeniero Civil Gustavo Sanhueza G.]

Para obtener los parámetros (pulsos) y (RPM) de cada sub-periodo se deben considerar los siguientes cálculos

Amplitud máxima en cm = 5

Amplitud máxima en vueltas del eje 5 cm/0.5 cm/vuelta = 10 vueltas

Amplitud en pasos del motor: (10 vueltas)*(200 pasos/vuelta) = 2000 pasos

Amplitud del movimiento en pulsos: (2000 pasos)*(4 pulsos/paso) = 8000 pulsos

Tiempo del Sub-periodo: 2 seg / 20 = 0.1 segundo

En la Figura 8 se puede observar que la cantidad de pulsos de cada sub-periodo no es la misma, sino que está determinada según la ley de la sinusoide que genera el movimiento. A destacar, la sinusoide continua se ha transformado en una serie de segmentos *Rectos* de pendientes diferentes.

Así, partiendo desde el instante 0, el primer valor a alcanzar en 0.1 segundos está dado por le ecuación (3).

$$S1 = 10 \cdot sen(2 \cdot \pi \cdot 0.1) = 3.09\ vueltas \quad (3)$$

Entonces, como se ha partido del reposo, la variación o *incremento de* vueltas es 3.09 – 0=3.09. Expresado en *pulsos* para generar en este sub periodo esto resulta ser 2472 pulsos en 0.1 segundo.

Transcurrido el primer sub periodo se debe generar el segundo, al cabo del cual la sinusoide habrá avanzado a la posición S2 de acuerdo a la ecuación (4)

$$S2 = 10 \cdot sen(2 \cdot \pi \cdot 0.2) = 5.88\ vueltas \quad (4)$$

Con esto, el incremento de vueltas para este segundo sub periodo es 5.88-3.09 =2.79 vueltas, lo que expresado en pulsos lleva a 2230 pulsos en 0.1 segundo.

Y así sucesivamente se calculan los incrementos de pulsos hasta completar los 20 sub-periodos que permiten generar la sinusoide de 2 segundos.

Por otra parte, para cada *incremento* de vueltas (*Delta* pulsos) es necesario calcular la velocidad a la que habrá de girar (RPM), cálculo que se hace según la ecuación (5).

$$RPM = 60 \cdot DELTAv/0.1s \quad (5)$$

Con los dos primeros valores ya calculados esto da:

RPM1=60*3.09/0.1 = 1854

RPM2=60*2.79/0.1 = 1673

Y así sucesivamente hasta calcular los 20 valores de RPM que se necesitan para las dos instrucciones que saldrán espaciadas cada 0.1 segundo, cosa que el software realiza en un LOOP de 20 iteraciones con el retardo respectivo.

El uso de una planilla EXCEL para estos cálculos permite facilitar la generación de matrices con estos valores, datos fundamentales para programar el Arduino- UNO en esta aplicación:

Tabla 2: Ejemplo de planilla Excel [Fuente: Ingeniero Civil Electrónico Gustavo Sanhueza G.]

		PULSOS	DELTA		VELOCIDAD
SUB PERIODO	Función SENO	Amplitud	PULSOS	TIEMPO (seg)	RPM
0	0,000	0		0	
1	0,314	2472	2472	0,1	1854
2	0,628	4702	2230	0,2	1673
3	0,942	6472	1770	0,3	1327
4	1,257	7608	1136	0,4	852
5	1,571	8000	392	0,5	294

Si bien es cierto en teoría los tramos generados con este método unidos al cabo de los 2 segundos asemejan y producen un movimiento sinusoide, se debe analizar más en detalle la Figura N° 4.

En ella, al término de cada sub periodo se ha dibujado un pequeño segmento horizontal que no está a escala (flecha roja). Este corresponde al *tiempo* que emplea el Arduino-UNO en ejecutar las instrucciones de máquina necesarias para ejecutar el *programa* y las instrucciones como las indicadas anteriormente, las que corresponden al *lenguaje* de Programación de Arduino-UNO y que es de alto nivel. Por esto en la Figura N°4 se ha querido mostrar que existe un tiempo que, aunque pequeño sea, hará que el periodo obtenido sea mayor en algunas decenas de mili segundos. En ensayos, se logró medir que un periodo de 2000 ms resultaba de 2045 ms. Para el movimiento mecánico a escala humana, este "alargue" del periodo no es perceptible, al igual que la deformación de la sinusoide al haberla obtenido por 20 tramos rectos.

En resumen, el Arduino producirá el movimiento requerido, dentro de las limitaciones del equipo, al poner el enfoque de la generación del movimiento en los *pulsos* que se deben generar.

Si se considera el primer sub periodo, se ve que el Arduino deberá generar un tren de 2472 pulsos en 0.1 segundos, lo que da una frecuencia de 24720 Hz. Si esta frecuencia se expresa en *tiempo*, se determina que para este caso (que es el más crítico por ser el más rápido) los pulsos se verán como en la Figura N° 9, esto es, la presencia de un cambio de voltaje que ocurre cada 20.225 microsegundos.

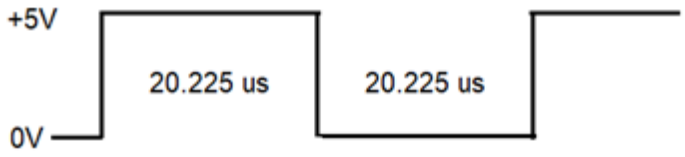

Figura N° 9: Función voltaje tiempo de un tren de pulsos [Fuente: Ing. Civil Electrónico G. Sanhueza G.]

Cada vez que se debe iniciar un sub periodo el software de alto nivel entrega el dato de pulsos a generar y RPM para esos pulsos, datos enteros de 16 bits con signo, en el caso de los pulsos. El procesador, como parte del programa debe "generar" la forma de onda vista en la figura 4, para lo cual debe transformar en forma precisa las RPM en microsegundos de duración de cada periodo.

La fórmula que debe calcular es

$$Tpulso = \frac{\frac{60}{RM}}{800} \quad (6)$$

Lo que en el ejemplo del primer sub periodo da 60/1854/800 = 40.45 useg, es decir la forma de onda de la Figura N° 5. Además, debe calcular y preparar el desfase entre PUL y DIR para el correcto sentido de giro, según el signo del dato "step". Con estos valores debe preparar su electrónica interna para lograr generar la forma de onda con los tiempos calculados. Si bien es cierto el micro controlador es muy rápido, se desconoce la cantidad de instrucciones de programa en lenguaje de máquina que genera la compilación del lenguaje de alto nivel, que es el disponible para el usuario. Sin embargo, vista la cantidad de bytes de código y memoria que informa como utilizados cuando carga, se aprecia que pueden ser 50 o más por instrucción del lenguaje C, lo que podría producir algún alargue en el periodo final generado, pero se estima que este no sería detectable a nivel humano y el actuador no se verá afectado. Estos retardos se indican con una flecha roja en la Figura N°5, la cual evidentemente no está a escala.

La forma de onda de la Figura N°5 se encuentra dentro del rango que el Arduino podría producir, el que podría extenderse en forma estimada hasta pulsos de unos 2 microsegundos o menos, debido a que las instrucciones que permiten "construir" estos pulsos se ejecutan en 0.06 microsegundos, de modo que teóricamente esos pulsos podrían emitirse. Sin embargo el fabricante del *Driver* especifica que los pulsos de entrada no deben superar los 200KHz, esto es 5 microsegundos(2.5 us alto y 2.5us bajo).

Enfocando la atención al motor, este definitivamente no será capaz de superar algún valor alto de RPM. Especialmente porque los pasos rápidos implican pulsos de frecuencia alta y falla la magnetización de las bobinas, debilitando el torque y finalmente dejando el motor sin movimiento.

Por lo anterior, podría haber limitaciones o problemas si el período de la señal a producir baja de 1 segundo (lo que dará intervalos de 50 milisegundos) y/o si se aumenta la *amplitud* de este movimiento sinusoidal, lo que obligará a girar más rápido al motor y podría alcanzar esa RPM crítica en la que desaparece el torque y se quedará inmóvil vibrando. Es difícil estimar teóricamente esta situación y sólo la experimentación podrá dar evidencia y establecer ese límite infranqueable e inherente al motor.

4. MATERIAL Y METODOS

En la construcción del actuador electromecánico se utilizó lo siguiente:

1. Motor Nema 34
2. Driver DM860
3. Fuente del poder
4. Placa Arduino UNO
5. Husillo de bolas de 16mm de diámetro y paso 4mm.
6. Carcasa donde se alojaron los componentes mecánicos

Para mejor compresión de método utilizado, este se desglosa en etapas:

Etapa 1: Definición de requerimientos específicos para las pruebas a ejecutar

1.1 Movimientos requeridos:

Los movimientos solicitados a ejecutar por el motor, corresponden a:

- Tipo “Movimiento armónico” (función seno).
- Tipo "Diente de sierra"

Se efectuaron cálculos con las fórmulas ya descritas, en una secuencia de hasta 20 segmentos o puntos de control correspondientes a fracciones del tiempo total en el que se desea ejecutar el ciclo de movimiento. Los cuales se verán visualizados en la graficas de los resultados.

Todos los cálculos se realizaron con ayuda de planilla Excel, donde fueron registrados en tablas. Como se observa en la Tabla N°3.

Tabla N°3: Excel con valores movimiento armónico [Fuente: Elaboración propia]

		$Asin(wt)$	
Segmento	tiempo (seg)	Giro (rad)	Giro (pasos)
1	0,000	0,000	0,000
2	0,100	0,013	0,411
3	0,200	0,025	0,782
4	0,300	0,034	1,076
5	0,400	0,040	1,265
6	0,500	0,042	1,330
7	0,600	0,040	1,265
8	0,700	0,034	1,076
9	0,800	0,025	0,782
10	0,900	0,013	0,411
11	1,000	0,000	0,000
12	1,100	-0,013	-0,411
13	1,200	-0,025	-0,782
14	1,300	-0,034	-1,076
15	1,400	-0,040	-1,265
16	1,500	-0,042	-1,330
17	1,600	-0,040	-1,265
18	1,700	-0,034	-1,076
19	1,800	-0,025	-0,782
20	1,900	-0,013	-0,411
	2,000	1,01E-16	3,22E-15

A través del movimiento armónico como eje principal, se puede ejecutar otros movimientos, como por ejemplo el movimiento "diente de sierra". El cual se obtiene utilizando las amplitudes máximas en sentido horario y anti horario. (Tabla N°4).

Tabla N°4: Excel con valores movimiento diente de sierra [Fuente: Elaboración propia]

		$Asin(wt)$	
Segmento	tiempo (seg)	Giro (rad)	Giro (pasos)
1	0,000	0,000	0,000
2	0,100	0,013	0,411
3	0,200	0,025	0,782
4	0,300	0,034	1,076
5	0,400	0,040	1,265
6	0,500	0,042	1,330
7	0,600	0,040	1,265
8	0,700	0,034	1,076
9	0,800	0,025	0,782
10	0,900	0,013	0,411
11	1,000	0,000	0,000
12	1,100	-0,013	-0,411
13	1,200	-0,025	-0,782
14	1,300	-0,034	-1,076
15	1,400	-0,040	-1,265
16	1,500	-0,042	-1,330
17	1,600	-0,040	-1,265
18	1,700	-0,034	-1,076
19	1,800	-0,025	-0,782
20	1,900	-0,013	-0,411
	2,000	1,01E-16	3,22E-15

Etapa 2: Desarrollo de Programas de Control con Arduino-UNO

Ya establecida la secuencia de movimientos requeridos de tipo "Diente de Sierra" y de tipo "armónico". Estos consignados en dos planillas Excel de confección propia. Se solicitó apoyo a un profesional con conocimientos de programación en lenguaje compatible con Arduino-UNO. El cual, hizo ver las limitaciones que presenta el programa, ya mencionadas en el marco teórico.

Los programas fueron diseñados en conjunto al programador con el menor número de instrucciones posibles que implicaran cálculos reiterativos por el microprocesador, con la finalidad de obtener los pulsos en los tiempos necesarios, optando por introducir valores en matrices unidimensionales. El detalle de la programación de ambos dos movimientos se especifican en **Anexo N°1 "Códigos de programación".**

La placa Arduino-UNO se encuentra conectada al PC mediante cable USB, una vez efectuada la compilación de la rutina desarrollada, utilizando los valores de tabla Excel, se procede a su carga en la memoria EEPROM del procesador del Arduino-UNO y se pasa al modo de ejecución , esto con monitoreo en pantalla del PC, permitiendo controlar el comportamiento del programa. Esta etapa finalizó con los programas depurados sin haber sido aún conectados a los motores.

Etapa 3: Conexiones

Con los materiales ya mencionados, se procede a realizar las conexiones. De la interacción con el ingeniero electrónico asesor, se obtuvo una guía operativa para el procedimiento de conexiones a realizar entre los elementos utilizados. Indica que se debe utilizar un voltaje de +5V CC, requeridos por los drivers y que debían ser proporcionados por Arduino-UNO.

La Figura N° 10 indica las conexiones que deberá hacerse para poner el motor en funcionamiento.

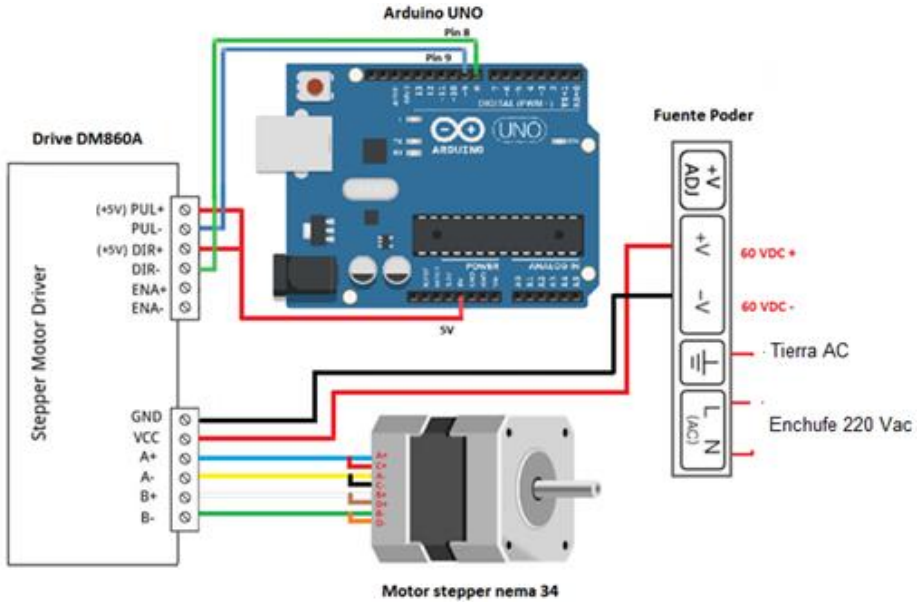

Figura N° 10: Conexiones placa Arduino-UNO y Motor Nema 34 [Fuente: Elaboración propia]

- **Driver y Motor:** El motor paso a paso se conecta al driver a través de sus salidas, que según las especificaciones del fabricante serían a través de los siguientes cables en paralelo: azul/rojo, amarillo/negro, blanco/café y verde/naranjo. Estas conexiones en paralelo se deben conectar a la vez a las salidas del controlador A+, A-, B+ y B- respectivamente. El driver está conectado a la fuente de alimentación DC, a su vez la fuente está conectada a la corriente alterna de 220v a través de sus salidas L (Fase), N (neutro) y G(tierra), esto para energizar el driver y por consiguiente el motor.

- **Arduino y Driver:** Las señales del driver son controladas por los pines 9 y 8 del Arduino que envían pulsos a entradas P- y D- del driver, además de energizar con 5V las entradas P+ y P-, según se especifica en la tabla 1, desde la placa Arduino. Desde el Arduino salen las secuencias de PULSOS de encendido de las bobinas para ejecutar cada PASO. El Arduino a su vez se energiza a través de cable USB.

- **Secuencia de encendido:** Al cargan el programa en el Arduino este queda guardado en la placa, por lo que al encender comienza la ejecución inmediata de programa que tiene en su EEPROM. Puede utilizarse botón Reset de Arduino para reinicio de programa en Arduino sin desconectar de fuente alimentación.

- **Funcionamiento general:** Como ya se mencionó, la ejecución se empieza en el Arduino luego de cargar el programa con las secuencias predefinidas, a través de sus pines 9 y 8 envía al driver la secuencia de pulsos para ejecutar ciertos movimientos, tanto a la señal de pulsos P- como a la de dirección D-. El driver procesa esta señales que irán al motor a través de A+,A-,B+ y B- lo que generaran los movimiento angular o PASOS en el motor. Así comenzará a girar el motor en un sentido u otro logrando obtener los desplazamientos requeridos y en este caso en particular, accionar un mecanismo con movimientos de tipo diente de sierra y armónico, que cuenta con un husillo de bolas, que transforma la rotación y torque del motor en desplazamiento y fuerza.

Etapa 4: Pruebas iniciales de programas con motor Stepper

Una vez que se dispuso de los programas desarrollados ya depurados, se procedió a hacer múltiples pruebas con motor NEMA34, utilizando la función de librería STEPPER de Arduino con valores de pasos físicos, se pudo comprobar que esta rutina NO considera el factor 4 de la fórmula de control por pulsos ya comentada en el marco teórico. Por lo que el giro real del motor observado en la práctica, era la cuarta parte de lo establecido para un intervalo de tiempo definido. Se midió la frecuencia de los pulsos y resultó ser efectivamente la calculada sin el factor 4 de la fórmula y se hizo la modificación para todos los programas denominándolo "factor diseño eléctrico" en planilla de cálculo Excel.

Etapa 5: Montaje y ejecución de pruebas de prototipo

Se montó el actuador junto al motor stepper Nema34, sobre una mesa de superficie nivelada de 2m x 1m. Este dispositivo motriz a su vez, se encuentra acoplado a un sistema de plataforma sobre rieles de movimiento lineal.

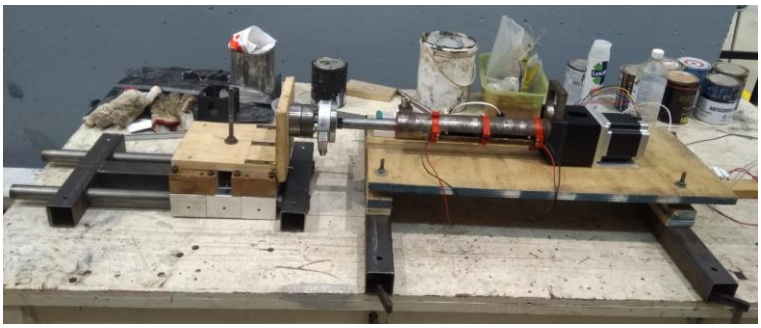

Figura N° 11: Montaje prototipo de ensayo [Fuente: Elaboración propia]

Etapa 6: Calibración del LVDT

Los LVDT (transformadores diferenciales variables lineales) son sensores de posición lineal. Se utilizan para medir el desplazamiento lineal y posición en distancias relativamente cortas. Un LVDT consiste en un tubo que contiene un eje que se mueve libremente. La base del tubo está montada en una posición fija y el extremo de la varilla se fija a un objeto cuya posición se moverá de forma lineal (hacia adelante y hacia atrás).

Este dispositivo al igual que la celda de carga para obtener un buen resultado es necesario aplicarle una calibración. Estos sensores de desplazamiento diferencial (LVDTs) se calibran mediante el uso de micrómetros; la relación del desplazamiento en milímetros (mm) genera una señal de voltaje de corriente directa de 0-10 Volt al aplicar los acondicionadores de señal. (Tabla N° 5)

Tabla N°5: Calibración LVDT [Fuente: Elaboración propia]

Calibración LVDT		
Escalón	Desplazamiento	Voltaje
0	0	10,42
1	20,13	9,324
2	40,69	8,347
3	59,84	7,338
4	80,51	6,294
5	100,61	5,242
6	121,09	4,179
7	140,12	3,179

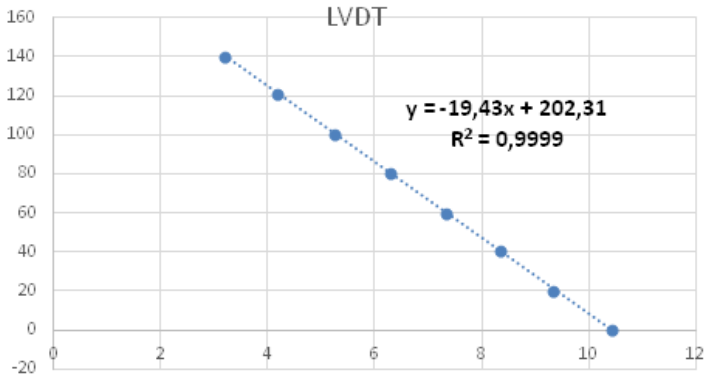

Figura N° 12: Calibración LVDT [Fuente: Elaboración propia]

Etapa 7: Calibración de celda de carga

Las celdas de carga son transductores, esto quiere decir que son mecanismos que transforman una energía de entrada de cierto tipo en una energía de salida de otro. En este ensayo, las celdas de carga transforman la energía mecánica que ejerce el motor stepper sobre la plataforma (resorte) en una señal eléctrica que es enviada a otro dispositivo que la procesa para mostrar un resultado. Visto de otra forma, las celdas de carga también pueden considerarse un tipo de sensor de fuerza, ya que toman un estímulo mecánico y lo transforman en una señal eléctrica medible por un sistema de control, que lo expresa de forma visual a través de un indicador.

Para lograr que la celda de carga funcione como corresponde, no solo basta con ajustarla y calibrarla en fábrica, sino que también es preciso verificarla en el lugar de instalación. La calibración es un proceso en el cual se aplican pesos conocidos a la celda de carga con fines de control. Por lo tanto, si se coloca un kilo sobre la celda de carga, el peso indicado debe ser de un kilo y no otro, con una desviación tanto menor cuanto mayor sea la clase de precisión.

A continuación, se entrega tabla de calibración.

Tabla N° 6: Calibración Celda de Carga [Fuente: Elaboración propia]

Calibración Celda de Carga					
Escalón	Carga adicional	Carga Acumulada (Kg)	Carga Acumulada (N)	Voltaje	
0	0	0	0	0,2879	0,03
1	4	4	39,24	0,8812	4,02
2	4	8	78,48	1,473	7,99
3	4	12	117,72	2,07	11,99
4	4	16	156,96	2,662	15,97
5	4	20	196,2	3,261	19,99
6	4	24	235,44	3,868	24,06
7	4	28	274,68	4,45	27,97
8	4	32	313,92	5,049	31,99
9	4	36	353,16	5,651	36,03
10	4	40	392,4	6,225	39,88
11	5	45	441,45	6,984	44,98
12	5	50	490,5	7,752	50,13
13	8	58	568,98	8,922	57,98

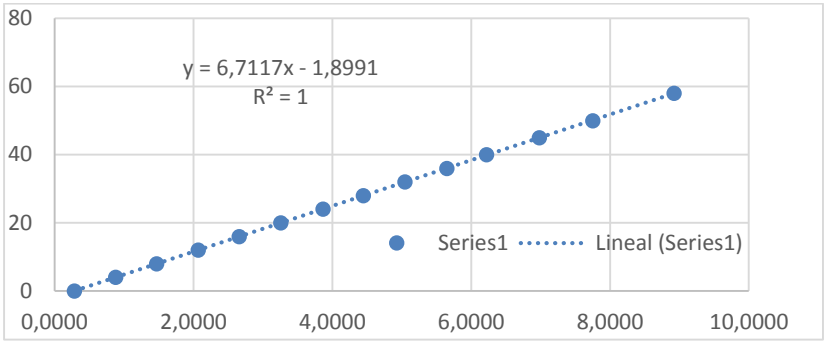

Figura N° 13: Calibración de celda de carga [Fuente: Elaboración propia]

En esta grafica se muestra la relación entre el peso y voltaje entregado por cada carga de peso, obteniendo un ajuste lineal perfecto.

5. RESULTADOS

Una vez realizada las calibraciones de los dispositivos utilizados en los ensayos, se dio inicio a esta experiencia.

El modelo de prueba fue ensayado sobre una plataforma impulsado por un actuador electromecánico, en el cual se utilizó un par de resortes para ver el comportamiento del motor con los movimiento programados en Arduino (Movimiento armónico y de tipo Diente de Sierra). Dicho resorte se instaló en la plataforma junto a los dispositivos de medición, los cuales son una Celda de carga y un LVDT, anteriormente descritos.

La información obtenida por cada sensor fue procesada utilizando el software Labview, para luego a través de Matlab aplicarle un filtro de frecuencia a los datos para eliminar la contaminación que se pueda producir durante un ensayo.

Se realiza el ensayo y se evidencio movimientos de tipo armónicos y de dientes de sierra con distintas amplitudes, velocidades, y distintos números de ciclos. Logrando desarrollar el desafío propuesto.

Ensayo con movimiento tipo Diente de Sierra

A continuación, se muestran las gráficas obtenidas:

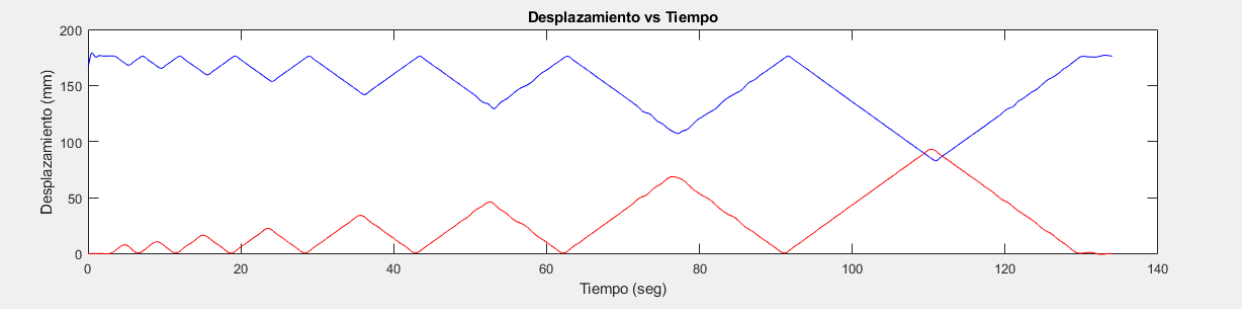

Figura N° 14: Desplazamiento vs tiempo "Movimiento Diente de Sierra" [Fuente: Elaboración Propia]

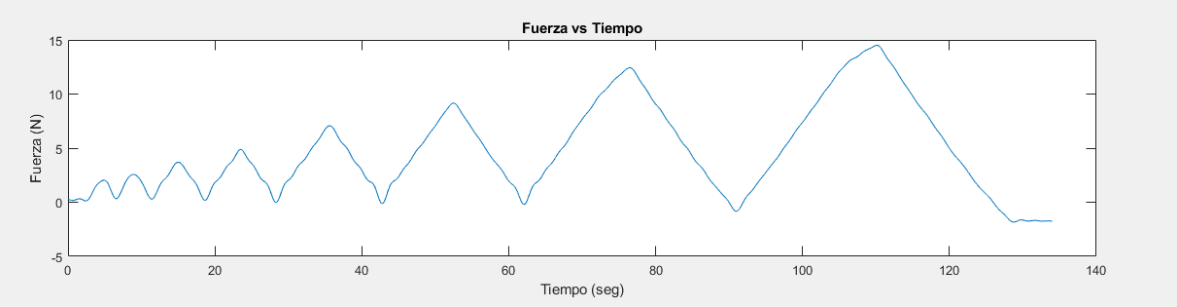

Figura N° 15: Fuerza vs tiempo "Movimiento Diente de Sierra" [Fuente: Elaboración propia]

En las Figuras N° 14 y N° 15 se muestran las mediciones realizadas por los dispositivos, en donde se ve desplazamientos y fuerza filtrados, correspondientes a la respuesta del dispositivo con distintas amplitudes de desplazamientos y fuerza a través del tiempo. Obteniendo un gráfico claro del movimiento solicitado.

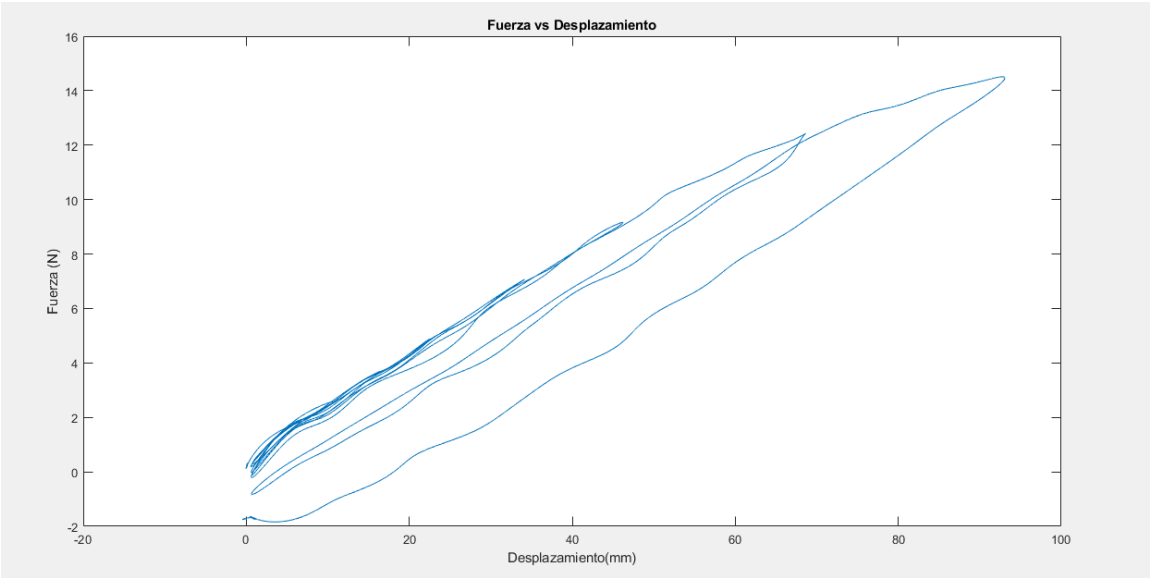

Figura N° 16: Fuerza vs Desplazamiento "Movimiento Diente de Sierra" [Fuente: Elaboración Propia]

En la Figura N° 16, se puede apreciar que al principio con desplazamientos pequeños, todas las curvas pasan por el mismo punto, pero al ir aumentando el desplazamiento, se ve un cambio de pendiente, entrando en fluencia el acero del resorte. Al cambiar la pendiente, la curva no vuelve por el mismo camino, es decir, el resorte tiene deformación plástica (permanente).

6. CONCLUSIONES

Es totalmente factible en la práctica, programar el control de un motor stepper tipo NEMA 34 para producir movimientos con un forzante controlado por movimiento uni-axial utilizando un dispositivo electrónico de la estructura tipo Arduino -UNO, existente en el mercado y a bajo costo.

Para este proyecto fue necesario, contar con orientación de ingeniero electrónico, para entender el desarrollo de un motor stepper y programación en conjunto a Arduino- UNO. Que a pesar de ser un programa gratuito y con información en su plataforma web, se debe tener algunos conocimientos básicos de programación.

Es posible y útil, contar una herramienta de apoyo planilla Excel, para evitar errores de cálculo y digitación de instrucciones modificables en cada segmento de control digital, según tipo de movimiento de prueba en programación.

El potencial del micro controlador de Arduino UNO, es limitado, pero útil para los rangos de movimientos tipo “diente de serrucho” y "armónico", como los desarrollados en este ensayo. Esto se debe a que usa un *lenguaje* de alto nivel (C++) y los programas del usuario son escritos en él, antes de cargar el programa en la *Memoria de programa* del ATmega328 el IDE realiza una compilación, es decir una transformación desde instrucciones de alto nivel a las que realiza la máquina. Para este caso *establece limitaciones a la velocidad de giro*, ya que se debe tener en cuenta que una aparentemente simple instrucción de lenguaje de Arduino puede dar origen a cientos de instrucciones de la máquina para lograr su ejecución.

Las limitaciones del motor nema34: Al aumentar de velocidad (RPM) debilita el torque del motor, lo que conlleva a una falla de la magnetización de las bobinas, debilitando el torque y finalmente dejando al motor sin movimiento. Por lo que se siguiere usar 20 subperiodos para desarrollar el movimiento, como se utilizó en este proyecto.

Adicionalmente esta experiencia podrá ser utilizado tanto por estudiantes como por académicos, como apoyo a la docencia o en el desarrollo de investigación. Ya que brinda una guía clara para su uso.

7. REFERENCIAS

[1] Academia de Ingeniería de México. (2016, 13 junio). *La investigación experimental en Ingeniería Estructural: Una opción para apoyar el desarrollo de la Ingeniería Mexicana*. Slideshare.Net. https://es.slideshare.net/AcademiaDeIngenieriaMx/la-investigacin-experimental-en-ingeniera-mexicana-una-opcin-para-apoyar-el-desarrollo

[2] Ávila-Valerio, J., & Zúñiga-Campos, A. (2020). Diseño, ensamblaje y validación de una mesa sísmica para el análisis estructural de modelos a escala reducida.

[3] Ingreso a página web el 31 de Mayo 2022: / https://www.arduino.cc/en/Guide

[4] Ingreso a página web el 30 de Mayo 2022: https://www.especificarmag.com.mx/todo-sobre-los-actuadores.html

[5] Ingreso a página web 30 de Mayo 2022: http://www.longs-motor.com/

[6] Ingreso a página web 30 de Mayo 2022: http://www.longs-motor.com/

[7] Ingreso a página web el 30 de Mayo 2022: https://es.slideshare.net/andrexlu/actuadores-2112343

[8] AguascaCabot, G. (2021). *" Desarrollo de una aplicación Java y de un manual para el control mediante Arduino de Maquetas Didácticas utilizadas en Teoría de Máquinas y Mecanismos y Proyecto II2* (Bachelor'sthesis, UniversitatPolitècnica de Catalunya).

[9] *Arduino, ¿Qué es y para que sirve? Aprende con Tutoriales y Proyectos*. (2021, 22 junio). Descubrearduino.com. https://descubrearduino.com/

[10] Torrente Artero, Óscar. ARDUINO Curso práctico de formación. México: Alfaomega Grupo Editor S.A de C.V. 2013. ISBN 978-607-707-648-3

[11] R. (2020, 6 agosto). *Cómo descargar e instalar Arduino y el IDE en nuestro ordenador*. Descubrearduino.com. https://descubrearduino.com/instalar-arduino/

[12] *Arduino Uno Rev3*. (s. f.). ArduinoOfficial Store. Recuperado 3 de octubre de 2021, de https://store.arduino.cc/arduino-uno-rev3

[13] Pe, I. (2021, 10 agosto). ▷ *Comparativa de todas las placas Arduino*. ComoHacer.eu. https://www.comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/

[14] *char - Arduino Reference*. (s. f.). Char [Data Types]. Recuperado 3 de octubre de 2021, de https://www.arduino.cc/reference/en/language/variables/data-types/char/

[15] *int - Arduino Reference*. (s. f.). Int [Data Types]. Recuperado 3 de octubre de 2021, de https://www.arduino.cc/reference/en/language/variables/data-types/int/

[16] *long - Arduino Reference*. (s. f.). Long [Data Types]. Recuperado 3 de octubre de 2021, de https://www.arduino.cc/reference/en/language/variables/data-types/long/.

ANEXO 1: CODIGOS DE PROGRAMACIÓN

- **Código movimiento armónico.**

```
//MOMBRE PCANema34P400T2000R1v20220617h12m59
#include <Stepper.h>
Stepper myStepper(400,9,8); //Velocidad fija
unsigned long Tiempofinal;  // definimos RELOJ
unsigned long Tiempo =0; // define auxiliar
// Valores velocidad según EXCEL
int veloArray [20] = {988, 888, 708, 452, 156, 156, 452, 708, 888, 988, 988, 888, 708, 452, 156,
156, 452, 708, 888, 988}  ; // PPR = 400  Periodo ms = 2000 Reduccion mecanica = 1
long pulsoArray [20] = {658, 593, 471, 302, 104, -104, -302, -471, -593, -658, -658, -593, -471, -
302, -104, 104, 302, 471, 593, 658}  ; // PPR = 400  Periodo ms = 2000 Reduccion mecanica = 1
void setup() {
Serial.begin(9600);
Serial.println("//MOMBRE PCANema34P400T2000R1v20220617h12m59");
}
void loop() {

 Tiempo=millis();   // mide el reloj del momento y lo guarda
Serial.println(Tiempo);
int  pulso = 0;
int  Velo = 0;
 for (int k=1; k <= 1; k++) // Establece iteración  tabla completa
 {
int j=0;
while(j <20)  // Ciclo “mientras que.. ejecute.
{
Velo = (veloArray [j]);
pulso = pulsoArray[j];
Serial.println("Velocidad RPM ");
 Serial.println(Velo / 4);
Serial.println("Pulso Motor = ");
Serial.println(pulso / 4);
myStepper.setSpeed(Velo);
myStepper.step(pulso);
j++;   // aquí se incrementa j y termina el ciclo con j=20, que no se hace
 }
 Tiempofinal=millis();  //mide el reloj del término del ciclo  (pos 20 datos)
Tiempofinal=Tiempofinal- Tiempo; // Calcula diferencia
Serial.print("me demore=");
Serial.println(Tiempofinal);  // Imprime lo que se demoró en el lazo
Serial.print(" millisegundos en emitir la secuencia");
```

```
 delay (0);
 }
 exit(0); // detenido para que no repita, ya que repite el ciclo cada 1 segundo al infinito
}
// Fin
```

- **Código movimiento Diente de sierra:**

```
//MOMBRE PCANema34P400T60R3v20220617h13m10
#include <Stepper.h>
Stepper myStepper(400,9,8); //Velocidad fija
unsigned long Tiempofinal;  // definimos RELOJ
unsigned long Tiempo =0; // define auxiliar
// Valores velocidad según EXCEL
int veloArray [20] = {0, 240, 240, 240, 240, 240, 240, 240, 240, 240, 240, 240, 0, 0, 0, 0, 0, 0, 0, 0}  ; // PPR = 400  Periodo ms = 60 Reduccion mecanica = 3
int pulsoArray [20] = {0, 573, 764, 1146, 1528, 2292, 3056, 4584, 6112, 7639, 9167, 10695, 0, 0, 0, 0, 0, 0, 0, 0}  ; // PPR = 400  Periodo ms = 60 Reduccion J27mecanica = 3
void setup() {
Serial.begin(9600);
Serial.println("//MOMBRE PCANema34P400T60R3v20220617h13m10");
}
void loop() {

 Tiempo=millis();   // mide el reloj del momento y lo guarda
Serial.println(Tiempo);
int  pulso = 0;
int  Velo = 0;
int j=0;
while(j <20)  // Ciclo “mientras que.. ejecute.
{
Velo = (veloArray [j]);
pulso = pulsoArray[j];
Serial.println("Velocidad RPM ");
 Serial.println(Velo / 4);
Serial.println("Pulso Motor = ");
Serial.println(pulso / 4);
myStepper.setSpeed(Velo);
myStepper.step(pulso);
j++;   // aquí se incrementa j y termina el ciclo con j=20, que no se hace
 }
 Tiempofinal=millis();  //mide el reloj del término del ciclo  (pos 20 datos)
Tiempofinal=Tiempofinal- Tiempo; // Calcula diferencia
```

```
Serial.print("me demore=");
Serial.println(Tiempofinal);  // Imprime lo que se demoró en el lazo
Serial.print(" millisegundos en emitir la secuencia");
 delay (1000);
 exit(0); // detenido para que no repita, ya que repite el ciclo cada 1 segundo al infinito
}
// Fin
```