

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería



**Diseño de un sistema de motores inteligentes para
aplicaciones de robótica de alta precisión y desempeño**

Trabajo de graduación presentado por Juan Pablo Isaac Valenzuela Saravia para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería



**Diseño de un sistema de motores inteligentes para
aplicaciones de robótica de alta precisión y desempeño**

Trabajo de graduación presentado por Juan Pablo Isaac Valenzuela Saravia para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

Vo.Bo.:

(f) _____
MSc. Miguel Zea

Tribunal Examinador:

(f) _____
MSc. Miguel Zea

(f) _____
PhD. Luis Rivera

(f) _____
Ing. Kurt Kellner

Fecha de aprobación: Guatemala, 3 de octubre de 2022.

Lista de figuras	VIII
Lista de cuadros	IX
1. Introducción	1
2. Antecedentes	3
3. Justificación	9
4. Objetivos	11
5. Alcance	13
6. Marco teórico	15
7. Diseño, manufactura y ensamblaje del actuado	27
7.1. Selección de componentes	27
7.1.1. Microcontrolador	27
7.1.2. <i>Encoder</i>	29
7.1.3. Sensor de efecto Hall	30
7.1.4. Motores <i>stepper</i>	30
7.2. Diseño CAD	31
7.3. Ensamblaje	35
7.4. Posición y velocidad	37
7.5. Diseño de PCB	38
7.6. Pruebas de motores	39
8. Protocolo de comunicación daisy chain	41
8.1. SPI	41
8.2. UART	42
8.3. Programación	42
8.3.1. Control de motores	43
8.3.2. Protocolo de comunicación final	44

8.4. Resultado final	45
9. Conclusiones	49
10. Recomendaciones	51
11. Bibliografía	53
12. Anexos	57
12.1. Proceso iterativo	57
13. Glosario	67

Lista de figuras

1.	Conexión de los motores y el microcontrolador maestro	5
2.	Conexión en cadena de Arduino Uno	7
3.	Conexión <i>Daisy-Chain</i> [11]	17
4.	Conexión SPI	18
5.	Conexión SPI en cadena	18
6.	Interfaz SMAART	19
7.	Transmisión planetaria [16]	19
8.	Partes del engranaje planetario [18]	20
9.	Partes del <i>Harmonic Drive</i> [21].	21
10.	Partes de la transmisión cicloidal [23]	22
11.	Elementos de un <i>encoder</i> [24].	23
12.	Proceso de diseño de Norton[27]	26
13.	Arduino pro mini[35].	28
14.	<i>Encoder</i> óptico[35].	29
15.	<i>Encoder</i> rotativo[35].	29
16.	Sensor Hall[35].	30
17.	DRV8825[35].	30
18.	Motor utilizado NEMA 17[35].	30
19.	Vista isométrica de los motores inteligentes.	31
20.	Parámetros del aceleración de diseño.	32
21.	Constantes de los engranajes.	33
22.	Recubrimiento de la carcasa.	34
23.	Configuración del acelerador de diseño en iteración 9	36
24.	Rotación de un <i>encoder</i> [38]	37
25.	Parámetros de diseño de PCB[40].	38
26.	Placa del actuador	39
27.	Resultados <i>Daisy-chain</i> con SPI.	42
28.	Rebote del <i>encoder</i>	43
29.	Prototipo final	46
30.	Comunicación con prototipo final	47

31.	Prototipo del motor	57
32.	Primera iteración	58
33.	Segunda iteración	59
34.	Tercera iteración	60
35.	Cuarta iteración	60
36.	Quinta iteración (variación 1)	61
37.	Quinta iteración (variación 2)	61
38.	Sexta iteración (variación 1)	61
39.	Sexta iteración (variación 2)	62
40.	Tapadera de carcasa	62
41.	Séptima iteración	63
42.	Octava iteración	63
43.	Novena iteración	64
44.	Décima iteración	64
45.	Primer Ensamble Físico	64
46.	Pruebas de los Motores	65
47.	Primer par de prototipos	65

Lista de cuadros

1.	Reducciones posibles del engranaje planetario [18].	20
2.	Reglas de diseño utilizadas.2	38
3.	Conexión óptima del motor unipolar	39
4.	Constantes finales de control	44
5.	Tiempos de comunicación con el protocolo	45
6.	Dispositivos máximos	45

CAPÍTULO 1

Introducción

Actualmente se está trabajando en la cuarta fase del desarrollo del brazo para HUMANA, pero se desea reemplazar los motores por actuadores inteligentes con un protocolo de comunicación de simple conexión que permita controlarlos todos desde un microcontrolador maestro[1]. Sin embargo, en la fase anterior no se logró cumplir con todas las expectativas, especialmente en el lado de control de posición para el motor *stepper*[2]. En este proyecto se desarrollarán actuadores inteligentes de alta precisión con el objetivo de utilizarlos próximamente en el proyecto del brazo robótico HUMANA y en otros proyectos de robótica en el departamento. Se busca que los actuadores inteligentes permitan su interconexión tipo *Daisy-chain*. Se desea que su construcción no represente un costo elevado y que sus componentes se encuentren localmente. Por esta razón se trabajó con el *atmega328p* para controlar los motores NEMA 17 y un *ESP32* como microcontrolador maestro que coordine todo el sistema.

Para lograrlo se utilizará el protocolo de comunicación desarrollado por Yau y se evaluará la facilidad de implementarlo en SPI o de lo contrario seguirlo usando basado en UART. [1]. Adicionalmente se emplearon sensores de efecto Hall y encodificadores para incrementar la precisión nativa de los motores.

CAPÍTULO 2

Antecedentes

En la fase previa del presente trabajo se llevó a cabo el desarrollo de motores inteligentes utilizando motores *stepper* Nema 23 y Nema 17. Yau, desarrolló la infraestructura de comunicación entre motores, la cual permite que los motores se interconecten unos con los otros formando un Lazo cerrado tipo *daisy-chain* con el microcontrolador maestro. Cada motor estuvo integrado con un microcontrolador *Arduiono UNO*, el cual fue el responsable de implementar el sistema de control del motor y de manejar el protocolo de comunicación. Desde el microcontrolador maestro se pudo ajustar los parámetros de los controladores individualmente y se monitoreó la información relevante de cada motor [1].

Motores inteligentes

Lynxmotion es uno de los fabricantes más antiguos de kits de robots, incluidos brazos robóticos, robots bípedos que caminan, cuadrúpedos, hexápodos, vehículos con orugas, ruedas y más. Anteriormente, esta empresa desarrolló un modelo de motor inteligente, el cual se denomina *LSS Servo*. Los servos inteligentes *Lynxmotion* (LSS) son actuadores nuevos, compactos, modulares y configurables diseñados para ser una evolución del servo RC estándar para su uso en robótica con varios grados de libertad y capacidad de interconectarse entre sí y con un microcontrolador maestro. Inteligente significa que los parámetros de cada servo son modificables y configurables por el usuario, hay retroalimentación del sensor, características de seguridad incorporadas y, aunque están destinados a conectarse a un microcontrolador, se pueden usar como actuadores RC avanzados. El protocolo de comunicación de los motores se basó en el protocolo SSC-32 / 32U y hace que la comunicación, el control y la configuración de los servos sean fáciles e intuitivos.[3]

DYNAMIXEL es una marca de actuador inteligente desarrollada por *ROBOTIS* para su uso en cualquier sistema robótico. El nombre *DYNAMIXEL* se deriva de dos palabras: *Dynamic* y *Cell*. El término hace referencia a un actuador inteligente todo en uno. Existen varios modelos de motores *DYNAMIXEL* con diferentes capacidades [4]. Lo que los diferencia de otros tipos de motores es:

- Diferentes modos para control de posición, control de velocidad, control de corriente (par), control PWM y más.
- La retroalimentación en tiempo real de posición, velocidad, corriente (torque/carga), temperatura, voltaje, estado del hardware y más están disponibles a través del firmware DYNAMIXEL.
- El control PID completo para posición y velocidad permite a los usuarios ajustar los movimientos.
- *Daisychainable*, es decir que se interconectan unos con otros y comparten un bus de datos.
- Soporta comunicación RS-485 y TTL (*Half-duplex*/asíncrona).
- Es compatible con C/C++, PYTHON, JAVA, MATLAB, LABVIEW, ROS, R+, SDK y librerías para Windows, Linux y Mobile.

Además del desarrollo de motores, la compañía *Trinamic* desarrolla *drivers* y módulos para el control de motores enfocado en el *hardware*. Su área de especialidad es la robótica avanzada, Internet de las cosas, dispositivos médicos (que funcionan con baterías), fabricación aditiva, prótesis entre otros. La tecnología que utilizan los *drivers* es llamada *MicroPlyer*. Por cada señal *STEP* enviada al controlador del motor paso a paso, *MicroPlyer* puede producir hasta 256 micropasos para un mayor rendimiento. La unidad interpoladora flexible de *Trinamic* lo hace interpolando el tiempo entre dos pulsos de paso, determinando la tasa de paso midiendo el intervalo de tiempo del período de paso anterior y dividiéndolo en partes iguales. Estos controladores también pueden realizar *Field Oriented Control*. El control orientado al campo es la forma más eficiente de impulsar motores. Transforma las corrientes de fase reales de sistemas de coordenadas fijos en el estator a sistemas de coordenadas síncronos de campo mediante el uso de dos transformaciones matemáticas simples. Se desarrolló originalmente para aplicaciones de gama alta y difíciles de implementar pero *Trinamic* ofrece control orientado al campo como un elemento básico de hardware fácil de usar [5].

Otra compañía que también ofrece soluciones para el control de motores es *Odrive*, pero sus controladores se enfocan más en el *software*. La comunicación con un ODrive consiste en una serie de operaciones de punto final. En teoría, una operación de punto final puede ser cualquier tipo de datos serializados de cualquier manera. Existe una implementación de serialización predeterminada, pero si se desea usar un tipo personalizado se debe crear. Los puntos finales disponibles se pueden enumerar leyendo el JSON desde el punto final 0 y pueden ser diferentes para cada interfaz de comunicación aunque en la práctica usualmente no lo son [6].

Cada operación de punto final puede enviar bytes a un punto final (referenciado por su ID) y al mismo tiempo recibir bytes del mismo punto final (comunicación *full Duplex*). La semántica de estas cargas útiles es específica para el tipo de cada operación punto final, cuyo nombre se indica en el *JSON*. Hay una versión basada en paquetes y una variante basada en flujo del protocolo. Cada variante se emplea según corresponda. Por ejemplo, USB ejecuta la variante basada en paquetes de forma predeterminada, mientras que UART ejecuta la variante basada en flujo [6].

Diseño de un sistema de motores inteligentes en conjunto con un sistema de control del brazo robótico asistencial para cirugías estereotácticas

El objetivo principal del proyecto fue diseñar un sistema de motores inteligentes que incorpore motores *stepper*, *drivers*, sensores y microcontroladores para luego implementarlo en el brazo robótico. También se buscó diseñar un protocolo cuya función principal fue entrelazar los módulos de reconocimiento de imágenes y control remoto. Asimismo, se diseñó un lazo de control manual a lazo abierto y otro a Lazo cerrado únicamente para los movimientos finos. Después se desarrolló un sistema de motores inteligentes que combinó motores *stepper* y microcontroladores con comunicación tipo *daisy-chain*.

Para llevar a cabo esa tarea se eligió el modelo Dynamixel AX-12A, los cuales se utilizaron para generar el movimiento rotacional de 6 juntas. Sin embargo, Yau afirma que se descartó el primer prototipo debido a la construcción asimétrica del brazo. Se detalla en su trabajo que el segundo prototipo tuvo un diseño más simétrico y eso aumentó la estabilidad del sistema. La problemática de esta iteración fue que los motores no eran suficientemente robustos para soportar el peso de las primeras tres juntas, las cuales estaban próximas a la base. Finalmente Yau explica que para el prototipo numero 3 se cambiaron los motores problemáticos por unos modelo MX-106, del mismo fabricante. Finalmente se utilizó un arduino Mega para controlar los motores porque éste tiene una alta cantidad de entradas y salidas digitales.

Para la segunda etapa, Yau decidió cambiar los servomotores de la primera etapa por motores paso a paso o *stepper*. Los motores stepper son motores eléctricos de corriente directa sin escobillas. Dividen una rotación completa en varios pasos iguales.[7]

Con base en los cálculos mecánicos de la construcción del robot, se determinó que los motores Nema fueron adecuados para las juntas principales, debido al torque y robustez requeridos, y los Nema 17 para las juntas en la parte superior del brazo.

Se seleccionó el *driver* TB67S128FTG y se colocó en el motor Nema 23 en la base. En el caso del TB67S249FTG, se le colocará al segundo Nema 23 porque puede entregar 4.5 A, lo cual es suficiente para su aplicación. Por último se utilizó el *driver* DRV8825 para el resto de motores porque no manejaron cargas elevadas y se utilizaron para realizar movimientos con la herramienta [1].

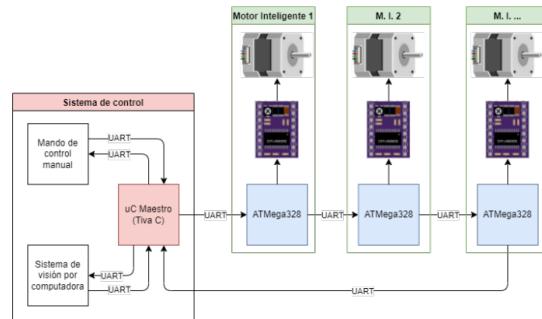


Figura 24: Diagrama de comunicación entre los dispositivos del brazo robótico

Figura 1: Conexión de los motores y el microcontrolador maestro

Luego se desarrolló un sistema de comunicación por UART para crear un Lazo cerrado entre los motores y el microcontrolador maestro. Se utilizó un sistema *daisy-chain* en el cual se conectó un ATMega328 al *driver* correspondiente y se utilizó su módulo UART para comunicarse con los microcontroladores adyacentes. Se eligió este microcontrolador por sus recursos, disponibilidad en el mercado, precio y facilidad de implementación. Se conectó la salida de un microcontrolador a la entrada del siguiente, el primer microcontrolador se conectó a la salida UART del microcontrolador maestro mientras que el último se conectó a la entrada del UART maestro. Para comunicarse con los motores se dispone de una serie de comandos en donde la información viaja de microcontrolador a microcontrolador, de modo que al llegar al destino correcto se genere un mensaje de confirmación el cual regresa al microcontrolador maestro. De esta forma se puede verificar el funcionamiento adecuado en todos los motores. Como microcontrolador maestro se implementó la TivaC (TM4C123GH6PM) debido a que en la fase previa al trabajo de Yau ya se le había considerado. Desde el controlador maestro se supervisa el estado de todos los motores en la cadena y se les puede enviar los parámetros de sus controladores adaptativos [1].

Yau menciona que se realizaron pruebas con la Tiva C y el sistema de reconocimiento óptico de caracteres (OCR) mediante un programa de python que permitió la comunicación serial entre el microcontrolador y la computadora. Sin embargo se encontraron problemas y se tomaron las siguientes medidas:

- Conexión serial en Python usando *ser.Serial()* y desconexión con *ser.close()*
- Verificación de la apertura del puerto serial con *ser.isOpen()*
- Uso de indicadores LED para verificar la ejecución de instrucciones

Al implementar el sistema OCR al sistema de control del brazo se resolvió el problema de latencia que se experimentó. Los resultados fueron satisfactorios, se obtuvo un error máximo de 0.2° en cada movimiento debido a que el número de pasos ejecutados era bajo. Por último, se realizaron pruebas con simulación del mando de control manual, en donde se armó un circuito sencillo utilizando un potenciómetro el cual intentaba acercarse lo más posible al funcionamiento real de un *encoder*. Luego se movió el motor paso a paso de la manera esperada pero solo cuando se rotaba lentamente el potenciómetro. El principal problema presentado fue la desincronización al mover el potenciómetro rápidamente. Yau concluyó que esto se podía deber al flujo de iteraciones utilizando la función “for” para llevar a cabo los pasos, en conjunto con el sistema de valores actuales y valores pasados implementado para evitar repeticiones de pasos.

Posteriormente se reemplazó el potenciómetro con un mando físico y se integró el sistema de reconocimiento de texto. Se conectaron los *drivers* con mayor capacidad de corriente a los motores Nema 23 mientras que se conectaron los *drivers* de segunda mayor capacidad a los motores Nema 17. Debido a los recursos limitados con los que se contaba, solo se utilizaron 3 motores en cada prueba, se cambiaron de posición para poder evaluar el rendimiento de las diferentes juntas. Los resultados permitieron verificar la exitosa recepción de datos del mando de control, tanto para ejecución de pasos como para cambio de modo de operación y la recepción de datos del sistema OCR. Adicionalmente, se logró confirmar el funcionamiento de los comandos creados para el sistema daisy chain empleado para el sistema de motores inteligentes [1].

Scott Lawrence desarrolló un trabajo similar como parte de su proyecto *Animatronic Avian*, cuyo objetivo era crear animatrónicos para un teatro para entretener a una audiencia. Se necesitaba tener una forma para que el *software/computadora* del controlador envíe datos sincronizados a través de todos los artistas en el escenario.[8] Los requisitos del proyecto fueron:

- Crear una forma de conectarlas en red sin usar ningún pin IO.
- Ser muy liviano para que pueda enviar datos a través de todos los artistas sin usar demasiados recursos del dispositivo.
- Debe ser barato, tanto con el costo de las piezas para la implementación como con los recursos del sistema en el firmware del chip.

La solución encontrada fue usar el UART0, que generalmente está configurado para una conexión RS232 estándar, y conectarlo en forma de *Daisy-chain*. Esto significa que el protocolo necesitaba de 4 líneas, alimentación, tierra, entrada y salida de datos. Lawrence afirmó en su trabajo que el código que coordina la comunicación en el dispositivo es lo suficientemente liviano como para no afectar el rendimiento del Arduino maestro.[8]

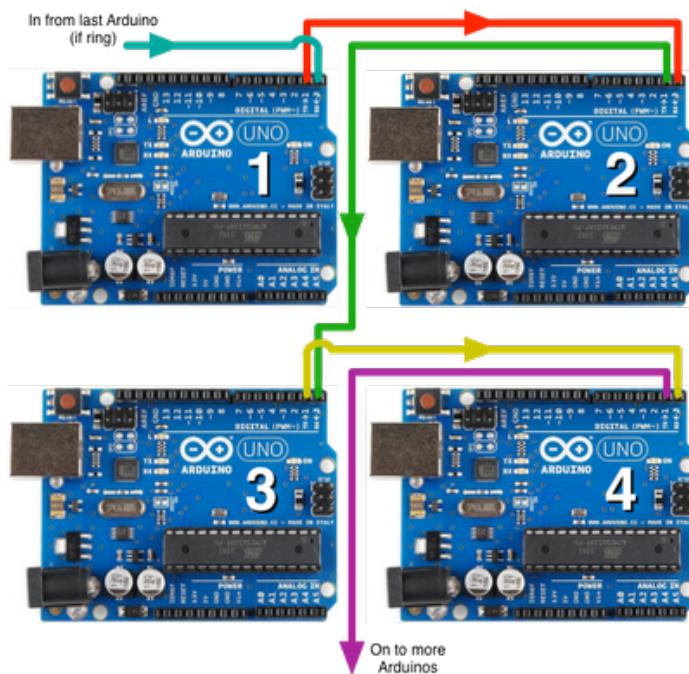


Figura 2: Conexión en cadena de Arduino Uno

El programa de prueba lee un carácter de la entrada, si es una "b", envía la "b.^al siguiente dispositivo inmediatamente y luego apaga el LED durante 1/4 de segundo. Si es un número, apagará el LED durante esa cantidad de segundos, luego restará uno de ese número y enviará ese nuevo número al siguiente nodo.

Entonces, si obtuvo un "6", apagará su LED durante 6 segundos y luego enviará un "5.^al siguiente.

El programa final funciona de la siguiente manera:

Primero, el maestro mira los datos del intervalo de tiempo actual. Para nuestro ejemplo, tendremos tres objetos animatrónicos en el bus serie, cada uno de los cuales tiene 5 bytes de datos asociados. Los datos están preparados y se verían así: 15, 1, 1, 1, 1, 200, 200, 200, 200, 30, 30, 30, 30

El primer número es la cuenta de los números que le siguen. En este caso, se incluirán 15 dígitos. Esto se envía desde la computadora a través de RS232, 9600 baudios. El primer microcontrolador animatrónico recibe esto y eliminará los números que necesita del final. En este caso, sacará los cinco "30" del final. También reducirá el conteo de 15 a 10. Esto se verá así: 10, 1, 1, 1, 1, 200, 200, 200, 200

Ahora envía esta serie de datos modificado al siguiente dispositivo en el bus. Ese dispositivo extraerá los cinco "200" del final y luego enviará el nuevo sobre. 5, 1, 1, 1, 1, 1

El último microcontrolador de la lista sacará los cinco "1", la cuenta se reduce a 0 y se abstiene de enviar algo más.

CAPÍTULO 3

Justificación

Se continuó con el desarrollo debido a que en la fase anterior quedó pendiente la implementación del código fuente como una librería. También se desea evaluar la implementación de *Type-Length-Value* para el protocolo de comunicación ya que éste permite que el usuario defina los comandos del tipo de dato que está transmitiendo y los parámetros de la comunicación. Yau menciona que para añadir flexibilidad al sistema, se recomienda reconfigurar el sistema para poder utilizar cualquier tipo de motor, especialmente motores DC y servos. Finalmente, una de las principales oportunidades que la fase anterior dejó fue la creación de un motor más completo. Se desea agregar un *encoder* para que el sistema tenga un cero global para obtener un sistema de lazo cerrado. Esto permite que el usuario pueda obtener información específica del sistema como la posición de los actuadores e implementar un mejor sistema de control que reduzca vibraciones, *overshoot* y ajustes cuando se reemplaza un motor [1].

Otra de las razones es porque en el trabajo presentado por Castillo, se menciona que una de las principales complicaciones fue la manufacturación del sistema. Debido a que los detalles de la carcasa eran tan pequeños que no se pudieron fabricar mediante impresión 3D. La segunda razón fue porque la tecnología de la Universidad del Valle de Guatemala, no fue capaz de fabricar placas de por lo menos 4 capas y la única forma de obtenerlas era mediante empresas externas. Éstas placas eran necesarias para disminuir drásticamente el tamaño del actuador ya que lo que ocupa más espacio después del motor, son las placas electrónicas. Sin embargo, Castillo detalla que se logró diseñar un actuador suficientemente compacto pero no se logró fabricar. En este trabajo se desea fabricar actuadores similares pero con base en motores *stepper*, los cuales son de mayor tamaño debido a que serán implementados en un brazo robótico en un futuro. Debido a que en esta propuesta el sistema es de mayor tamaño, se podrá fabricar con mayor facilidad [2].

CAPÍTULO 4

Objetivos

Objetivo general

Desarrollar un sistema de motores inteligentes, basado en motores *stepper* que se controle mediante un protocolo tipo *daisy-chain* desde un microcontrolador maestro.

Objetivos específicos

1. Fabricación de una carcasa para contener el motor y la electrónica de control
2. Implementación de una caja reductora acorde a las necesidades de cada motor
3. Integración de *encoders* al sistema
4. Validación del protocolo de comunicación

CAPÍTULO 5

Alcance

Como se mencionó antes, el enfoque principal de este proyecto es la creación de actuadores inteligentes, que permitan la conexión tipo *Daisy-chain* con otros de la misma clase. Se limitarán los componentes a utilizar para poder manufacturarlos a bajo costo y con componentes/materiales locales. Las tecnologías principales a utilizar en la manufactura serán el corte láser e impresión 3D, ya que son las que cuentan con mayor disponibilidad en las instalaciones de la Universidad. Además de un *driver*, estos motores contarán con un encodificador y un sensor de efecto Hall para poder centrarlos en la posición 0 de forma precisa.

Adicionalmente, el diseño de una caja reductora se sale del alcance de este trabajo y se investigará en el trabajo *Diseño y análisis de cajas reductoras fabricadas por medio de Impresión 3D y corte láser* por Darío Marroquín.

CAPÍTULO 6

Marco teórico

Motores *stepper*

Como se mencionó anteriormente los motores *stepper* son máquinas eléctricas que convierten la energía eléctrica en energía mecánica. Además, es un motor eléctrico síncrono sin escobillas que puede dividir una rotación completa en muchos pasos. La rotación del eje se puede controlar con precisión sin utilizar retroalimentación, siempre que se utilice el motor adecuado para la aplicación [9].

El motor *stepper* utiliza imanes permanentes en el rotor y electroimanes en el estator para hacer que el eje del motor gire una distancia precisa cuando se proporciona un pulso de electricidad. Generalmente, el estator tiene ocho polos y el rotor tiene seis polos. El rotor requerirá 24 pulsos de electricidad para mover los 24 pasos para hacer una revolución completa. Cuando se le suministra una corriente a las bobinas del estator, se genera un campo magnético alrededor de las mismas. Éste campo crea fuerzas electromotrices que interactúan con las fuerzas del campo magnético del rotor y lo hacen girar [9].

Cuando el rotor gira se alinea con el estator o para tener el menor espacio a través del estator. De esta forma, los estatores se activan en serie para hacer girar el motor paso a paso. En este motor, el devanado del estator está hecho de láminas de hierro al silicio para asegurar una magnetización adecuada. Los polos del estator y del rotor son independientes del tipo de motor [7].

Los tipos de motores *stepper* son:

- De imán permanente.
- Síncrono híbrido.
- De reluctancia variable.

Anteriormente se mencionó que los motores de imán permanente, lo utilizan en el rotor

y funcionan con la atracción o repulsión entre el rotor y los electroimanes del estator. Este motor es el más común en comparación con los demás y también se conoce como motor de lata/lata. Su principal beneficio es que tiene el menor costo de fabricación [7].

Los motores de reluctancia variable tienen un rotor de hierro y funcionan según el principio de que la reluctancia mínima se produce con un entrehierro mínimo, por lo que los puntos del rotor se atraen hacia los polos magnéticos del estator. Se utiliza desde hace muchos años y como sugiere el nombre, la posición angular del rotor depende principalmente de la reluctancia del circuito magnético que se puede formar entre los dientes del estator y del rotor [7].

Los motores *stepper* híbridos reciben este nombre porque utilizan una combinación de técnicas de imán permanente (PM) y reluctancia variable (VR) para lograr la máxima potencia. Es el más popular porque ofrece un buen rendimiento en comparación con un rotor de imanes permanentes en términos de velocidad, resolución de paso y par de retención. Su principal desventaja es su elevado costo en comparación con los motores *stepper* de imanes permanentes. Estos motores se utilizan cuando se requiere un ángulo de paso menor, como 1.5, 1.8 y 2.5 grados [7].

A continuación se detalla las diferentes formas para operar un motor *stepper*:

1. El método básico para impulsar un motor paso a paso es el modo de excitación único. Es un método antiguo y actualmente ya no se usa, pero es importante conocerlo. En esta técnica cada fase frente al estator adyacente, se activará una por una alternativamente con un circuito especial. Esto magnetizará y desmagnetizará el estator para mover el rotor hacia adelante [7].
2. La siguiente técnica es conocida como paso completo o *full step*. En esta técnica, dos estatores se activan a la vez en lugar de uno en un intervalo casi instantáneo. El resultado es un par alto que permite que el motor impulse una carga alta sin problemas [7].
3. Después viene la técnica del medio paso o *half step*. Esta técnica está relacionada con la de paso completo porque los dos estatores se colocarán uno al lado del otro para que se activen primero, mientras que el tercero se activará luego. Esta técnica dará como resultado una resolución mejorada del motor paso a paso mientras se reduce el par [7].
4. Por último, viene la técnica conocida como *Micro Stepping*. Esta técnica es la más utilizada debido a su precisión. El principio fundamental consiste en dividir cada paso completo en pasos más pequeños para suavizar la rotación del motor a velocidades bajas. Por ejemplo, un paso de 1.8 grados se puede dividir hasta 256 veces, logrando un ángulo de paso de 0.007 grados ($1.8 \div 256$), o 51,200 micropasos por cada revolución. El *Micro Stepping* se logra mediante el uso de voltaje modulado por ancho de pulso (PWM) para controlar la corriente a los devanados del motor. El controlador envía dos ondas sinusoidales de voltaje, desfasadas 90 grados, a los devanados del motor. Mientras que la corriente aumenta en un devanado, disminuye en el otro devanado. Esta transferencia gradual de corriente da como resultado un movimiento más suave y una producción de par más constante que el control de paso completo o de medio paso [7].

Daisy-Chain

Una conexión tipo *Daisy-Chain* se parece un poco a una flor de margarita. En electrónica, una conexión en cadena tipo margarita es donde se conecta múltiples dispositivos secuencialmente en una conexión de cable ininterrumpida. En la red Daisy Chain, una computadora está conectada a la siguiente sin ningún dispositivo que intervenga, por lo tanto, el mensaje se envía de una computadora a la siguiente y luego a la siguiente, y así sucesivamente. Esto implica que el último dispositivo del bucle puede o no conectarse al primero. Esto quiere decir que estas conexiones pueden ser lineales o tipo anillo [10].



Linear Daisy Chain Topology

Figura 3: Conexión *Daisy-Chain* [11]

En una conexión en cadena lineal, una computadora se conecta a la siguiente mediante la conexión de cables de dos vías entre ellas, como se muestra en la figura 2.

Lo opuesto a la conexión *Daisy-Chain* es la conexión en estrella. En ésta cada dispositivo esclavo está conectado al maestro con un cable individual. El cableado en cadena se encuentra comúnmente en las comunicaciones en serie como RS485, donde los dispositivos individuales se conectan secuencialmente [12].

Una de las ventajas de las conexiones tipo *Daisy-Chain* es que utilizan menos longitud de cable en comparación con las conexiones en estrella, esto equivale a un tiempo y costo de instalación más rápidos. En teoría, una conexión *Daisy-Chain* puede parecer un anillo pero en la práctica, difícilmente lo es. Es posible que tenga varios dispositivos repartidos a distancias irregulares y colocados en un área cerrada, lo que eleva el costo de la instalación de conexiones en estrella [12].

Este tipo de conexión facilita bastante la comunicación entre dispositivos pero también presenta desventajas. Mientras que se ahorra costos y tiempo de instalación, si ocurre una falla en un solo cable puede causar una interrupción total del sistema. Por ejemplo, si se rompe un cable entre el maestro y el primer dispositivo provocaría una pérdida de comunicación en todos los dispositivos [10].

Daisy-chain con SPI

Muchos dispositivos SPI no se pueden direccionar individualmente. En consecuencia, la comunicación entre esos dispositivos y el dispositivo maestro en un bus requiere una organización adicional de hardware o software. [13]

En la figura 3, el microcontrolador maestro utiliza una salida de reloj (SCK) y una línea de entrada/salida maestra (MOSI) para escribir a todos los esclavos. El maestro asigna

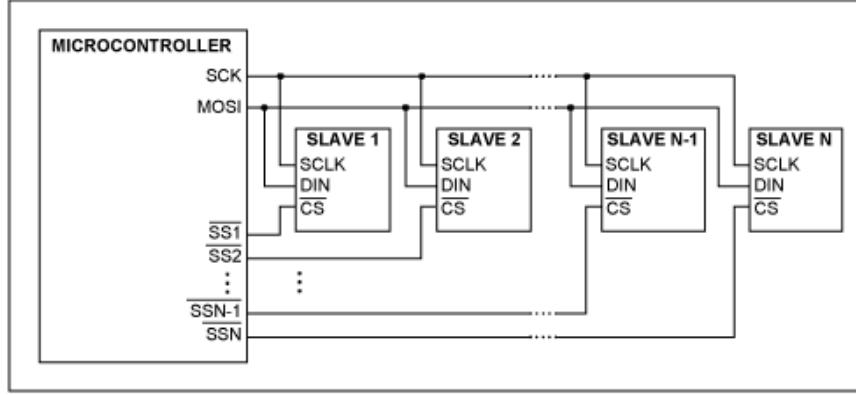


Figura 4: Conexión SPI

una señal independiente para seleccionar a los esclavos (activo-bajo SS) con los que desea comunicarse. Debido a que todos los esclavos comparten un reloj único y las líneas de datos, solo los esclavos con sus entradas CS en cero lógico reconocerán y responderán a la actividad en el reloj serial y las líneas de datos. Este sistema es simple de implementar cuando hay muy pocos dispositivos esclavos en el sistema. En sistemas con muchos esclavos, se necesitará tantas salidas SS activas bajas como el número de esclavos. Esta arquitectura aumenta la complejidad del diseño y el hardware.[13]

Un método alternativo para las aplicaciones de interfaz en serie es la conexión en cadena o *Daisy-Chain*, que propaga comandos a través de dispositivos conectados como se muestra en la figura 4.

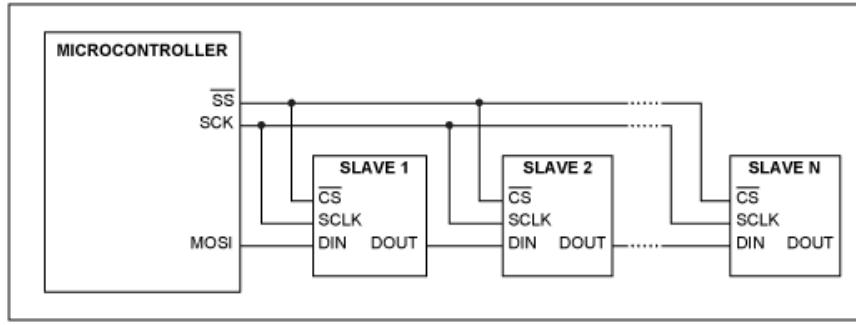


Figura 5: Conexión SPI en cadena

Una sola señal de SS activo-bajo (o CS activo-bajo) controla todas las entradas de CS activo-bajo de los esclavos; todos los esclavos reciben la misma señal de reloj. Solo el primer esclavo de la cadena (ESCLAVO 1) recibe los datos de comando directamente del microcontrolador. Cada otro esclavo en la red recibe sus datos DIN de la salida DOUT del esclavo anterior en la cadena.

Para que la conexión en cadena funcione correctamente, el esclavo debe poder ingresar un comando en DIN durante un ciclo de comando determinado (definido por la cantidad de pulsos de reloj necesarios para sincronizar un comando) y generar el mismo comando en DOUT durante el ciclo de comando subsiguiente. En pocas palabras, hay un retraso de DIN a DOUT de un ciclo de comando. Además, el esclavo solo debe ejecutar el comando

escrito en él en el flanco ascendente de CS activo-bajo. Esto significa que mientras el CS bajo activo permanezca bajo, el esclavo ignora el comando y lo envía a DOUT en el siguiente ciclo de comando. Si el CS bajo activo sube después de un ciclo de comando dado, todos los esclavos ejecutan los comandos recién escritos en sus respectivas entradas DIN. Si el CS bajo activo sube, los datos no se emiten en DOUT. Este proceso hace posible que cada esclavo de la cadena ejecute un comando diferente. Siempre que se cumplan estos requisitos de conexión en cadena, el microcontrolador solo necesita tres señales (SS activo-bajo, SCK y MOSI) para controlar todos los esclavos en la red.[14]

Daisy-chain con UART

Es posible relizar la conexión *Daisy-chain* utilizando como base la arquitectura del módulo *UART* que está disponible en la mayoría de microcontroladores. *Texas Instruments* desarrolló el protocolo SMAART, el cual está basado en la comunicación serial por UART y permite conectar varios dispositivos en cadena, como por ejemplo los TMP104. Asimismo, la interfaz admite acceso a múltiples dispositivos (MDA) mediante comandos que permiten al maestro comunicarse con múltiples dispositivos en el bus simultáneamente, eliminando la necesidad de enviar comandos individuales a cada TMP104 en el bus.[15]

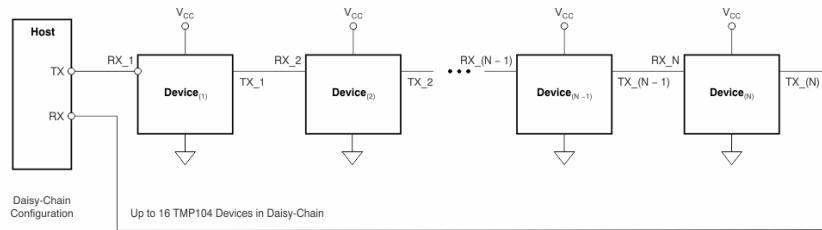


Figura 6: Interfaz SMAART

Cajas reductoras

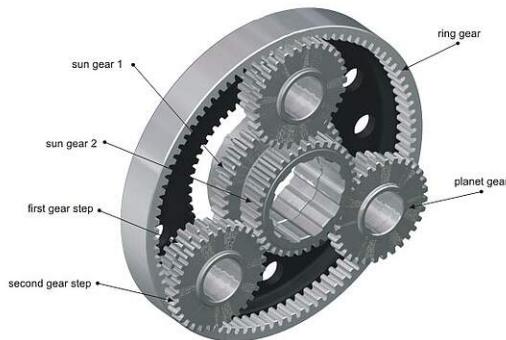


Figura 7: Transmisión planetaria [16]

Un sistema de transmisión planetaria (o sistema epicicloidal), consta normalmente de un engranaje solar de pivote central, una corona y varios engranajes planetarios que giran alrededor del solar [16].

La ventaja de una transmisión planetaria está determinada por la distribución de la carga entre múltiples engranajes planetarios. Por lo tanto, es posible transferir pares elevados utilizando un diseño compacto. Si se coloca la entrada de potencia a un miembro del sistema planetario y se aplica un mecanismo de freno para evitar la rotación de un segundo miembro, el tercer miembro girará y se convertirá en la salida [17]. Ésta permite tener varias reducciones de velocidad en un mismo mecanismo dependiendo de dónde sea la entrada de potencia y dónde sea la salida. Estas reducciones se detallan en el Cuadro 1, en donde N_s es el número de dientes del sol, N_A el número de dientes del anillo y N_p el número de dientes del planeta [18].

Detenido	Entrada	Salida	Reducción
Anillo	Sol	Portasatélite	$\frac{N_A}{N_s} + 1$
Portasatélites	Sol	Anillo	$\frac{N_A}{N_s}$
Sol	Anillo	Portasatélite	$\frac{N_s}{N_A} + 1$

Cuadro 1: Reducciones posibles del engranaje planetario [18].

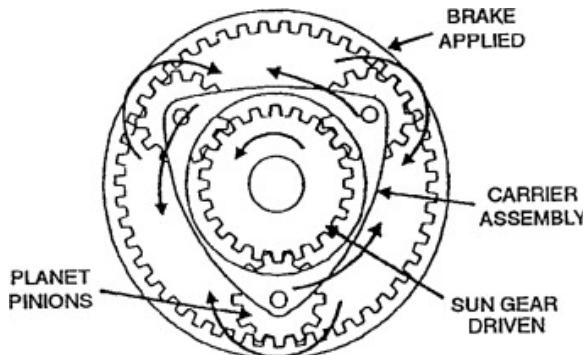


Figura 8: Partes del engranaje planetario [18]

Otra forma de lograr una transmisión de potencia efectiva es mediante el uso de un engranaje de transmisión armónica o *Harmonic Drive*. Es un dispositivo mecánico de cambio de velocidad, inventado en la década de 1950, que reduce la relación de transmisión de una máquina rotativa para aumentar el par. Funciona según un principio diferente al de las transmisiones de velocidad convencionales. El dispositivo consta de un anillo delgado que se desvía elásticamente mientras rueda dentro de un anillo circular rígido un poco más grande [19].

Hay tres elementos en un *Harmonic Drive*: un *spline* circular, un *flexspline* y un generador de ondas. El *spline* circular tiene dientes internos que engranan con dientes externos en el *flexspline*, que tiene menos dientes y un diámetro efectivo menor que el *spline* circular. El generador de ondas tiene forma elíptica y actúa como un enlace con dos rodillos que giran dentro del *flexspline*, haciendo que se engrane progresivamente con el *spline* circular en puntos diametralmente opuestos. Si el generador de ondas, (entrada de potencia) gira en el sentido de las agujas del reloj mientras el *spline* circular está fijo, el *flexspline*, que actúa

como salida de potencia, girará a un ritmo mucho más lento y en sentido contrario a las agujas del reloj[20].

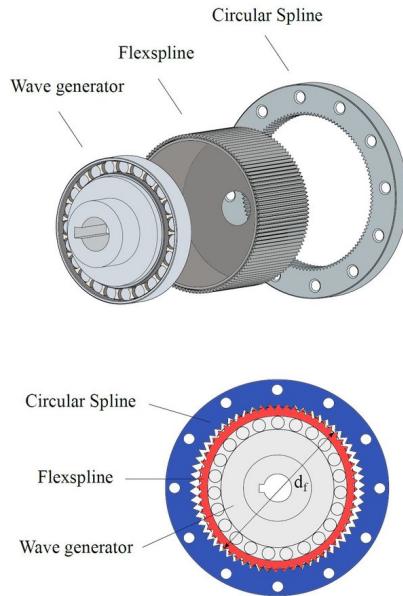


Figura 9: Partes del *Harmonic Drive* [21].

La relación entre la velocidad de entrada y la de salida depende de la diferencia en el número de dientes en la ranura circular y en la ranura flexible. Es posible producir reducciones de velocidad de hasta 320 a 1 en estas transmisiones, las cuales son más ligeras, más pequeñas y más eficientes que las transmisiones convencionales de alta relación [20].

Entre las ventajas de esta transmisión se encuentran la falta de juego, porque el movimiento de acoplamiento de los dientes (cinemática) del engranaje impulsor armónico es muy diferente al de los engranajes planetarios o rectos. El diseño permite que un 30 % de los dientes esté engranado en todo momento en comparación con seis dientes para un engranaje planetario o uno o dos para una transmisión tradicional recta. Otra ventaja es el rendimiento consistente ya que como parte del diseño, los dientes del engranaje de la ranura flexible están precargados contra los de la ranura circular en el eje principal de la elipse. A medida que los dientes del engranaje se desgastan, la deformación radial elástica actúa como un resorte muy rígido para compensar el espacio entre los dientes que, de lo contrario, provocaría un aumento en la holgura. También relaciones de torque a peso y de torque a volumen más altas que otras tecnologías de engranajes. Por último, ofrecen una excelente precisión en las posiciones debido a la combinación del principio de transmisión armónica y la tecnología de fabricación, lo que permite una precisión de posición de 30 segundos de arco (0.008°) [19].

La transmisión cicloidal es otro sistema eficiente que se utiliza frecuentemente en muchas áreas industriales enfocadas en velocidad y conversión de par, debido a su gran relación de transmisión, tamaño compacto, gran capacidad de carga y alta eficiencia. En los últimos años se utilizan cada vez más en la transmisión de precisión, principalmente en el área de robótica y aeroespacial. Se caracteriza por engranar simultáneamente la mitad de sus dientes, además, tienen el excelente efecto de promediación de los errores lo que aporta una alta precisión. En comparación con el *Harmonic Drive*, no hay ningún elemento flexible en

la transmisión cicloidal que pueda generar una gran rigidez [22].

Esta transmisión consiste en un eje de transmisión excéntrico que impulsa un disco cicloidal con pasadores anulares. Los pasadores están dispuestos en un círculo alrededor del eje excéntrico, en el que se acopla el disco cicloidal. Debido al movimiento excéntrico, el disco cicloidal gira alrededor de estos pasadores para que el disco cicloidal gire alrededor de su eje de simetría. Hay agujeros en el disco cicloidal que, a diferencia del eje excéntrico, ahora giran en el sentido de las agujas del reloj. Los pasadores de rodillos de un disco de pasadores encajan en estos orificios. De esta manera, el disco cicloidal acciona el disco de pasadores, al que se une el eje de salida montado en el centro y que es coaxial con el eje de entrada [22].

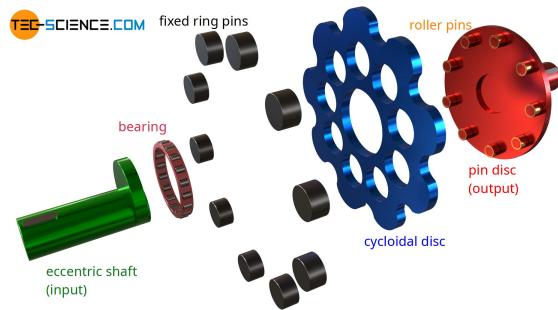


Figura 10: Partes de la transmisión cicloidal [23]

La relación de transmisión de un accionamiento cicloidal está determinada por el número de pines fijos del anillo (N) y el número de lóbulos del disco cicloidal (n). El número de lóbulos en el disco cicloidal siempre debe ser menor que el número de pasadores que lo rodean, de lo contrario, el disco sería más grande que el círculo de referencia de los pasadores y el disco ni siquiera encajaría entre los pasadores. En la mayoría de los casos, el disco cicloidal tiene un lóbulo menos que el número de pines [22].

La relación de transmisión de un accionamiento cicloidal se puede determinar de la siguiente manera, sobre la base del número de lóbulos del disco cicloidal n y la diferencia con el número de pines N :

$$i = \frac{n}{N-n}$$

Encoders

Un *encoder* es un dispositivo que proporciona retroalimentación negativa si se implementa en un lazo de control. Convierten el movimiento en una señal eléctrica que puede ser leída por algún dispositivo en un sistema de control de movimiento, como un contador o un PLC. El codificador envía una señal de retroalimentación que se puede usar para determinar la posición, el conteo, la velocidad o la dirección. Un dispositivo de control puede usar esta información para enviar un comando para una función particular [24].

Utilizan diferentes tipos de tecnologías para crear una señal, que incluyen: mecánica, magnética, resistiva y óptica, siendo la óptica la más común. En la detección óptica, el

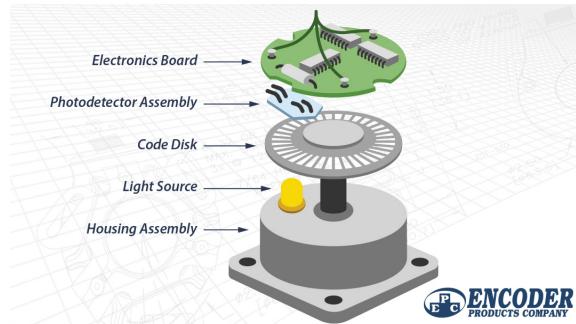


Figura 11: Elementos de un *encoder*[24].

encoder proporciona información basada en la interrupción de la luz. En la detección óptica un haz de luz emitido por un LED pasa a través del *Code Disk*, que tiene un patrón de líneas opacas (parecido a los radios de una rueda de bicicleta). A medida que gira el eje del motor en donde se instaló, el haz de luz se ve interrumpido por las líneas opacas del disco de codificación antes de ser captado por el conjunto del fotodetector. Esto produce una señal de pulso: luz = encendido y sin luz = apagado. La señal se envía al contador o controlador, que luego enviará la señal para producir la función deseada [24].

La implementación de *encoders* para controlar motores trae muchas ventajas. Son precisos y confiables, tienen bajo costo, alta resolución, son compactos y además tiene electrónica integrada que hace todo el proceso. Sin embargo, presentan algunas desventajas como interferencia en *encoders* magnéticos debido a ondas electromagnéticas o de radio. En los ópticos, las fuentes externas de luz pueden interferir con el proceso. En general, son susceptibles a la suciedad, contaminantes de polvo y aceite dependiendo del ambiente en donde estén [25].

Algunas aplicaciones de estos dispositivos son en robótica, taladros de todo tipo, bandas transportadoras, **equipo médico**, líneas de ensamblaje, automatización y control de compuertas [25].

El *encoder* lineal convierte la posición lineal en una señal de salida eléctrica. Sus partes principales son una escala de cinta métrica, una fuente de luz y un fotorreceptor. Generalmente, la fuente de luz y el fotorreceptor se encuentran juntos. Éste dispositivo puede detectar el movimiento a partir de los cambios de posición con el tiempo.

Los *encoders* lineales pueden ser de tipo óptico, magnético, inductivo, capacitivo y de *Corriente de Eddy*. Los magnéticos usan un sensor magnético y una escala magnética para producir los canales de salida analógicos A y B, de modo que cuando el sensor magnético pase a lo largo de la escala magnética, el sensor detectará un cambio en el campo magnético y emitirá una señal. Esta señal de salida será proporcional a la velocidad de medición y al desplazamiento del sensor. La principal ventaja del *encoder* lineal es que funciona mediante un cambio en el campo magnético, por lo que no se verá afectado por la interferencia de la luz, el aceite, los desechos, etc [24].

Los de tipo capacitivo funcionan detectando la capacitancia entre un lector y una báscula. Las aplicaciones comunes son calibradores en digitales o *vernier*. Una de las desventajas es la sensibilidad a la suciedad irregular, que puede cambiar localmente la permitividad relativa e introducir errores en las mediciones [25].

Los de tipo inductivo son resistentes a los contaminantes, por lo que se pueden utilizar en calibres y otras herramientas de medición a prueba de refrigerantes. Una aplicación muy popular del principio de medición inductivo es el Inductosyn [25].

Los de tipo de *corriente de Eddy* de alto rendimiento se caracterizan por no tener contacto y medir la posición y el movimiento de los objetivos conductores. Estos sensores usan una bobina de referencia para producir un campo magnético en un objetivo. A medida que el objetivo se acerca al sensor, los campos magnéticos del sensor y del objetivo se distorsionan entre sí. Frecuentemente se les confunde con los de tipo inductivo debido a principios de funcionamiento similares. Estos últimos, se utilizan en los interruptores de proximidad industriales, pero no son adecuados para aplicaciones de metrología de precisión. Por el contrario, los sensores de *corriente de Eddy* tienen salidas muy lineales, alta resolución y son estables con la temperatura [25].

Otro tipo de *encoder* es el rotatorio, el cual es capaz de convertir la posición angular del movimiento del eje en una señal digital. En éste, el eje del dispositivo estará acoplado al eje del motor a monitorear. La revolución del eje luego se convierte en una señal eléctrica y se hace con la ayuda de un sistema fotoeléctrico y uno de procesamiento. Por último, esta señal eléctrica se transfiere al PLCs.

Los *encoders* rotatorios también pueden ser de tipo óptico, magnético y mecánico. Los últimos también se conocen como *encoders* conductivos. Para su implementación, se utiliza una serie de pistas de cobre circunferenciales grabadas en una PCB para codificar la información a través de cepillos de contacto que detectan las áreas conductoras. Son económicos pero susceptibles al desgaste mecánico. Son comunes en las interfaces humanas, como los multímetros digitales, ratones optomecánicos y trackballs, reómetros de tensión controlada y plataformas de radar giratorias [26].

Todos los tipos anteriores se refieren a la forma en que funcionan estos dispositivos, pero esto se puede resumir en dos grandes clasificaciones que se refieren a la forma en la que toman los datos: los absolutos y los incrementales.

Los absolutos pueden proporcionar la ubicación específica. Pueden ser de una o varias vueltas. Los de una sola vuelta son adecuados para aplicaciones de recorrido corto, mientras que los de varias vueltas son adecuados para aplicaciones de posiciones largas. Los componentes de un *encoder* absoluto son casi los mismos que los de un incremental.

Los absolutos están compuestos por una fuente de luz que sería LED, un fotodetector y un disco. Los codificadores absolutos también se denominan *encoders* de ángulo de eje y se utilizan para detectar la posición angular. El disco o placa que tiene segmentos opacos o transparentes pasa entre una fuente de luz y esta podría ser un LED y el haz de luz sería interrumpido por el detector. La señal electrónica generada se transferirá luego a un controlador donde la posición y la velocidad se calculan de acuerdo con la señal recibida [26].

Por el otro lado, los incrementales se utilizan principalmente para determinar la posición angular o lineal. Tienen sensores para hacer la medición angular, los cuales pueden ser mecánicos o magnéticos. El codificador incremental utiliza un disco transparente y tendría secciones opacas que están igualmente espaciadas para conocer el movimiento. El LED pasaría la luz a través de un disco de vidrio y sería detectado por un fotodetector, por lo que

el codificador generaría pulsos que son pulsos igualmente espaciados [26].

Metodología de diseño de Norton

En el libro *diseño de máquinas* de Norton [27] se detalla la metodología para el proceso de diseño. En ingeniería, se define como el proceso de aplicación de varias técnicas y principios científicos con la finalidad de definir un dispositivo, un proceso o sistema, con el detalle suficiente que permita su realización. Norton considera que las piezas de una máquina se deben diseñar una a la vez, y que es importante reconocer el funcionamiento y desempeño de cada una. Éstos dependen de muchas otras porque están interrelacionadas dentro de la misma máquina. Él también establece que para llegar a la etapa en donde se dimensionan las piezas, se deben definir primero sus movimientos cinemáticos. Sin embargo, como no es posible que el diseñador prediga con exactitud las cargas ambientales a las que el usuario someterá la máquina. En tales casos, el análisis estático con datos empíricos, reunidos a partir de la prueba real brindan información para los propósitos de diseño.

El dilema se resuelve con iteración, lo cual quiere decir *regresar a una fase anterior o repetir*. La mayoría de veces se verá que el diseño de la primera prueba falla debido a que los materiales no pueden mantener los niveles de esfuerzo presentes, entonces se deben rediseñar con la finalidad de obtener un diseño aceptable que cumpla los requisitos. En esencia el proceso de diseño es un ejercicio de creatividad aplicada. En general, Norton define los pasos del proceso de diseño de la siguiente manera:

Tabla A-2 Planteamiento y cálculo del problema

1	Definición del problema		Etapa de definición
2	Establecimiento de las especificaciones		
3	Hacer supuestos adecuados		
4	Decisiones preliminares de diseño		Etapa de diseño preliminar
5	Diseño de esbozos		
6	Modelos matemáticos		
7	Análisis del diseño		Etapa de diseño detallado
8	Evaluación		
9	Documentar resultados		

Figura 12: Proceso de diseño de Norton[27]

CAPÍTULO 7

Diseño, manufactura y ensamblaje del actuado

Se inició con el desarrollo de la carcasa de los actuadores inteligentes ya que era de gran importancia tener una pieza que sostuviera los motores y los componentes en sus respectivos lugares. Se favoreció el corte láser debido a que es más rápido y sus tolerancias son menores que la impresión 3D. En el diseño final se tiene solo una pieza pequeña impresa en 3D y el resto se fabricó con MDF de 6.35 mm de grosor.

7.1. Selección de componentes

7.1.1. Microcontrolador

Como uno de los requisitos del proyecto era que los materiales debían estar disponibles localmente, por lo que se tenían las siguientes opciones:

- Arduino Uno.[28]
- Arduino Mega.[29]
- ESP32.[30]
- Arduino Nano.[31]
- Arduino pro mini.[32]
- Arduino pro micro.[33]
- Attiny85.[34]

Se eligió el Arduino pro mini para controlar los motores principalmente porque tiene un tamaño compacto de 33 mm x 18 mm. Además, también tiene los módulos UART, SPI,

I2C, 5 entradas analógicas fácilmente accesibles y 14 pines IO digitales. Esta tarjeta tiene el microcontrolador ATMEGA328P el cual es el mismo que tiene el Arduino Uno, pro micro y Nano, el cual es capaz de tener un buen rendimiento en este proyecto. Entre sus especificaciones más importantes están:

- 32kB de memoria flash.
- 2kB de SRAM.
- 1kB de EEPROM.
- 16MHz de reloj.

Estas características son más que suficientes para lo que se necesitó en este proyecto.

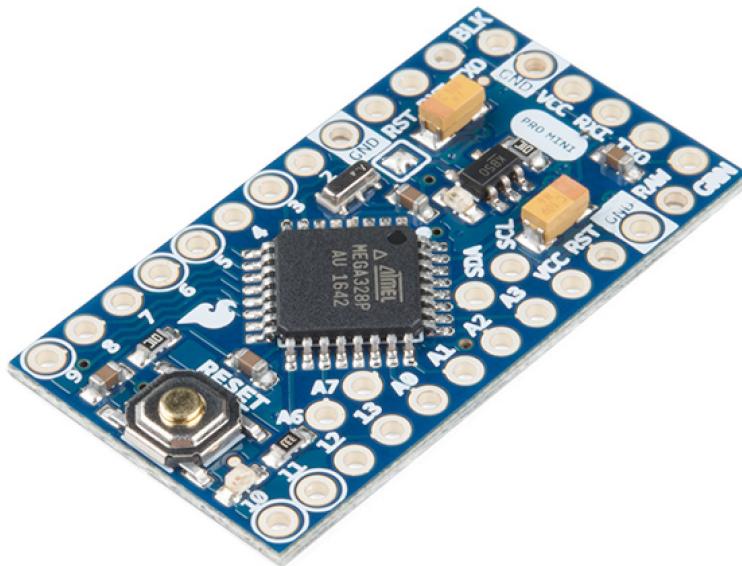


Figura 13: Arduino pro mini[35].

No se eligió el Arduino Mega principalmente por su tamaño, además que se desea que el microcontrolador tenga pines macho para insertarlo en una placa dentro del motor. El Attiny85 tampoco fue una opción adecuada porque solo tenía 6 pines IO. El Arduino Nano y el pro micro fueron buenas opciones ya que tienen suficientes pines IO y son más baratos en general.



Figura 14: *Encoder* óptico[35].

7.1.2. *Encoder*

Al momento de elegir un *encoder* se tenían dos principales opciones: óptico y rotativo. El *encoder* óptico disponible localmente consistía de un disco metálico con divisiones y un módulo de leds infrarrojos que generan un pulso cuando gira el disco. El disco va montado sobre el eje para que cuando gire el eje se generen los pulsos correspondientes a la salida del módulo infrarrojo. Por el otro lado, el *encoder* rotativo se asemeja a un potenciómetro pero con la diferencia de que puede girar 360°. Éste consta de un disco con varios contactos de cobre que al girar hacen contacto con *switches* y producen señales lógicas en los pines de salida correspondientes. Estas señales se leen en el microcontrolador y al realizar los cálculos respectivos se puede averiguar la velocidad de rotación, sentido de giro y la posición del eje. Simplemente se necesita acoplar el eje del *encoder* al eje del motor para obtener su información.

Se eligió el encoder rotativo debido a que está montado sobre una placa de 20 x 30 mm y tiene agujeros de 3 mm para tornillos M3, lo cual permitió montarlo fácilmente sobre una plancha de MDF o una superficie impresa en 3D. La otra principal razón fue porque resulta fácil de acoplar con el eje del motor, solo es necesario añadir un par de poleas o engranajes idénticos para que cada rotación del eje sea equivalente a una rotación en el *encoder*.



Figura 15: *Encoder* rotativo[35].

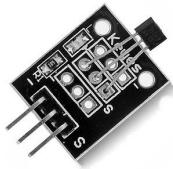


Figura 16: Sensor Hall[35].

7.1.3. Sensor de efecto Hall

Se seleccionó este sensor para la detección de cero del motor *stepper*, debido a que el *encoder* empleado es incremental y no absoluto. Otra de sus ventajas es que venía soldado en una pequeña placa de 10 x 15 mm con agujeros para tornillos M3. Al igual que para el *encoder* esto simplificó mucho el proceso porque solo fue necesario crear una superficie para atornillarlo en el montaje final.

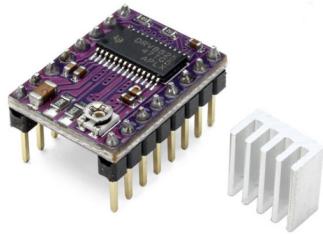


Figura 17: DRV8825[35].

7.1.4. Motores *stepper*

Se seleccionó el motor NEMA 17 debido a que en la Universidad se cuenta con varios modelos en caso de que se necesiten fabricar más de 5 en un futuro. El *driver* elegido para este motor es el DRV882517 principalmente porque tiene capacidad de entregar hasta 2.2A y maneja de 8.2V a 45V, además que puede realizar pasos de 1/32 e incluye un disipador de calor.

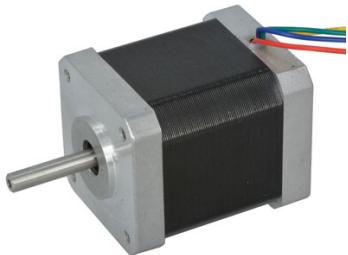


Figura 18: Motor utilizado NEMA 17[35].

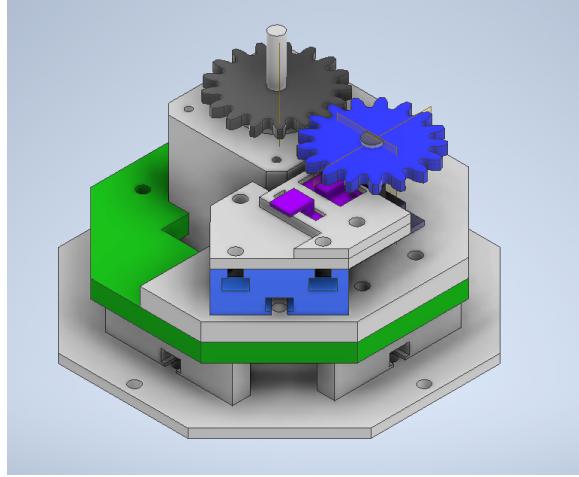


Figura 19: Vista isométrica de los motores inteligentes.

7.2. Diseño CAD

Para el diseño en CAD, se utilizó el *software* Inventor 2022 de Autodesk.[36] Para la primera fase se tomaron medidas de los componentes utilizando un vernier, después se realizaron modelos 3D de cada uno con la mayor exactitud posible. Luego, se diseñaron 2 engranajes idénticos basados en la norma AGMA-2001[27] con 17 dientes, módulo de 1 mm y un espesor de 3.125mm. Para lograr esta tarea se utilizaron los valores por defecto del acelerador de diseño de Inventor porque estos engranajes no transmitirán la potencia del motor, sino que únicamente se encargarán de darle la vuelta al eje del *encoder*, el cual no representa una resistencia considerable al giro.

La razón de velocidad de los engranajes es unitaria ya que tanto el piñón como el engrane tienen la misma cantidad de dientes. Esto simplifica el uso del *encoder* porque una rotación del mismo es igual a 1 rotación del eje del motor pero en sentido contrario.

La corriente medida del motor en operación fue de 1.1 A a 10 V, por lo que la potencia consumida se determinó mediante:

$$P = VI = (10V)(1.1A) = 11W. \quad (1)$$

Se asumió una eficiencia de 0.80 típica para motores eléctricos y se calculó la salida de potencia mecánica en el eje.[37]

$$P_{eje} = P_e(\eta) = (11W)(0.8) = 8.8W \quad (2)$$

De esto se calcula el torque nominal mediante:[27]

$$T = \frac{P_{eje}}{w} = \frac{8.8W}{100 \frac{rev}{min} \frac{2\pi rad}{1rev} \frac{1min}{60s}} = 0.84Nm. \quad (3)$$

Como se observa en 19, el diseño favorece el corte láser por el tiempo de manufactura y exactitud que ofrece en comparación a la impresión 3D. Luego de ensamblar el primer

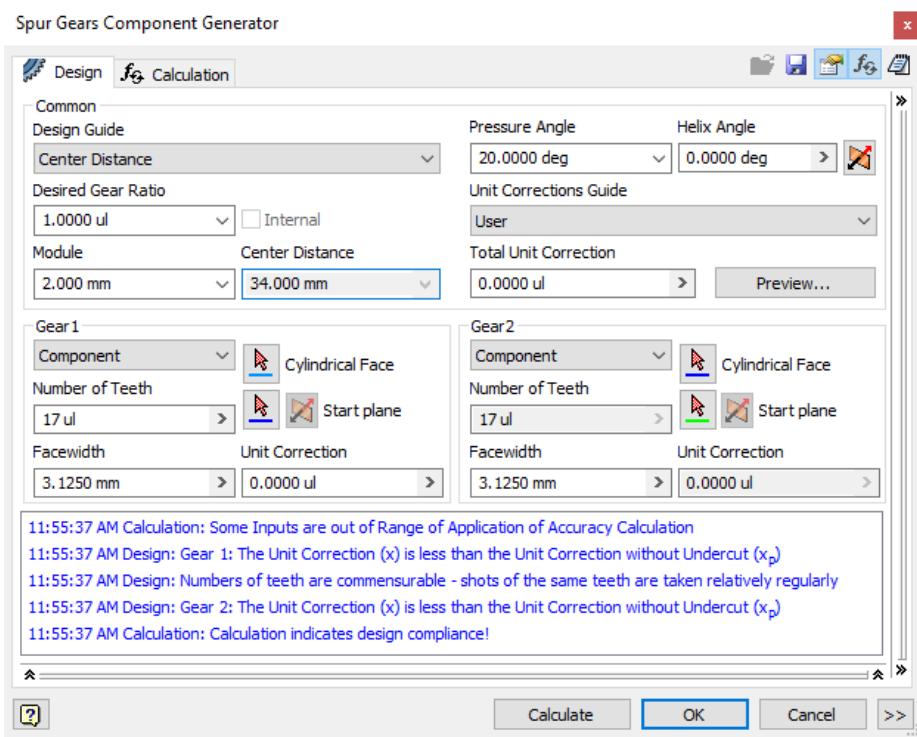


Figura 20: Parámetros del aceleración de diseño.

actuador se determinó a prueba y error que la mejor tolerancia según las capacidades de manufactura locales era de 0.1 mm ya que al aplicarla se logró un ajuste a presión perfecto del motor. Otra ventaja que ofrece el corte láser es que los agujeros se pueden dejar marcados por si luego se desea colocar el componente en la posición final y luego taladrar los agujeros de sujeción correspondientes. Este método no es muy recomendado debido a que la superficie del MDF se puede dañar severamente si no se colocan piezas de sacrificio, pero a cambio se obtiene la flexibilidad de realizar el agujero según sea necesario.

Para cerrar la carcasa del motor, se imprimió por partes una serie de anillos octagonales con el objetivo de encavarlos como si fueran piezas de *LEGO* y adicionalmente atornillarlos a la base para una mayor firmeza y rigidez.

Se logró llegar al diseño final del prototipo luego de 7 iteraciones. En cada una se fue mejorando un aspecto crítico de diseño. En la primera iteración principalmente se buscó colocar los componentes en una carcasa básica para identificar la distancia más adecuada entre los mismos. También, se consideró la posibilidad de utilizar engranajes metálicos a 90° para montar el *encoder* en las paredes de la carcasa y sujetarlo simplemente con tornillos. Los engranajes que se iban a utilizar tenían un agujero central del mismo tamaño que el diámetro de los ejes del motor y *encoder* respectivamente. El inconveniente que se dió fue que se necesitaba una pared para montar los componentes debido al ángulo de montaje de los engranajes, lo cual no aportaba mucha firmeza al diseño debido a que se deseaba que tuviera una forma no rectangular.

Durante la segunda iteración, se cambió la forma de la base para que fuera octagonal para disminuir el área ocupada respecto a una forma rectangular. Durante esta iteración se

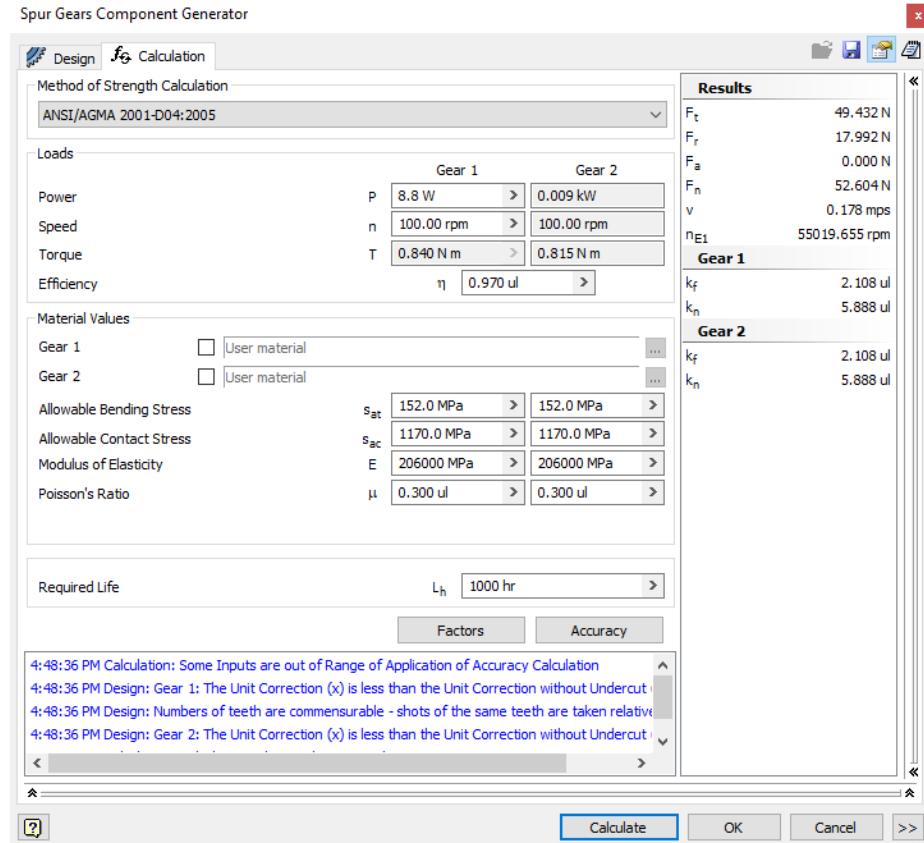


Figura 21: Constantes de los engranajes.

optó por un diseño alargado con el fin de añadirle un pequeño compartimiento para colocar una placa con el *driver* y el Atmega328P. Uno de los principales problemas encontrados fue la sujeción del circuito impreso ya que la placa iba montada verticalmente y se necesitaba una pieza plana para sujetarla. Una de las soluciones propuestas fue la creación de una tapa impresa en 3D en la parte inferior del montaje para montarle verticalmente una pieza que sujetara la placa. Se cambió en las iteraciones posteriores debido a que se observó que si se reduce de tamaño la placa se podría atornillar en el suelo del sujetador de piezas y quedaría más asegurada. Durante la tercera iteración se decidió utilizar engranajes rectos porque de esta forma se simplificaba el montaje del *encoder*, además que se tenía control total sobre las dimensiones de los engranajes al fabricarlos. Esto permitió realizar varias pruebas con diferentes tamaños de juego para el agujero central y el que dió mejor resultado fue el de 0.1 mm.

Para la cuarta iteración se modificó el anclaje del *encoder*. Anteriormente se fabricó una pequeña caja montada a la pared de la carcasa en donde se atornilló ese componente pero acá se optó por montarlo directamente en la pared de la carcasa. Adicionalmente, se diseñó una carcasa cuadrada principalmente para evitar el uso de piezas de acople que fueron necesarias con el diseño original. Un problema del cambio principal de esta iteración fue que la forma cuadrada aumentó el volumen que ocupaba la carcasa. Por último, se realizó una prueba de montaje con los engranajes a 90° pero se decidió descartarlos porque se observó que al montar el *encoder* horizontalmente sobresalían los tornillos en las paredes. Como se desea

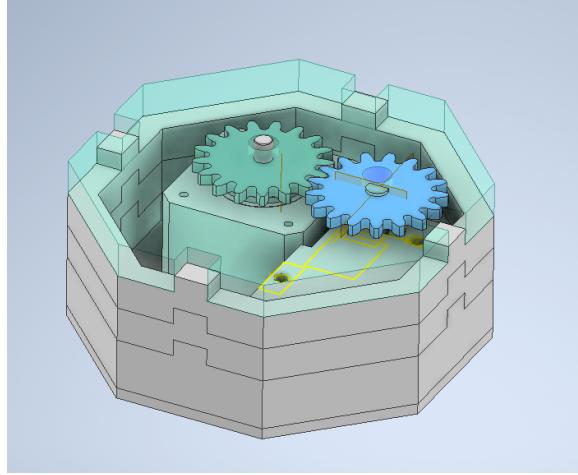


Figura 22: Recubrimiento de la carcasa.

que estos motores sean lo más compactos posibles, se decidió montarlo verticalmente con los engranajes mencionados en la iteración anterior. De esta forma, los tornillos sobresalen verticalmente pero esto no afecta debido a que en la parte inferior se tiene una recámara para el montaje de los circuitos impresos que cuenta con suficiente espacio.

En la quinta iteración se mejoró el montaje de los componentes añadiendo agujeros y ranuras para los tornillos y demás elementos de sujeción. También se hizo una revisión respecto a las tolerancias de la iteración anterior para arreglar los ajustes defectuosos y la interferencia. Como se mencionó en anteriormente, se montó el *encoder* verticalmente al lado del motor usando piezas impresas en 3D con agujeros para los tornillos. Se le colocó un engranaje de los diseñados en la sección de engranajes y adicionalmente se le hizo una ranura para introducir un imán de neodimio pequeño. Se deseaba montar el imán de neodimio en el engranaje del *encoder* para reducir el tamaño del diseño y colocar el sensor debajo de éste. Esto permite la detección del cero absoluto por medio del sensor de efecto *hall* para calibrar los motores. También se fabricaron dos propuestas para la sujeción del sensor *HALL* de modo que estuviera casi pegado al engranaje. La primera consistió en una pieza rectangular sostenida verticalmente con otras piezas pequeñas de la misma forma. La pieza vertical tenía dos agujeros para montar el sensor con tornillos M3 y una ranura rectangular para introducir un tornillo M4 y sujetarla a la base. La segunda consistió en un par de piezas rectangulares con ranuras de 4 mm con el fin de montarlas a la base con tornillos. Éstas tenían la función de sostener la cama del sensor montado horizontalmente. La segunda propuesta fue la que se adoptó para la siguiente iteración por la facilidad que aportó para conectar el sensor con el resto del sistema.

La sexta iteración consistió en una pieza octagonal con 4 agujeros en los extremos superior, inferior, izquierdo y derecho. Sobre esta se colocaron dos placas rectangulares con la forma del motor y con agujeros correspondientes para tornillos M4. También se eliminaron los soportes del *encoder* de la iteración anterior y se cortó una pieza de 0.25 pulgadas en donde se atornilló. Con este cambio el *encoder* adquirió mayor firmeza y no tuvo problemas cuando se giró el motor a 1000 pasos/segundo. Esta iteración también tuvo 2 variaciones, en la primera se diseñó una pieza plana larga para sostener el sensor *HALL* horizontalmente. Ésta contaba con una ranura rectangular para deslizar el sensor y acomodarlo en la posi-

ción deseada para que detectara el imán colocado en el engranaje del *encoder*. La segunda consistió en una pieza plana más corta que la de la primera variación. Esta pieza contaba con un agujero rectangular para acomodar los pines sobresalientes del sensor para que al acomodarlo no se apoyara sobre ellos. Esta pieza también contaba con agujeros de 4 mm para montarla con tornillos sobre los soportes. Los soportes fueron los mismos en ambas variaciones debido a que solo se buscó cambiar la pieza soportada por ellos. Éstos se montaron a la base mediante tornillos M4. Finalmente, se diseñó una serie de anillos octagonales para colocarlos uno encima de otro y ajustarlos con tornillos. El diseño de la tapa fue una pieza octagonal plana con agujeros para que pasara el eje del motor y del *encoder*.

Durante la séptima iteración se buscó mejorar la forma de montar el sensor ya que los agujeros tenían un tamaño real de 2.83 mm y los tornillos M3 no entraron. Se optó por realizar una pieza impresa en 3D con la forma del sensor para acomodarlo en los soportes mediante tornillos M3. El objetivo principal de esta iteración fue adaptar el diseño anterior a las dimensiones de los motores NEMA17 unipolares disponibles en la Universidad. Como estos motores tenían las mismas dimensiones en su base que los bipolares, sólo se fabricaron unos soportes de 14.33 mm de altura para levantar la base que sostenía los sensores. También se fabricó una pieza octagonal con agujeros y ranuras para los soportes, la cual sostenía todo el sistema. Por último se fabricó un circuito impreso que sostuviera todos los componentes, el cuál se atornilló en la parte inferior.

En la octava iteración se cambió totalmente el diseño de la carcasa. El nuevo diseño consistió en una forma circular con paredes y techo porque esta forma ofreció un mayor espacio interno. También se cambió de lugar el imán, el sensor y el módulo del engranaje a 0.8 mm para darle más exactitud al sistema de orientación. Utilizando el acelerador de diseño de Inventor se calculó automáticamente el módulo dejando una distancia entre centros de 36 mm. De esta forma se logró *mapear* 200 pasos del encoder con los 200 pasos del motor mediante una relación de engranajes 5:1.

En la novena iteración se cambió el módulo a 1.250 mm y la distancia entre centros a 37.589 mm para aumentar el diámetro de paso del engranaje sin cambiar drásticamente las dimensiones. Por esta razón se tuvo que volver a fabricar todas las piezas internas.

Durante la décima iteración se diseñaron dos anillos sujetadores para montar la placa. El montaje de la placa consistió en atornillarla a dos piezas de madera con agujeros de 4 mm. Después, se atornilló el montaje al anillo sujetador. Por último, se insertó un anillo que sostiene el motor y le da soporte mediante uniones a presión. Ésta iteración tuvo dos variaciones en el anillo sujetador, la primera con uniones atornilladas y la segunda con uniones a presión. Se eligieron los ajustes a presión porque fueron mucho más convenientes al momento de ensamblar, además no iban a estar sujetados a ninguna fuerza externa por lo que no presentaron ningún riesgo a la integridad del sistema.

7.3. Ensamblaje

Para ensamblar la mayoría de las piezas se emplearon tornillos y tuercas M4, sin embargo, sí se utilizaron tornillos y tuercas M3 para montar el *encoder*. Se observa en 19 que las piezas no sólo presentan agujeros sino que también ranuras con el objetivo de introducir la

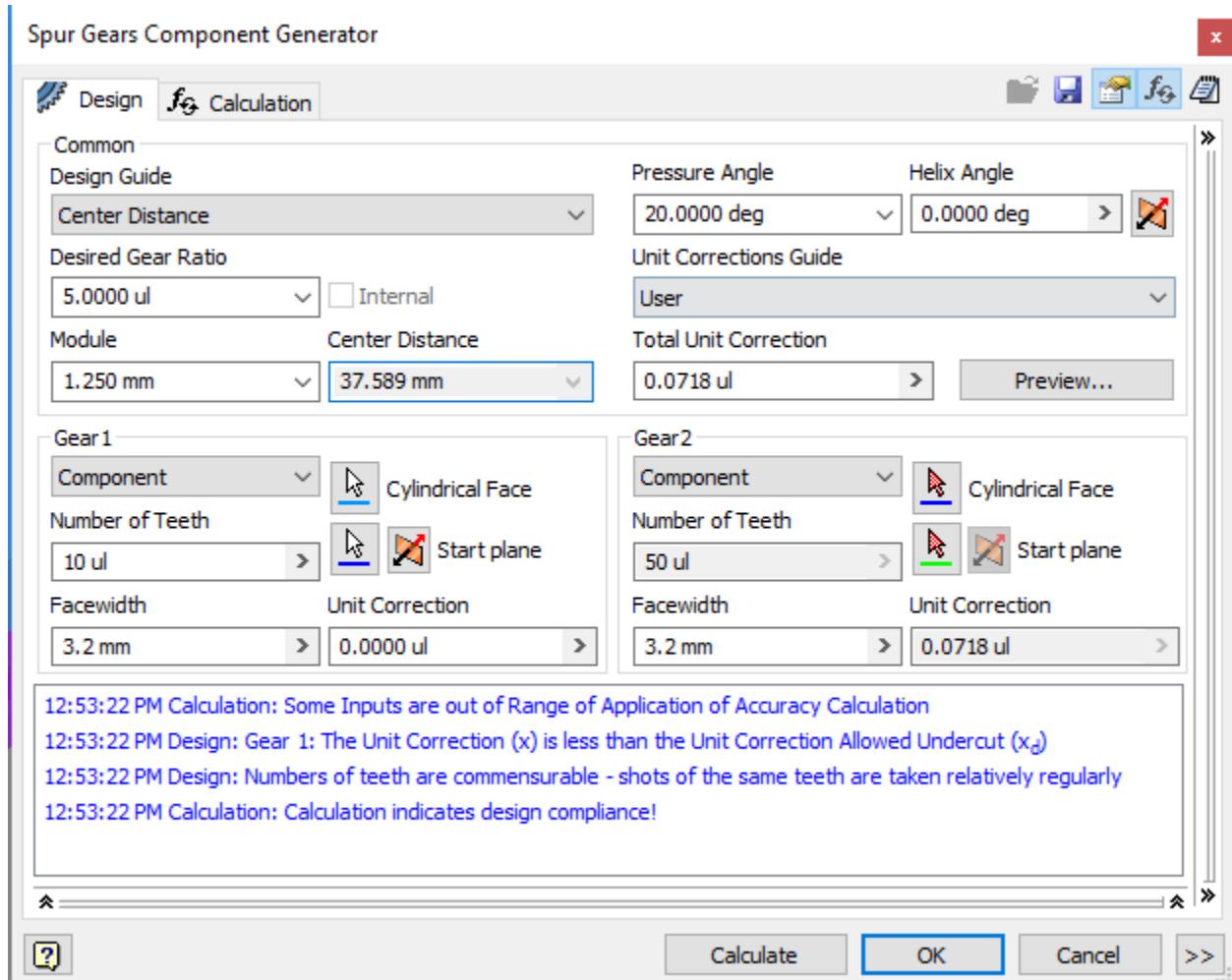


Figura 23: Configuración del acelerador de diseño en iteración 9

cabeza del tornillo y aprovechar su forma para aplicar una pequeña fuerza de sujeción que mantenga todo el sistema unido sin necesidad de pegamento. Esto resulta ser más caro que utilizar pegamento o simples uniones a presión, pero la ventaja es que aportan al sistema una robustez mayor a la del pegamento, además de la facilidad de montar y desmontar todo a voluntad.

7.4. Posición y velocidad

Para la medición de velocidad y posición se utilizó el encoder rotativo mencionado anteriormente. Primero, para calibrar los motores se giran hacia la izquierda hasta que el sensor de efecto HALL detecta el campo magnético de un imán montado en el engranaje del *encoder*. Después, se regresa paso por paso en la dirección contraria hasta que el sensor deja de detectar el campo magnético. En ese momento se detiene el movimiento y se establece el 0 de una variable que guarda la posición del encoder.

Para la medición de la posición, se utilizan las señales de los pines DT Y CLK del *encoder* para detectar la dirección de giro y el número de pasos que se avanzó, como se observa en 24. Si se detecta una dirección positiva en el movimiento se le suma a la variable de posición, de lo contrario se le resta una unidad. El cálculo de la posición angular se realiza por medio de $Pos = \frac{360}{(200\text{step}/\text{Rev})(\text{GearRatio})}$.

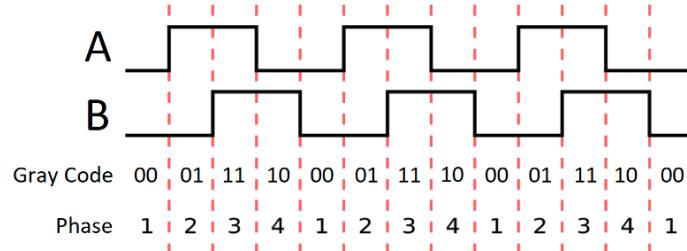


Figura 24: Rotación de un *encoder* [38]

La velocidad se puede medir con la frecuencia con la que se detecta el imán ya que esto indica que se ha completado 1 revolución. Sin embargo, este método no es tan preciso ya que no se tiene un período de muestreo constante para llevar a cabo la tarea.

Para medir la velocidad de una forma más precisa, se utilizó un temporizador con un período constante de 10 ms. Dentro de éste período se midió el cambio de posición restando la posición actual con la del período anterior. Después, se dividió entre el período de muestreo para obtener la velocidad. Esto nos permite saber tanto la magnitud como la dirección de la velocidad. Por último, se implementó un controlador PID para asegurarse de obtener la velocidad deseada. Sin embargo, para el control se debe convertir la velocidad deseada de RPM a pasos por segundo por medio de $Step/s = \frac{RPM(200\text{step}/\text{rev})}{60s/min}$.

7.5. Diseño de PCB

Para el diseño de la placa impresa se utilizó la calculadora de ancho de trazas ANSI debido a que ésta toma en cuenta el estándar IPC-2221/IPC-2221A. Asimismo, se utilizaron las mismas reglas de diseño expuestas en el documento ‘Especificaciones LPKF S103’ por Pablo Mazariegos.[39]

ANSI PCB TRACE WIDTH CALCULATOR				
Input Data		Results Data		
Field	Value	Units	Trace Data	
			Internal Traces	External Traces
Current (max. 35A)	1	Amps ▾	Required Trace Width [0.8] mm ▾	[0.31] mm ▾
Temperature Rise (max. 100°C)	10	°C ▾	Gross-section Area [0.03] mm ² ▾	[0.01] mm ² ▾
Cu thickness	1	oz/in ² ▾	Resistance [0.07] Ω Ohms	[0.17] Ω Ohms
Ambient Temperature	25	°C ▾	Voltage Drop [0.07] Volts	[0.17] Volts
Conductor Length	10	cm ▾	Loss [0.07] Watts	[0.17] Watts
Peak Voltage	12	Volts	Required Track Clearance [0.65] mm ▾	

Figura 25: Parámetros de diseño de PCB[40].

Parámetro	mm	mil
Ancho mínimo de pista	0.254	10
Ancho mínimo recomendado de pista	0.508	20
Espacio mínimo entre pistas	0.254	10
Ancho mínimo recomendado entre pistas	0.508	20
Tamaño mínimo de pad	Tamaño agujero + 0.508	Tamaño agujero + 20
Tamaño mínimo entre pads	0.254	10
Tamaño mínimo de pad para Vias	1.524	60
Tamaño recomendado para pad de resistores y capacitores	1.5748 y agujero de 0.9	62 y agujero de 36

Cuadro 2: Reglas de diseño utilizadas.2

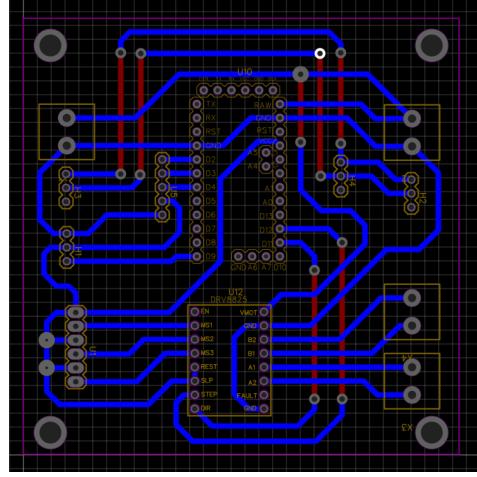


Figura 26: Placa del actuador

7.6. Pruebas de motores

Se realizaron pruebas de funcionamiento con el motor NEMA17 bipolar a 12 V, el cual consumió un mínimo de 0.84 A y un máximo de 1.15 A. Manejó velocidades de hasta 1200 pasos/segundo en ambas direcciones, lo cual se debe a la rapidez del Atmega328[32]. Éstos son conocidas por usar frecuencias de reloj desde 60 MHz hasta 600 MHz en modelos más recientes, además que una de sus aplicaciones más comunes es controlar motores stepper a altas velocidades (160,000 pasos/s). Al correr los experimentos con el motor bipolar, se notó que el motor se calentó al igual que el *driver*.

El motor unipolar se probó al mismo voltaje, sin embargo la corriente que consumió estuvo en el rango de 0.170 A a 0.345 A. Esto se debe principalmente a la resistencia de las bobinas, las cuales fueron de 57.0 Ω y 55.7 Ω, mientras que las resistencias del motor bipolar fueron de 16.1 Ω y 15.9 Ω. Éste motor, alcanzó una velocidad máxima de 600 pasos/segundo al conectarlo de la siguiente manera:

Pin	Color
B1	Azul
B2	Rojo
A1	Verde
A2	Negro

Cuadro 3: Conexión óptima del motor unipolar

Cuando no se siguió la conexión descrita, el motor alcanzó una velocidad máxima de 150 pasos/segundo al controlarlo con el *driver* DRV8825. Cuando no se siguió la configuración encontrada, el motor presentaba muchas vibraciones y luego de dar 2 vueltas regresaba 10 pasos aproximadamente. Cuando se conectó adecuadamente y se superó la velocidad establecida, el motor presentó muchas vibraciones y produjo un sonido agudo. El driver se calentó más de lo normal al correr este experimento, sin embargo el motor no se calentó y permaneció frío al tacto.

CAPÍTULO 8

Protocolo de comunicación daisy chain

En la fase anterior se desarrolló un protocolo de comunicación basado en la comunicación serial por UART[1]. Originalmente Yau desarrolló este protocolo para el ARM CORTEX M4 presente en el modelo TM4C123GH6PM de la TivaC de Texas Instruments y varios ATmega328P. En esta fase se continua trabajando con el ATmega328P como esclavo pero se evaluó la posibilidad de utilizar el ESP32 como maestro debido a que en un futuro se podría controlar a los motores de forma inalámbrica por medio de *WiFi* o *Bluetooth*.

8.1. SPI

Como se observó en 5, es posible interconectar varios microcontroladores via SPI para obtener una estructura tipo *Daisy-chain*. La ventaja de este método es que los datos automáticamente pasan por todo el circuito debido que cuando el maestro transmite un dato, la señal de reloj y la señal de SS le llegan a todos los dispositivos al mismo tiempo. La comunicación se vuelve bastante simple, si por ejemplo se desea enumerar cada dispositivo, se puede enviar un carácter conocido hasta que regrese al *Buffer* del maestro. Luego, cuando el maestro reciba el carácter que envió, significa que ha terminado de comunicarse con todos los dispositivos y el número de esclavos es el número de veces que envió el carácter.

En esta fase se intentó traducir el protocolo de Yau para utilizarse en SPI pero surgieron algunos problemas de compatibilidad principalmente debido a que el ESP32 es de 32 bits y sus registros no son fáciles de acceder. Las librerías nativas de Arduino no tuvieron un funcionamiento estable y únicamente se logró una comunicación exitosa al reiniciar a los esclavos. Se utilizó una librería llamada *ESP32DMASPIMaster* [41] para transmitir datos desde el maestro, sin embargo la comunicación duraba solo unos minutos antes de fallar. Después de recibir el dato por SPI, se envió por comunicación serial a un puerto USB de la computadora para mostrarlo en el monitor serial de Arduino como se muestra en 27.

The screenshot shows the Arduino IDE interface. On the left is the code editor with the file name 'Daisy-chain.ino'. The code is written in C++ and defines a class 'Master' with methods for initializing SPI, reading from slave 0, and performing calibration. It also includes a main loop for reading data from slave 0 and printing it to the serial port. On the right is the 'Serial Monitor' window titled 'COMM' which displays the received data from the slaves.

```

1 // DAIY-CHAIN.ino
2 // This sketch demonstrates how to connect multiple
3 // Arduino boards in a Daisy-chain configuration
4 // and read data from them sequentially via SPI.
5 // It uses the Master-Slave mode of the SPI library
6 // to have one Arduino act as the master and others as slaves.
7 // The master reads data from slave 0 and then moves on to slave 1, etc.
8 // The slaves are connected in a chain where the CS pin of one slave
9 // is connected to the SS pin of the next slave in the chain.
10 // The master connects to the first slave's CS pin.
11 // The slaves have their SCK and MOSI pins connected to the master's SCK and MOSI pins.
12 // The slaves have their MISO pins connected to the master's MISO pin.
13 // The master has its MISO pin connected to ground.
14 // The master has its CS pin connected to digital pin 10.
15 // The slaves have their CS pins connected to digital pins 9, 8, 7, 6, 5, 4, 3, 2, 1, and 0 respectively.
16 // The master has its SS pin connected to digital pin 11.
17 // The slaves have their SS pins connected to digital pins 10, 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
18 // The master has its SCK pin connected to digital pin 13.
19 // The slaves have their SCK pins connected to digital pins 12, 11, 10, 9, 8, 7, 6, 5, 4, and 3 respectively.
20 // The master has its MOSI pin connected to digital pin 11.
21 // The slaves have their MOSI pins connected to digital pins 10, 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
22 // The master has its MISO pin connected to digital pin 10.
23 // The slaves have their MISO pins connected to digital pins 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
24 // The master has its SS pin connected to digital pin 11.
25 // The slaves have their SS pins connected to digital pins 10, 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
26 // The master has its SCK pin connected to digital pin 13.
27 // The slaves have their SCK pins connected to digital pins 12, 11, 10, 9, 8, 7, 6, 5, 4, and 3 respectively.
28 // The master has its MOSI pin connected to digital pin 11.
29 // The slaves have their MOSI pins connected to digital pins 10, 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
30 // The master has its MISO pin connected to digital pin 10.
31 // The slaves have their MISO pins connected to digital pins 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
32 // The master has its SS pin connected to digital pin 11.
33 // The slaves have their SS pins connected to digital pins 10, 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
34 // The master has its SCK pin connected to digital pin 13.
35 // The slaves have their SCK pins connected to digital pins 12, 11, 10, 9, 8, 7, 6, 5, 4, and 3 respectively.
36 // The master has its MOSI pin connected to digital pin 11.
37 // The slaves have their MOSI pins connected to digital pins 10, 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
38 // The master has its MISO pin connected to digital pin 10.
39 // The slaves have their MISO pins connected to digital pins 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
40 // The master has its SS pin connected to digital pin 11.
41 // The slaves have their SS pins connected to digital pins 10, 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
42 // The master has its SCK pin connected to digital pin 13.
43 // The slaves have their SCK pins connected to digital pins 12, 11, 10, 9, 8, 7, 6, 5, 4, and 3 respectively.
44 // The master has its MOSI pin connected to digital pin 11.
45 // The slaves have their MOSI pins connected to digital pins 10, 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.
46 // The master has its MISO pin connected to digital pin 10.
47 void setup() {
48   // Set digital pin 10 as an output for CS
49   pinMode(MOSI, OUTPUT);
50   // Set digital pin 11 as an output for SS
51   digitalWrite(SS, HIGH); //Disable communications for now
52   //SPI.begin();
53   //SPI.setBitOrder(LSBFIRST);
54 }
55
56 void loop() {
57   // processing app.Serial.<init>()
58   ...
59   // note
60   Error opening serial port 'COM4'.
61 }
```

Figura 27: Resultados *Daisy-chain* con SPI.

8.2. UART

El protocolo estaba desarrollado originalmente para utilizarse basado en serial. Esta comunicación es bastante sencilla ya que solo se necesita 1 cable de datos entre cada esclavo. Un detalle importante fue que durante las pruebas fue necesario conectar la salida del último esclavo con la entrada del maestro para completar el lazo de modo que el maestro obtenga cierta retroalimentación de los esclavos.

Uno de los mayores cambios que se realizó fue cambiar la librería por otra llamada *HardwareSerial* que está hecha para trabajar con el ESP32 [42]. Otro cambio notable fue la traducción del código de C a Arduino ya que hay algunos tipos de datos que no están reconocidos o que cambian de nombre. Finalmente se optó por utilizar la computadora como maestro debido a que era mucho más versátil para manejar los datos obtenidos durante la experimentación. Además, ésto permite una conexión más universal al protocolo, ya que éste es independiente del hardware con el que se trabaje. Al ser basado en serial, se puede trabajar con cualquier microcontrolador o computadora que soporte esa funcionalidad.

8.3. Programación

Uno de los desafíos más grandes aparte de la coordinación del protocolo de comunicación fue la implementación del antirrebote de los interruptores. Como se mencionó anteriormente el *encoder* rotativo rebotó mucho al girar, por lo que en el código fue necesario implementar dos lecturas de sus pines para compensar los efectos no deseados. Primero se lee cuando hay un cambio en el pin, después se utiliza la función *millis()* para realizar una segunda lectura aproximadamente 10 ms luego de la primera. Esto se realiza de esta manera porque ese es el intervalo promedio que tarda el fenómeno del rebote. [43] Luego de este tiempo, la lectura que se realiza es del estado estable del interruptor, el cual no volverá a cambiar hasta el próximo paso del eje. Por esta razón, se recomienda implementar el antirrebote por medio de hardware usando un *Schmitt trigger* en conjunto con un circuito RC para capturar esas

fluctuaciones de voltaje y suavizarlas.

8.3.1. Control de motores

Como se mencionó anteriormente, el control de los motores estará a cargo de un *Atmega328*. Como se observa en 28 el *encoder* presentó un efecto conocido como rebote. Éste consiste en el comportamiento no ideal de cualquier interruptor que genera múltiples transiciones de una sola entrada. El rebote no es un problema importante cuando tratamos con los circuitos de potencia, pero causa problemas cuando tratamos con los circuitos lógicos o digitales porque ocasiona lecturas erróneas de sensores o interruptores.[43]



Figura 28: Rebote del *encoder*

Para llevar a cabo la lectura correcta de los sensores se implementó un sistema antirrebote digital. Éste consiste en leer constantemente el pin en donde se conectan las terminales del *encoder* y al detectar un cambio en su estado se ejecuta un algoritmo que maneja el antirrebote. Éste algoritmo se encarga de hacer una nueva lectura 10 ms después de la inicial, lo que garantiza que se capture el estado estable del pin.

Luego de obtener la información y asegurarse de que sea la correcta se compara el estado actual del *encoder* con el estado anterior. Esto permite obtener la posición del eje, el sentido y la velocidad de rotación. Con estos valores, se vuelve más sencillo implementar un controlador para rotar el eje y colocarlo en la posición deseada.

A continuación, se programaron dos controladores PID digitales, uno para la posición y otro para la velocidad. El de posición tiene como entrada la posición actual, la posición deseada como referencia y la velocidad de rotación como salida. El de velocidad tiene como entrada la velocidad deseada, como referencia la velocidad medida del *encoder*, como salida la velocidad que se debe enviar al *driver* para que la velocidad deseada sea igual a la medida. Por consiguiente, se limitó la velocidad para que al adoptar un valor menor a 1 paso/segundo el motor se pare. Esto se debe a las vibraciones observadas al enviarle velocidades menores a 1 paso/segundo al motor por medio de su *driver*.

Para calibrar el motor, se realizó una rutina que se ejecuta al encenderlo. Ésta mueve

Controlador	Kp	Kd	Ki
Velocidad (1)	1.3	0	0.0024
Posición (1)	1.6	0	0
Velocidad (2)	1.71	0	0.0014
Posición (2)	1.45	0	0

Cuadro 4: Constantes finales de control

el motor con una velocidad constante de 10 pasos/segundo hasta que se activa el sensor de efecto *hall*. Después, mueve el motor a la misma velocidad pero en sentido contrario y se detiene cuando el sensor se desactiva. Por último, se establece esa posición como el cero absoluto junto con una velocidad máxima de 1000 pasos/segundo y una aceleración máxima de 100 pasos/segundo².

8.3.2. Protocolo de comunicación final

Se optó por utilizar la librería *ArduinoJSON* para manejar los paquetes de datos que se envían en el bus de datos porque ésta ofrece una funcionalidad de serialización y deserialización de datos de forma nativa. La sintaxis *JSON* se deriva de la sintaxis de notación de objetos de JavaScript, pero el formato *JSON* es solo texto. Además, el formato *JSON* es un formato que se ha convertido en un estándar en el mundo de la programación por lo que se puede utilizar en varios lenguajes de programación.

La forma en la que trabaja el protocolo es la siguiente:

- Primero se actualiza el documento para envío de datos con el comando deseado.
- Después se serializa el paquete *JSON* y se envía automáticamente.
- Se recibe en el primer dispositivo y se verifica si llegó a su destino.
- Si llegó a su destino se deserializa el paquete completo y se ejecuta el comando
- Si no llegó a su destino se reenvía al siguiente dispositivo.

A continuación se resumen los comandos y sus funciones:

- 0: No hace nada
- 1: Cambio de id
- 2: Modo de control
- 3: Escribir posición/velocidad
- 4: Leer posición
- 5: Leer velocidad
- 6: Cambiar relación de engranajes

- 7: Cambiar Kp
- 8: Cambiar Kd
- 9: Cambiar Ki
- 10: Leer Kp
- 11: Leer Kd
- 12: Leer Ki
- 13: Leer relación de engranajes

Como los dispositivos están conectados en serie, la información debe darle la vuelta entera al sistema para regresar al maestro. A continuación se detallan los tiempos máximos en microsegundos de comunicación:

Baudrate	Serializar	Deserializar	Envío	Total	Fallo
115200	1180	3488	1666.7	6334	4632
0.5M	1220	936	384	2540	2096
1M	1048	220	192	1460	1284

Cuadro 5: Tiempos de comunicación con el protocolo

Con estos datos se puede obtener la frecuencia de comunicación mediante la siguiente operación matemática $f_{coms} = 1/t_{total}$ y el número de dispositivos máximos con $Max = f_{coms}/f_{actualizacin}$

Baudrate	Comunicación (Hz)	Actualización (Hz)	Dispositivos
115200	205	10	20
0.5M	394	10	39
1M	685	10	68

Cuadro 6: Dispositivos máximos

8.4. Resultado final

El resultado final fue un actuador capaz de interconectarse con otros de su mismo tipo en una conexión tipo *Daisy Chain*. Los actuadores son capaces de ejecutar un algoritmo de control de posición y velocidad en un *Atmega328p* que funciona como microcontrolador principal.



Figura 29: Prototipo final

```
1  {"command":10,"id":1,"payload":1.6}
2  {"command":11,"id":1,"payload":0}
3  {"command":12,"id":1,"payload":0}
4  {"command":10,"id":0,"payload":1.45}
5  {"comnd":11,"id":0,"payload":0}
6  {"command":12,"id":0,"payload":0}
7  {"command":13,"id":1,"payload":1}
8  {"command":13,"id":1,"payload":1}
9  {"command":13,"id":1,"payload":15}
10 {"command":13,"id":0,"payload":18}
11 {"command":13,"id":1,"payload":1}
12 {"command":13,"id":0,"payload":1}
13 {"command":4,"id":1,"payload":59.4}
14 {"command":4,"id":0,"payload":315}
15 {"command":4,"id":1,"payload":156.6}
16 {"command":4,"id":1,"payload":201.6}
17 {"command":4,"id":0,"payload":239.4}
18 {"command":4,"id":1,"payload":201.6}
19 {"command":4,"id":1,"payload":201.6}
20 {"command":4,"id":0,"payload":12.6}
21 {"command":4,"id":1,"payload":201.6}
22 {"command":4,"id":0,"payload":-178.2}
23 {"command":4,"id":1,"payload":84.6}
24 {"command":4,"id":0,"payload":-178.2}
25 {"command":4,"id":1,"payload":-86.39999}
26 {"command":4,"id":0,"payload":-178.2}
27 {"command":4,"id":1,"payload":-45}
28 {"command":4,"id":0,"payload":-149.4}
29 {"command":10,"id":1,"payload":1}
30 {"command":11,"id":1,"payload":1}
31 {"command":12,"id":1,"payload":1}
32 {"command":10,"id":0,"payload":0}
33 {"command":11,"id":1,"payload":1}
34 {"command":12,"id":0,"payload":0}
35 {"command":5,"id":1,"payload":100}
36 {"command":5,"id":0,"payload":0}
```

Figura 30: Comunicación con prototipo final

CAPÍTULO 9

Conclusiones

- Se observó que la comunicación más rápida fue la basada en SPI, pero su desventaja es el número de cables que se necesita para la interconexión.
- Se observó que las uniones utilizadas fueron muy exitosas y convenientes debido a que el usuario puede controlar la fuerza de sujeción según se necesite.
- Se observó que al favorecer el corte láser sobre la impresión 3D se pudo manufacturar más rápido e iterar el diseño más eficientemente. Por lo que se llegó al diseño final en 7 iteraciones y sólo se tuvo que manufacturar 3 veces.
- Se controló un motor unipolar conectando los extremos de sus bobinas al *driver* para motores bipolares. Sin embargo, se observó que su velocidad máxima estaba limitada al no conectar el centro de las bobinas al voltaje de alimentación del motor.
- Se observó que la forma más conveniente de acoplar los tornillos en la carcasa es por medio de ajustes a presión ya que no se necesitan herramientas.
- Se determinó que los pines de reinicio de los microcontroladores se deben dejar desconectados para evitar interferencias.

CAPÍTULO 10

Recomendaciones

- Se recomienda implementar un método antirebote por *hardware* debido a que es más efectivo y sencillo que por *software*. Ya que se observó que el *encoder* rebota bastante al hacer contacto con los switches y el código adquirió complejidad que podría haber comprometido el rendimiento general de los microcontroladores esclavos.
- Se recomienda fabricar las placas con alguna empresa como JLCPCB o PCBWAY debido a que ellos sí tienen la capacidad de realizar vías que conecten dos o más capas. De esta forma se evita el uso de alambres soldados lo que disminuye el ruido. Además que cuentan con el servicio de serigrafía para añadir gráficos indicadores.
- Se recomienda utilizar microcontroladores con el mismo nivel de voltaje como maestros y esclavos para evitar el uso de adaptadores en la comunicación serial. A pesar de que los microcontroladores como el atmega328 puedan funcionar a 3.3V sin problemas es mejor que todo funcione al voltaje recomendado.
- Respecto al diseño de los motores, se recomienda acoplar el eje a una pieza metálica más larga debido a que los motores bipolares utilizados tienen un eje demasiado corto. Esto podría solucionarse al utilizar los motores NEMA17 unipolares de la universidad pero las pruebas realizadas con éstos indican que se necesita un driver especializado para controlarlos adecuadamente.
- Se recomienda utilizar microcontroladores más rápidos como la serie teensy 3.0 y 4.0 [44] si se desea alcanzar velocidades mayores a 1200 pasos/segundo.
- Se observó que la capacitancia parásita de la placa causó interferencia con el pin de reinicio del microcontrolador, por lo que se recomienda dejarlo desconectado de la PCB.

CAPÍTULO 11

Bibliografía

- [1] P. Yau, «Diseño de un sistema de motores inteligentes en conjunto con un sistema de control del brazo robótico asistencial para cirugías estereotácticas,» Tesis de Ingeniería Mecatrónica, Universidad del Valle de Guatemala, Departamento de Ingeniería Biomédica, Mecatrónica y Electrónica, ene. de 2021.
- [2] J. D. Castillo, «Desarrollo de actuadores altamente dinámicos con aplicaciones en robótica bio-inspirada,» Tesis de Ingeniería Mecatrónica, Universidad del Valle de Guatemala, Departamento de Ingeniería Biomédica, Mecatrónica y Electrónica, ene. de 2021.
- [3] Lynxmotion. «LSS Servos.» (2021), dirección: <http://www.lynxmotion.com/>.
- [4] Dynamixel. «What is DYNAMIXEL?» (2022), dirección: <http://www.dynamixel.com/whatisdxl.php>.
- [5] Trinamic. «Motor Control Technology.» (2022), dirección: <https://www.trinamic.com/technology/motor-control-technology/>.
- [6] O. Robotics. «ODrive Communication Protocol.» (2021), dirección: <https://docs.odriverobotics.com/v/latest/protocol.html>.
- [7] D. Jones. «Control of Stepper Motors.» (2021), dirección: <http://homepage.divms.uiowa.edu/~jones/step/>.
- [8] S. Lawrence. «Daisy Chaining Serial Connections.» (2012), dirección: <http://www.wintergroundfairlands.com/2012/12/daisy-chaining-serial-connections.html>.
- [9] Elprocus. «What is a Stepper Motor: Types and Working.» (2013), dirección: <https://www.elprocus.com/stepper-motor-types-advantages-applications/>.
- [10] A. Jahejo. «Daisy Chain Network | Advantages and Disadvantages.» (2022), dirección: <https://computernetworktopology.com/daisy-chain-network/>.
- [11] D. Asturias. «Network Topology Guide: Types, Mapping, and Design.» (2021), dirección: <https://www.pcwld.com/network-topology-guide>.

- [12] A. Designer. «Keep the Petals on: Optimizing Daisy Chain Wired Communication.» (2018), dirección: <https://resources.altium.com/p/keep-petals-optimizing-daisy-chain-wired-communication>.
- [13] M. integrated. «How to Daisy-Chain SPI Devices.» (2021), dirección: <https://www.maximintegrated.com/en/design/technical-documents/app-notes/3/3947.html>.
- [14] J. L. Ishtiaque Amin. «Daisy Chain Implementation for Serial Peripheral Interface.» (2021), dirección: https://www.ti.com/lit/an/slvae25a/slvae25a.pdf?ts=1663425894130&ref_url=https%253A%252F%252Fwww.google.com%252F.
- [15] T. Instruments. «TMP104 Low-Power, Digital Temperature Sensor With SMAART Wire™ Interface (Rev. B).» (2018), dirección: <https://www.ti.com/document-viewer/TMP104/datasheet/device-images-dv#dv>.
- [16] Rohloff. «Planetary gear system.» (2022), dirección: <https://www.rohloff.de/en/experience/technology-in-detail/planetary-gear-system>.
- [17] «International Gear Conference 2014,» en *Forsthoffer's Best Practice Handbook for Rotating Machinery*, W. Forsthoffer, ed., Boston: Butterworth-Heinemann, 2014, pág. ix, ISBN: 978-0-08-096676-2. DOI: <https://doi.org/10.1016/B978-0-08-096676-2.10038-4>. dirección: <https://www.sciencedirect.com/science/article/pii/B9780080966762100384>.
- [18] F. Ren y D. Qin, «Investigation of the effect of manufacturing errors on dynamic characteristics of herringbone planetary gear trains,» en *International Gear Conference 2014: 26th–28th August 2014, Lyon*, P. Vexel, ed., Oxford: Chandos Publishing, 2014, págs. 230-239, ISBN: 978-1-78242-194-8. DOI: <https://doi.org/10.1533/9781782421955.230>. dirección: <https://www.sciencedirect.com/science/article/pii/B9781782421948500276>.
- [19] A. Lauletta, «Basics of harmonic drive gearing,» *Gear product news*, págs. 32-36, abr. de 2006. DOI: <https://web.archive.org/web/20160303211059/http://www.gearproductnews.com/issues/0406/gpn.pdf>.
- [20] J. P. Rafferty. «Harmonic Drive.» (2021), dirección: <https://www.britannica.com/technology/Harmonic-Drive>.
- [21] E. Saerens. «Exploded view of the different parts of a harmonic drive (top) and a detailed cross-section view of it (bottom).» (2019), dirección: https://www.researchgate.net/figure/Exploded-view-of-the-different-parts-of-a-harmonic-drive-top-and-a-detailed_fig12_334030874.
- [22] B. Chen, H. Zhong, J. Liu, C. Li y T. Fang, «Generation and investigation of a new cycloid drive with double contact,» *Mechanism and Machine Theory*, vol. 49, págs. 270-283, 2012, ISSN: 0094-114X. DOI: <https://doi.org/10.1016/j.mechmachtheory.2011.10.001>. dirección: <https://www.sciencedirect.com/science/article/pii/S0094114X11001935>.
- [23] TecScience. «How does a cycloidal drive work?» (2019), dirección: <https://www.tec-science.com/mechanical-power-transmission/planetary-gear/how-does-a-cycloidal-gear-drive-work/>.
- [24] EPC. «What is an Encoder?» (2022), dirección: <https://www.encoder.com/article-what-is-an-encoder>.

- [25] S. Basler, *Encoder und Motor-Feedback-Systeme*, 1.^a ed. Springer Fachmedien, 2016.
- [26] M. Murray. «HOW ROTARY ENCODERS WORK – ELECTRONICS BASICS.» (2021), dirección: <https://www.thegeekpub.com/245407/how-rotary-encoders-work-electronics-basics/>.
- [27] R. L. Norton, *Diseño de máquinas un enfoque integrado*. Pearson Education, 2011, ISBN: 9786073205894.
- [28] Arduino. «Arduino Uno Rev3.» (2022), dirección: <https://store-usa.arduino.cc/products/arduino-uno-rev3>.
- [29] Arduino. «Arduino Mega 2560 Rev3.» (2022), dirección: <https://store-usa.arduino.cc/products/arduino-mega-2560-rev3>.
- [30] Espressif. «ESP32.» (2022), dirección: <https://www.espressif.com/en/products/socs/esp32>.
- [31] Arduino. «Arduino Nano.» (2022), dirección: <https://store.arduino.cc/products/arduino-nano>.
- [32] Arduino. «Arduino Pro Mini.» (2022), dirección: <https://docs.arduino.cc/retired/boards/arduino-pro-mini>.
- [33] Arduino. «Arduino Micro.» (2022), dirección: <https://store.arduino.cc/products/arduino-micro>.
- [34] Microchip. «ATtiny85.» (2022), dirección: <https://www.microchip.com/en-us/product/ATTiny85>.
- [35] LaElectrónica. «Arduino y boards.» (2022), dirección: <https://laelectronica.com.gt/arduino-y-boards>.
- [36] autodesk. «Innovate it. Manufacture it. Autodesk it.» (2022), dirección: <https://www.autodesk.com/>.
- [37] BUN-CA. «Motores eléctricos: Buenas prácticas en eficiencia energética.» (2010), dirección: <https://www.bun-ca.org/wp-content/uploads/2019/02/FasciculoMotoresElectricos.pdf>.
- [38] S. Goldwasser. «Construction Guidelines for Basic Quadrature-Sin-Cos Decoder and Quad-A-B Interface Kits.» (2021), dirección: <https://www.repairfaq.org/sam/manuals/quadins1.htm>.
- [39] P. Mazariegos. «Especificaciones técnicas del servicio de fresado de circuitos.» (2022), dirección: <https://sites.google.com/uvg.edu/makerlabuvg/?pli=1&authuser=3>.
- [40] N. de Smith. «ANSI IPC-2221A PCB Trace Width Calculator.» (2021), dirección: <https://www.desmith.net/NMdS/Electronics/TraceWidth.html>.
- [41] hideakitai. «ESP32DMASPI.» (2022), dirección: <https://github.com/hideakitai/ESP32DMASPI>.
- [42] Espressif. «Arduino core for the ESP32, ESP32-S2, ESP32-S3 and ESP32-C3.» (2022), dirección: <https://github.com/espressif/arduino-esp32>.
- [43] A. Barela. «Debouncing.» (2020), dirección: <https://learn.adafruit.com/make-it-switch/debouncing>.

- [44] PJRC. «Teensy® 4.1 Development Board.» (2019), dirección: <https://www.pjrc.com/store/teensy41.html>.
- [45] N. Nisan y S. Schocken, *The Elements of Computing Systems: Building a Modern Computer from First Principles (History of Computing S.)* The MIT Press, 2008, ISBN: 9780262640688.
- [46] D. Lancaster. «Typewriter Cookbook.» (2010), dirección: <https://www.tinaja.com/ebooks/tvtcb.pdf>.
- [47] S. Safris. «A Deep Look at JSON vs. XML, Part 1: The History of Each Standard.» (2021), dirección: <https://www.toptal.com/web/json-vs-xml-part-1>.
- [48] R. M. M. Karl Johan Åström. «Feedback Systems: An Introduction for Scientists and Engineers.» (2010), dirección: https://books.google.com/books?id=cdG9fNqTDS8C&q=%22This+makes+reasoning+based+on+cause+and+effect+tricky%22&pg=PA1&redir_esc=y.
- [49] T. Instruments. «RS-422 and RS-485 Standards Overview and System Configurations.» (2010), dirección: <https://www.ti.com/lit/an/sl1a070d/sl1a070d.pdf>.

CAPÍTULO 12

Anexos

12.1. Proceso iterativo

A continuación se muestran las figuras que detallan el proceso iterativo:

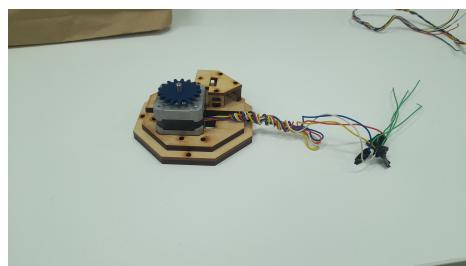


Figura 31: Prototipo del motor

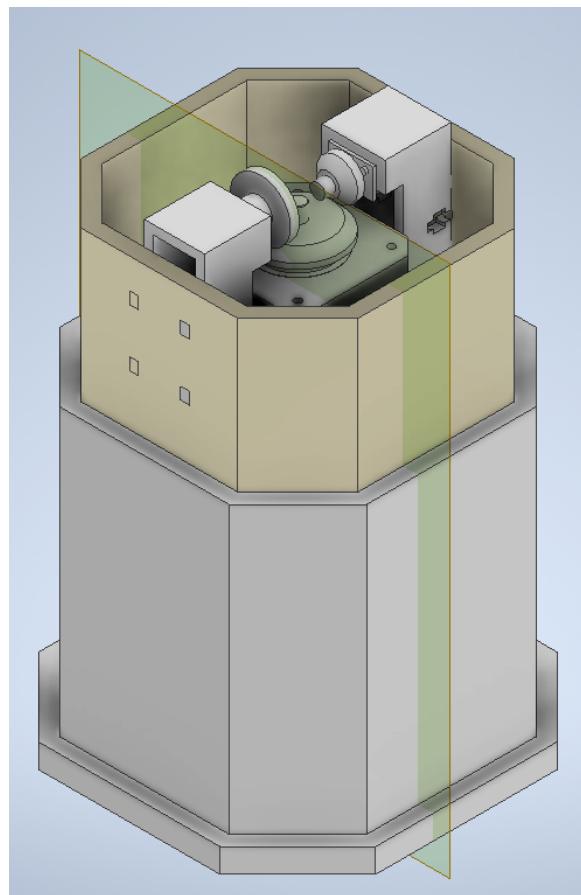


Figura 32: Primera iteración

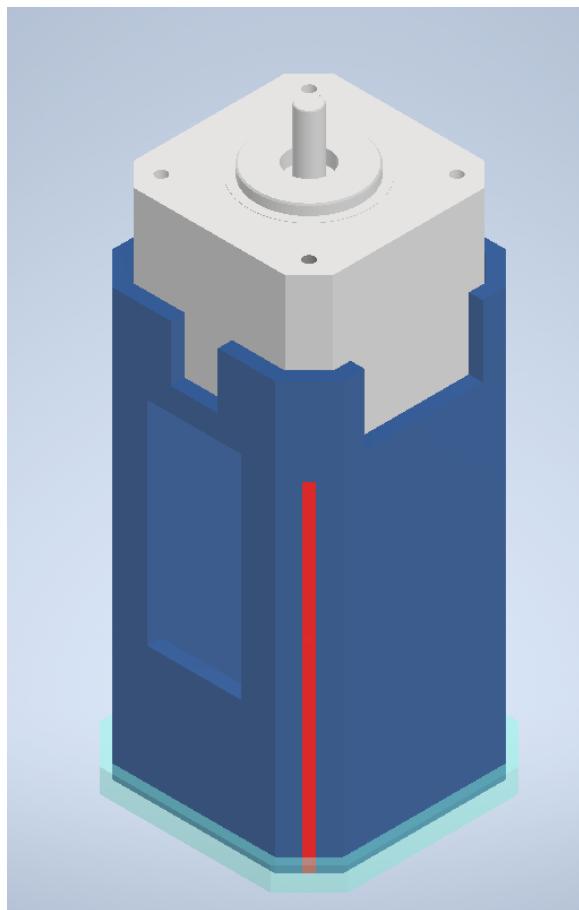


Figura 33: Segunda iteración

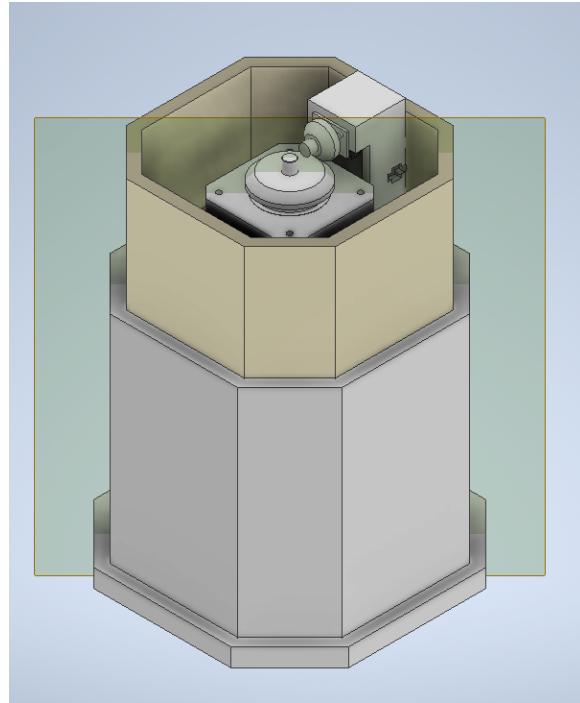


Figura 34: Tercera iteración

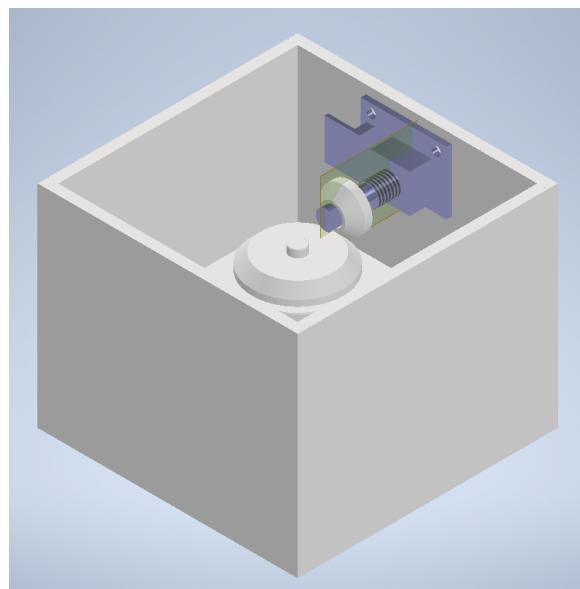


Figura 35: Cuarta iteración

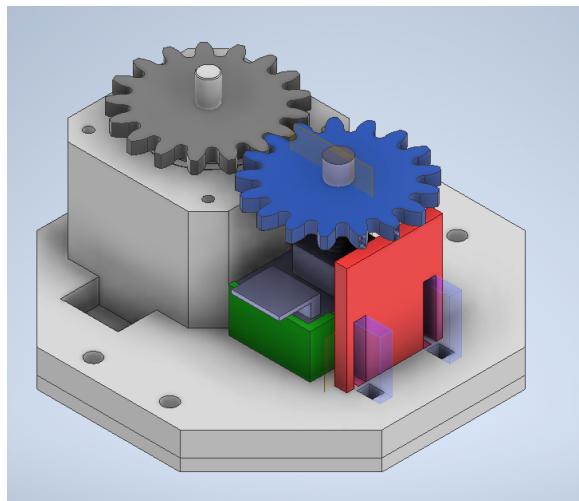


Figura 36: Quinta iteración (variación 1)

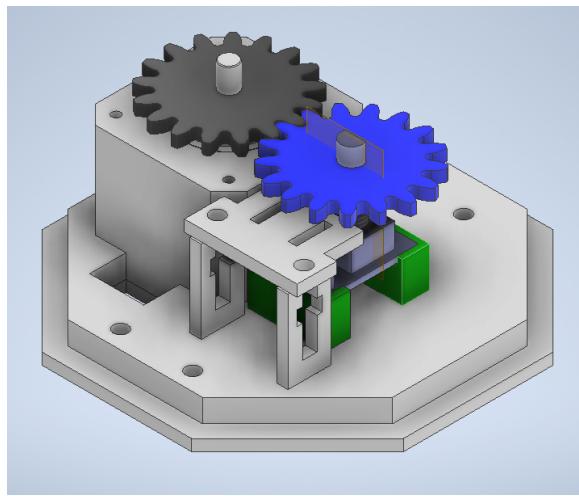


Figura 37: Quinta iteración (variación 2)

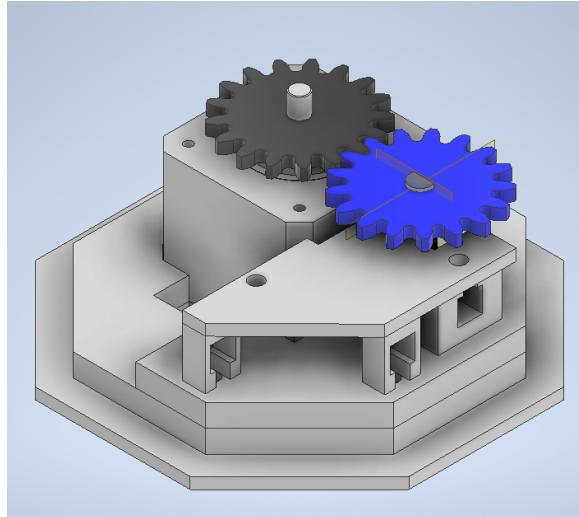


Figura 38: Sexta iteración (variación 1)

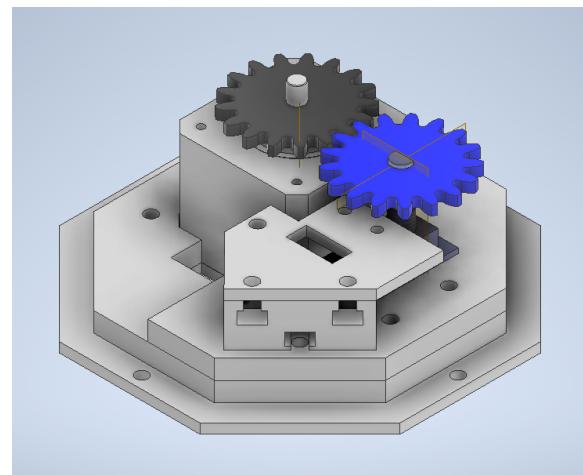


Figura 39: Sexta iteración (variación 2)

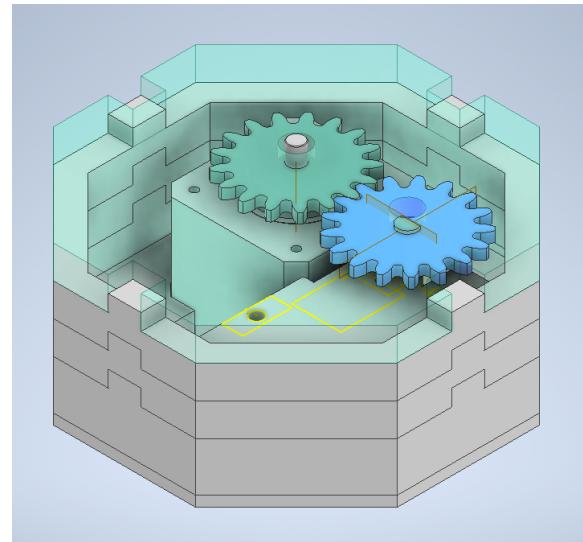


Figura 40: Tapadera de carcasa

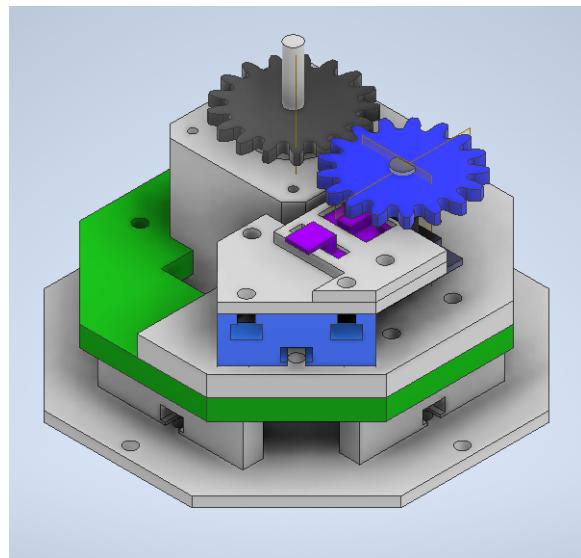


Figura 41: Séptima iteración

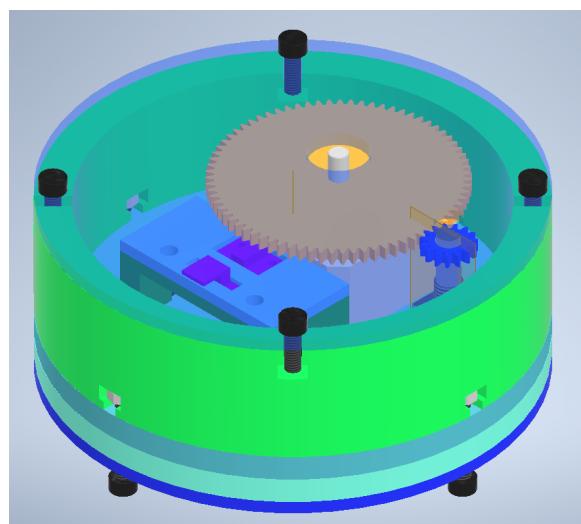


Figura 42: Octava iteración

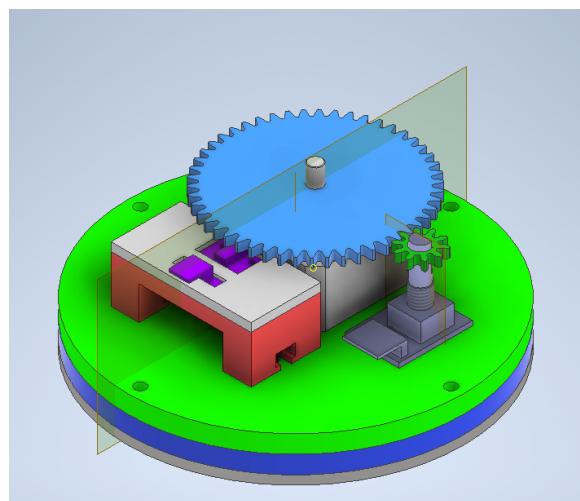


Figura 43: Novena iteración

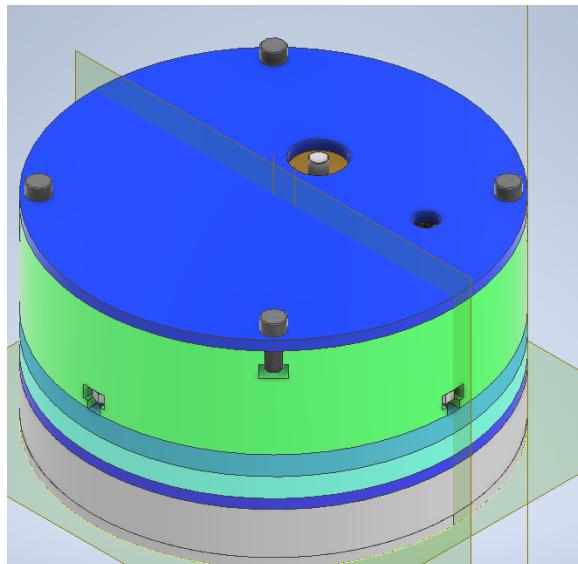


Figura 44: Décima iteración

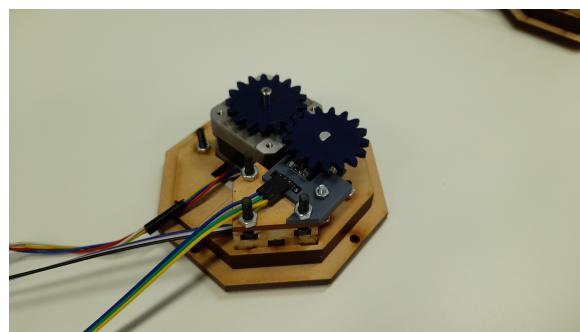


Figura 45: Primer Ensamble Físico



Figura 46: Pruebas de los Motores

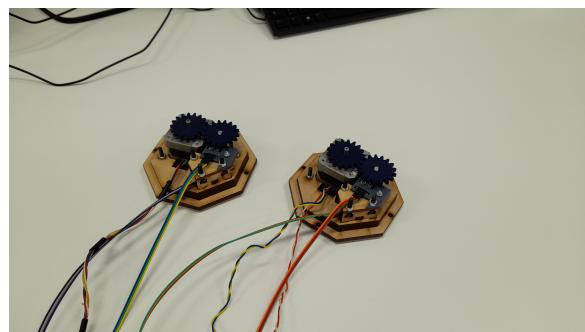


Figura 47: Primer par de prototipos

CAPÍTULO 13

Glosario

Buffer Es una región de una memoria utilizada para almacenar temporalmente datos mientras se mueve de un lugar a otro.[45]. 41

Daisy-chain Se define como una conexión en cadena donde se conecta múltiples dispositivos secuencialmente en una conexión de cable ininterrumpida.. 1, 13

Half-duplex En un sistema *half-duplex*, ambas partes pueden comunicarse entre sí, pero no simultáneamente; es decir la comunicación es una dirección a la vez.[46]. 4

JSON Es un formato de archivo estándar abierto y un formato de intercambio de datos que utiliza texto legible por humanos para almacenar y transmitir objetos de datos que consisten en pares de atributo-valor y matrices (u otros valores serializables).[47]. 4

Lazo cerrado La retroalimentación en lazo cerrado ocurre cuando las salidas de un sistema se vuelven a enrutar como entradas como parte de una cadena de causa y efecto que forma un circuito o bucle.[48]. 3, 5, 6

RS-485 Es un estándar que define las características eléctricas de los controladores y receptores para su uso en sistemas de comunicaciones serial. Las redes de comunicaciones digitales que lo implementan se pueden usar efectivamente en largas distancias y en entornos eléctricamente ruidosos.[49]. 4

SPI Es una interfaz de comunicación serial síncrona utilizada para la comunicación a corta distancia, principalmente en sistemas embebidos. La interfaz fue desarrollada por Motorola a mediados de la década de 1980 y se ha convertido en un estándar de facto. Las aplicaciones típicas incluyen tarjetas *Secure Digital* y pantallas LCD.[14]. 17