Diseño e implementación de un sistema de motores inteligentes con funcionalidades básicas

Daniela del Carmen Godínez Castillo

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

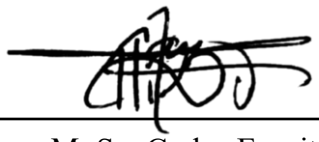
Diseño e implementación de un sistema de motores inteligentes con funcionalidades básicas

Trabajo de graduación presentado por Daniela del Carmen Godínez Castillo para optar al grado académico de Licenciada en Ingeniería Mecatrónica

Guatemala,

2024

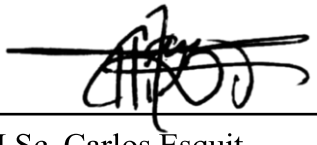
Vo.Bo.:



(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:



(f) _____

M.Sc. Carlos Esquit

(f) _____

M. Sc. Miguel Enrique Zea Arenales

(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

La realización de este trabajo ha sido un camino lleno de desafíos y aprendizajes, marcado por momentos de alegría, esfuerzo y apoyo de quienes estuvieron a mi lado. A lo largo de este proceso, he adquirido experiencias que me han enriquecido profundamente. Sobre todo, quiero agradecer a Dios, mi guía y fortaleza, por haberme sostenido en cada paso y por ser el pilar que me permitió alcanzar esta meta. Sin su presencia y dirección, este logro no habría sido posible.

Quiero expresar mi más profundo agradecimiento a mi familia, especialmente a mis padres, Eloisa Castillo y Yuri Godínez, por ayudarme a alcanzar mis metas. Sus palabras de aliento, sus sabios consejos, su constante presencia y la confianza que han depositado en mí para culminar este trabajo han sido fundamentales durante todo este proceso.

Quiero expresar mi sincero agradecimiento a la Ing. Joseline Ortiz, el MSc. Miguel Zea y el Ing. Pedro Castillo por su paciencia, orientación y dedicación durante la elaboración de este trabajo. Sus valiosos aportes han sido fundamentales para su culminación y me han inspirado para seguir avanzando en este camino.

Quiero expresar mi más sincero agradecimiento a todos mis amigos que me han acompañado en este camino, especialmente a Alba Rodas. Su amistad ha sido un pilar fundamental durante mis años universitarios, brindándome apoyo tanto en los momentos de alegría como en aquellos en los que sentía que no podía continuar. Su compañía ha sido una fuente constante de motivación y fortaleza, por ello, siempre estaré profundamente agradecida.

Por último, quiero dedicar una mención especial a quienes me acompañaron en mis noches de desvelo y momentos de locura: mis mascotas. Su presencia fue un consuelo invaluable durante este proceso.

Índice

Lista de figuras

Lista de cuadros

Resumen

En este trabajo, se diseñó e implementó un sistema de motores inteligentes con funcionalidades básicas, enfocado en aplicaciones de robótica y sistemas que requieren precisión. El proyecto aborda la integración de microcontroladores, drivers y sensores en una conexión tipo Daisy-Chain utilizando el protocolo SPI.

Se definieron las especificaciones de diseño, se seleccionaron los componentes adecuados, y se desarrollaron librerías en lenguaje C para la configuración y el control de los motores. Además, se diseñaron y fabricaron placas de circuito impreso (PCB) optimizadas y una carcasa compacta para garantizar robustez y eficiencia. Los resultados demuestran que el sistema es capaz de operar de manera confiable, con un protocolo de comunicación rápido y eficiente, y modularidad que permite futuras mejoras.

Palabras clave: motores inteligentes, comunicación serial, comunicación SPI, protocolo JSON.

Abstract

This study presents the design and implementation of a smart motor system with basic functionalities, focusing on robotics and applications requiring precision. The project integrates microcontrollers, drivers, and sensors in a Daisy-Chain connection using the SPI protocol.

Design specifications were defined, suitable components were selected, and configuration and control libraries were developed in C. Additionally, optimized printed circuit boards (PCBs) and a compact housing were designed and manufactured to ensure robustness and efficiency. The results demonstrate that the system operates reliably, with a fast and efficient communication protocol, and modularity enabling future improvements.

Keywords: smart motors, serial communication, SPI communication, JSON protocol.

CAPÍTULO 1

Introducción

El presente proyecto continúa con el desarrollo de motores inteligentes, siendo el sucesor del trabajo de Peter Yau y el antecesor de Juan Valenzuela. Aunque se ha avanzado en la implementación de protocolos de comunicación y mejoras en el diseño de motores inteligentes, aún quedan aspectos pendientes, por lo que este proyecto se centra en generalizar el uso de motores inteligentes para diversas aplicaciones, haciéndolos más versátiles y compatibles con una variedad de componentes, como sensores y sistemas de control. Se busca optimizar la implementación del código fuente, establecer librerías en lenguaje C y ampliar las funcionalidades mediante la adición de nuevos componentes. El objetivo es ofrecer una solución más completa y fácil de usar, que no solo sea eficiente, sino adaptable a distintos contextos, como la robótica y sistemas que requieren retroalimentación precisa de motores.

CAPÍTULO 2

Antecedentes

2.1. Motores inteligentes

Con los avances tecnológicos, la robótica ha comenzado a formar parte de la cotidianidad. En este contexto, el diseño del cableado y las conexiónes juegan un papel decisivo. Un buen cableado mejora la estética, optimiza el espacio y reduce las interferencias [1]. En la robótica es común el uso de microcontroladores y actuadores para el control de movimiento en el mecanismo. Sin embargo, la mayoría de los microcontroladores son incapaces de soportar más de tres actuadores. Por esta razón, se han buscado formas para usar varios actuadores con un microcontrolador [2].

Los motores inteligentes ofrecen una comunicación por medio de Daisy-Chain, que es un tipo de conexión para protocolos de comunicación serial síncronos en forma de bucle secuencial. En esta jerarquía, un microcontrolador es el cerebro de la cadena y transmite comandos a los demás microcontroladores, los cuales contienen un motor cada uno. Esto evita el cableado excesivo y soluciona la limitante de actuadores en un mecanismo robótico [2].

Por lo que, un motor inteligente tiene una placa integrada al motor. Esta placa contiene un microcontrolador para manejar el motor, además de otros dispositivos que permiten lecturas como posición y velocidad, arranque del motor, corriente inducida al motor y sensores de posicionamiento por si el usuario lo requiere.

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

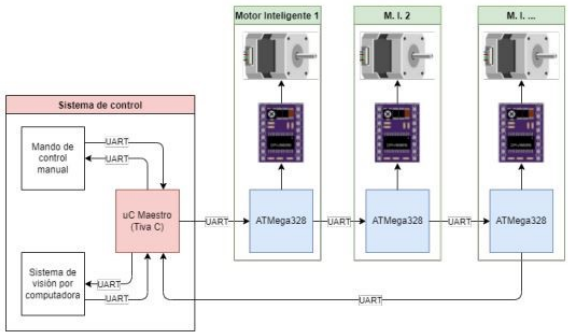
Con el avance de la robótica médica, los manipuladores seriales han emergido como herramientas fundamentales para la asistencia en cirugías, ofreciendo precisión y eficiencia. En este contexto, el ingeniero mecatrónico Peter Yau buscó desarrollar un brazo robótico asistencial con el objetivo de facilitar las operaciones quirúrgicas. Sin embargo, la metodología del proceso de manufactura de este brazo tuvo que adaptarse significativamente.

Peter Yau lideró el diseño de un sistema que integró motores stepper, drivers, sensores y un microcontrolador para optimizar el funcionamiento del brazo robótico. Como parte del desarrollo, se diseñó un protocolo de comunicación que enlazó un módulo de reconocimiento de imágenes y un sistema de control remoto. El módulo permitió reconocer caracteres ópticamente desde imágenes y convertirlos en texto legible para el brazo robótico, mientras que el sistema de control remoto ofreció ajustes manuales según las necesidades del usuario [3].

Además, Yau implementó un sistema de control dual: Un control manual de lazo abierto para movimientos básicos y un control de lazo cerrado para ajustes finos en los grados de libertad del brazo. Entre los avances más notables se encuentra el diseño de un sistema de motores inteligentes con una comunicación tipo daisy-chain, un método serial en bucle que facilitó la interconexión de los microcontroladores del sistema.

Para seleccionar los componentes adecuados, Yau realizó un análisis comparativo (trade study) de microcontroladores y drivers, optando por una comunicación UART dentro del esquema daisy-chain (figura 1). Este diseño incluyó un conjunto de comandos que permitían al usuario realizar configuraciones como la asignación de direcciones a los microcontroladores, el control del número de pasos, la implementación de microstepping y la definición de un punto cero relativo para medir giros acumulados del motor.

Figura 1: *Conexión Daisy-Chain.*



Sin embargo, se identificaron desafíos durante el desarrollo, como la confusión generada por el direccionamiento numérico de los microcontroladores, que se resolvió adoptando un esquema alfabético para mayor claridad y modularidad. Asimismo, el sistema se ajustó para utilizar códigos ASCII en la transmisión de comandos, mejorando la compatibilidad y precisión en la operación.

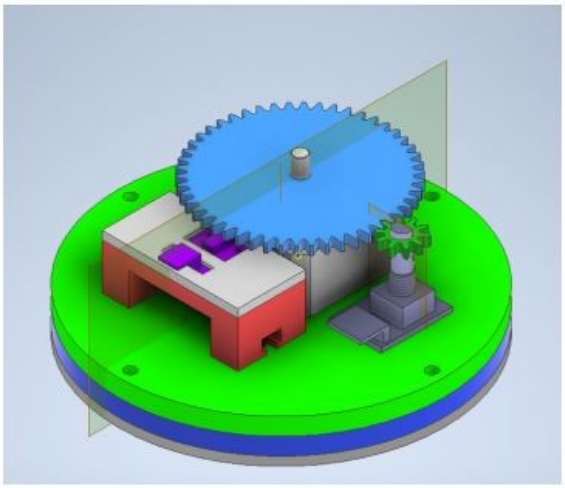
Finalmente, Yau recomendó ampliar el alcance de este sistema para su aplicación en motores DC y servo-motores, con el fin de hacerlo más versátil para futuros proyectos. También sugirió incorporar funciones avanzadas como monitoreo de posición y velocidad, un punto cero global y un control de lazo cerrado para mejorar la modularidad y flexibilidad del sistema. Estos desarrollos representan un paso importante hacia el diseño de sistemas robóticos más adaptables y eficientes.

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

Como continuación del trabajo desarrollado por Peter Yau, el ingeniero Juan Pablo Valenzuela amplió las capacidades del sistema de motores inteligentes con comunicación tipo daisy-chain para motores stepper [4]. Su enfoque incluyó no solo la mejora del sistema de comunicación, sino también el diseño de una carcasa para la electrónica de control que se integra al motor, la implementación de una caja reductora y la incorporación de un encoder para medir posición y velocidad. Además, Valenzuela buscó validar el protocolo de comunicación para garantizar la funcionalidad y robustez del sistema.

En su proyecto, Valenzuela trabajó con motores NEMA17, seleccionados previamente por Yau, eligió un driver y un sensor de efecto Hall, permitiendo determinar un cero absoluto para los motores. Para los microcontroladores, optó por usar un Arduino Pro Mini como esclavo y un ESP32 como maestro, con el objetivo de probar una posible transición del protocolo de comunicación de UART a SPI. Esta transición buscaba que cada esclavo de la cadena pudiera ejecutar comandos específicos. Sin embargo, debido a problemas de compatibilidad del ESP32 y la inestabilidad observada con las librerías de Arduino, Valenzuela decidió mantener el protocolo original (UART).

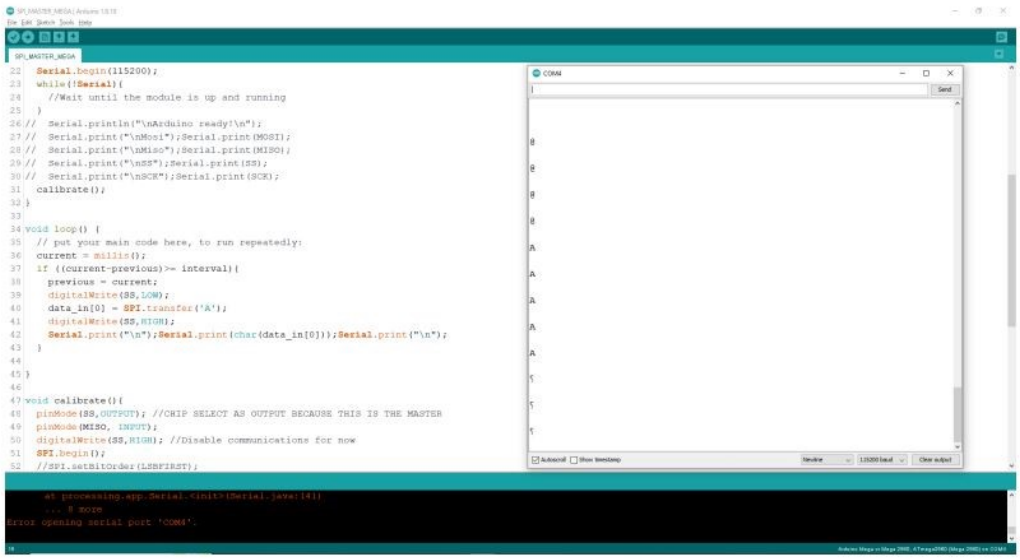
Figura 2: *Novena iteración para la retroalimentación del encoder rotativo al motor.*



El diseño de los motores incluyó una caja de transmisión con relación uno a uno, necesaria para obtener lecturas precisas de posición y velocidad mediante el encoder (figura 2). Además, se realizaron diez iteraciones en el diseño de la carcasa, buscando optimizar su funcionalidad y adaptación al motor.

Entre las limitaciones identificadas, Valenzuela destacó el problema del rebote en el encoder, el cual afectaba el rendimiento al generar pulsos erróneos. Para mitigar este problema, recomendó implementar un método antirrebote. Asimismo, sugirió que los microcontroladores involucrados en el sistema compartieran el mismo nivel de voltaje para evitar incompatibilidades eléctricas.

Figura 3: *Error al usar comunicación SPI.*



Este trabajo representa un avance significativo en el desarrollo de motores inteligentes y resalta los retos asociados a la integración de sistemas complejos con protocolos de comunicación en aplicaciones de robótica.

2.4. Motores inteligentes con cable reducido que utilizan comunicación por línea de alimentación de DC

En los documentos mencionados anteriormente, se ha buscado el desarrollo de motores inteligentes para motores stepper y se menciona que en un futuro se quisiera implementar para motores DC o servo motores [5]. En esta investigación se desarrollaron motores inteligentes para servo motor. La conexión de este tipo de motor requiere tres cables: alimentación, tierra y señal. Este cableado resulta tedioso para proyectos con varios motores, ya que tienden a enredarse en ejes o en la estructura. Por lo que los investigadores diseñaron una forma de eliminar los cables de conexión para consolidarlos en un solo cable.

Los motores inteligentes normalmente utilizan cables para la transmisión de datos y para la alimentación de cada microcontrolador (figura 4). Según los investigadores, estos motores emplean un sistema conocido como DC Power Supply Bus (CPS Bus), que se encarga de suministrar únicamente voltaje al bus para la transmisión de datos. Sin embargo, este enfoque también ayuda a reducir las perturbaciones electromagnéticas generadas durante la transmisión [5].

Con el objetivo de minimizar la cantidad de cableado necesario, los investigadores han propuesto usar un único bus, como se muestra en la figura 5. Este bus se emplearía exclusivamente para proporcionar energía a los motores, dejando de lado la transmisión de mensajes a través de este medio, lo cual contribuiría a un diseño más eficiente y con menos cables.

Por esto mismo, buscaron otra manera de transmitir los datos entre microcontroladores. A lo que propusieron utilizar el espectro ensanchado, que es un método de transmisión de datos por medio de radiofrecuencias. Con esto se eliminaría el cable de señal. Tomando en cuenta que algunos métodos de radiofrecuencia son interferidos por los mismos circuitos integrados del motor. El protocolo de telecomunicaciones implementado fue CDMA: (Acceso múltiple por división de código), el que permite que muchas señales comparten un canal de transmisión para optimizar la banda ancha. Además que soluciona la interferencia y el ruido del ambiente. Esto debido a que cada par de emisor y receptor utiliza una secuencia de señales aleatorias para la decodificación y codificación [5].

Figura 5: *Conexión de bus de alimentación propuesta.*

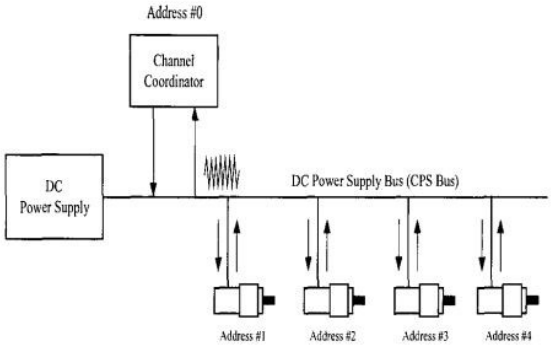
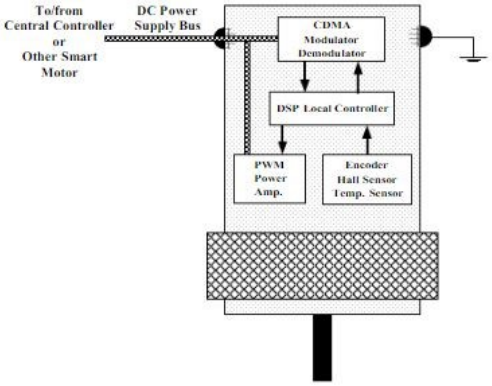
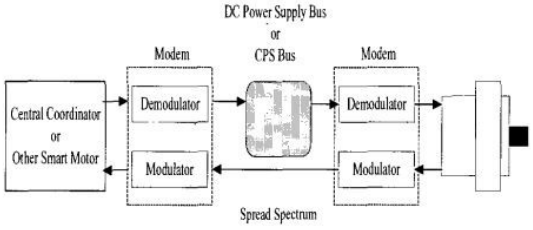


Figura 4: *Servo-motor inteligente propuesto.*



Por último, recomiendan diseñar un protocolo de comunicación específico para este sistema de motores inteligentes, figura 6, dado que no pudieron utilizar los buses propuestos en otras investigaciones, ya que no están diseñados para su bus de alimentación.

Figura 6: *Motor inteligente con protocolo CDMA propuesto.*



CAPÍTULO 3

Justificación

Este proyecto se plantea como una continuación y mejora de los trabajos realizados por Peter Yau y Juan Valenzuela, quienes realizaron importantes aportes en el desarrollo de motores inteligentes con comunicación tipo daisy-chain y sistemas de control integrados. Aunque sus investigaciones representan avances significativos en este campo, todavía existen áreas que necesitan ser fortalecidas para lograr motores inteligentes más versátiles y aplicables a diversas situaciones.

Yau desarrolló un protocolo de comunicación serial daisy-chain, pero quedó pendiente convertir el código en una librería, traducirlo a lenguaje C y añadir componentes para aumentar su funcionalidad. Valenzuela, por su parte, mejoró el diseño mecánico y de control, optimizó el protocolo de comunicación y añadió componentes que facilitaron el uso de los motores. Sin embargo, es necesario integrar estos aportes para lograr un sistema más completo.

El presente proyecto busca establecer un protocolo de comunicación definido, implementar librerías en lenguaje C y reducir la cantidad de cables necesarios, lo que es crucial en aplicaciones móviles y robóticas. Además, se propone incorporar encoders para medir y controlar posición, velocidad y aceleración, mejorando el rendimiento en aplicaciones que requieren alta precisión.

El impacto esperado incluye motores inteligentes versátiles y adaptables a múltiples aplicaciones, la simplificación de su uso en sistemas complejos y la apertura de nuevas líneas de investigación en robótica dentro del entorno universitario. Con estas mejoras, se busca consolidar los avances previos y crear una base sólida para el desarrollo de sistemas de motores inteligentes eficientes y flexibles.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar e implementar un sistema de motores inteligentes con funcionalidades básicas.

4.2. Objetivos específicos

- Definir las especificaciones de diseño y funcionalidad del motor inteligente.
- Diseñar, fabricar y ensamblar el motor inteligente.
- Diseñar y desarrollar librerías de configuración y control de los motores inteligentes.

CAPÍTULO 5

Alcance

El proyecto tiene como alcance principal el diseño e implementación de un sistema de motores inteligentes con funcionalidades básicas que permitan su uso en diferentes aplicaciones. Incluye la definición y optimización del protocolo de comunicación, la programación en lenguaje C, y el desarrollo de librerías de control específicas para cada microcontrolador involucrado.

También se busca reducir la cantidad de cables de operación, lo que es especialmente beneficioso en sistemas móviles, como los usados en robótica, donde es importante controlar con precisión la posición, velocidad y aceleración de los motores. Además, se incorporará un encoder para proporcionar retroalimentación al sistema. Finalmente, este proyecto pretende abrir nuevas oportunidades de investigación en el área de robótica dentro de la Universidad.

CAPÍTULO 6

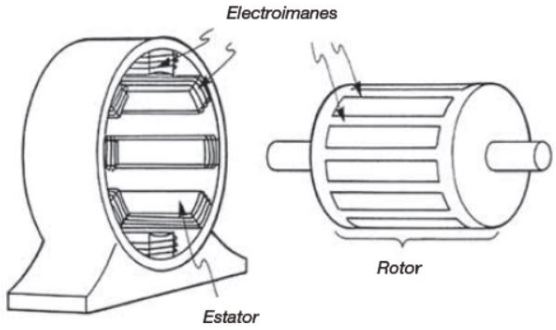
Marco teórico

Para comprender el desarrollo y la implementación de los motores inteligentes, es necesario situar la investigación dentro del contexto teórico. A continuación, se presenta lo que sustenta el estudio mientras se proporciona una base sólida para el análisis de los resultados, facilitando la comprensión de los conceptos clave y su aplicación en la actualidad.

6.1. Motores eléctricos de corriente continua (DC)

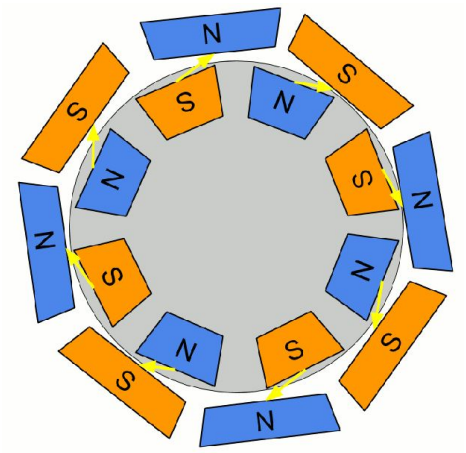
Los motores eléctricos de corriente continua permiten la conversión de energía eléctrica a movimiento. Al ser de corriente continua, también llamada corriente directa, el par del motor depende del campo magnético [6], generado por un flujo de corriente en los devanados (conductores). Está compuesto por un estator, rotor, eje y bobinas. El estator es la parte estacionaria mientras que el rotor, la parte móvil. Al interactuar entre sí, permiten que se genere un campo magnético y, por lo tanto, que el eje del motor gire.

Figura 7: *Partes principales de un motor eléctrico de corriente continua.*



Funciona generando un campo magnético cambiante que varía con la posición del rotor. Este campo magnético es producido por el flujo de corriente en los devanados del motor. Para generar el cambio del campo magnético, es necesario que la dirección del flujo de corriente en los devanados también cambie, esto se logra por el conmutador, el cuál altera la polaridad de la tensión aplicada a los devanados. De este modo, el conmutador actúa como un inversor mecánico que convierte la corriente continua a corriente alterna para los devanados, permitiendo el funcionamiento continuo del motor [6].

Figura 8: *Representación de atracción de polos opuestos del estator y rotor.*



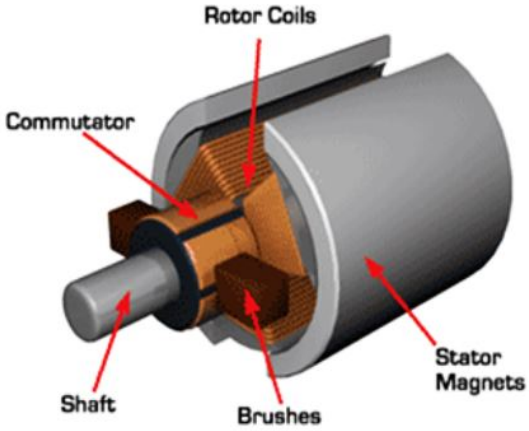
Hay dos tipos de motores eléctricos de corriente continua, los cuáles son:

6.1.1. Brushed DC Motors

Se les conoce también como motor de corriente continua de escobillas. Como indica su nombre, utiliza escobillas en el conmutador para realizar el cambio de polaridad en el rotor [7]. Las escobillas son las que se encargan de conmutar la corriente de las bobinas. Los puntos

de contacto entre las escobillas y el conmutador son importantes debido a que permiten la rotación continua del eje, alternando el encendido y apagado de los puntos.

Figura 9: *Componentes de un motor con escobillas.*

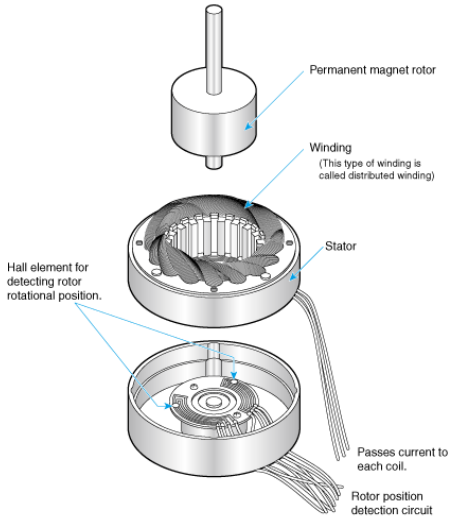


Sin embargo, estos motores se han caracterizado por su corto límite de vida útil debido a la fricción provocada por cambiar de polaridad lo que hace que las escobillas se desgasten. Además, esta fricción produce pérdida de calor lo que repercute en la eficiencia del motor. Por otro lado, al aumentar la velocidad disminuye el torque del motor; la relación de torque y velocidad no es proporcional [6].

6.1.2. Brushless DC Motors

Este tipo de motor opera sin necesidad de conmutador o escobillas ya que la conmutación de las bobinas se hace electrónicamente a través de un controlador [7]. Al evitar el uso de un conmutador mecánico se reduce el peso del motor y, a su vez, evita la fricción y pérdida de calor [6]. Sin embargo, su costo es elevado comparado con los motores de escobillas. El torque en estos motores es lineal, por lo que el torque será proporcional a la corriente inducida al motor. Además, la relación entre torque y velocidad es proporcional.

Figura 10: *Componentes de un motor sin escobillas.*

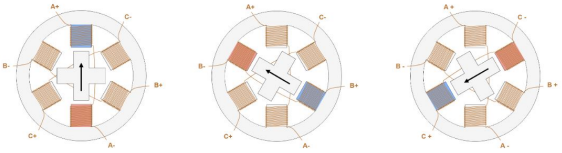


Los motores de corriente continua sin escobillas más comunes son los siguientes:

1. Motor Stepper

   Dentro de los motores sin escobillas se encuentra el motor paso a paso, también conocido como motor stepper. Este tipo de motor eléctrico realiza la rotación del eje en pequeños incrementos, denominados pasos [8]. Son ideales para mecanismos que requieran precisión, debido a que se pueden mover un paso a la vez por cada pulso aplicado. Estos pasos suelen variar entre 1.8° a 90°.

Figura 11: *Funcionamiento de los pasos.*

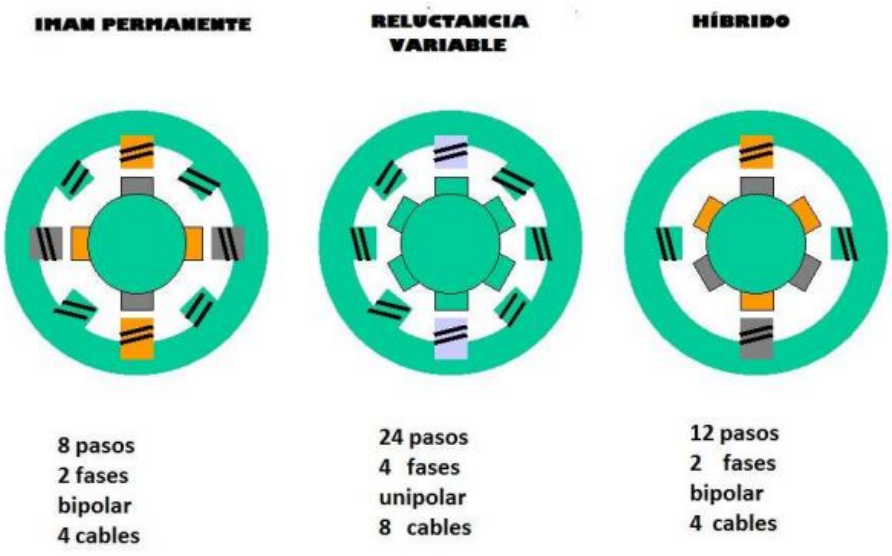


Hay tres tipos de motores stepper 12 :

- De reluctancia variable, el cual está compuesto por un rotor en forma de engranaje con dientes sobresalientes y el estator es una serie de electroimanes los cuales se deben de alimentar para crear una rotación.
- De imán permanente, el rotor es un imán permanente que tiene franjas alternantes de polaridad positiva y negativa. Los campos electromagnéticos son creados por terminales externos.

- Híbridos, que son una combinación de reluctancia variable e imanes permanentes. Siendo el rotor, el engranaje con dientes sobresalientes y los campos electromagnéticos son creados por los dos terminales externos.

Figura 12: *Tipos de motor stepper.*



2. Servo Motor

   Otro tipo de motor que se encuentra dentro de la familia de motores sin escobillas son los servo motores, el cuál permite controlar con precisión la posición del eje. La principal característica de este motor es que dan retroalimentación correctiva al control del motor. Esta retroalimentación corrige el error de posición del sistema. Cuenta con tres conexiones, siendo de un par de alimentación y la tercera es de señal, la que permite saber el ángulo del actuador y retroalimentar el control [6].

6.2. Módulo controlador de motor (driver)

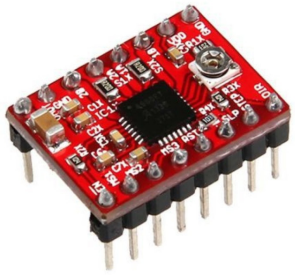
Se le conoce como controlador de motor. Se caracteriza por funciones tales como: Arranque/detención del motor y establecer la velocidad, par y dirección. Los motores requieren corriente, algo que un microcontrolador no proporciona debido a su bajo nivel de operación [6]. Los controladores de motor, conocidos como drivers, actúan como intermediarios entre el microcontrolador y el motor. Su función principal es incrementar la corriente suministrada al motor, ya que los microcontroladores no pueden proporcionar suficiente energía directa-

mente para hacerlo funcionar correctamente. De esta manera, el driver toma las señales del microcontrolador, las amplifica y las envía al motor para controlar su movimiento de manera eficiente.

Para la elección de un driver, se debe tomar en cuenta lo siguiente:

- Voltaje y corriente que requiere el motor.
- Compatibilidad entre motor y driver.
- Protocolo de comunicación del driver.

Figura 13: *Driver DRV4988.*



6.3. Encoder

Es un dispositivo que convierte el movimiento rotatorio en un señal eléctrica, la cual permite la retroalimentación negativa del movimiento para poder aplicar un controlador [9]. Esto permite monitorear la posición y velocidad del motor.

Los encoders se dividen en dos categorías de seguimiento:

- Lineales: permiten obtener información del movimiento lineal.
- Rotativos: permiten obtener información del movimiento rotacional.

Los encoders rotativos, se dividen en dos categorías de medición que son absolutos e incrementales. Los absolutos son los que proporcionan información actual de la posición y velocidad angular [9]. Mientras que las señales de salida de los incrementales son proporcionales al desplazamiento efectuado por el eje.

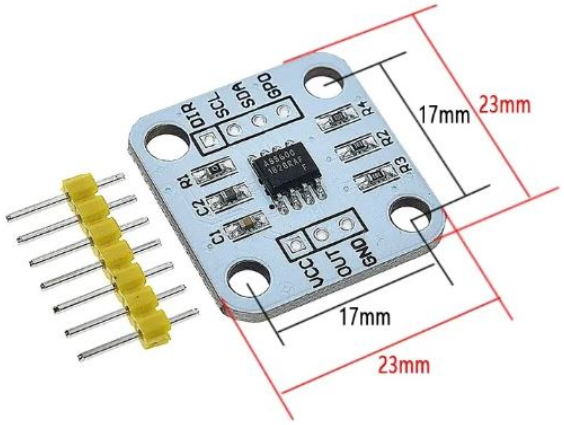
La última categoría de los encoders, depende de la tecnología de seguimiento:

- Óptico: su lectura es precisa y exacta, sin embargo, son susceptibles a factores ambientales como el calor y suciedad.
- Magnético: no es afectado por factores ambientales. Pero su lectura es menos precisa. Son absolutos.
- Capacitivo: combina los aspectos del encoder magnético y óptico.

6.3.1. Encoder magnético AS5600

Este sensor se clasifica como absoluto y magnético. Lo que permite obtener alta precisión en el desplazamiento angular del eje del motor [10].

Figura 14: *Encoder magnético AS5600.*



Sus características principales son:

- Alimentación de 5 V.
- Salida como Protocolo I2C o PWM para obtener la posición angular.
- Requiere un imán especial que tiene que estar entre 0.5 a 3 milímetros de distancia con el sensor.
- Maneja datos de 12 bits.
- Resolución de 0.08789 °/bit.

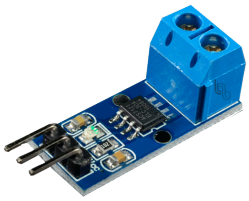
6.4. Sensor hall effect

Como se mencionó con anterioridad sobre el principio de efecto Hall, este dispositivo trabaja bajo ese principio donde permite aumentar la tensión inducida en un conductor por el campo magnético generado.

Su funcionamiento permite que sea candidato para la detección de movimiento y proximidad ya que ofrece una alta precisión por su alta sensibilidad y resistencia a condiciones ambientales variables [11]. En la detección de movimiento se puede utilizar como encoder, generando retroalimentación para el control. Por el lado de la proximidad, se puede utilizar como final de carrera.

Además, este sensor permite medir corriente de manera eficiente gracias a su capacidad para detectar los campos magnéticos generados por el flujo de corriente a través de un conductor. Esto lo hace ideal para monitorear la corriente de forma no invasiva, sin interrumpir el circuito, proporcionando una solución práctica y segura en sistemas eléctricos y de control.

Figura 15: *Sensor Hall Effect.*



6.5. Final de carrera

Son dispositivos electrónicos o mecánicos que permiten detectar el final de un movimiento, ya sea rotativo o lineal [6]. Su funcionamiento es por medio de pulsos, cuando detecta el final del movimiento por accionamiento mecánico (palanca o émbolo) o accionamiento electrónico (botones) envía un bit lógico para el controlador donde es procesado.

El accionamiento mecánico se acciona al entrar en contacto con el objeto en movimiento, generando una señal binaria. Sin embargo, con el tiempo puede llegar a sufrir desgaste físico.

Por otro lado, el accionamiento electrónico emplea el uso de sensores ópticos o capacitivos, que no requieren contacto físico. Ya que estos solo detectan la proximidad del objeto.

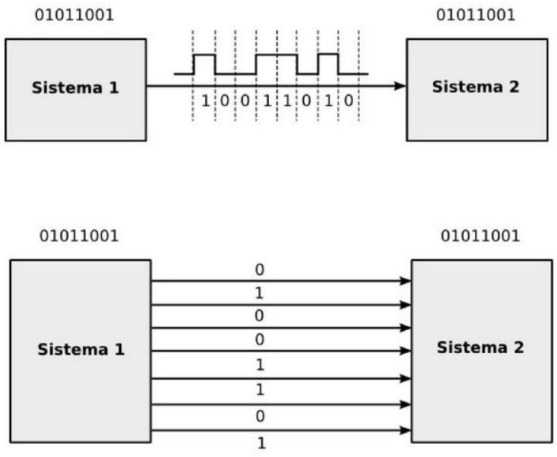
Su conexión eléctrica es por dos pines, donde se cierra y abre el circuito dependiendo de su configuración. Puede ser normalmente cerrado o abierto.

Figura 16: *Final de carrera mecánico.*

6.6. Transmisión de datos y comandos

En la comunicación serial, la transmisión de bits se da de forma sucesiva y de uno a uno, utilizando solo un canal [12]. Lo que permite que el receptor junte los bits enviados por el transmisor para formar el mensaje.

Figura 17: *Diferencia de conexión entre comunicación serial y paralela.*

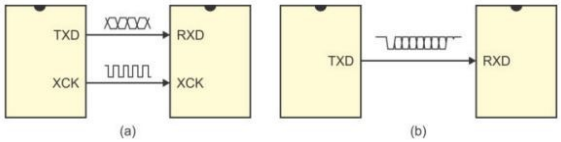


Cuando se transmite una señal del reloj, un pulso generado para sincronizar sistemas, se le conoce como comunicación serial síncrona [13]. Para lograrlo, el transmisor utiliza un elemento de sincronización conocido como "bandera". Este elemento actúa como un marcador que indica el inicio o fin de una transmisión de datos, permitiendo al receptor identificar claramente los límites del mensaje y sincronizarse correctamente para procesar la información.

Ahora bien, en la comunicación serial asíncrona la transmisión no depende del reloj. Por lo que cada dispositivo tiene una señal de reloj independiente [13]. En esta, se utiliza un

bit de start y stop para indicar cuando inicia y finaliza la comunicación. Y el transmisor y receptor se deben de configurar con la misma velocidad de transferencias (bit/segundo) y formato de datos.

Figura 18: *Representación de comunicación síncrona y asíncrona.*



Una de las ventajas de utilizar este tipo de comunicación es que se puede realizar entre microcontroladores, microcontrolador a computadora o cualquier dispositivo que tenga puerto de comunicación serial.

Por otro lado, el tipo de transmisión se clasifica de la siguiente manera:

- Simplex: donde se transmite en una dirección.
- Half-duplex: se puede transmitir en ambos sentidos pero en una dirección al mismo tiempo.
- Full-Duplex: la transmisión es en ambos sentidos. Cuenta con dos canales diferentes para la transmisión y recepción de datos.

Existen diferentes protocolos de transmisión que utilizan como base la comunicación serial, algunos de los cuales son los siguientes:

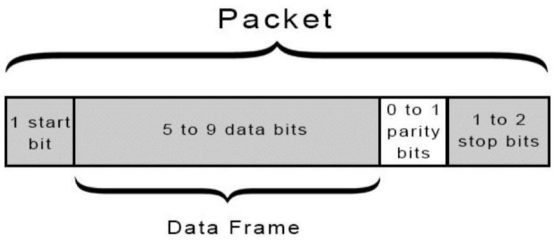
6.6.1. Universal synchronous and asynchronous serial receiver and transmitter (USART)

El protocolo USART, por sus siglas en inglés Universal Synchronous and Asynchronous Serial Receiver and Transmitter, fue uno de los primeros protocolos de comunicación serial. Este permite la comunicación asíncrona o síncrona, según lo requiera el usuario.

En el modo síncrono y asíncrono se utilizan dos líneas de datos, la de transmisión (TX) y recepción (RX), que se conectan entre sí en los dispositivos. Sin embargo, como se mencionó, la comunicación síncrona necesita de reloj. Por lo que, en ese modo se necesita una línea más, la cual conectará los relojes de los dispositivos. La forma de conexión se puede observar en la figura 18 .

El envió de datos está compuesto por un bit de start, nueve bits de datos, un bit de paridad (opcional) y un bit de stop [13]. El bit de paridad se utiliza para verificar si la comunicación se está realizando correctamente. El usuario decide si se lee como bit alto (1) o bit bajo (0).

Figura 19: *Transmisión de paquete de data.*



Algunas ventajas que USART proporciona son:

- Disponibilidad de varias velocidades de transmisión (bit/segundos).
- El bit de paridad permite verificar errores en la comunicación.
- Puede ser síncrona o asíncrona, dependiendo del uso que se requiera.

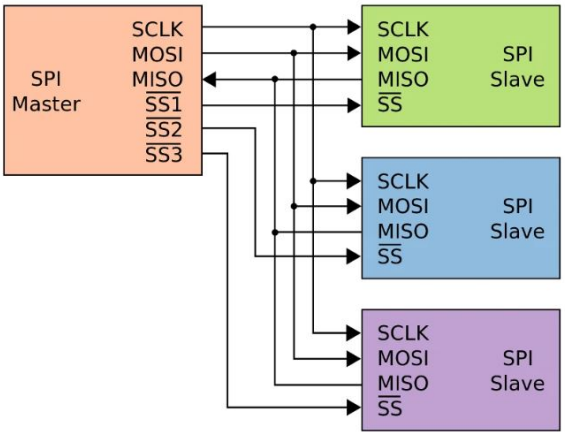
6.6.2. Serial peripheral interface (SPI)

La característica principal de este protocolo es que opera con un dispositivo maestro y otro como esclavo. Su única forma de operación es síncrona, por lo que es necesario el reloj para la transmisión de datos [13].

Usa cuatro líneas de transmisión los cuáles son:

- Serial data out (SDO) o Master output slave input (MOSI), es la línea de transmisión.
- Serial data in (SDI) o Master input slave output (MISO), es la línea de recepción.
- Serial clock (SCK-CLK), es la línea de sincronización controlada por el reloj.
- Slave select (SS), es la línea de selección de dispositivo esclavo. Esta puede ser opcional si el usuario lo desea.

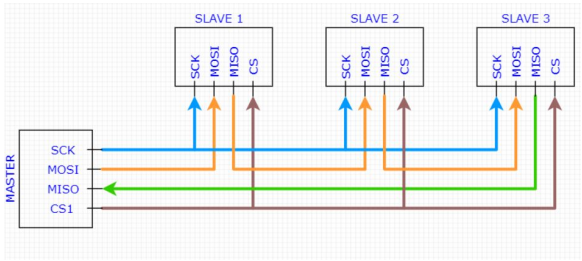
Figura 20: *Conexión paralela (tradicional) de comunicación serial para protocolo SPI.*



Este protocolo permite que la comunicación sea transmitida para varios dispositivos, por lo que se pueden conectar varios esclavos a la vez. El maestro inicia la comunicación debido a que tiene el control del reloj y determina cuándo necesita una respuesta del esclavo.

Los modos de comunicación dependen del slave select [14]. Una forma es que el maestro seleccione a qué esclavo dirigirse, esto sería una conexión paralela entre dispositivos (figura 20). Ahora bien, puede ser por medio de daisy-chain, donde la única salida de selección es la que activa sucesivamente cada dispositivo. Esta conexión es en serie (figura 21).

Figura 21: *Conexión en serie (Daisy-Chain) de comunicación serial para protocolo SPI.*



Las ventajas que ofrece el SPI son las siguientes:

- Es full-duplex, ya que la transmisión y recepción de datos son dos líneas independientes entre sí.
- No hay bit de inicio o paridad, lo que permite que el mensaje sea directo.
- Es el protocolo más rápido, comparado con USART.

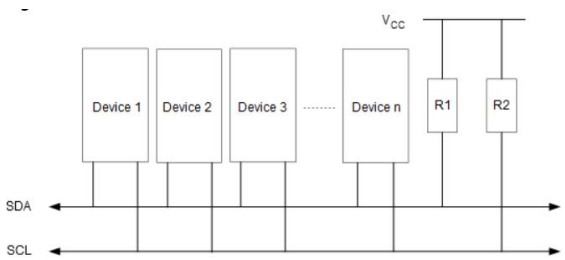
6.6.3. Two-wire serial interface (TWI)

Este protocolo síncrono es común para intercambio de datos con circuitos integrados o microcontroladores. Sin embargo, externamente se necesita conectar dos resistencias en forma de pull-up (elevan la tensión del circuito), debido a que es open-drain [14] que usa una resistencia para hacer la transición digital entre 0 y 1.

Utiliza dos líneas de conexión:

- Serial data (SDA) es la señal de datos.
- Serial clock (SCL) es la línea de señal del reloj.

Figura 22: *Conexión del protocolo de comunicación TWI.*



La conexión puede tener múltiples maestros y esclavos. El reloj, que depende del maestro, solo se usa para sincronizar el funcionamiento de los dispositivos. La transmisión de datos se da por un bit de arranque, siete bits de mensaje, y un bit que indica si se necesita que el dispositivo receptor lea o escriba el dato enviado, se puede observar en la figura 23 .

Figura 23: *Envió de datos del protocolo TWI.*

1 Byte							
B7	B6	B5	B4	B3	B2	B1	B0
Dirección dispositivo							R/W

Los esclavos al compartir la misma línea de señal de datos reciben simultáneamente el primer byte. Sin embargo, solo el que contiene la dirección recibirá el mensaje.

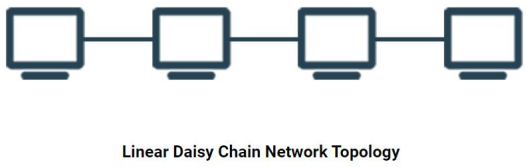
Las ventajas de este protocolo son:

- Soporta múltiples maestros y esclavos.
- Utiliza solo dos líneas de comunicación.

6.7. Conexión daisy-chain para protocolos de comunicación serial

Es una forma de conexión que se puede implementar en protocolos de comunicación serial, en la cúal los dispositivos están conectados en forma de secuencia o de anillo.

Figura 24: *Conexión Daisy-Chain en topología de red.*

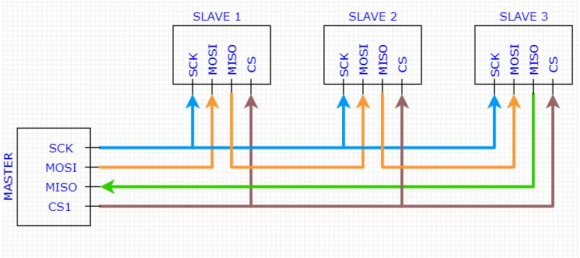


Sin embargo, en este tipo de conexión no se usan los términos de maestro o esclavo, por lo que el nombre de jerarquías cambia. El término utilizado por Microchip es anfitrión y clientes, siendo el anfitrión el microcontrolador maestro o cerebro de la cadena y los clientes son los microcontroladores esclavos [15]. En la ingeniería computacional llaman al microcontrolador maestro como Central Process Unit (CPU) y a los esclavos como dispositivos [16]. Otros textos mencionan al maestro como árbitro y a los esclavos como maestros [14].

A lo largo del proyecto, se implementarán los términos de Microchip para la conexión Daisy-Chain. Con lo anterior, el dispositivo anfitrión es el cerebro de la cadena, envía comandos o solicita información. Los demás dispositivos, clientes, tienen la capacidad de recibir y enviar información [15]. Cada cliente tiene una dirección o nombre, que es denominado por el anfitrión. Esta dirección permite conocer si el comando enviado por el anfitrión está dirigido a ellos. Por lo que, el cliente recibe la información y tiene la opción de ejecutar o enviar al siguiente cliente. Si la dirección es correcta, el cliente ejecuta el comando y manda un mensaje indicando que la orden se ejecutó, el cual se transmite a los clientes restantes para llegar al anfitrión quien recibe el mensaje y lo procesa [3]. Lo mismo pasa, si el anfitrión solicita información de algún cliente.

Este tipo de conexión puede implementarse para protocolo SPI o USART. Microchip recomienda utilizar este tipo de conexión con protocolos síncronos, debido a que el reloj permite que se reciba y transmita el mensaje al mismo tiempo en cada dispositivo, lo que hace que la transmisión sea de forma eficaz [15].

Figura 25: *Conexión Daisy-Chain en protocolo SPI propuesto por Microchip [15].*



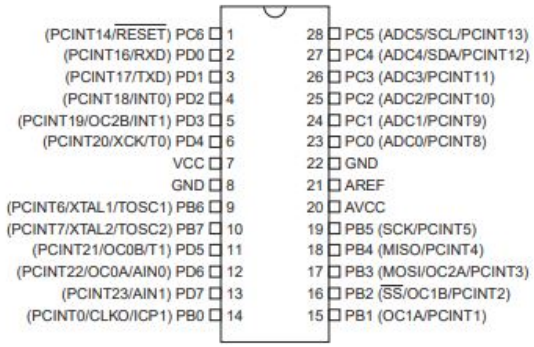
6.8. Microcontrolador ATMEGA28P para comunicación serial y control de motores

Se conoce como microcontrolador a los circuitos integrados programables que tienen CPU, memoria, interrupciones, temporizadores, entre otros.

Los microcontroladores AVR (Alf-Vegard-RISC) deben su nombre a Alf-Egil Bogen y Vegard Wollan, quienes propusieron su arquitectura en 1992. El término RISC (Reduced Instruction Set Computers) se refiere a la organización interna del CPU, que, en otras palabras, se centra en tareas simples para permitir que el hardware opere a frecuencias más altas. La tipología ofrece pocas instrucciones, las cuales son del mismo tamaño. La arquitectura fue comercializada por Atmel, quien fue comprada por Microchip [14]. Existe tres categorías de microcontroladores en la familia AVR, los cuales son:

- Gama Tiny, miembros con menos recursos.
- Gama Mega, versión intermedia.
- Gama XMega, miembros con más recursos.

Figura 26: *ATMEGA28P Pinout.*



Con lo mencionado anteriormente, el ATMEGA28P (figura 26) es un microcontrolador de la familia AVR clasificado en la gama de los Mega. Y cuenta con las siguientes características [17]:

- Memoria de código de 32 kbytes de memoria Flash.
- Memoria de datos de 2 kbytes de SRAM y 1 kbyte de EEPROM.
- 23 Terminales para entrada/salida.
- Frecuencia máxima de trabajo de 20 MHz.
- Voltaje de alimentación entre 1.8 a 5.5 Volts.
- Temporizadores: 2 de 8 bits y 1 de 16 bits.
- 6 Canales PWM.
- Interfaz SPI como Maestro o Esclavo.
- Transmisor/Receptor Universal Síncrono/Asíncrono (USART).
- Interfaz serial de dos hilos (TWI).
- Oscilador interno configurable.

6.9. Librería en lenguaje de programación

Son colecciones de código diseñadas previamente para que el programador desarrolle software eficientemente [18]. En otras palabras, es como un formulario prescrito. Permiten que el trabajo sea eficaz debido a que ahorran tiempo.

Hay dos tipos de librerías:

- Dinámica, son en tiempo real. No es necesario una copia dentro del archivo, basta con correrlo y el programa la buscará.
- Estática, se debe tener un copia para utilizarlo.

Las principales razones por las que un programador busca el uso de librerías de programación son:

- Estandarización de procesos para su implementación práctica.

- Eficiencia y optimización para evitar sobrecargas y facilitar el trabajo del desarrollar.
- Modularidad para la organización de código, lo que permite un mantenimiento y actualización eficientes.
- Reutilización del código para ahorrar tiempo y espacio.
- Compatibilidad entre programas.

Además de permitir que el trabajo sea eficaz, permiten que la programación se realice de forma fluida y ordenada. Haciendo que el código principal no se sature. Y por último, ayuda con el acceso de funcionalidades avanzadas, evitando la repetición de procedimientos complicados y permite la colaboración de proyectos [18].

CAPÍTULO 7

Definición de especificaciones de diseño del motor inteligente

Se inició con las especificaciones de diseño que tendrá cada cliente, debido a su papel principal en el sistema de motores inteligentes. Recordemos que el sistema de motores tendrá un tipo de comunicación daisy-chain, la cuál es una red en forma de anillo, en donde cada nodo o punto de conexión es el motor inteligente. Por lo que fue imperativo establecer los requisitos, características y funciones que tendrá el motor inteligente.

Con esto dicho, se prosiguió con la selección de microcontrolador, componentes y protocolo de comunicación, además de la definición del sistema de alimentación.

7.1. Selección de microcontrolador

Se comenzó con la selección del microcontrolador debido al impacto directo con el desarrollo y desempeño del proyecto, permitiendo futuras modificaciones sin necesidad de cambiar el hardware.

Para dicha selección, se tomó en cuenta la disponibilidad en el país y Universidad del Valle de Guatemala, por lo que se redujo el número de opciones a microcontroladores de 8 bits de procesamiento. Dichas opciones se dividen en dos familias de microcontroladores, las cuales son:

- PIC

- AVR

Ahora bien, la familia AVR se ha utilizado en los trabajos previos a este proyecto, por lo que por familiaridad y reducción de curva de aprendizaje y posibles errores se redujo la lista a solo microcontroladores ATMEGA328P que pertenecen a la familia AVR. Estos se encuentran en 2 diferentes placas las cuáles son:

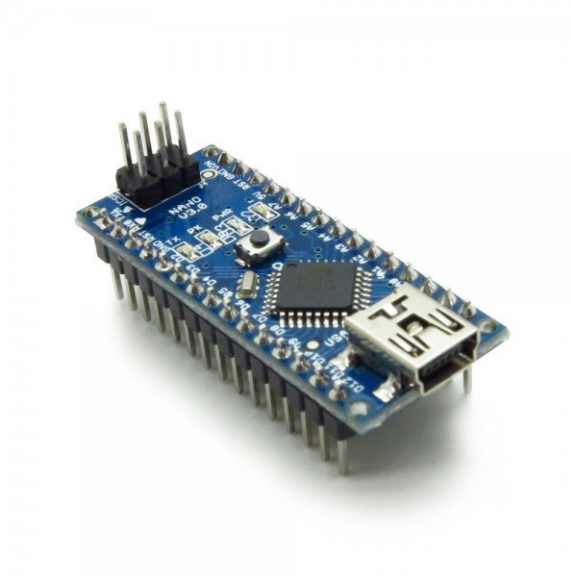
- Arduino Nano V3
- Arduino Uno

Sin embargo, el Arduino UNO se tuvo que descartar debido a que se busca microcontroladores con placa de tamaño compacto por la aplicación que tendrán.

Por lo que el microcontrolador seleccionado fue el ATMEGA328P con la placa del Arduino Nano V3. Entre sus especificaciones más importantes están:

- Memoria de código de 32 Kbytes de memoria Flash.
- Memoria de datos de 2 kbytes de SRAM.
- Memoria de datos de 1 kbyte de EEPROM.
- Frecuencia máxima de trabajo de 20 MHz.
- Interfaz SPI.
- USART/UART.
- Interfaz serial de dos hilos (TWI).
- Oscilador interno configurable.

Figura 27: *Arduino nano V3.*



7.2. Selección de componentes

Como se mencionó anteriormente, para que un motor sea considerado inteligente es necesario que tenga ciertos componentes que lo ayuden en su control y diagnóstico. A continuación se muestran los componentes seleccionados.

7.2.1. Motor stepper

En la selección del motor, se decidió utilizar un tipo de Brushless DC Motors, los cuáles son los motores stepper. Al ser un proyecto destinado para trabajos del campo de robótica, este tipo de motores son ideales para mecanismos que requieran precisión y permitan durabilidad y robustez para diferentes entornos. Además, ofrecen compatibilidad con drivers y microcontroladores.

El motor stepper seleccionado fue el NEMA17 bipolar debido a su disponibilidad dentro de la Universidad del Valle de Guatemala. Este presenta un alto par de torsión y precisión, por otro lado, tiene un tamaño compacto lo que facilita su utilización en espacios reducidos. Mientras que su facilidad de control varía sus aplicaciones.

Figura 28: *Motor stepper - Nema17.*

7.2.2. Driver

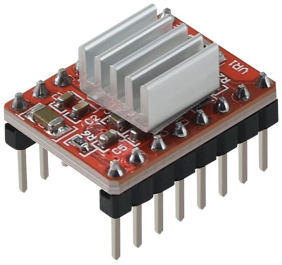
Para la selección del driver se consideró el motor elegido, con esto se busco un driver que lograra alcanzar los requisitos necesarios para controlar un motor stepper NEMA17. La característica más importante del motor es que necesita 2 amperios para trabajar, por lo que esto redujo las opciones de driver, ya que no todos proporcionan esta cantidad de amperaje.

Como se ha mencionado, se buscan componentes con tamaño compacto, por lo que el driver también tendrá que tener esta característica. Esto redujo la búsqueda a placas pequeñas con drivers, también llamadas módulos controlador de motor. Entre las opciones de módulos, se tuvo:

- Módulo con driver DRV8825.
- Módulo con driver A4988.

Sin embargo, por disponibilidad en el país y Universidad del Valle de Guatemala, se eligió el módulo con driver A4988. Cabe destacar que el módulo elegido es un sustituto del módulo con driver DRV8825, por lo que en futuras iteraciones se puede cambiar el módulo del driver de ser necesario.

Figura 29: *Módulo controlador de motor con driver A4988.*

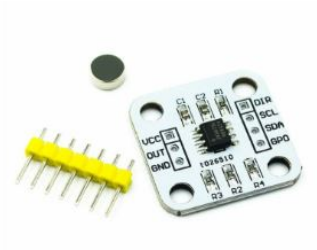


7.2.3. Encoder

Al tener diferentes tipos de encoder, se escogió un encoder rotativo magnético que fuera absoluto debido a que proporciona una posición única para cada ángulo de giro, incluso después de una pérdida de energía o reiniciarse. Esto permite una lectura y resolución más exacta. Asímismo, evita realizar una retroalimentación al sistema.

El encoder rotativo absoluto AS5600 al tener las características mencionadas previamente, por su tamaño compacto y disponibilidad en la Universidad del Valle de Guatemala, será utilizado en este proyecto.

Figura 30: *Encoder magnético absoluto - AS5600.*



7.2.4. Sensor hall effect

En la selección del sensor hall effect, que medirá la corriente, se buscó que fuera preciso al detectar cambios de corriente y que tuviera aislamiento eléctrico para proteger al circuito de cualquier pico de corriente.

Con lo mencionado, el sensor hall effect seleccionado fue el ACS712-5A. El cuál nos ofrece un tamaño compacto, simplicidad de uso y lectura analógica del sistema. Además, permitirá que en futuras iteraciones se pueda utilizar en dado caso se use corriente alterna.

Figura 31: *Sensor hall effect - ACS712-5A.*

7.2.5. Finales de carrera

Los finales de carrera, al ser de uso opcional para el usuario, no es necesario definirlos a un tipo. Sin embargo, para ejemplificar su uso, se usarán finales de carrera de tipo SPDT.

Figura 32: *Final de carrera - SPTD.*

7.3. Selección del protocolo de comunicación

Como se mencionó anteriormente, el sistema de motores inteligentes tendrá una tipología de conexión daisy-chain. Existe una jerarquía en este tipo de conexión, el que manda y solicita los comandos se denomina anfitrión y el que recibe y envía información se denomina cliente.

Ahora bien, dependiendo de la compatibilidad, la conexión daisy-chain se puede utilizar en protocolos de comunicación serial. Por lo que fue necesario realizar un trade study para determinar el protocolo de comunicación para el sistema de motores inteligentes.

Por lo tanto, en esta sección, se definirá el protocolo de comunicación para una conexión daisy-chain entre microcontroladores ATMEGA328P. El protocolo debe proporcionar una comunicación eficiente y rápida que permita conectar múltiples clientes. Con el microcontrolador seleccionado, se puede trabajar con los siguientes protocolos de comunicación:

- SPI (serial peripheral interface).
- Usart (universal synchronous asynchronous receiver transmitter).
- TWI (two wire interface).

Los criterios de evaluación para los protocolos de comunicación serán:

- Número de dispositivos

- Número de cableado.
- Tipo de transmisión.
- Velocidad de transmisión.
- Distancia de transmisión.
- Síncrono o asíncrono.
- Uso (daisy chain).

A cada criterio de evaluación se le definió un factor de peso en un rango entre 1 a 3, como se observa en el cuadro 1.

Definición de factores de peso	
Número de dispositivos	3
Número de cableado	2
Tipo de transmisión	2
Distancia de transmisión	3
Velocidad de transmisión	3
Síncrono o Asíncrono	1
Uso (Daisy-Chain)	3

Cuadro 1: *Factores de peso. Determinan el valor que tiene cada criterio de evaluación.*

Lo siguiente que se realizó fue una escala de normalización con cada criterio de evaluación, que permite establecer las características que debe cumplir el protocolo de comunicación (cuadro 2).

Definir la escala de normalización	
Número de dispositivos	Escala
200 a más	3
100-200	2
0-100	1
Número de cableado	Escala
2	3
3	2
4	1
Tipo de transmisión	Escala
Full-Duplex	3
Half-Duplex	2
Simplex	1
Distancia de transmisión	Escala
15	3
5-10	2
0-5	1
Velocidad de transmisión	Escala
1M a +	3
100K-1M	2
0-100K	1
Síncrono o Asíncrono	Escala
Asíncrono	2
Síncrono	1
Uso (Daisy-Chain)	Escala
Comun	2
Casi nulo	1

Cuadro 2: *Escala de normalización. Determina la normalización que tendrá cada una de las características de los criterios de evaluación.*

Se realizó una investigación para cada protocolo de comunicación y se distribuyó en el cuadro 3.

Por último, a la información obtenida se le aplicó una sumatoria del producto de la normalización con los factores de peso. El resultado es la ponderación que obtuvo cada protocolo de comunicación. Y como se observa (cuadro 4), el protocolo con mayor ponderación es el SPI, ya que cumple con la mayoría de los criterios de evaluación.

Basándose en el trade study realizado, el protocolo de comunicación para la conexión Daisy-Chain en el sistema de motores inteligentes que se realizará en este proyecto será el SPI (Serial Peripheral Interface).

Información		
Soluciones	Número de dispositivos	Número de cableado
USART	256	3
SPI	128 (1024)	3
TWI	127	4
Soluciones	Distancia de transmisión (m)	Velocidad de transmisión (bps)
USART	15	115200
SPI	10	4000000
TWI	4	100000
Soluciones	Síncrono o Asíncrono	Uso (Daisy-Chain)
USART	Síncrono	Casi Nulo
SPI	Síncrono	Común
TWI	Asíncrono	Casi Nulo

Cuadro 3: *Protocolos de comunicación. Información de cada protocolo de comunicación serial según los criterios de evaluación.*

Factores de peso aplicados	
Soluciones	Ponderación
USART	38
SPI	41
TWI	23

Cuadro 4: *Resultado del trade study. El protocolo de comunicación con mayor ponderación es el SPI.*

7.4. Definición del sistema de alimentación

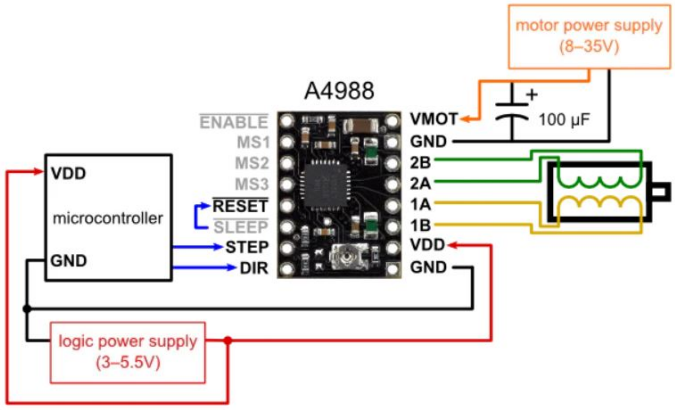
En esta sección se describe la selección el sistema de alimentación para el motor inteligente. Esta parte es crucial para la funcionalidad y seguridad del sistema.

Como primer paso, se realizó un sondeo del amperaje y voltaje que solicita cada uno de los componentes del motor inteligente. Los componentes seleccionados requieren un voltaje de operación de 5V con un amperaje menor a 1A (cuadro 5). Sin embargo, el módulo controlador del motor necesita dos voltajes de operación, uno para el driver y el otro para el motor (figura 33).

Alimentación requerida		
Componente	Voltaje (V)	Amperaje (A)
A4988 - Driver	5	0,05
A4988 - Motor	9	1,5
AS5600	5	0,05
ACS712-5A	5	0,05
Final de carrera	5	0,05

Cuadro 5: *Alimentación requerida por cada componente.*

Figura 33: *Diagrama de conexión del módulo A4988.*



El voltaje de operación para el motor es de 8 a 35V, sin embargo, se estableció que su voltaje requerido será entre 8 a 9V. Esto debido a la ley de Ohm, la cuál nos indica que la relación entre el voltaje y la corriente es proporcional. Por lo tanto, entre más voltaje requiera el circuito, la corriente aumentará. Teniendo este rango de voltaje establecido, se obtuvo un rango de amperaje entre 1 a 1.5A.

Sabiendo que el voltaje y amperaje de operación máxima del circuito es de 9V con 1.5A, se sugirió un sistema de alimentación con baterías. Esto permitiría que el sistema fuera independiente a la red eléctrica lo que genera autonomía, portabilidad y reducción de ruido. Además, se planeó el uso de un boost converter para reducir el voltaje de 9V a 5V en cada cliente. No obstante, al necesitar un voltaje y amperaje alto, se encontraron ciertas limitaciones con este tipo de sistema, los cuáles fueron:

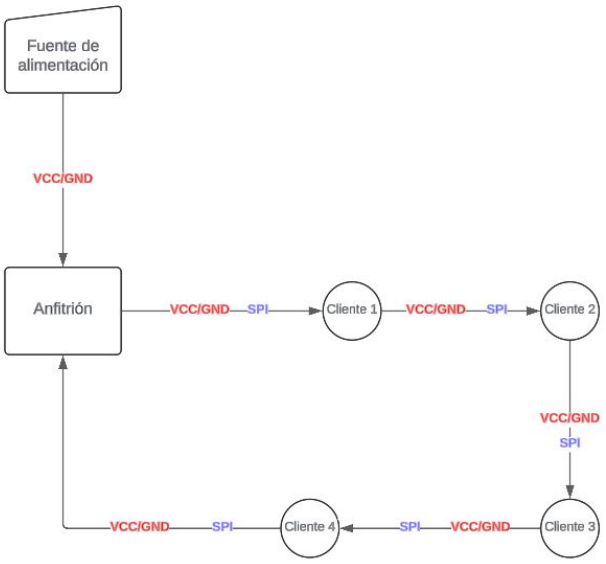
- Las baterías de ion, litio y alcalino que proporcionan dicha capacidad de alimentación tienen un gran tamaño, por lo que no cumple con el requisito de tamaño compacto que se busca en el diseño del motor.
- Se puede alcanzar el requerimiento de voltaje y amperaje con conexiones en serie o paralelo entre las baterías con tamaño de AA y AAA, sin embargo, no cumplen con el

requisito de tamaño compacto ya que se necesitaría utilizar más de 5 baterías.

- Al hacer el cálculo de vida del sistema de alimentación usando baterías, genera un resultado menor a 1 hora, lo cuál haría que el sistema de motores inteligentes fuera ineficiente.

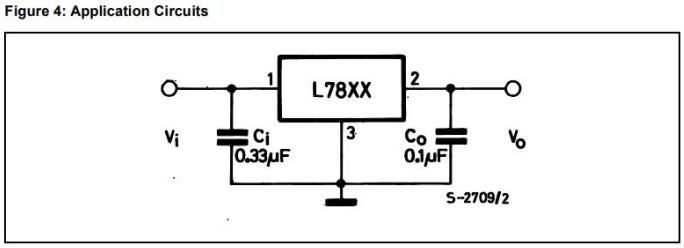
Conociendo lo anterior, se descartó el uso de baterías para el sistema de alimentación. Por lo que se optó por un sistema de alimentación dependiente a una fuente de poder. Siendo el anfitrión del sistema de motores inteligentes el que proporcionará el voltaje para cada cliente, como se observa en la figura 34.

Figura 34: *Diagrama de sistema de alimentación por fuente de poder.*



Mientras que para la reducción de voltaje de 9 a 5V, se utilizará un regulador de voltaje 7805.

Figura 35: *Diagrama del regulador de voltaje [19].*



CAPÍTULO 8

Diseño, fabricación y ensamblado del del motor inteligente

En el desarrollo de los motores inteligentes se requirió la integración de diversos componentes electrónicos y mecánicos que trabajen en conjunto para garantizar un funcionamiento eficiente y confiable. En este sentido, el diseño del circuito impreso (PCB) y de la carcasa que alberga los componentes se convierte en un aspecto fundamental para asegurar la protección, organización y funcionalidad del sistema.

En este capítulo se aborda el diseño del PCB, que incluye la distribución de los elementos electrónicos necesarios para el control y comunicación del motor inteligente, como el microcontrolador, sensores y driver. Además, se describe el proceso de diseño y fabricación de la carcasa, la cual tiene como objetivo proteger los componentes electrónicos y facilitar su instalación.

El objetivo principal de este capítulo es documentar los pasos seguidos en la creación del PCB y la carcasa, justificando las decisiones de diseño con base en las necesidades del sistema y las restricciones del proyecto. Asimismo, se presentan las pruebas realizadas para verificar la funcionalidad y durabilidad de ambos elementos en condiciones reales de operación.

8.1. Diseño del printed circuit board (PCB)

En esta sección se describe el proceso de diseño del PCB, considerando las especificaciones eléctricas establecidas previamente. Se detalla cómo se calcularon los tracks con base en

los requerimientos de corriente y las características de los componentes, siguiendo estándares del MakerLab para garantizar un diseño confiable y funcional. Asimismo, se aborda la disposición de los componentes, la asignación de capas y la implementación de técnicas para minimizar interferencias electromagnéticas y optimizar el rendimiento del sistema.

El objetivo es desarrollar un PCB que integre todos los elementos necesarios para el control del motor inteligente, manteniendo un equilibrio entre compactación, funcionalidad y facilidad de manufactura.

8.1.1. Identificación de corriente de cada componente

Como se mencionó con anterioridad, es necesario definir la corriente en la PCB debido a la importancia de garantizar que cada pista (track) pueda soportar las corrientes requeridas por los componentes electrónicos, sin comprometer la integridad del circuito.

Cada componente en el diseño, como microcontrolador, drivers, sensores y el motor, tiene demandas específicas de corriente que deben ser consideradas para evitar sobrecalentamientos, caídas de voltaje o fallos en el sistema.

Por lo tanto, es fundamental calcular con precisión la corriente que atraviesa cada elemento y dimensionar las pistas de acuerdo con estas necesidades, siguiendo los estándares de diseño establecidos. Este análisis no solo asegura el correcto funcionamiento del sistema, sino que también contribuye a la durabilidad y confiabilidad del PCB.

Componente	**Amperaje Máx (Datasheet)**		**Origen**
Encoder	100	mA	Entrada alimentación
Encoder	10	uA	Entrada de cada pin
Driver	20	uA	Entrada lógica
Driver	3	uA	Entrada alimentación
LM805	1,5	A	Salida
ACS712	11	mA	Salida
Arduino Nano	40	mA	Salida

Cuadro 6: *Amperaje máximo de cada componente según datasheet.*

En el cuadro anterior 6, se muestra el amperaje máximo y la fuente de alimentación de cada componente. Esta información es crucial, ya que nos da una idea clara del rango de corrientes que se manejarán en las PCBs, lo que facilita el diseño de las pistas y garantiza que puedan soportar estas corrientes sin problemas.

Con las corrientes máximas definidas, se identificó un rango de amperaje variado que no supera los 1.5 amperios. Esto requirió realizar cálculos específicos para dimensionar las pistas (tracks) de la placa de circuito impreso (PCB). Además, fue necesario considerar las

especificaciones de diseño del MakerLab, donde se fabricará la PCB, ya que utilizan un proceso de fresado con ciertos límites técnicos (figura 36). Por este motivo, se llevó a cabo un análisis nodal para estandarizar el diseño de las pistas y garantizar su compatibilidad con el proceso de fabricación.

Figura 36: *Especificaciones de tracks en el MakerLab.*

Track sizes (Tamaños de pista)

Usted debe ser cuidadoso al elegir el ancho de las pistas y tomar en cuenta la **corriente** que pasa a través de ella antes de determinar el tamaño de la pista.

En el documento del link superior, Generic Standard on Printed Board Design encontrará cómo realizar los cálculos correspondientes con el ancho de pista según la corriente que pasa a través de ella

ANSI IPC-2221A PCB Trace Width Calculator http://www.desmith.net/NMdS/Electronics/TraceWidth.html

Ancho mínimo de pista:	0.254 mm	10 mil
Ancho mínimo recomendado de pista:	0.508 mm	20 mil
Espacio mínimo entre pistas:	0.254 mm	10 mil
Ancho mínimo recomendado entre pistas:	0.508 mm	20 mil

En este análisis, se identificaron los nodos principales de voltaje de la PCB: el voltaje de alimentación de 8 voltios, el voltaje regulado de 5 voltios y la conexión a tierra. Los valores detallados de voltaje y corriente asociados a estos nodos se presentan en el cuadro a continuación.

NET	**Voltaje (V)**	**Amperaje (A)**
8V	8	2
5V	5	1,5
GND	0	0

Cuadro 7: *Voltaje y amperaje de cada nodo.*

En el diseño de la PCB se identificaron tres nodos principales: dos de alimentación y uno de tierra. Por razones de diseño, solo se calcularon las pistas de los nodos de alimentación, ya que estas son críticas para garantizar el suministro de corriente adecuado. En el caso del nodo de tierra, se optó por diseñarlo como un área en forma de pentágono, lo que simplifica el diseño de la placa y mejora la distribución de la conexión a tierra.

8.1.2. Cálculo de pistas (tracks)

Para el cálculo del ancho de las pistas, se utilizó como referencia la página de PCBWay, que ofrece una guía basada en el estándar IPC-2221. Esta guía proporciona una fórmula ampliamente utilizada en el diseño de PCBs para dimensionar correctamente las pistas según la corriente máxima esperada y las condiciones térmicas del sistema. La elección de esta fuente se debe a su claridad en la explicación de los parámetros involucrados y su adecuación a las necesidades específicas de este proyecto, permitiendo garantizar la integridad eléctrica y térmica de la PCB.

Para el diseño de las pistas de la PCB, se utilizó la fórmula recomendada en la página de PCBWay, para calcular tanto el área transversal como el ancho de las pistas. Este método asegura que las pistas puedan manejar la corriente máxima sin superar un incremento de temperatura predefinido.

La fórmula para el cálculo del área transversal (A) es:

$$A = \left(\frac{I}{K \times T_{\text{Rise}}^{b}} \right)^{\frac{1}{c}}$$

donde:

- I: corriente máxima en amperios.
- T_{Rise}: incremento de temperatura permitido en $°C$.
- K, b, c: constantes definidas por el estándar IPC-2221, que dependen de si las pistas están en capas internas o externas.

Con el área calculada, se determina el ancho de la pista (W) mediante la fórmula:

$$W = \frac{A}{t \times 1.378}$$

donde t representa el espesor del cobre en la PCB, siendo de 1 oz (o 35 μm) para este proyecto.

En este trabajo, se asumieron valores típicos para el diseño, como un incremento de temperatura permitido de 10 $°C$, una temperatura ambiente de 25 $°C$, y un espesor estándar de cobre de 1 oz. Las pistas de alimentación y tierra fueron las principales a dimensionar. Este enfoque garantiza un diseño térmico y eléctrico robusto, adaptado a las especificaciones del proyecto.

Teniendo lo anterior, el ancho de pista para cada nodo de alimentación quedo de la siguiente manera:

Width (1,5 A)	20,688	mil
Width (2 A)	30,764	mil

Cuadro 8: *Ancho de pista.*

8.1.3. Implementación de PCB

En un inicio, se planeaba realizar el diseño de una sola placa que integrara todos los componentes necesarios para el motor inteligente, incluyendo el encoder, el driver y el microcontrolador. Sin embargo, debido a las limitaciones de espacio impuestas por el tamaño del motor NEMA17 y con el objetivo de mantener un diseño compacto y funcional, se decidió dividir el sistema en tres placas independientes. Esta distribución permite optimizar el espacio disponible, mejorar la organización de los componentes y facilitar el ensamblaje y mantenimiento del sistema.

La primera placa se destinó exclusivamente al encoder, encargándose de capturar la posición y velocidad del motor. La segunda placa se diseñó para albergar el driver, el cual es responsable de gestionar la corriente requerida por el motor stepper. Por último, la tercera placa se dedicó al microcontrolador, encargado del procesamiento y control general del sistema.

Figura 37: *Placa de circuito impresa del Encoder.*

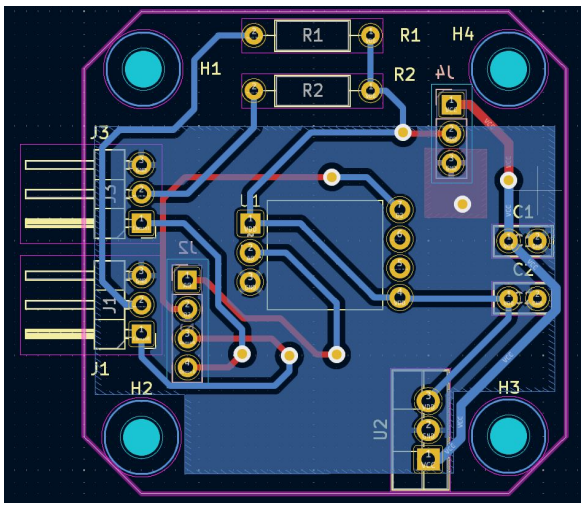


Figura 38: *Placa de circuito impresa del Driver.*

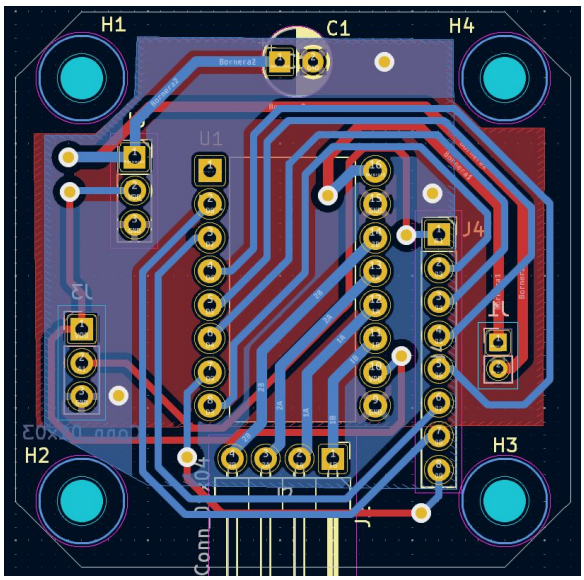
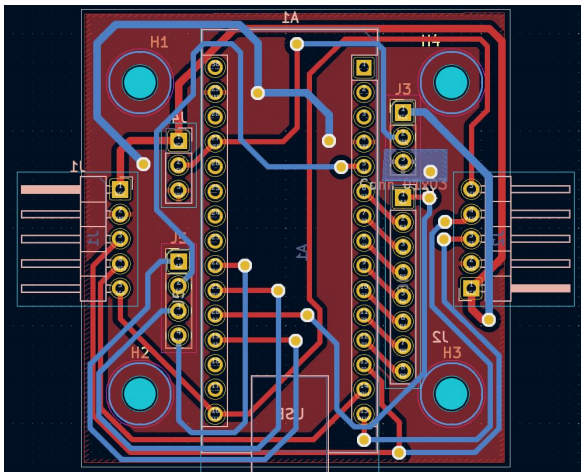


Figura 39: *Placa de circuito impresa del Arduino nano.*



Para garantizar el correcto funcionamiento del conjunto, cada placa cuenta con una conexión específica que permite el acoplamiento eléctrico y lógico entre ellas, asegurando la transmisión de datos y la alimentación de manera eficiente. Esta segmentación no solo permite un diseño más compacto sino que también facilita futuras modificaciones o actualizaciones en los módulos individuales, aumentando la flexibilidad y la escalabilidad del sistema.

8.2. Diseño mecánico de la carcasa del motor

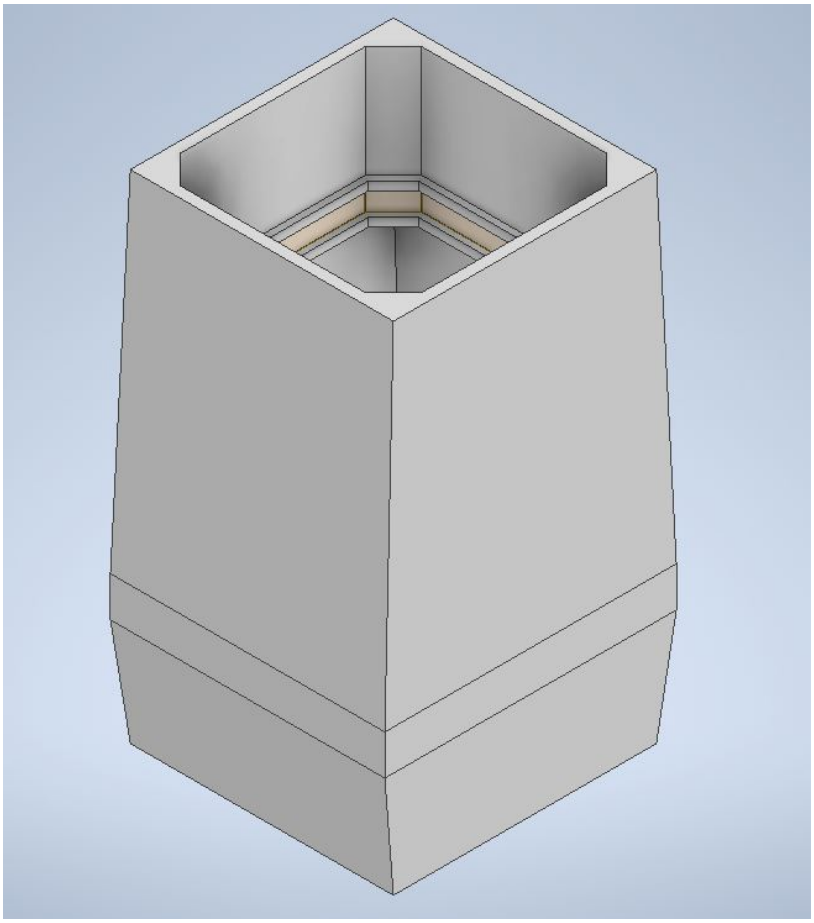
Para completar el diseño del motor inteligente, se desarrolló una carcasa mediante fabricación 3D. Este diseño tuvo que considerar las características específicas de las tres placas independientes: la placa del encoder, la del driver y la del Arduino nano. Dado que estas placas se conectan entre sí para garantizar un funcionamiento integrado, la carcasa debía proporcionar un espacio adecuado para cada una, asegurando tanto su protección como su correcto ensamblaje.

El diseño de la carcasa se centró en mantener la compactación general del sistema, respetando las dimensiones del motor Nema17. Se incluyeron ranuras y soportes internos para asegurar el ajuste preciso de cada placa, así como canales y aberturas que faciliten las conexiones eléctricas entre ellas. Además, se diseñaron puntos de fijación para garantizar la estabilidad de las placas durante el funcionamiento del motor, minimizando las vibraciones y permitiendo un mantenimiento sencillo. Este enfoque modular no solo optimiza el uso del espacio, sino que también simplifica el proceso de ensamblaje y facilita futuras modificaciones en el diseño del sistema.

8.2.1. Primera iteración

En la primera iteración del diseño de la carcasa (Figura 40), se buscó una estructura sencilla y compacta que pudiera albergar las tres placas del sistema. Sin embargo, este diseño fue descartado debido a varias limitaciones.

Figura 40: *Primera iteración de carcasa del motor inteligente.*



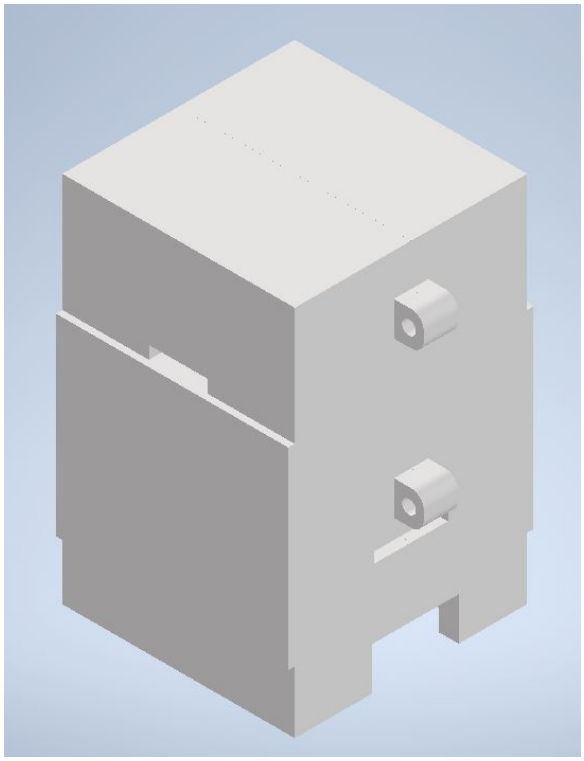
En primer lugar, las paredes de la carcasa resultaron ser demasiado delgadas, lo que comprometía su rigidez y durabilidad, especialmente bajo condiciones de uso regular. Además, no se incluyeron aperturas para los pines de conexión, lo que dificultaba tanto el ensamblaje como la interacción con los componentes internos. Estas deficiencias llevaron a realizar ajustes en el diseño para mejorar su funcionalidad y resistencia estructural en iteraciones posteriores.

8.2.2. Segunda iteración

La segunda iteración del diseño de la carcasa (Figura 41) fue la versión final, ya que cumplía con todas las especificaciones de diseño previamente establecidas. A diferencia de la primera iteración, esta carcasa fue optimizada para garantizar mayor rigidez, incorporando paredes de mayor grosor para asegurar su durabilidad. Además, se incluyeron aperturas para los pines de conexión, lo que facilitó el ensamblaje de las tres placas y su interconexión adecuada. Con estos ajustes, la carcasa no solo cumplió con los requisitos mecánicos, sino

que también permitió un diseño más compacto y funcional, cumpliendo con las necesidades del sistema.

Figura 41: *Segunda iteración de carcasa del motor inteligente.*



CAPÍTULO 9

Implementación de librerías de configuración y control

Para el desarrollo del sistema de motores inteligentes, es fundamental implementar dos librerías principales: una destinada a la configuración y otra al control. La librería de configuración estará disponible para los módulos clientes y será responsable de gestionar elementos clave del sistema, como el controlador del motor (driver), el encoder, el sensor de efecto Hall y los finales de carrera. Esta librería permitirá a los módulos clientes establecer los parámetros necesarios para el funcionamiento adecuado de cada componente.

Por otro lado, la librería de control será implementada en el dispositivo anfitrión y estará enfocada en manejar el protocolo de comunicación SPI. A través de esta librería, el anfitrión podrá enviar comandos específicos a los módulos clientes, permitiendo el control centralizado de las funciones asociadas a cada uno de ellos. Estas dos librerías trabajarán de manera conjunta para garantizar la sincronización y el desempeño óptimo del sistema de motores inteligentes.

9.1. Definición del lenguaje de programación para anfitrión y clientes

Para el desarrollo del sistema de motores inteligentes, se seleccionó el microcontrolador ATMEGA328P. Este microcontrolador ofrece compatibilidad con dos lenguajes de programación principales:

- Lenguaje ensamblador (ASM).
- Lenguaje C.

La diferencia entre ambos lenguajes radica en su nivel de abstracción. El lenguaje ensamblador es considerado de bajo nivel porque su estructura está directamente relacionada con las instrucciones del microcontrolador, permitiendo una interacción más directa con el código máquina. Esto produce un código compacto y eficiente en términos de memoria y velocidad. Sin embargo, la cercanía al hardware hace que la organización del código sea más compleja, lo que puede incrementar los errores si no se tiene una adecuada planificación del desarrollo del programa.

Por otro lado, el lenguaje C es considerado de alto nivel, ya que su estructura es más cercana al lenguaje humano. Esto facilita la escritura y comprensión del código gracias a las estructuras de control que ofrece. No obstante, su uso genera un mayor consumo de memoria SRAM en comparación con el lenguaje ensamblador. A pesar de esto, el lenguaje C permite desarrollar un código más legible, menos propenso a errores y con una ejecución rápida, lo cual resulta ideal para proyectos que requieran escalabilidad o iteraciones futuras.

La elección del lenguaje de programación se basó en dos criterios principales:

- Preferencia del programador: Elegir un lenguaje familiar para el programador aumenta la calidad del desarrollo, mejora la eficiencia y reduce la probabilidad de errores. Además, permite un flujo de trabajo más ágil y confiable.
- Facilidad de iteraciones futuras: Usar un lenguaje de alto nivel como C facilita el mantenimiento, la extensión y la comprensión del código por parte de otros desarrolladores o del mismo programador en el futuro.

Con base en estos aspectos, se seleccionó el lenguaje C como la herramienta principal para programar el sistema de motores inteligentes. Esta decisión permite un desarrollo más fluido, asegura una mayor claridad en el código y facilita la realización de modificaciones o mejoras en el sistema a largo plazo.

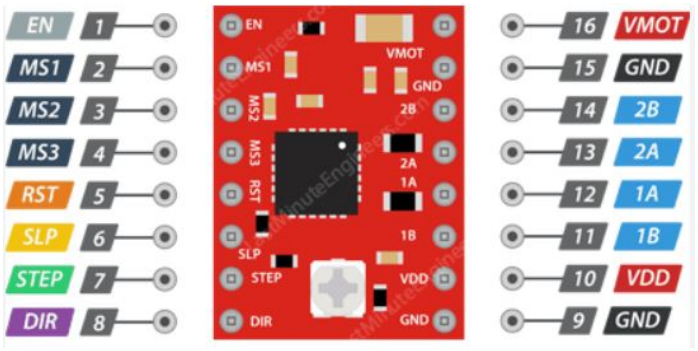
9.2. Librería de configuración - cliente

La librería de configuración incluye los elementos necesarios para la operación del sistema, los cuales son: el módulo controlador del motor, el encoder, el sensor de efecto Hall y los finales de carrera.

9.2.1. Librería del driver A4988

El desarrollo de esta librería comenzó con la configuración del módulo controlador del motor, ya que este es el encargado de regular la velocidad del motor, una función esencial en el sistema. Para ello, se realizó un diseño preliminar identificando los pines que utilizará este módulo, como se muestra en la figura 42. Este bosquejo sirvió como base para estructurar la conexión del controlador con el resto de los componentes del sistema.

Figura 42: *Pinout del módulo controlador del motor - driver A4988.*



El módulo controlador de motores A4988 está compuesto por 16 pines que se dividen en entradas (lado izquierdo) y salidas (lado derecho), como se muestra en la Figura 42. A continuación, se detallan los pines y su función:

- **Entradas (lado izquierdo):**
  - **EN (Enable):** activa o desactiva el driver.
  - **MS1, MS2, MS3:** configuran el nivel de microstepping del motor.
  - **RST (Reset):** restaura el estado inicial del driver. Debe estar en bajo para el funcionamiento normal.
  - **SLP (Sleep):** pone el driver en modo de reposo. Debe estar en bajo para activarlo.
  - **STEP:** recibe los pulsos que determinan los pasos del motor.
  - **DIR (Direction):** determina la dirección de rotación del motor.
- **Salidas (lado derecho):**
  - **VMOT:** alimentación para las bobinas del motor, para este proyecto es de 8V.
  - **GND:** tierra para el circuito de potencia.
  - **2A, 2B, 1A, 1B:** conexiones a una de las bobinas del motor.
  - **VDD:** alimentación lógica del driver, en este proyecto es de 5V.

- **GND:** tierra para el circuito lógico.

La mayoría de las entradas del módulo se programarán mediante pines del microcontrolador. Sin embargo, hay una excepción: la entrada correspondiente al pin STEP. Este pin es responsable de generar los pasos del motor, los cuales se producen mediante pulsos enviados al driver.

Es importante explicar el concepto de microstepping. Este método permite dividir los pasos del motor para lograr un movimiento más suave y preciso. Según el datasheet del driver A4988 [20], el paso completo del motor equivale a un avance de 1.8 grados, lo que significa que se requieren 200 pasos (*steps/rev*) para completar una vuelta completa del eje del motor.

El microstepping permite disminuir el tamaño de cada paso, aumentando la precisión del movimiento del motor. Sin embargo, esta mayor precisión tiene un costo: al reducir el tamaño de los pasos, disminuye la velocidad del motor y aumenta el consumo de torque.

En el cuadro 9 se muestra el tipo de microstepping que permite el driver, junto con los grados de avance y el número de pasos necesarios para completar una revolución.

	Full step	**Half step**	**1/4 step**	**1/8 step**	**1/16 step**
Grado	1,8	0,9	0,45	0,225	0,1125
Rev/step	200	400	800	1600	3200

Cuadro 9: *Se muestra el grado de avance y pasos que genera cada microstepping.*

Aunque el pin STEP es un pin digital, no se configuró como tal. Esto se debe a que este pin es responsable de indicar los pasos del motor mediante pulsos o transiciones entre niveles lógicos alto (1) y bajo (0). Si se requiere que el motor realice una cantidad específica de pasos (n), es necesario enviar n pulsos al pin STEP.

Por esta razón, la configuración del pin STEP se realizó mediante un generador de pulsos, los cuales son producidos por el microcontrolador. En este caso, el ATMEGA328P ofrece una ventaja significativa, ya que dispone de tres temporizadores (timers) [14]:

- Eltimer0 y timer2, ambos de 8 bits.
- El timer1, que es de 16 bits

Dado que los temporizadores de 8 bits tienen un rango más limitado antes de desbordarse (es decir, antes de reiniciarse al contar valores altos), se optó por usar el timer1. Este temporizador de 16 bits permite generar un rango más amplio de pulsos antes de alcanzar

el límite, lo que garantiza mayor precisión y eficiencia en la generación de pulsos para el motor.

Con el timer1 seleccionado como generador de pulsos para el pin STEP, fue necesario elegir el modo de operación adecuado para este temporizador. Las opciones disponibles son:

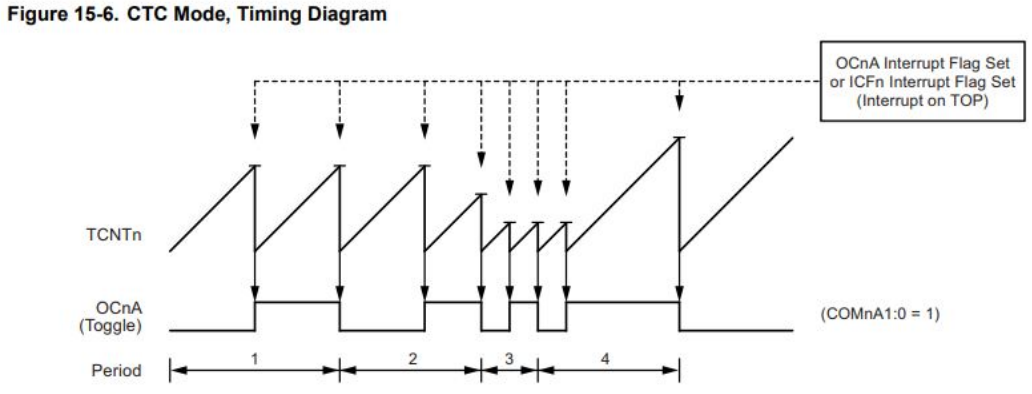
- Modo normal.
- Modo Clear time on compare match (CTC)
- Modo Fast PWM
- Modo Phase correct PWM

Estas configuraciones permiten adaptar el funcionamiento del temporizador a las necesidades del sistema, garantizando que los pulsos enviados al pin STEP sean precisos y confiables.

Inicialmente, se intentó utilizar el modo Fast PWM para generar los pulsos necesarios en el pin STEP. Sin embargo, este modo presentó un inconveniente: el cambio de velocidad del motor ocurría de forma demasiado rápida, dificultando un control preciso y estable sobre el motor.

Por esta razón, se optó por utilizar el modo Clear timer on compare match (CTC). Este modo permite generar una señal cuadrada que, aunque no es completamente simétrica, ofrece un cambio más controlado y estable entre pulsos. Esto hace que sea más adecuado para ajustar la velocidad del motor de forma precisa, garantizando un funcionamiento más confiable y eficiente.

Figura 43: *Diagrama de tiempo generado por el modo CTC [17].*



El funcionamiento del modo CTC (Clear Timer on Compare Match) se basa en la comparación de un contador con un valor preestablecido en el registro OCR1A. Cuando el contador

alcanza el valor de OCR1A, se produce un desbordamiento y el estado del pin configurado cambia, generando una señal cuadrada, como se muestra en la Figura 43. La frecuencia de esta señal depende del valor almacenado en OCR1A, que se calcula utilizando la siguiente ecuación:

$$f_{OCnA} = \frac{f_{clk_{I/O}}}{2 \cdot N \cdot (1 + OCRnA)} \tag{1}$$

Para determinar el valor de OCR1A necesario para obtener una frecuencia específica, se despeja la ecuación anterior:

$$OCRnA = \frac{f_{clk_{I/O}}}{2 \cdot N \cdot f_{OCnA}} - 1 \tag{2}$$

Donde:

- $f_{clk_{I/O}}$ es la frecuencia del oscilador del microcontrolador.
- f_{OCnA} es la frecuencia deseada.
- N es el valor del factor del preescalador.
- $OCRnA$ es el valor del registro de comparación.

En este proyecto, se utilizó un Arduino Nano V3 con un oscilador externo de 16 MHz para el microcontrolador ATMEGA328P, y el preescalador fue configurado con un valor de 8.

Dado que el módulo de control del motor requiere ajustar su velocidad en función de las revoluciones por minuto (RPM) especificadas por el usuario, fue necesario convertir las RPM a la frecuencia deseada (f_{OCnA}). Para esto, se utilizó la siguiente ecuación, que incluye el número de pasos por revolución según el microstepping configurado:

$$f_{OCnA} = \frac{rev}{min} \cdot \frac{steps}{rev} \cdot \frac{min}{sec} \tag{3}$$

Con la frecuencia deseada calculada, el valor del registro OCR1A se determina usando la fórmula anterior. Esto permite generar la señal necesaria para controlar los pasos del motor y ajustar su velocidad.

Implementación de la rampa de velocidad

Para garantizar un cambio gradual en la velocidad del motor, se implementó una rampa de aceleración. El usuario define el tiempo total de la rampa (t_{rampa}), y el cambio en el registro de comparación ($\Delta OCRnA$) se calcula con la siguiente ecuación:

$$\Delta OCRnA = \frac{OCRnA_{MAX} - OCRnA_{MIN}}{t_{rampa} \cdot f_{OCnA_{MAX}}} \tag{4}$$

El valor $\Delta OCRnA$ se suma o resta al registro OCR1A en cada desbordamiento del temporizador hasta alcanzar el valor deseado, logrando así una transición suave hacia la velocidad objetivo.

Configuración del microcontrolador

Finalmente, el microcontrolador fue configurado de la siguiente manera:

- **Generación de pasos:** Utilización del timer1 en modo CTC para generar la señal en el pin STEP, ajustando la velocidad del motor.
- **Control del microstepping y la dirección del motor:** Configuración de pines digitales para manejar los pines MS1, MS2, MS3 (para el microstepping) y DIR (para la dirección).

Esta configuración permite un control preciso de la velocidad y dirección del motor, asegurando el cumplimiento de los requerimientos del sistema.

Capacidad de la velocidad del motor según Microstepping

El control de las revoluciones por minuto (RPM) del motor se basa directamente en la configuración del nivel de microstepping y el valor del registro OCR1A, que determina la frecuencia de la señal enviada al driver A4988. El nivel de microstepping afecta el número de pasos necesarios para completar una revolución del motor mientras que el valor de OCR1A ajusta la frecuencia de los pulsos generados por el temporizador.

A medida que se aumenta el nivel de microstepping, el motor requiere más pasos para completar una vuelta, lo que mejora la precisión del movimiento, pero puede reducir la velocidad máxima alcanzable. Por otro lado, el valor de OCR1A define el intervalo entre pulsos, controlando directamente la frecuencia y, en consecuencia, las RPM del motor.

En las tablas que se presentan a continuación, se detallan las RPM del motor en función del nivel de microstepping configurado y el valor del registro OCR1A. Estas tablas permiten visualizar cómo varía la velocidad del motor con los diferentes parámetros del sistema, facilitando su configuración según las necesidades del usuario.

RPM	Full Step	Frecuencia	Periodo	OCR
10	200	33,33	0,03000	29999
20	200	66,67	0,01500	14999
30	200	100,00	0,01000	9999
40	200	133,33	0,00750	7499
50	200	166,67	0,00600	5999
60	200	200,00	0,00500	4999
70	200	233,33	0,00429	4285
80	200	266,67	0,00375	3749
90	200	300,00	0,00333	3332
100	200	333,33	0,00300	2999
200	200	666,67	0,00150	1499

Cuadro 10: *RPM según full-step.*

RPM	Half Step	Frecuencia	Periodo	OCR
10	400	66,67	0,01500	14999
20	400	133,33	0,00750	7499
30	400	200,00	0,00500	4999
40	400	266,67	0,00375	3749
50	400	333,33	0,00300	2999
60	400	400,00	0,00250	2499
70	400	466,67	0,00214	2142
80	400	533,33	0,00188	1874
90	400	600,00	0,00167	1666
100	400	666,67	0,00150	1499

Cuadro 11: *RPM según half-step.*

RPM	1/4	Frecuencia	Periodo	OCR
10	800	133,33	0,00750	7499
20	800	266,67	0,00375	3749
30	800	400,00	0,00250	2499
40	800	533,33	0,00188	1874
50	800	666,67	0,00150	1499
60	800	800,00	0,00125	1249
70	800	933,33	0,00107	1070

Cuadro 12: *RPM según quater-step.*

RPM	1/8	Frecuencia	Periodo	OCR
10	1600	266,67	0,00375	3749
20	1600	533,33	0,00188	1874
30	1600	800,00	0,00125	1249
40	1600	1066,67	0,00094	937

Cuadro 13: *RPM según eight-step.*

RPM	1/16	Frecuencia	Periodo	OCR
10	3200	533,33	0,00188	1874
20	3200	1066,67	0,00094	937

Cuadro 14: *RPM según sixteenth-step.*

Figura 44: *Diagrama de flujo del código del A4988.*

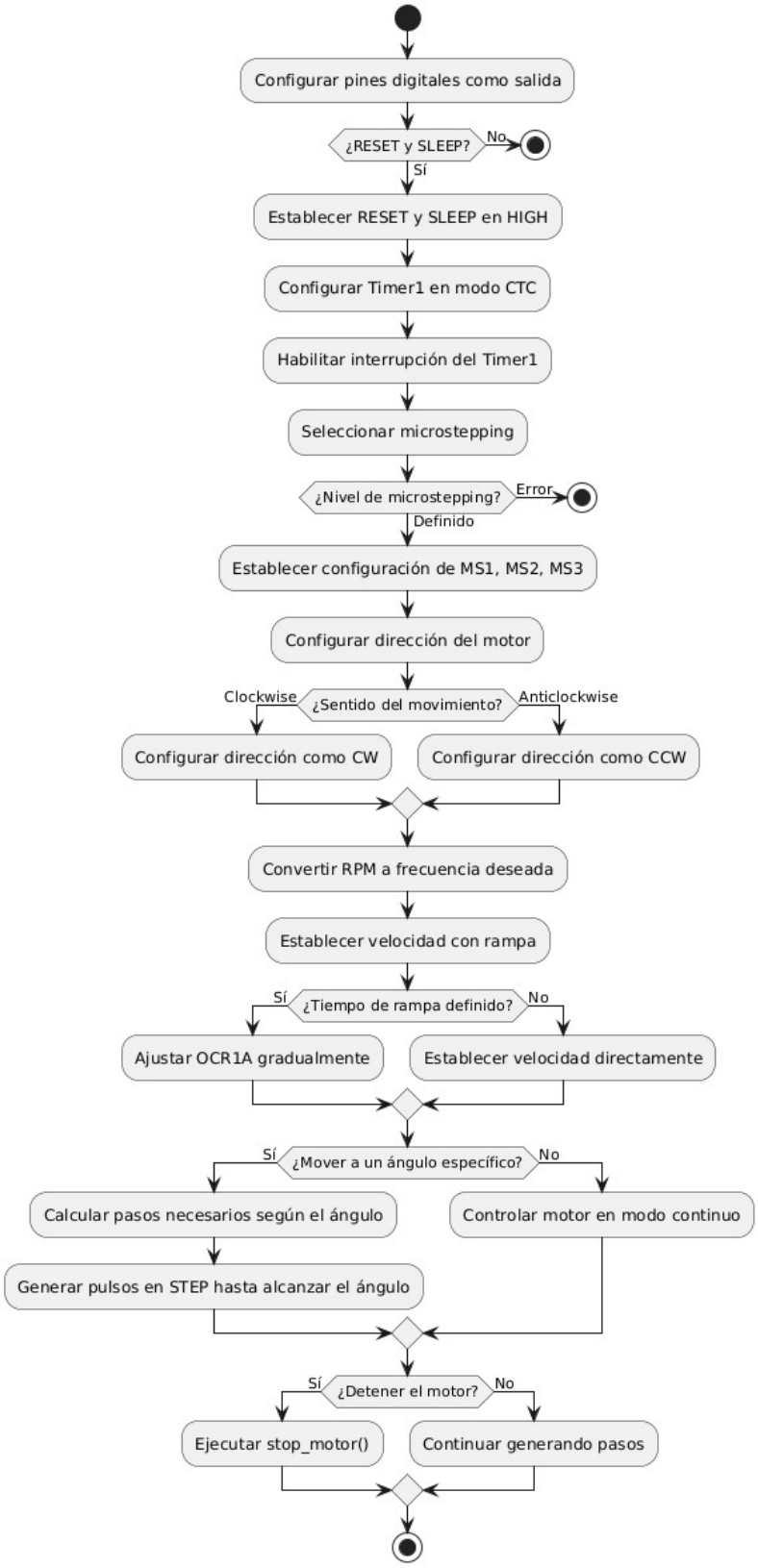


Diagrama de flujo del driver A4988

El procedimiento para configurar y operar el motor paso a paso utilizando el microcontrolador ATmega328P y el driver A4988 consta de varias etapas clave que garantizan el control eficiente del sistema. Este proceso abarca desde la configuración inicial del hardware hasta la implementación de funciones específicas para control de velocidad y posición (figura 44).

1. **Configuración inicial del sistema:** El primer paso consiste en configurar los pines del microcontrolador que se conectarán al driver A4988. Los pines necesarios incluyen:

   a) **STEP:** Para la generación de pulsos que controlan los pasos del motor.

   b) **MS1, MS2 y MS3:** Para configurar el nivel de microstepping.

   c) **RESET y SLEEP:** Para habilitar el funcionamiento del driver.

   d) **DIR:** Para definir la dirección de giro del motor.

   Los pines se configuran como salidas digitales, asegurando que puedan enviar las señales adecuadas al driver. Adicionalmente, los pines RESET y SLEEP se establecen inicialmente en estado alto (HIGH) para habilitar el driver.

2. **Configuración del Temporizador:** El control del motor se realiza mediante el Timer1, que se configura en el modo Clear Timer on Compare Match (CTC). Este modo permite generar una señal cuadrada en el pin STEP, necesaria para controlar los pasos del motor. Se habilita la interrupción asociada al temporizador para manejar eventos de comparación y cambios de velocidad.

3. **Selección de microstepping:** El nivel de microstepping se selecciona mediante la función Microstepping(). Esta función ajusta los pines MS1, MS2 y MS3 según el nivel deseado (por ejemplo, paso completo, medio paso, un cuarto de paso, etc.). El microstepping permite aumentar la precisión del movimiento, aunque disminuye el torque a medida que se incrementa el número de pasos por revolución.

4. **Configuración de Dirección:** El sentido de giro del motor se configura utilizando el pin DIR. Dependiendo de la dirección deseada, el sistema establece el pin en estado alto para movimiento antihorario (CCW) o en estado bajo para movimiento horario (CW).

5. **Conversión de Velocidad:** La velocidad del motor, indicada en revoluciones por minuto (RPM), se convierte en una frecuencia de pasos por segundo mediante la función RPM2Counter(). Esta función calcula el valor adecuado para el registro de comparación (OCR1A), que determina la frecuencia de la señal cuadrada generada por el temporizador.

6. **Establecimiento de Velocidad con Rampa:** La función *setmotor*() permite configurar la velocidad del motor con una rampa de aceleración o desaceleración. Esto se logra ajustando el valor de OCR1A gradualmente, evitando cambios abruptos que puedan afectar la estabilidad del sistema. El cálculo de la rampa considera el tiempo especificado por el usuario y la diferencia entre la velocidad inicial y la velocidad objetivo.

7. **Movimiento Basado en Ángulo:** Para mover el motor a un ángulo específico, se utiliza la función move2angle(). Esta función calcula los pasos necesarios según la diferencia entre el ángulo actual y el ángulo deseado, considerando el nivel de microstepping configurado. El motor se mueve en la dirección correspondiente hasta alcanzar el objetivo, deteniéndose automáticamente al completar los pasos calculados.

8. **Control en la Interrupción:** El control del motor durante rampas de velocidad o movimientos angulares se realiza dentro de la rutina de servicio de interrupción (ISR). En esta función:

   a) Se ajusta el valor de OCR1A para lograr cambios graduales de velocidad.

   b) Se controla el conteo de pasos durante movimientos angulares, deteniendo el motor una vez alcanzado el objetivo.

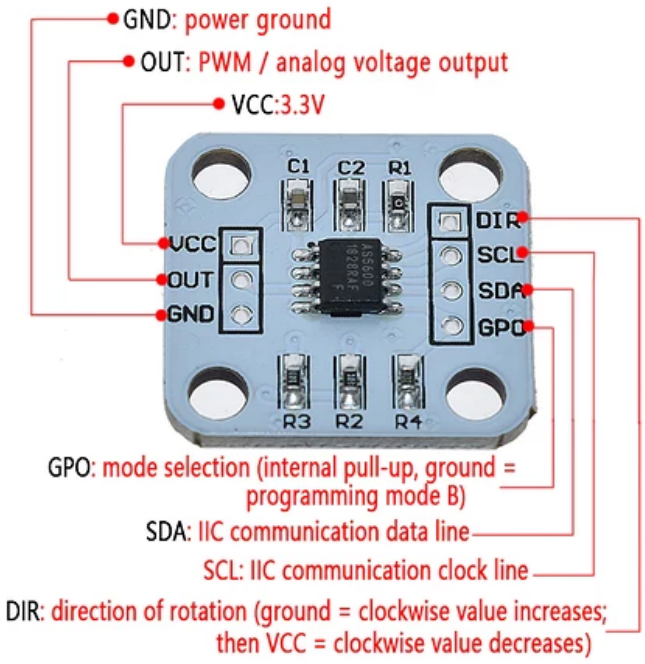
9. **Finalización del Movimiento:** El motor se detiene utilizando la función stop motor(), que deshabilita el temporizador y limpia los registros asociados. Esto garantiza que el motor deje de recibir pulsos y quede en estado de reposo.

9.2.2. Librería del encoder y sensor hall effect

Para los componentes analógicos, como el encoder y el sensor hall effect, se realizó una conversión de señal de analógica a digital para integrarlos con el microcontrolador.

En el caso específico del encoder, los pines se utilizaron de acuerdo al pinout mostrado en la figura 45.

Figura 45: *Pinout del Encoder AS5600.*



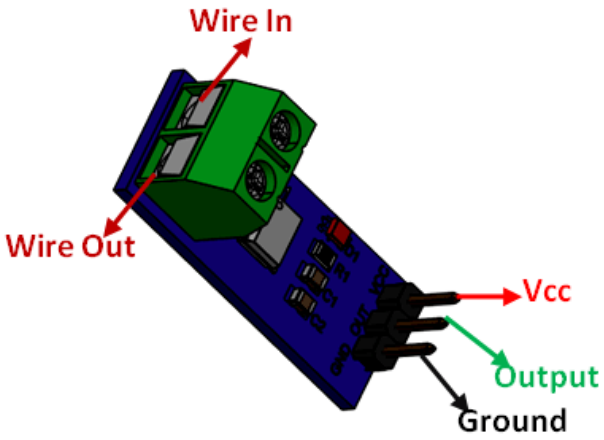
A continuación, se describen sus funcionalidades principales:

- **VCC:** entrada de alimentación para el encoder, con un voltaje de 5V
- **GND:** tierra.
- **OUT:** salida analógica que indica el ángulo del motor. Este pin es clave para el proyecto, ya que proporciona los datos necesarios para determinar la posición angular.
- **GPO:** pin digital que debe mantenerse en estado alto (a voltaje) para que el encoder funcione correctamente. Esto se debe al modo de uso seleccionado. En este caso, se configuró en el modo de programación B, que permite habilitar la salida analógica en el pin OUT [10].
- **DIR:** pin digital que determina la dirección de rotación. Si está conectado a tierra, la dirección es en sentido horario, y si está conectado a voltaje, la dirección es en sentido antihorario.

Aunque el encoder cuenta con pines para comunicación I2C (SDA y SCL), esta funcionalidad no fue necesaria en este proyecto. En su lugar, se utilizó exclusivamente la salida analógica del pin OUT, lo que simplificó la implementación y permitió el control adecuado del sistema sin la complejidad adicional de configurar la comunicación I2C.

Con el sensor de efecto Hall, los pines utilizados corresponden a los mostrados en la figura 46. Este sensor permite medir la corriente eléctrica al requerir únicamente una conexión de voltaje para realizar su lectura.

Figura 46: *Pinout del Sensor Hall Effect - ACS712.*



Para llevar a cabo la medición, se abrió el circuito y se conectaron los terminales de entrada y salida de corriente (Wire In y Wire Out) a través de las borneras del sensor. Esto permitió que el flujo de corriente pasara a través del sensor, facilitando así su lectura mediante el pin de salida analógica (Output), mientras que los pines de alimentación (VCC) y tierra (GND) se conectaron al microcontrolador.

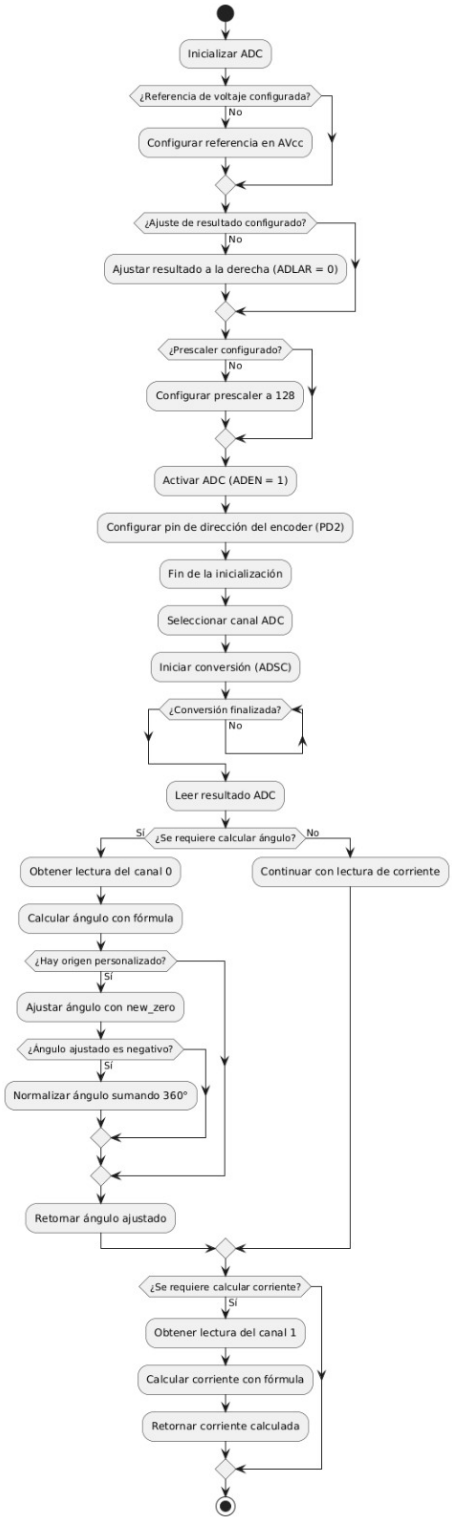
Ambos sensores generan señales analógicas, por lo que es necesario convertirlas a digitales para que puedan ser procesadas por el microcontrolador. El ATmega328P cuenta con un pin de referencia de voltaje para la conversión analógica a digital. Este pin se configuró para que utilizara el mismo voltaje de entrada que alimenta al microcontrolador. Con esta configuración, la conversión de la señal analógica a digital se realiza utilizando la siguiente fórmula:

$$Resolucion = \frac{V_{ref}}{1024} \tag{5}$$

- *Resolucion* es el valor digital obtenido de la conversión.
- V_{ref} es el voltaje de referencia, en este proyecto de 5V.
- 1024 el valor de conversión analógico a digital.

Diagrama de flujo del encoder AS5600 y sensor hall effect ACS712

Figura 47: *Diagrama de flujo del código del ADC.*



En esta sección se explica de manera detallada el procedimiento realizado para configurar y utilizar el módulo de conversión analógica-digital (ADC) integrado en el microcontrolador ATmega328P.

El procedimiento incluye la configuración inicial del ADC, que establece las bases para una medición precisa, y las funciones desarrolladas para leer los valores del ángulo y la corriente.

El diagrama de flujo (figura 47) ilustra el proceso completo del manejo del ADC, desde la inicialización hasta la obtención de las mediciones de los sensores. Este diagrama proporciona una visión clara y estructurada de los pasos realizados, facilitando la comprensión del procedimiento.

1. **Inicialización del ADC:** el primer paso consiste en configurar el módulo ADC para operar correctamente. Esto incluye:

   a) Configurar la referencia de voltaje como AVCC para que el rango de entrada del ADC sea la misma que la alimentación del microcontrolador.

   b) Ajustar el resultado de la conversión hacia la derecha, asegurando que los valores leídos sean interpretados correctamente.

   c) Seleccionar un prescaler de 128, el cual divide la frecuencia del reloj para garantizar que el ADC opere dentro de su rango especificado.

   d) Habilitar el módulo ADC mediante el registro correspondiente.

   e) Configurar el pin digital PD2 como salida y mantenerlo en alto para controlar la dirección del encoder.

2. **Proceso de Conversión y Lectura:** Con el ADC inicializado, se realizaron las lecturas de los sensores de la siguiente manera:

   a) **Selección del Canal ADC:** se selecciona el canal que corresponde al sensor que se desea leer. El canal 0 se utiliza para el encoder, mientras que el canal 1 se utiliza para el sensor de corriente.

   b) **Inicio de Conversión:** se activa el proceso de conversión analógica a digital configurando el bit correspondiente en el registro ADC.

   c) **Espera por Finalización:** el sistema espera a que la conversión se complete verificando el estado del bit de conversión.

   d) **Obtención del Resultado:** Una vez finalizada la conversión, se lee el valor resultante del registro ADC.

3. **Cálculo de Ángulo:** para obtener el ángulo proporcionado por el encoder:

a) Se lee el valor del canal 0 del ADC.

b) Se convierte el valor digital leído en un ángulo utilizando la fórmula:

$$\text{Ángulo} = \frac{\text{Lectura\_ADC} \times 360°}{\text{Resolución\_ADC}} \tag{6}$$

c) Si se definió un nuevo origen (new zero), el ángulo calculado se ajusta restando este valor. En caso de que el ángulo ajustado sea negativo, se normaliza sumando 360° para mantenerlo dentro del rango (0°, 360°).

4. **Cálculo de Corriente:** para medir la corriente del sistema:

a) Se lee el valor del canal 1 del ADC.

b) Se convierte el valor digital en una corriente utilizando la fórmula:

$$\text{Corriente} = \frac{\text{Lectura\_ADC} \times \text{Voltaje Máximo}}{\text{Resolución\_ADC}} \tag{7}$$

9.2.3. Librería del comunicación SPI configurada para daisy-chain

Como se mencionó en los capítulos anteriores, la comunicación seleccionada para la configuración en Daisy-Chain es el protocolo SPI (Serial Peripheral Interface). Este protocolo permite que varios dispositivos se comuniquen de manera eficiente y rápida a través de una línea de datos compartida.

En este sistema, se utiliza un microcontrolador maestro (master) y varios microcontroladores esclavos, en donde los microcontroladores de los motores están configurados como esclavos. Esto significa que los motores reciben las órdenes del maestro y responden a ellas de acuerdo con los comandos que se les envían.

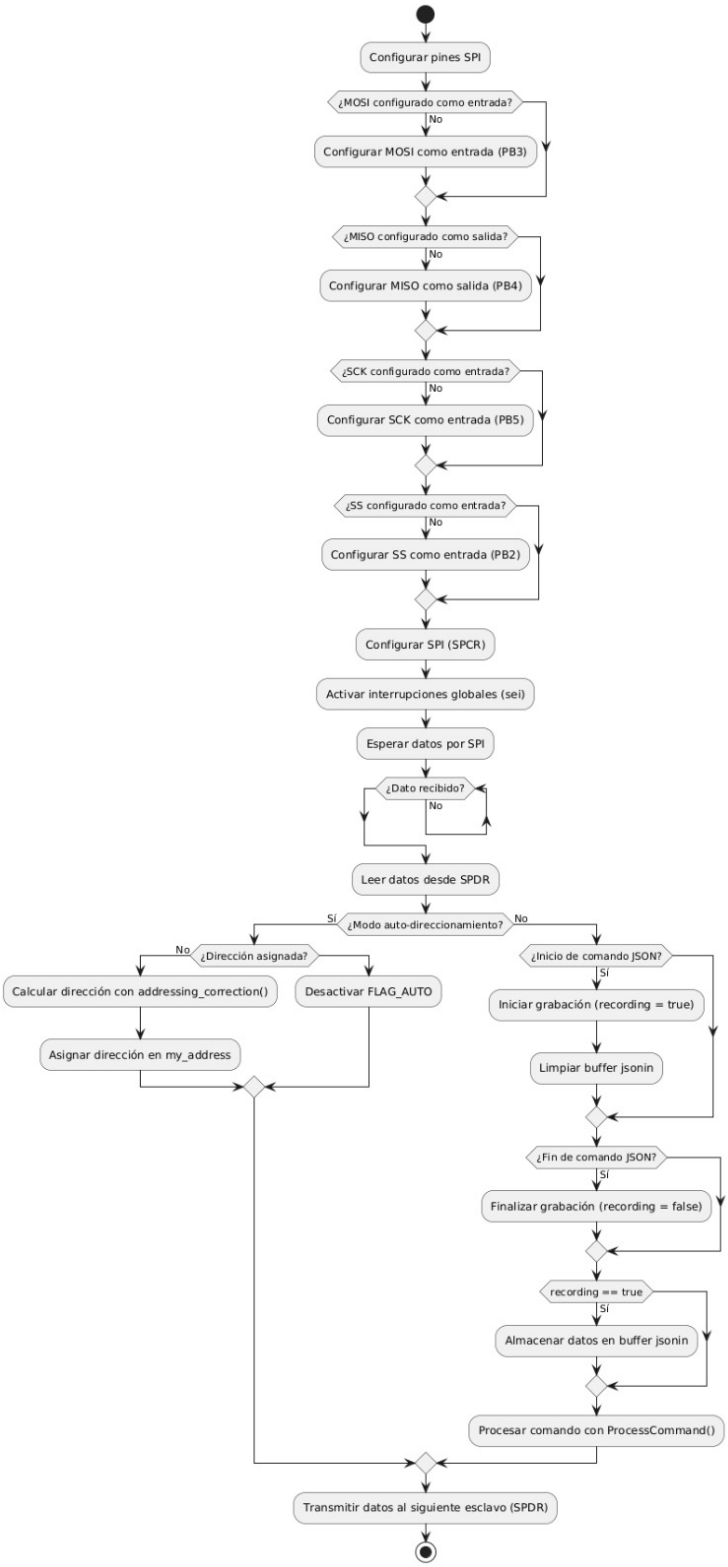
La comunicación entre el maestro y los esclavos no solo implica el envío de datos, sino también su correcta interpretación. Para esto, se decidió utilizar el formato JSON (JavaScript Object Notation) para el intercambio de mensajes. El protocolo JSON es un formato de texto ligero que estructura los datos en paquetes de caracteres fácilmente legibles, lo que permite transmitir información de manera eficiente y comprensible [21].

Cuando un motor esclavo recibe un mensaje a través del protocolo SPI, este mensaje en formato JSON es almacenado temporalmente en el microcontrolador del motor. Posteriormente, el microcontrolador decodifica el mensaje para verificar que el paquete recibido corresponde a una orden válida. Esta decodificación asegura que el motor reciba la acción correcta que debe ejecutar, como un cambio de dirección, ajuste de velocidad o cualquier otra función especificada en el mensaje.

Este proceso de recibir, almacenar y decodificar los mensajes en formato JSON permite que el sistema de motores funcione de manera coordinada, asegurando una comunicación precisa entre el maestro y los motores esclavos en el daisy-chain.

Diagrama de flujo del SPI del Periférico

Figura 48: *Diagrama de flujo del código del SPI del Periférico.*



La Figura 48 muestra los pasos clave, desde la inicialización del módulo SPI hasta el manejo de interrupciones para procesar y transmitir datos entre los microcontroladores involucrados. Cada paso asegura que los comandos enviados por el anfitrión sean interpretados y ejecutados correctamente por los microcontroladores esclavos. A continuación, se describe el procedimiento implementado:

1. **Configuración del SPI:**
   - *a*) **Definir los pines SPI:**
     1) MOSI (PB3) como entrada para datos enviados por el maestro.
     2) MISO (PB4) como salida para datos enviados al maestro o al siguiente esclavo.
     3) SCK (PB5) como entrada para la señal de reloj generada por el maestro.
     4) SS (PB2) como entrada para la selección del esclavo.
   - *b*) **Habilitar el SPI:** configurar el registro de control del SPI (SPCR) para habilitar el módulo y permitir interrupciones SPI.
   - *c*) **Activar interrupciones globales:** llamar a la función sei() para permitir que las interrupciones globales sean procesadas.
2. **Envío y eecepción de datos:**
   - *a*) **Envío de datos:** para enviar datos al maestro o al siguiente esclavo, se utiliza la función SPI send(uint8 t data). Los datos se cargan en el registro SPDR y la transmisión se completa cuando el bit SPIF en el registro SPSR se activa.
   - *b*) **Recepción de datos:** para recibir datos enviados por el maestro, se utiliza la función SPI receive(), que lee el valor del registro SPDR después de que el bit SPIF indique la finalización de la recepción.
3. **Autodireccionamiento en daisy-chain:**
   - *a*) **Asignación automática de dirección:** durante la inicialización, cada esclavo recibe una dirección única basada en los datos enviados por el maestro y procesados con la función addressingcorrection(uint8t autoaddressed).
   - *b*) **Confirmación de dirección:** una vez asignada la dirección, se almacena en la variable `my_address` y se desactiva la bandera `FLAG_AUTO` para completar el proceso de direccionamiento.
4. **Procesamiento de comandos:**
   - *a*) **Almacenar datos JSON:** los datos enviados por el maestro se interpretan como mensajes JSON. La rutina de interrupción SPI `(ISR(SPI_STC_vect))` identifica los límites del mensaje (y) y almacena los datos en un buffer jsonin.

b) **Decodificar y ejecutar comandos:** una vez recibido un mensaje completo, se procesa con la función `ProcessCommand(uint8_t command, uint16_t parameter1, uint16_t parameter2)`, que ejecuta acciones como las siguientes:

1) Configuración de microstepping.
2) Ajuste de dirección y velocidad del motor.
3) Lectura de parámetros como ángulo, velocidad o corriente.

5. **Retransmisión de datos:**

a) **Enviar al siguiente esclavo:** los datos procesados son retransmitidos al siguiente esclavo en la cadena mediante el registro SPDR.

9.3. Librería de control - anfitrión

En este capítulo se detalla el proceso de diseño de las librerías desarrolladas para el anfitrión del sistema. El anfitrión es el componente central encargado de controlar los motores conectados en cadena, enviando los comandos necesarios para que realicen las acciones correspondientes de manera coordinada y eficiente.

El microcontrolador seleccionado para el anfitrión es el ATmega328P, debido a su compatibilidad con los microcontroladores utilizados en los módulos controladores de los motores. Esta elección garantiza una comunicación fluida y un diseño uniforme en todo el sistema, facilitando el desarrollo y mantenimiento de las librerías.

A continuación, se detallan los pasos realizados para diseñar e implementar las librerías encargadas de gestionar la comunicación, configurar las operaciones de los motores y ejecutar los comandos enviados por el anfitrión. Estas librerías son esenciales para garantizar que el sistema funcione de forma precisa y confiable, cumpliendo con los requisitos de control y sincronización de los motores conectados en cadena.

El sistema se basa en dos librerías principales: SPI y UART, cada una con un propósito específico dentro del sistema.

9.3.1. Librería SPI

Esta librería está diseñada para implementar la comunicación en cadena (Ddaisy-chain) entre el anfitrión y los microcontroladores que controlan los motores. Utiliza el protocolo SPI, que permite al anfitrión enviar comandos a múltiples motores conectados en serie, asegurando una transferencia de datos rápida y sincronizada. En esta configuración, cada

microcontrolador actúa como un esclavo, recibiendo, interpretando y retransmitiendo los datos al siguiente nodo hasta que llegan al motor correspondiente.

La librería codifica los mensajes en formato JSON, lo que facilita la organización y transmisión de la información. Los datos enviados incluyen una cadena de caracteres con la dirección del motor, la función a ejecutar y los parámetros necesarios (si aplican). Esto permite que cada motor interprete de manera clara las acciones que debe realizar.

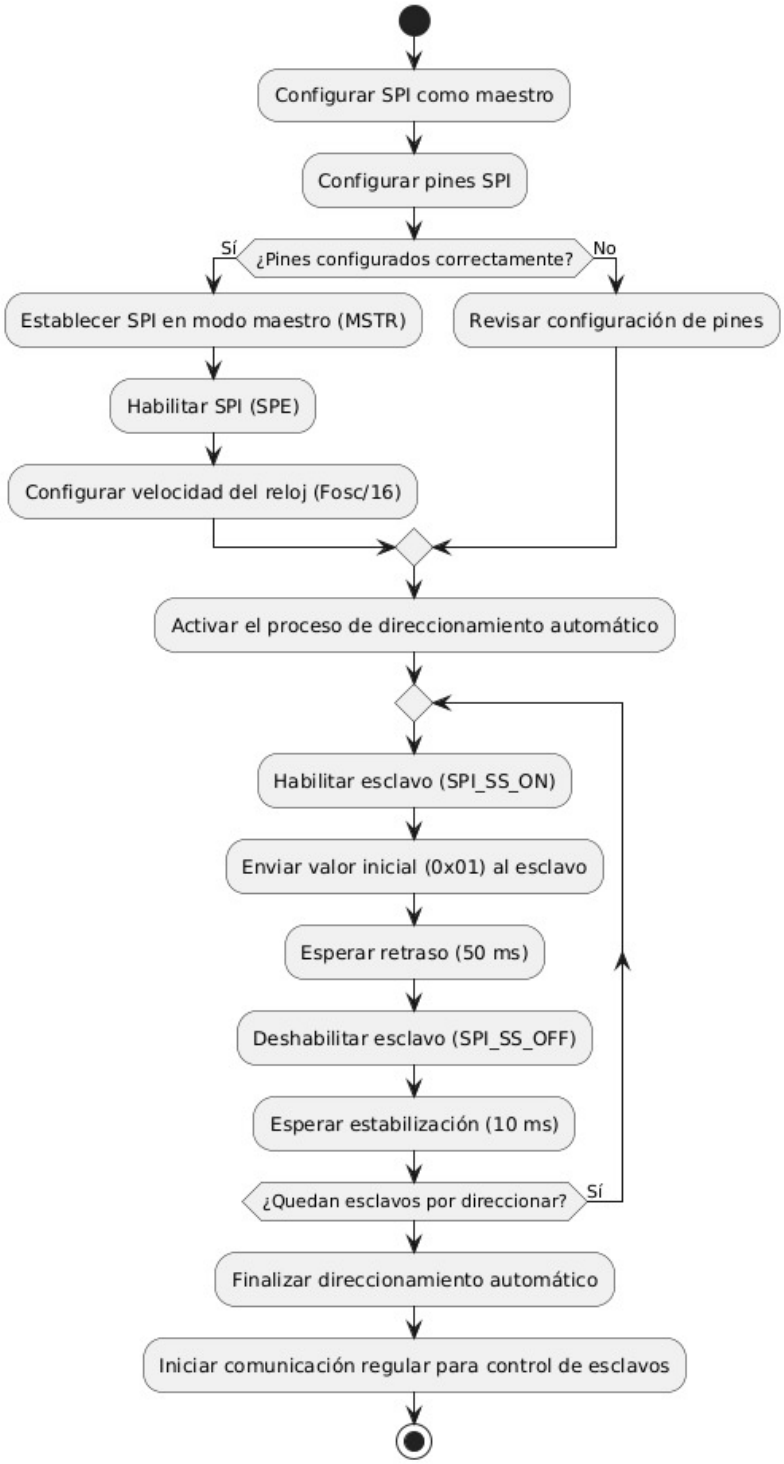
Al inicializar el sistema, se realiza un proceso de direccionamiento automático. El anfitrión envía un valor inicial de 1, que cada motor en la cadena incrementa en 1 antes de transmitirlo al siguiente. De esta forma, los motores reciben una numeración única que va de 1 a n, donde n es la cantidad total de motores conectados. Una vez completado el direccionamiento, el sistema comienza a enviar las acciones y comandos para controlar los motores.

Las funciones disponibles para el usuario incluyen operaciones como configurar el microstepping, ajustar la dirección y velocidad, mover el motor a un ángulo específico, y leer parámetros como el ángulo, la velocidad o la corriente. Estas funciones están detalladas en el siguiente cuadro 15, proporcionando una referencia clara de las capacidades del sistema.

Nombre	**Función**	**Parámetros**
Tipo de microstepping	Microstepping	Microstepping
Dirección horaria	Clockwise	—
Dirección antihorario	Anticlockwise	—
Paro del motor	stop_motor	—
Movimiento del motor	set_motor	RPM Ramping time
Movimiento hacia ángulo	move2angle	RPM Angle
Lectura de velocidad	read_speed	—
Establecer origen	set_origin	—
Lectura de ángulo	read_angle	—
Lectura de corriente	read_current	—

Cuadro 15: *Funciones para el control de los motores.*

Figura 49: *Diagrama de flujo del código del SPI para el controlador.*



El siguiente procedimiento describe la configuración del protocolo SPI en el microcontrolador como maestro y el proceso de direccionamiento automático de los dispositivos esclavos en una comunicación Daisy-Chain. Este mecanismo garantiza que cada dispositivo en la

cadena reciba una dirección única para permitir un control ordenado y eficiente.

1. **Configuración inicial del SPI:**
   - *a*) **Definir los Pines SPI:**
     1) Se configuró los pines del microcontrolador según las funciones requeridas por el protocolo SPI:
        - *a′* MOSI (PB3), SCK (PB5) y SS (PB2) como salidas.
        - *b′* MISO (PB4) como entrada.
     2) Establezca el pin SS (PB2) en alto para indicar el estado inactivo del esclavo.
   - *b*) **Configurar el Registro SPI (SPCR):**
     1) Active el módulo SPI habilitando el bit SPE.
     2) Configure el microcontrolador como maestro activando el bit MSTR.
     3) Ajuste la frecuencia del reloj SPI a $Fosc/16$ configurando los bits SPR1, SPR0 y SPI2X.
2. **Proceso de direccionamiento automático:**
   - *a*) **Inicio del proceso:** Inicie el direccionamiento llamando a la función `auto_addressing()`. Esta función se encargará de asignar direcciones únicas a cada dispositivo esclavo en la cadena.
   - *b*) **Envió de dirección inicial:** En un bucle que recorre todos los esclavos conectados (13 iteraciones en este caso).
     1) Habilite el esclavo llamando a la función `SPI_SS_ON()`, colocando el pin SS en bajo.
     2) Envíe un valor inicial (0x01) al primer esclavo utilizando la función `SPI_send()`.
     3) Espere un breve intervalo de tiempo (`_delay_ms(50)`) para permitir que el esclavo procese el dato.
     4) Deshabilite el esclavo llamando a `SPI_SS_OFF()`, colocando el pin SS en alto.
     5) Agregue un pequeño retraso adicional (`_delay_ms(10)`) para estabilizar la comunicación antes de pasar al siguiente esclavo.
   - *c*) **Asignación de direcciones:** Cada esclavo en la cadena incrementa el valor recibido en 1 antes de transmitirlo al siguiente dispositivo. Esto garantiza que cada esclavo reciba una dirección única que va de 1 a n, donde n es el número total de dispositivos conectados.
   - *d*) **Finalización del direccionamiento:** Una vez que todos los esclavos han sido direccionados, el proceso concluye y el sistema queda listo para iniciar la comunicación regular.

3. **Envío y recepción de datos:**

   a) **Envío de datos:** Para transmitir datos, utilice la función `SPI_send(uint8_t data)`:

      1) Cargue los datos en el registro SPDR.
      2) Espere a que la transmisión se complete verificando el bit SPIF.

   b) **Recepción de Datos:** Para recibir datos desde los esclavos, utilice la función `SPI_receive()`:

      1) Escriba un valor predeterminado (0xFF) en SPDR para iniciar la comunicación.
      2) Lea el valor recibido del registro SPDR una vez completada la recepción.

9.3.2. Librería UART

Esta librería está diseñada para que el usuario pueda enviar comandos directamente al sistema. Esta interfaz se utiliza para la comunicación entre el usuario y el anfitrión, permitiendo que se especifiquen parámetros como velocidad, dirección del motor, y configuraciones de microstepping. Los datos enviados por UART son procesados por el anfitrión, que luego utiliza la librería SPI para transmitirlos a los motores en la configuración daisy-chain.

La siguiente figura 50 muestra la estructura de los comandos que deben enviarse a través de UART. Los datos incluidos en cada comando son los siguientes: la dirección del motor, la función a ejecutar y, si es necesario, el parámetro 1 y el parámetro 2.

Figura 50: *Estructura de comando para envió en UART.*

(Address, function, parameter 1, parameter 2)

En el cuadro 16 se presentan las funciones disponibles para el sistema, junto con su nombre, los parámetros requeridos y el número asignado a cada función. Este número se utiliza para identificar la función al enviarla a través de UART, permitiendo que el sistema interprete y ejecute las acciones correspondientes de manera precisa. Además, los parámetros especificados facilitan la configuración detallada de las operaciones a realizar.

Diagrama de flujo de UART

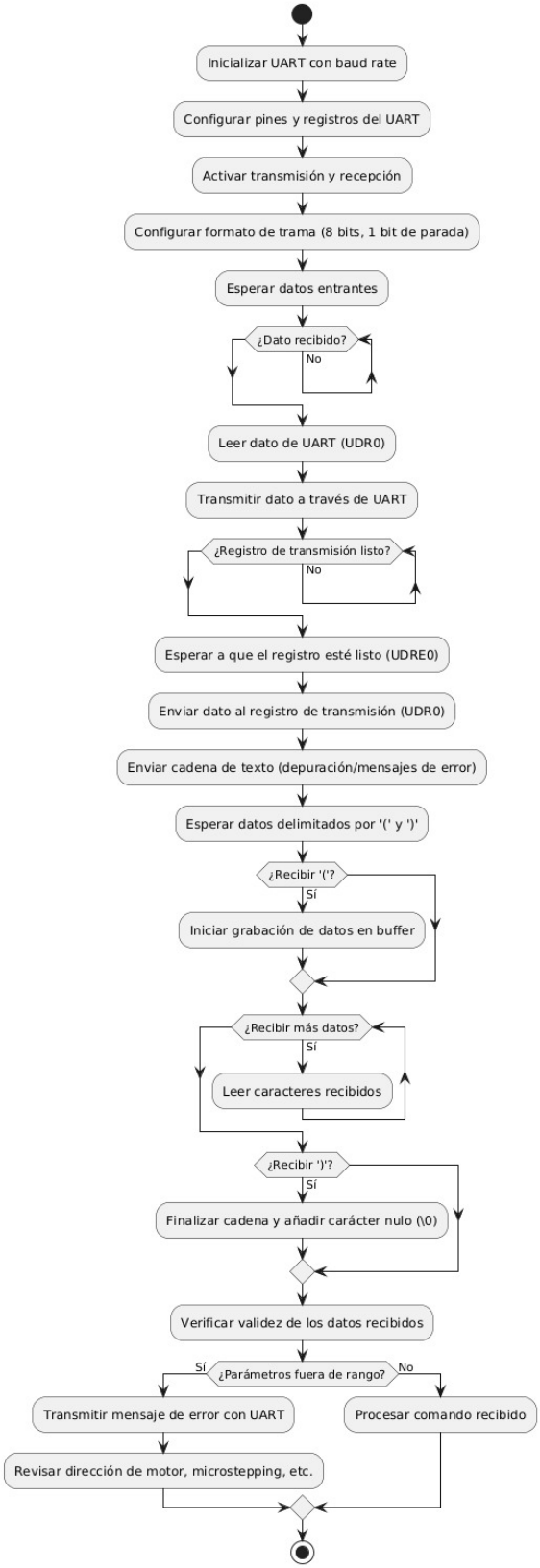
En este procedimiento (figura 51), se detalla la configuración y el uso del módulo UART del microcontrolador para la transmisión y recepción de datos, así como el proceso de vali-

Numero de Función	**Nombre**	**Función**	Parámetros
1	Tipo de microstepping	Microstepping	Microstepping
2	Dirección horaria	Clockwise	—
3	Dirección antihorario	Anticlockwise	—
4	Establecer origen	set_origin	—
5	Movimiento del motor	set_motor	RPM Ramping time
6	Paro del motor	stop_motor	—
7	Movimiento hacia ángulo	move2angle	RPM Angle
8	Lectura de ángulo	read_angle	—
9	Lectura de velocidad	read_speed	—
10	Lectura de corriente	read_current	—

Cuadro 16: *Funciones para el usuario con su respectivo número.*

dación de los comandos recibidos.

Figura 51: *Diagrama de flujo del código del UART para el controlador.*



A continuación se explica el procedimiento:

1. **Inicialización del UART:** se configura el módulo UART con un baud rate determinado, ajustando los registros y habilitando tanto la transmisión como la recepción de datos. El formato de la trama se establece con 8 bits de datos y 1 bit de parada, lo que permite una comunicación estándar y confiable.

2. **Recepción de datos:** el microcontrolador espera la llegada de datos a través de UART. Cuando los datos son recibidos, el sistema los lee desde el registro UDR0 y los almacena para su procesamiento.

3. **Transmisión de datos:** cuando se recibe un dato o comando, el microcontrolador puede enviarlo a través de UART a otro dispositivo o al usuario. La transmisión se realiza asegurando que el registro de transmisión esté listo, verificando el bit UDRE0 antes de enviar el dato.

4. **Recepción de cadenas de texto:** El sistema es capaz de recibir cadenas de texto delimitadas por los caracteres (y). Durante la recepción, los caracteres se almacenan en un buffer hasta que se detecta el carácter), lo que indica el final de la cadena.

5. **Validación de datos recibidos:** Antes de procesar cualquier comando, el sistema valida que los parámetros recibidos estén dentro de los rangos aceptables. Si algún dato está fuera de rango, se envía un mensaje de error al usuario para indicar el problema. Los parámetros validados incluyen direcciones de motor, configuraciones de microstepping, y valores de velocidad o ángulo.

CAPÍTULO 10

Conclusiones

- Se definieron las características del motor inteligente, incluyendo la selección de componentes clave como el microcontrolador, driver, encoder y un protocolo de comunicación confiable.
- Se diseñó un motor inteligente con una PCB optimizada, integrando subsistemas mecánicos y electrónicos en una carcasa compacta, asegurando robustez y eficiencia en el manejo del motor.
- Se desarrollaron librerías para configurar y controlar el motor inteligente, permitiendo ajustar microstepping, velocidad, dirección y lectura parámetros clave como ángulo y corriente.
- Se comprobó que el protocolo SPI es una solución rápida y eficiente para la comunicación en configuración daisy-chain. Su alta velocidad de transferencia y sincronización permite que los datos se transmitan de forma confiable entre el anfitrión y múltiples dispositivos esclavos conectados en cadena.
- El correcto funcionamiento del sistema requiere que todos los componentes compartan una misma referencia de tierra, asegurando la precisión de las señales y la estabilidad en la comunicación y operación.

CAPÍTULO 11

Recomendaciones

- Fortalecer el protocolo SPI para garantizar que los datos se transfieran de manera más segura y confiable.
- Implementar la capacidad del controlador (master) para recibir datos de los dispositivos esclavos (slaves).
- Utilizar cables más robustos para mejorar la calidad de la transferencia de datos en la comunicación SPI.
- Evaluar si el ancho de los tracks en la PCB afecta la transferencia de datos para realizar ajustes si es necesario.
- Diseñar y probar diferentes rampas de aceleración (exponencial, cuadrática, etc.) para complementar la rampa lineal implementada.
- Optimizar la estabilidad del microstepping y ampliar el rango de velocidad del motor para mejorar su desempeño.
- Rediseñar el motor para hacerlo más modular y compacto, integrando todas las funciones en una sola placa en lugar de tres, facilitando su uso en aplicaciones de robótica.
- Probar la placa diseñada para el controlador, verificando su funcionamiento y asegurando que cumpla con los requerimientos del sistema.

CAPÍTULO 12

Referencias

[1] T. Dale. «Smart motors drive motion system design.» (2000), dirección: https://www.machinedesign.com/motors-drives/article/21827189/smart-motors-drive-motion-system-design.

[2] T. Schiazza y F. Mielli, «New trends for intelligent motor control centers,» en *2012 IEEE-IAS/PCA 54th Cement Industry Technical Conference*, 2012, págs. 1-8. DOI: 10.1109/CITCON.2012.6215681.

[3] P. A. Y. Pan, «Diseño de un sistema de motores inteligentes en conjunto con un sistema de control del brazo robótico asistencial para cirugías estereotácticas.,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[4] J. P. I. V. Saravia, «Diseño de un sistema de motores inteligentes para aplicaciones de robótica de alta precisión y desempeño.,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2023.

[5] C.-H. Liu, E. Wade y H. Asada, «Reduced-cable smart motors using DC power line communication,» en *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, IEEE, vol. 4, 2001, págs. 3831-3838.

[6] S. Westcott y J. R. Westcott, *Basic Electronics*, 2nd. David Pallai., 2018.

[7] P. Millett. «Brushless Vs Brushed DC Motors: When and Why to Choose One Over the Other.» (2022), dirección: https://www.monolithicpower.com/en/brushless-vs-brushed-dc-motors#:~:text=There%20are%20two%20types%20of,commutation%20function%20with%20electronic%20control..

[8] C. Fiore. «Stepper Motors Basics: Types, Uses, and Working Principles.» (2023), dirección: https://www.monolithicpower.com/stepper-motors-basics-types-uses#:~:text=The%20basic%20working%20principle%20of,rotor%20aligns%20with%20this%20field..

[9] Edwards. «Types of Encoders: Rotary, Linear, Position, and Optical Encoder Types.» (2021), dirección: https://www.thomasnet.com/articles/automation-electronics/types-of-encoders-a-thomasnet-buying-guide/.

[10] *12-bit Programmable Contactless Potentiometer*, AS5600, Rev. 1, ams, ago. de 2014.

[11] G. Torres. «Principios básicos del sensor de efecto Hall.» (2022), dirección: https://urany.net/blog/adquiere-precisi%C3%B3n-con-efecto-hall.

[12] RayMingPCB. «The Serial Communications Protocols Comparison.» (2024), dirección: https://medium.com/@raymingpcb/the-serial-communications-protocols-comparison-b96c9acf14f7.

[13] D. S. Dawoud y P. Dawoud, «1 Serial Communication,» en *Serial Communication Protocols and Standards RS232/485, UART/USART, SPI, USB, INSTEON, Wi-Fi and WiMAX*. 2020, págs. 1-46.

[14] F. S. Espinosa. 2020.

[15] Microchip. «35.4.5 Daisy-Chain Configuration.» (2024), dirección: https://onlinedocs.microchip.com/oxy/GUID-61202E6F-BC03-4D81-AAB9-269E87F9B49C-en-US-22/GUID-A303F2B6-8BBA-4702-B548-AFF957E0B12D.html.

[16] H. Ashtari. «What Is Network Topology? Definition, Types With Diagrams, and Selection Best Practices for 2022.» (2022), dirección: https://www.spiceworks.com/tech/networking/articles/what-is-network-topology/.

[17] *ATmega328P*, 7810D, Rev. 3, ATMEL, jun. de 2016.

[18] B. V. Ridge. «El funcionamiento de las librerías en programación: una guía detallada.» (2023), dirección: https://www.mediummultimedia.com/apps/como-funcionan-las-librerias-en-programacion/.

[19] *L7800 Series*, Rev. 12, STMicroelectronics, nov. de 2004.

[20] *DMOS Microstepping Driver with Translator and Overcurrent Protection*, 4988-DS, Rev. 1, Allegro MicroSystems, Inc., ago. de 2010.

[21] J. Papoušek, «Traffic Flow Processing in JSON Format.,» Bachelor's Thesis, Masaryk University, 2020.

CAPÍTULO 13

Anexos

A continuación se proporciona el hipervínculo para acceder a github:

https://github.com/Daniela-Godinez/Motores_Inteligentes