UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIA Y TECNOLOGÍA

ESCUELA DE INGENIERÍA ELECTRÓNICA

IMPLEMENTACIÓN DEL PROTOCOLO CAN-BUS PARA EL CONTROL DE MOVIMIENTO DE UN BRAZO ROBÓTICO DE 3 GRADOS DE LIBERTAD

Trabajo de graduación previo a la obtención del título de:

INGENIERA ELECTRÓNICA

Autora:

EVELYN DANIELA OCHOA PINTO

Director:

ING. HUGO TORRES, PhD

CUENCA, ECUADOR

2025

DEDICATORIA

Este trabajo se lo dedico a mis papás, Patricio Ochoa y Yolanda Pinto, y a mi hermana, Adriana Ochoa, porque siempre creyeron en mí, incluso cuando yo no lo hacía. Gracias por estar ahí en cada momento difícil, por levantarme cuando sentía que no podía más y por no dejar que me rinda. Les amo con todo mi corazón.

AGRADECIMIENTOS

Agradezco profundamente a todas las personas que hicieron posible este trabajo. A mis padres, por su apoyo incondicional, tanto emocional como económico, durante toda la carrera; a mi hermana, por sus palabras de aliento en los momentos más difíciles; a mi primo, Adrián Pacheco, por sus conocimientos técnicos en la parte mecánica del brazo robótico; a mis amigos Mateo Estévez, Francisco Gálvez, William Goercke y Berenice Guerrero, por sus consejos y motivación constante a lo largo de estos años. Por último, agradezco de manera especial al Ing. Hugo Torres, PhD, por haberme proporcionado la estructura física del brazo robótico, lo cual fue clave para que pudiera enfocarme en el desarrollo del sistema de control y las interfaces de visualización.

IMPLEMENTACIÓN DEL PROTOCOLO CAN-BUS PARA EL CONTROL DE MOVIMIENTO DE UN BRAZO ROBÓTICO DE 3 GRADOS DE LIBERTAD

Este trabajo tuvo como objetivo principal la implementación del protocolo de comunicación CAN-bus para el control del movimiento de un brazo robótico de tres grados de libertad, donde luego se agregó un grado adicional de libertad para poder expandir el área de trabajo, mejorando de esta manera la capacidad de manipulación del mismo. Este protocolo, ampliamente utilizado en sistemas embebidos, se seleccionó debido a sus características de alta fiabilidad, baja latencia, resistencia a interferencias electromagnéticas y eficiencia en la transmisión de datos. Estas propiedades lo convirtieron en una opción idónea para aplicaciones en robótica, donde se requiere un control distribuido y funcionamiento en tiempo real. En primer lugar, se realizó un análisis exhaustivo del funcionamiento del protocolo CAN-bus, detallando su arquitectura y ventajas en comparación con otras tecnologías de comunicación. Posteriormente, se diseñó y desarrolló un prototipo de brazo robótico, el cual incorporó un microcontrolador por cada motor, lo que permitió la transmisión y recepción de datos a través de la red CAN. Se realizaron pruebas de funcionamiento del sistema, verificando la comunicación efectiva entre nodos dentro del bus CAN, lo que se tradujo en la correcta ejecución de trayectorias y la sincronización de los movimientos del manipulador. Los resultados obtenidos evidenciaron que el uso de CAN-bus fue una alternativa efectiva y adecuada para el control distribuido en sistemas robóticos, garantizando un desempeño adecuado y estable. En conclusión, la implementación del protocolo CAN-bus en este contexto resultó una solución viable y eficiente para el control de movimiento en sistemas robóticos.

Palabras clave: 3 GDL, Brazo Robótico, CAN-Bus, Comunicación Industrial, SavvyCAN

IMPLEMENTATION OF CAN-BUS PROTOCOL FOR MOTION CONTROL OF A 3-DEGREE-OF-FREEDOM ROBOT ARM

The main objective of this work was to implement the CAN-bus communication protocol in order to control the movement of a robotic arm with three degrees of freedom, to which an additional degree of freedom was later added in order to expand the workspace, thereby improving its manipulation capabilities. This protocol, widely used in embedded systems, was selected due to its high reliability, low latency, resistance to electromagnetic interference, and efficiency in data transmission. These characteristics made it an ideal option for robotic applications, where distributed control and real-time response are required. First, an in-depth analysis of the CAN-bus protocol was carried out, detailing its architecture and advantages compared to other communication technologies. Subsequently, a robotic arm prototype was designed and developed, incorporating microcontrollers that enabled data transmission and reception over the CAN network. System functionality tests were conducted, verifying if the communication between nodes was successful, which resulted in the correct execution of trajectories and synchronization of the manipulator's movements. The results demonstrated that the use of CAN-bus was an effective and suitable alternative for distributed control in robotic systems, ensuring a suitable and stable performance. In conclusion, the implementation of the CAN-Bus protocol in this context proved to be a viable and efficient solution for motion control in a robotic system.

Keywords: 3 DOF, Robotic Arm, CAN-Bus, Industrial Communication, SavvyCAN

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Implementación del protocolo CAN-Bus para el Control de Movimiento de un Brazo Robótico de 3 Grados de Libertad

Evelyn Daniela Ochoa Pinto
Escuela de Ingeniería Electrónica
Universidad del Azuay
Cuenca, Ecuador
evelyn.ochoa@es.uazuay.edu.ec

***Resumen*—Este trabajo tuvo como objetivo principal la implementación del protocolo de comunicación CAN-bus para el control del movimiento de un brazo robótico de tres grados de libertad, donde luego se agregó un grado adicional de libertad para poder expandir el área de trabajo, mejorando de esta manera la capacidad de manipulación del mismo. Este protocolo, ampliamente utilizado en sistemas embebidos, se seleccionó debido a sus características de alta fiabilidad, baja latencia, resistencia a interferencias electromagnéticas y eficiencia en la transmisión de datos. Estas propiedades lo convirtieron en una opción idónea para aplicaciones en robótica, donde se requiere un control distribuido y funcionamiento en tiempo real. En primer lugar, se realizó un análisis exhaustivo del funcionamiento del protocolo CAN-bus, detallando su arquitectura y ventajas en comparación con otras tecnologías de comunicación. Posteriormente, se diseñó y desarrolló un prototipo de brazo robótico, el cual incorporó un microcontrolador por cada motor, lo que permitió la transmisión y recepción de datos a través de la red CAN. Se realizaron pruebas de funcionamiento del sistema, verificando la comunicación efectiva entre nodos dentro del bus CAN, lo que se tradujo en la correcta ejecución de trayectorias y la sincronización de los movimientos del manipulador. Los resultados obtenidos evidenciaron que el uso de CAN-bus fue una alternativa efectiva y adecuada para el control distribuido en sistemas robóticos, garantizando un desempeño adecuado y estable. En conclusión, la implementación del protocolo CAN-bus en este contexto resultó una solución viable y eficiente para el control de movimiento en sistemas robóticos.**

***Palabras clave*—3 GDL, Brazo Robótico, CAN-Bus, Comunicación Industrial, SavvyCAN**

I. Introducción

A lo largo del desarrollo de diversas arquitecturas de control para robots industriales, se ha recurrido con frecuencia a enfoques centralizados utilizando protocolos como Modbus y Profibus [1]. Estas arquitecturas centralizadas presentan una estructura rígida y dependen en gran medida del cableado, lo que genera problemas de fiabilidad y aumenta los costos. Los protocolos utilizados en sistemas centralizados son susceptibles a interferencias electromagnéticas, lo que puede provocar fallos en la comunicación y afectar la precisión en el control de los robots [2]. Además, la cantidad de cableado necesaria para conectar punto a punto cada componente periférico, como sensores y actuadores, incrementa tanto los costos de implementación como la complejidad del mantenimiento del sistema [2]. En aplicaciones que requieren modularidad y portabilidad, este tipo de arquitectura presenta grandes limitaciones, ya que el cableado voluminoso restringe la flexibilidad del diseño robótico y dificulta su transporte o reconfiguración en diferentes entornos [2]. Además, diversas investigaciones han documentado que los sistemas basados en estos protocolos tienden a sufrir problemas de fiabilidad en entornos industriales donde predominan las interferencias electromagnéticas [1], [2].

Algunas de las aplicaciones que requieren de la característica de modularidad son robots en el área médica [1], [3]–[7] tomando como ejemplo un robot exoesqueleto, mencionado en [5] , en donde se tiene el diseño de un robot de rehabilitación de extremidades inferiores que reconoce la intención de movimiento humano a través de sensores de fuerza, desplazamiento y velocidad. En estos proyectos se empleó la comunicación CAN-bus en donde se conecta las unidades de hardware en paralelo consiguiendo reducir así el cableado y mejorar la flexibilidad del diseño; además de conseguir comunicación en tiempo real como beneficio de la arquitectura CAN, proporcionando estabilidad y robustez.

En aplicaciones en el sector productivo e industrial también se ha optado por CAN-bus [8]–[13] , tanto por su modularidad como por su baja inversión económica, fiabilidad y capacidad de trabajar en tiempo real. En [13] se menciona el diseño y prueba de un robot de desmalezado para aplicarlo en el sector agrícola. Su objetivo principal es subsanar dificultades que se presentan en las máquinas de desmalezado tradicionales como es la alta sensibilidad a las interferencias y la falta de automatización en estos sistemas, lo que resulta en menor productividad y mayores costos operativos. Mediante el uso de múltiples microcontroladores STM32 interconectados mediante el CAN-bus se consigue una automatización estable, económica y fiable, teniendo en cuenta que se realizaron pruebas de comunicación que resultaron en un desempeño efectivo de las acciones necesarias para la tarea de desmalezado.

Por otro lado, los protocolos de Ethernet industrial, como Profinet y EtherCAT, son ampliamente utilizados en la automatización industrial debido a su capacidad para manejar grandes volúmenes de datos y su interoperabilidad con diversos dispositivos [14], [15]. Estos protocolos permiten comunicación en tiempo real eficiente, pero para lograr baja latencia, requieren mecanismos adicionales que incrementan la complejidad [16]. Esto plantea desafíos en la escalabilidad y la gestión a largo plazo [1], además de que se necesita una inversión considerable [4] .

Debido a todas las dificultades de los sistemas industriales tradicionales y aplicaciones de CAN-bus presentadas se puede afirmar que CAN-bus es un protocolo adecuado para aplicaciones en el campo de la robótica, resaltando sus principales beneficios: la reducción de complejidad de implementación y mantenimiento; ahorro económico; modularidad; fiabilidad; comunicación en tiempo real y alta inmunidad ante interferencias electromagnéticas [4] , [17]. La manera en la que este protocolo consigue todos estos beneficios se debe, primeramente, a su carácter determinista, su sistema de arbitraje y de detección de errores, ya que mediante estos se puede garantizar la fiabilidad y comunicación en tiempo real de todos los nodos de un sistema [18], [19]. Además, la utilización de señal diferencial y trenzado de los cables es lo que proporciona una alta inmunidad a las interferencias electromagnéticas [15]. Finalmente, CAN-bus tiene un ancho de banda de hasta 1 Mbps, lo cual es bajo, pero se caracteriza por tener una latencia baja (menos de 1 ms), lo que lo convertiría en un protocolo ideal para aplicaciones donde se maneja un volumen bajo de datos, como en los sistemas robóticos, y donde la rapidez, fiabilidad y estabilidad del sistema son esenciales para su operación óptima [20].

II. Metodología

A. Protcolo CAN

El protocolo CAN es un bus de campo digital y un sistema de comunicación descentralizado diseñado para permitir la interacción eficiente entre múltiples dispositivos en un entorno distribuido. Este protocolo utiliza un esquema de multicast, en el cual los mensajes se transmiten a todos los nodos de la red, pero solo el nodo destinatario procesa el mensaje. Cada mensaje tiene asignada una prioridad y función específica, determinadas por un número de identificación único de 11 bits. Este identificador también se utiliza durante el proceso de arbitraje, un mecanismo que permite resolver conflictos cuando varios nodos intentan transmitir simultáneamente. Durante el arbitraje, los nodos compiten por el acceso al bus comparando los bits de sus identificadores. El nodo con el identificador de menor valor domina el bus y continúa transmitiendo, mientras que los demás cesan su transmisión. Este enfoque garantiza una resolución determinista de conflictos y reduce la carga de procesamiento en los nodos [18], [21]. Todo mensaje transmitido en la red CAN sigue la estructura de su trama, la cual se detalla a continuación y se representa en la Fig. 1:

- **Start of Frame (SOF):** Este es un único bit dominante que marca el inicio de la transmisión de una trama en el bus CAN. Su función principal es sincronizar a todos los nodos en la red, asegurando que reconozcan el comienzo de una nueva comunicación [22].
- **Arbitration Field (Campo de Arbitraje):**
  - **11-bit identifier:** Determina la prioridad del mensaje en la red y contiene el identificador del mensaje [22].
  - **Remote Transmision Request (RTR):** Mensaje que un nodo usa para solicitar datos a otro nodo en lugar de enviar datos por sí mismo [22].
- **IDE:** Determina si es que se utiliza el identificador estándar de 11 bits (0) o el identificador extendido de 29 bits (1) [22].
- **Reserved Bit (Bit Reservado (r0)):** Es un bit reservado para posibles implementaciones futuras o características adicionales del protocolo CAN. En las versiones actuales, se transmite como recesivo y no desempeña una función activa [22].
- **DLC (Longitud del Campo de Datos):** También conocido como Data Length Code (DLC), este campo especifica la cantidad de bytes de datos que contiene la trama. Su valor puede variar entre 0 y 8 bytes [22].
- **Data Field (Campo de Datos):** Es el campo que transporta los datos útiles que se desean transmitir entre los nodos [22].
- **CRC Field (Campo CRC):** Contiene un código de verificación generado mediante un algoritmo de redundancia cíclica (CRC). Este campo permite a los nodos receptores detectar errores en la transmisión, garantizando la integridad del mensaje [22].
- **Acknowledge Field (Campo de Reconocimiento o ACK):** Este campo confirma que los nodos receptores han recibido la trama correctamente. Los nodos receptores envían un bit dominante en el ACK Slot para indicar la recepción exitosa [22].
- **End of Frame (EOF):** Consiste en una secuencia de 7 bits recesivos que marcan el final de la transmisión de la trama. Este campo señala a todos los nodos que la transmisión ha concluido y que el bus está disponible para la siguiente comunicación [22].

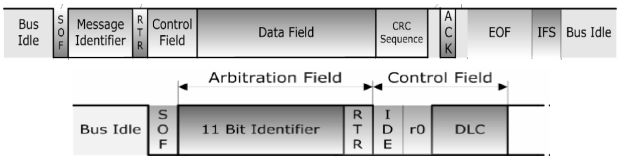


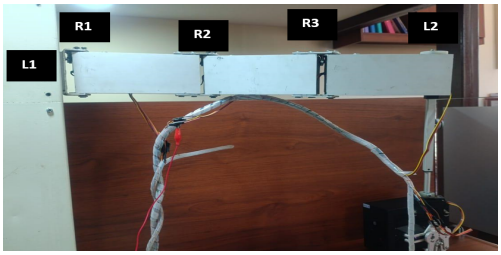
Fig. 1: Estructura del Mensaje CAN [21]

Además, otro aspecto importante es la señalización de los mensajes en el protocolo CAN. Esto se realiza mediante dos estados lógicos: dominante y recesivo que se representan utilizando señalización diferencial, que mide la diferencia de voltaje entre las líneas CANH (CAN-Alto) y CANL (CAN-Bajo). En el estado dominante, que corresponde a un ”0 lógico”, la diferencia de voltaje entre las líneas es alta. Por otro lado, en el estado recesivo, que representa un ”1 lógico”,

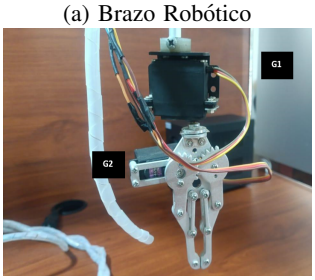
la diferencia de voltaje es baja. Este método no solo facilita la interpretación de los datos, sino que también ayuda a cancelar el ruido, ya que cualquier interferencia que afecte ambas líneas de manera similar no altera la diferencia de voltaje utilizada para transmitir la información [18].

B. Inspección del Robot

Luego de haber profundizado en la teoría y en la aplicación del protocolo CAN-bus en el área de la robótica se procedió a evaluar el estado del robot, etiquetando sus actuadores como se indica en la Fig. 2. Primero se comenzó determinando el número de actuadores presentes en el brazo robótico y luego se realizó un mantenimiento de la estructura. Después, se procedió a realizar una documentación de las conexiones de cada servomotor al conector DB-25 hembra y de cada actuador lineal al DB-9 específico, como se indica en la Tabla I vinculada a la Fig. 3 y la Tabla II vinculada a la Fig. 4, respectivamente.



(a) Brazo Robótico



(b) Garra

Fig. 2: Brazo Robótico y su Garra Etiquetados

La importancia de documentar todas las conexiones de los conectores DB9 y DB25 del brazo robótico está dada por el objetivo de optimizar el proceso de prueba y programación de los motores. Este enfoque permite una gestión más eficiente y fluida durante las fases de verificación y ajuste del sistema, asegurando que cada conexión esté correctamente identificada y asignada en su correspondiente puerto. La documentación detallada facilita la resolución de posibles incidencias y mejora la organización del proceso de inspección del robot.

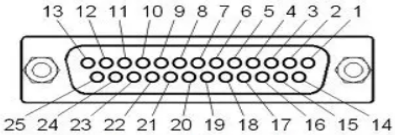


Fig. 3: Esquemático del Conector Hembra DB25

TABLA I: Conexiones de los Servomotores

Número del Puerto	**1**	**2**	**3**	**4**	**5**	**17**	**18**	**19**	**24/25**
Función	G2 PWM	G1 PWM	R3 PWM	R2 PWM	R1 PWM	R3/R1 VCC	R2/G2 VCC	G1 VCC	GND

Con esta documentación se procedió a probar el funcionamiento de los actuadores del brazo robótico, proporcionando una señal y alimentación específica para cada actuador como se indica en [22]–[25], manuales de cada actuador. Por medio de un microcontrolador (ESP32) y una fuente de alimentación variable se realizó la programación de un código básico encargado de mover el servomotor desde su posición inicial hasta su posición final con el fin de configurar los anchos de pulso exactos en los que el servomotor se mueve en un rango completo de 180°.

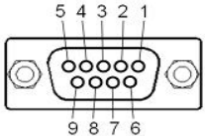


Fig. 4: Esquemático del Conector Hembra DB9

TABLA II: Conexiones de los Motores Lineales

Motor Lineal L1					
Número del Puerto	**1**	**2**	**3**	**4**	**5**
Función	Feedback VCC	GND	VCC	Feedback Posición	Feedback GND
Motor Lineal L2					
Número del Puerto	**1**	**2**	**3**	**4**	**5**
Función	Feedback Posición	GND	VCC	Feedback VCC	Feedback GND

C. Implementación de Hardware

Para mejorar la capacidad de manipulación y ampliar el área de trabajo del brazo robótico, se implementó un motor lineal de movimiento prismático, añadiendo un grado adicional de libertad. Anteriormente, el sistema solo permitía el movimiento de la garra en el plano XY mediante servomotores rotacionales, lo que limitaba la manipulación a lo largo del eje Z. Esta restricción dificultaba el alcance de objetos fuera de una distancia específica sobre dicho eje. Con la incorporación del motor lineal, el robot adquirió mayor versatilidad y amplió su rango operativo. Además, para garantizar un funcionamiento eficiente, se integraron diversos componentes y módulos, como se enlistan a continuación:

- Transceptores CAN (TJA1051 y MCP2551)
- Convertidor de nivel bidireccional (3.3V – 5V)
- ESP32 Dev Module

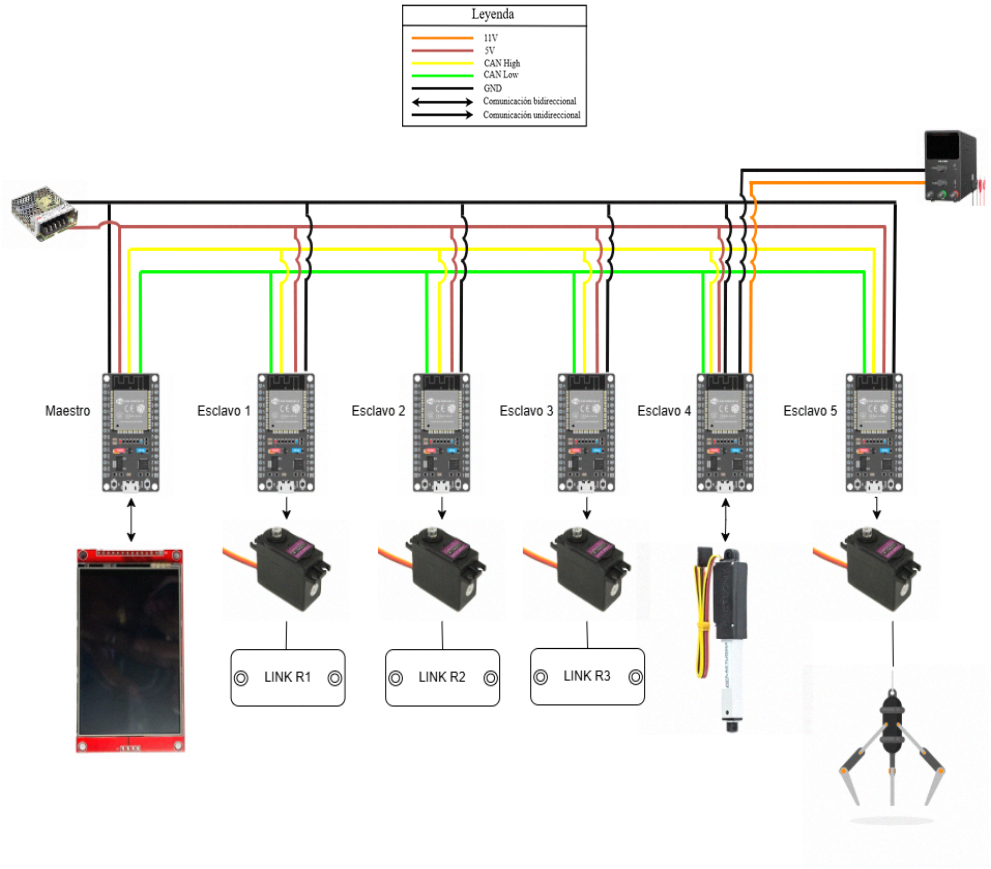


Fig. 5: Topología Física del Sistema Robótico

- Driver de motor DVR8833 (únicamente para el motor lineal L2)
- Resistencia de 120 Ω para la terminación del bus CAN
- Pantallas OLED para la visualización de mensajes CAN
- Pantalla LCD TFT táctil de 4”

En cuanto a la elección del microcontrolador, se tomó en consideración las capacidades de comunicación presentes tanto en microcontroladores Arduino como en los ESP32. Tras un análisis comparativo, se determinó la superioridad del ESP32 en aplicaciones CAN debido a su controlador CAN integrado, a diferencia de los modelos de Arduino, así como por su mayor velocidad de procesamiento. Luego, se realizó el prototipado del circuito en un protoboard para facilitar ajustes en el diseño. A continuación, se presenta el diagrama general de conexiones y su respectiva leyenda en la Fig. 5.

En la Fig. 5 se muestra cada nodo o articulación del brazo robótico, junto con el nodo maestro encargado de procesar la información de la pantalla LCD táctil mediante comunicación SPI. Los esclavos 1, 2, 3 y 5 controlan la posición de los servomotores de 5V encargados del movimiento de los eslabones R1, R2, R3 y la garra, respectivamente. El esclavo 4, por su parte, controla un motor lineal de corriente continua (L2), que opera a 11V. Además, todos los nodos están interconectados mediante un bus de comunicación CAN con las líneas CAN-Alto y CAN-Bajo, mientras que la alimentación se distribuye

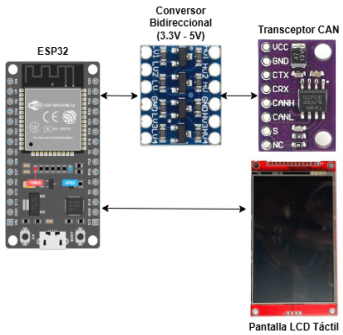


Fig. 6: Nodo Maestro: Interfaz Táctil para Usuario

a través de un bus de 5V para los ESP32 y una línea de 11V exclusiva para el motor lineal. A continuación, se muestra un diagrama general de conexiones por cada nodo dentro de la red CAN en la Fig. 6, 7 y 8 que provee una idea general de las conexiones y módulos necesarios para conseguir una comunicación y control exitosa.

En la Fig. 6 se destacan los componentes claves para el funcionamiento del nodo maestro, como el microcontrolador principal, encargado de gestionar la comunicación y procesar las señales entrantes y salientes, así como el transceptor CAN, que facilita la interacción con los nodos esclavos a través del

bus de datos. Además, para asegurar la compatibilidad entre los diferentes niveles de voltaje de los componentes, se incorporó un módulo de conversión bidireccional de 3.3V a 5V, permitiendo la correcta transmisión de datos entre dispositivos que operan con distintos voltajes lógicos. Asimismo, se integró una pantalla LCD táctil para comandar los nodos esclavos a realizar diferentes acciones programadas.

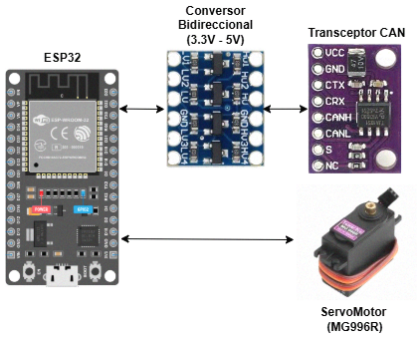


Fig. 7: Nodo Esclavo (1,2,3,5): Control de Servomotores

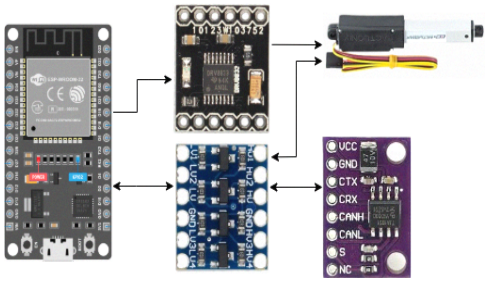


Fig. 8: Nodo Esclavo 4: Control del Motor Lineal

En la Fig. 7 se muestra el diagrama de conexiones generales de las placas que manejan los servomotores rotacionales, en donde además de los módulos de conversión de nivel lógico bidireccional y de comunicación CAN se tiene un servomotor. El mismo que controlará el movimiento rotacional de un eslabón o la apertura y cierre de la garra. En cambio, la Fig. 8 muestra el diagrama de conexiones de la placa que controla el motor lineal en el que se tienen como componentes adicionales a los principales módulos (conversor y transceptor) un driver de motor, encargado de invertir el sentido de giro del motor lineal y el motor lineal.

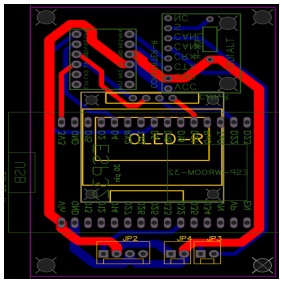


Fig. 9: Diseño PCB del Esclavo 1,2 y 3

Tras finalizar el prototipado del circuito, se diseñaron las placas PCB correspondientes a cada motor utilizado, así como una placa dedicada a la interfaz de usuario. Esta última tiene el propósito de servir como nodo maestro, encargado de enviar las órdenes de acción a los demás nodos mediante comunicación CAN. Para el diseño de las placas PCB se utilizó la plataforma EasyEDA. A continuación, se presentan el diseño PCB y la vista 3D de las placas encargadas de controlar el movimiento de los servomotores R1, R2 y R3.

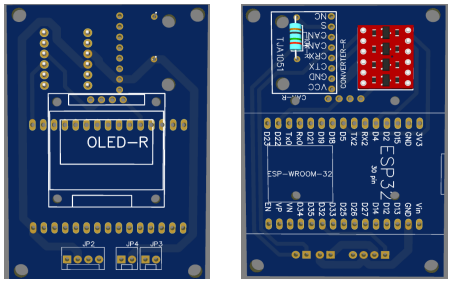


Fig. 10: Perspectiva 3D del PCB - Vista delantera y posterior

La Fig. 9 presenta la vista esquemática de la placa PCB diseñada, la cual cuenta con un diseño de doble cara para optimizar la distribución de los componentes y las conexiones eléctricas. Dado que la placa aprovecha ambas caras para la disposición de las pistas y elementos electrónicos, se proporciona una representación detallada de cada lado en la Fig. 10. Aquí se muestra la vista delantera, donde se pueden apreciar los componentes visibles para el usuario como la pantalla OLED y su disposición en la superficie. Además, se muestra la vista trasera, permitiendo analizar la disposición de las conexiones y el enrutamiento de las pistas en la misma.

El diseño PCB presentado se encarga del funcionamiento del servomotor R1, y se replicará para controlar R2 y R3. Es un diseño de doble cara, optimizando el espacio y la distribución de componentes para facilitar su montaje en cada eslabón del brazo robótico.

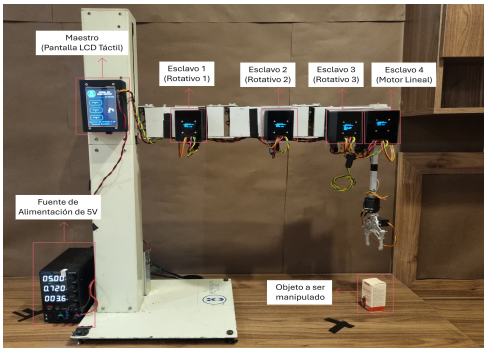


Fig. 11: Vista Frontal del Robot

Además, se tuvo en cuenta la demanda de corriente de las pistas encargadas de la alimentación y los servomotores para evitar caídas de voltaje y sobrecalentamiento. Por ello, se usaron pistas de 2 mm para la alimentación del servomotor (máximo 2A) y pistas de 1 mm para las señales de

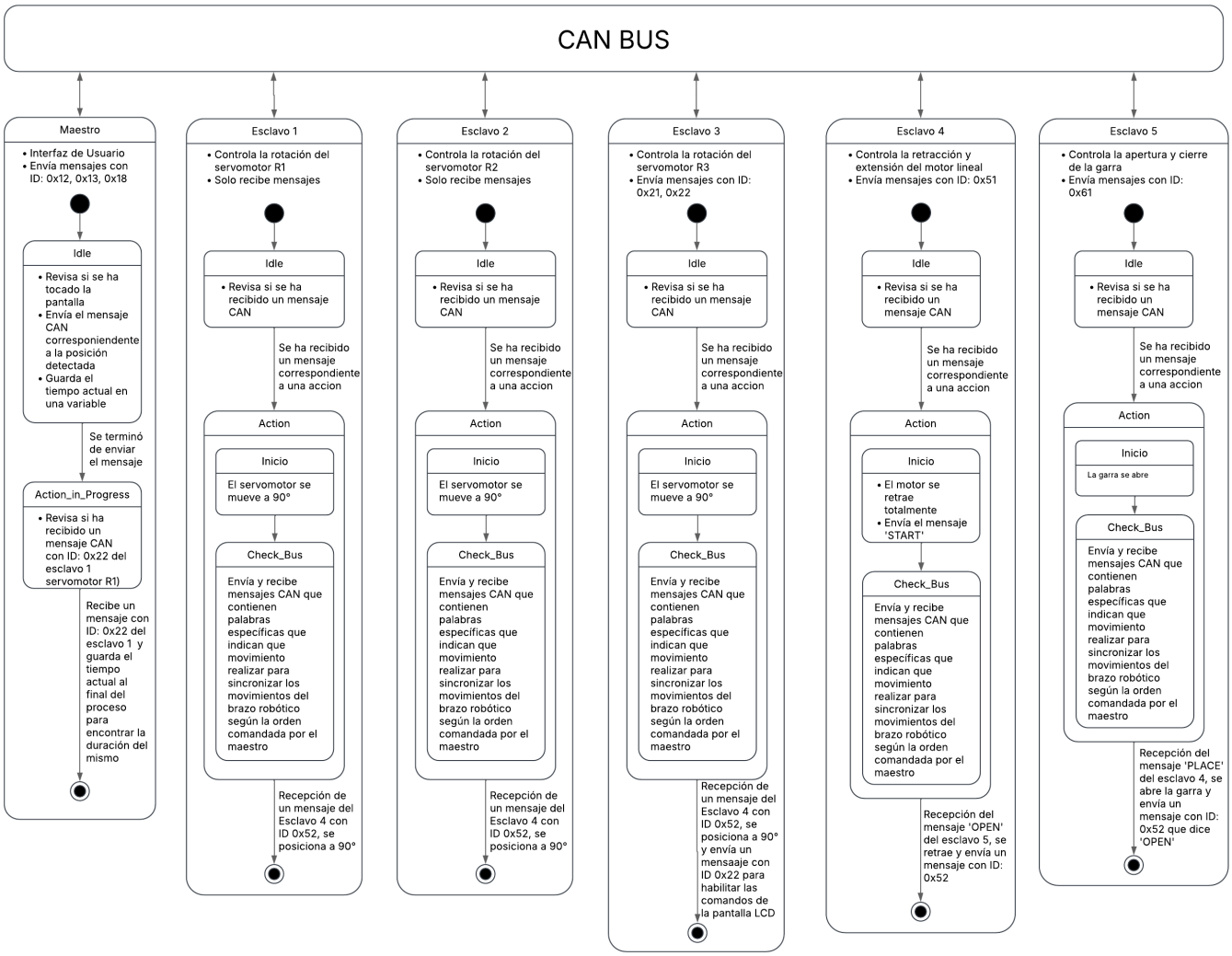


Fig. 12: Lógica de Programación del Sistema Robótico

comunicación y control PWM (máximo 1A) ya que estas no demandan mucha corriente.

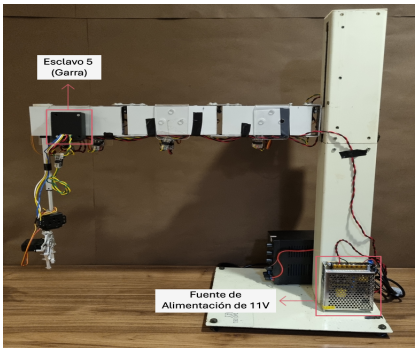


Fig. 13: Vista Posterior del Robot

Una vez terminados los diseños se mandó a realizar impresiones 3D de cajas para que escondan las placas y dejen ver las pantallas OLED. En la Fig. 11 se puede observar la vista frontal del robot con los módulos implementados, incluyendo la fuente de alimentación. Desde esta perspectiva se observan el nodo maestro, conformado por la pantalla LCD, los nodos esclavos 1 (R1), 2 (R2), 3 (R3) y 4 (Motor Lineal), además de la fuente de alimentación de 5V, que suministra energía a los microcontroladores y servomotores. Por otra parte, en la Fig. 13, se identifican un módulo adicional (Esclavo 5 - Garra) y la fuente de alimentación de 11V.

D. Implementación de Software

1) Lógica General del Sistema: Para obtener movimientos fluidos y coordinados del brazo robótico, se implementó, parcialmente, una lógica por máquina de estados pero también mediante el envío y recepción de mensajes CAN que provocan una acción específica en un nodo esclavo. Cada nodo esclavo, analiza primero el identificador del mensaje, luego analiza el mensaje o contenido como tal para realizar una acción determinada. Como se indica en la Fig. 12, un nodo maestro actúa como interfaz de usuario y envía mensajes a través del

bus CAN con diferentes identificadores (ID), según el botón táctil que se presione en la pantalla LCD, para coordinar las acciones de los esclavos. Además, el esclavo trabaja con estados para evitar que se envíe otro comando mientras se está realizando una acción determinada. Por otra parte, cada nodo esclavo tiene una función específica: los esclavos 1, 2 y 3 controlan la rotación de los servomotores R1, R2 y R3 respectivamente; el esclavo 4 maneja la retracción y extensión del motor lineal; y el esclavo 5 controla la apertura y cierre de la garra. La estructura interna de cada nodo sigue una estructura de estados donde cada esclavo entra en un estado de espera **(Idle)** hasta recibir un mensaje del maestro. Al recibir un mensaje con un ID específico, el esclavo ejecuta una acción programada, mueve el actuador correspondiente y responde con un mensajes de confirmación ("DONE", "PLACE" y otros mensajes que no se mencionan por cuestión de brevedad) al siguiente nodo en la secuencia. La utilización de un Bus CAN en esta arquitectura permite comandar 2 o más nodos a la vez, incrementando así la velocidad de accionamiento comparado con un sistema totalmente secuencial. Es necesario mencionar que la parte secuencial del código se debe a la función que realiza el robot, ya que en este caso se tiene un robot de tipo *Pick and Place*, lo que genera la necesidad de esperar a que se extienda totalmente el motor lineal para cerrar la garra (recoger o asentar).

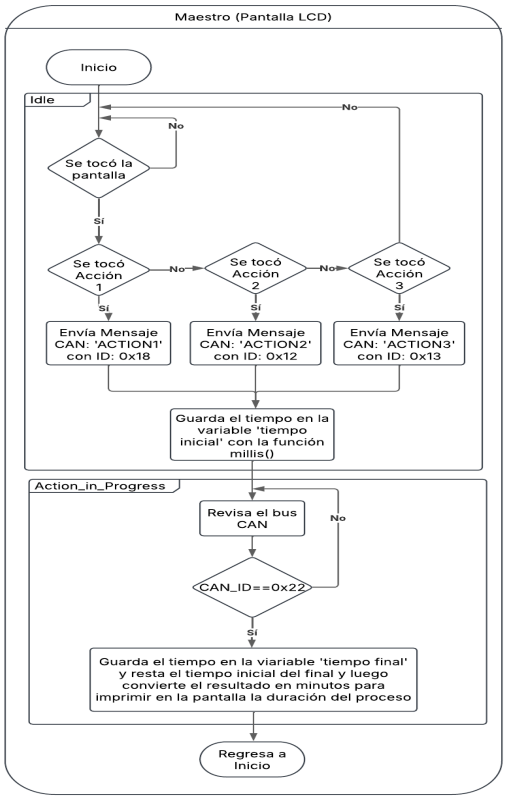


Fig. 14: Programación Lógica del Maestro

2) Lógica Detallada del Sistema: A continuación, se describe de forma más específica el funcionamiento del sistema

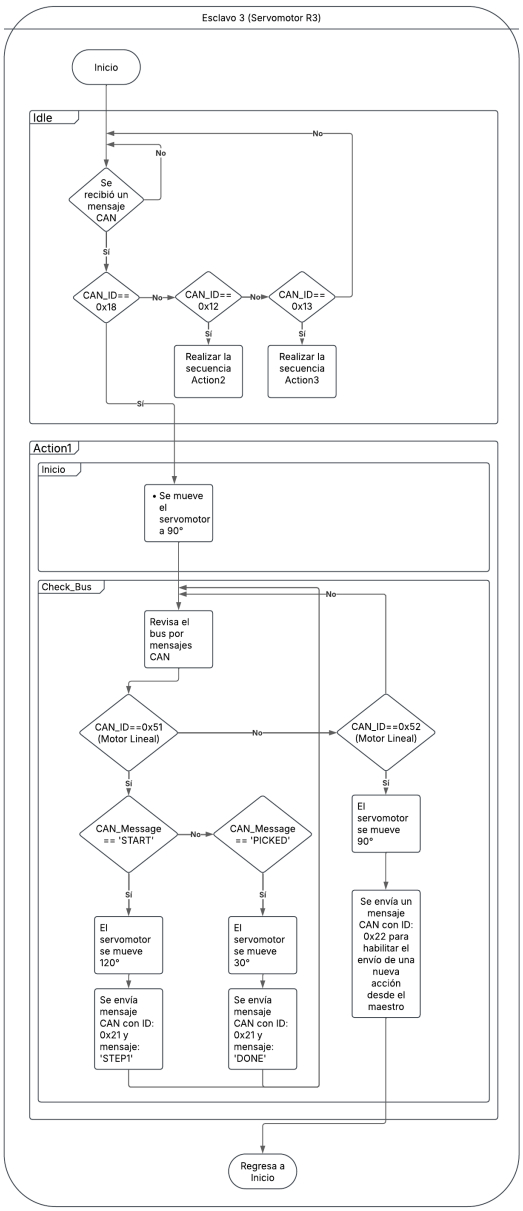


Fig. 15: Progrmación Lógica del Esclavo 3 (Servomotor R1)

robótico, comenzando con el nodo maestro. En la Fig. 14 se muestran dos estados en donde el primer estado **(Idle)** se encarga de revisar si se ha tocado la pantalla para luego, determinar qué botón táctil se presionó basado en la posición detectada. El segundo estado **(Action_in_Progress)** se encarga de constantemente revisar el bus CAN por un mensaje con el identificador 0x22 que será enviado por parte del nodo esclavo 3 encargado del movimiento del servomotor rotativo R1 y de enviar un mensaje que indica la finalización total de la secuencia comandada. Esto provocará el regreso al estado inicial **(Idle)** en donde se volverá a esperar que alguien presione un botón.

En el esclavo 3 en cambio, se tienen 2 estados principales y

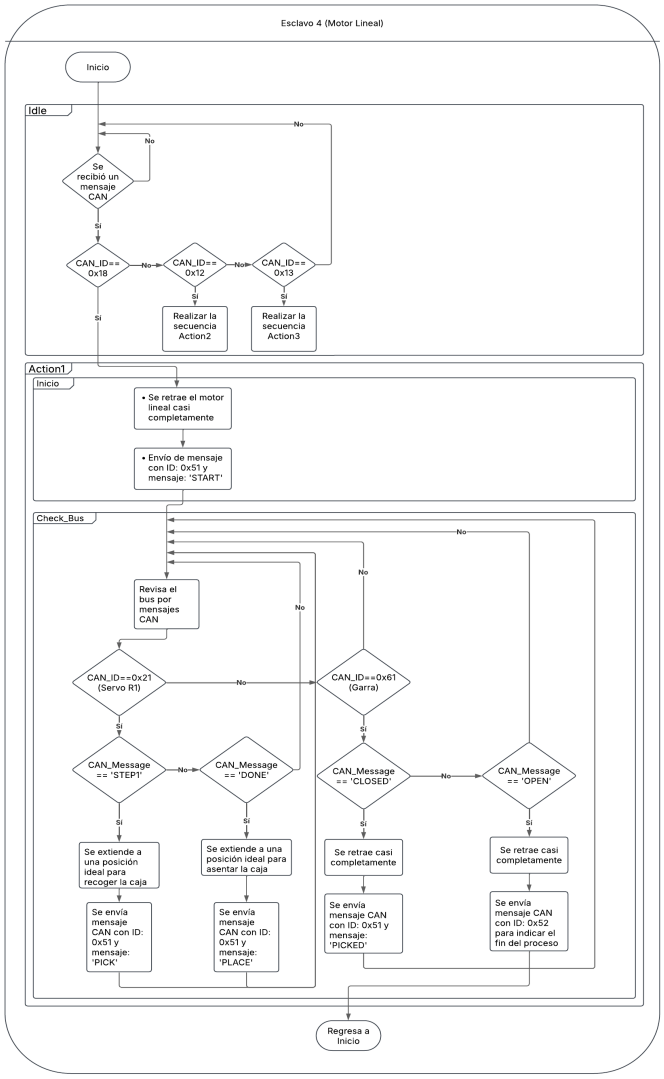


Fig. 16: Progrmación Lógica del Esclavo 4 (Motor Lineal)

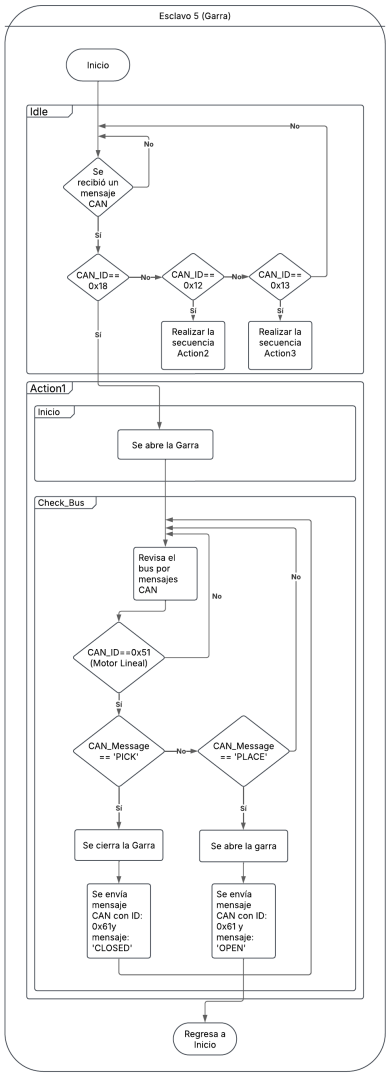
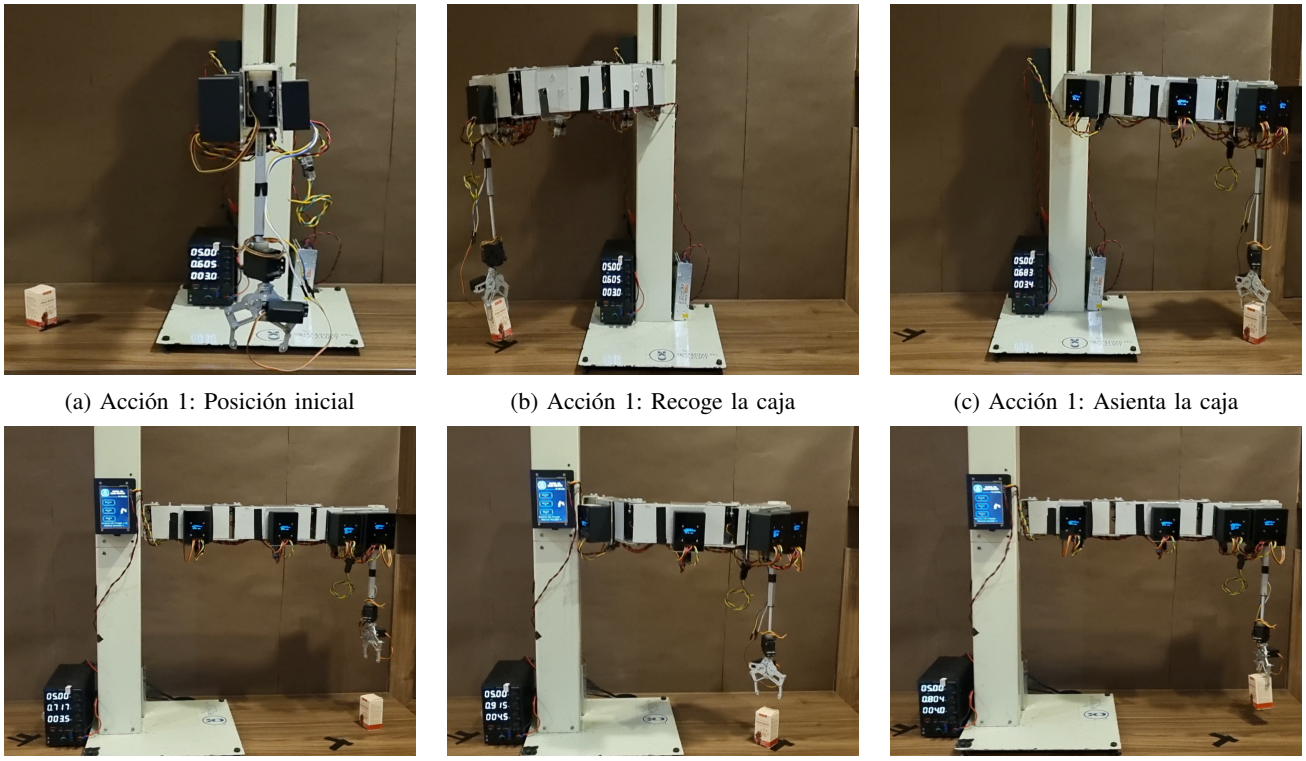


Fig. 17: Progrmación Lógica del Esclavo 5 (Garra)

2 subestados dentro de uno de ellos para controlar las acciones del servomotor R1, como se muestra en la Fig. 15. En el primer estado (Idle) se revisa el bus CAN por mensajes del nodo maestro para comenzar una secuencia. Una vez que se detecta una acción/secuencia determinada, se pasa al siguiente estado que en este ejemplo es Action1.

Dentro de este estado se tienen 2 subestados en donde se mueve el servo a su estado inicial **(Inicio)** y luego un estado **(Check_Bus)** donde se revisa los mensajes en el bus CAN y se filtra según su identificador y mensaje. Cuando se recibe un mensaje del esclavo 4 (motor lineal) se mueve a 120° y envía un mensaje que avisa que ha realizado el primer movimiento. Una vez que el motor lineal y la garra han realizado los movimientos coordinados para recoger una caja y el motor lineal ha subido, el motor lineal envía un mensaje (“PICKED”) que informa al esclavo 3 que debe realizar el segundo movimiento a 30°, luego de moverse, el esclavo 3 enviará un mensaje (“DONE”) avisando que se ha realizado su último movimiento de la secuencia. El esclavo 3 sigue revisando el bus hasta que llega un mensaje con identificador 0x52 del motor lineal que indica que la secuencia finalizó y esto provoca que el servomotor se mueva a 90° (posición inicial) y envía un mensaje con identificador 0x22 que avisa al maestro que ya puede regresar a revisar si se ha tocado la pantalla para realizar otra secuencia.

En la Fig. 16 se muestran los mismos estados del esclavo 1, pero con diferencias en las acciones y mensajes filtrados en el subestado inicio y el estado **Check_Bus**. El motor lineal se retrae inicialmente y envía el mensaje “START” (ID 0x51) para indicar el inicio de la secuencia. En el estado **Check_Bus**, se filtran los mensajes del esclavo 3; si se recibe “STEP1”, el motor se extiende y, al llegar a la posición de recolección, envía “PICK” (ID 0x21) para que la garra cierre y recoja el objeto. Luego de recibir “CLOSED” (ID 0x61) de la garra, el motor se retrae casi completamente y envía “PICKED” (ID 0x51) para que el esclavo 3 se mueva al destino. Cuando el



(a) Acción 1: Posición inicial (b) Acción 1: Recoge la caja (c) Acción 1: Asienta la caja

(d) Acción 2: Posición inicial (e) Acción 2: Recogiendo la caja (f) Acción 2: Asentando la caja

Fig. 18: Secuencia de Movimiento de cada Acción

esclavo 3 envía "DONE" (ID 0x21), el motor extiende y envía "PLACE" (ID 0x51) para que la garra abra y deje el objeto. Al recibir "OPEN" (ID 0x61) de la garra, el motor se retrae y envía el mensaje final con ID 0x52, indicando que el proceso ha terminado y todos los esclavos regresan a su posición inicial (90°). Finalmente, el esclavo 3 (servo R3) envía un mensaje con ID 0x22 y el maestro lo filtra de manera que consigue regresar a su estado inicial en el que está revisando si es que se ha tocado la pantalla. Los procesos del motor lineal y de la garra se pueden observar en la Fig. 16 y la Fig. 17. Es importante mencionar que la lógica específica de los esclavos 2 y 3 no se menciona debido a su similitud casi exacta a la lógica del esclavo 3, con la diferencia que los grados de movimiento son distintos y no envían mensajes después de realizar dichos movimientos. Esto sucede ya que al realizar los movimientos al mismo tiempo que el servomotor R1, se puede descartar la necesidad de que estos comuniquen que ya se han movido. Esto se realizó con el objetivo de reducir tiempos de procesamientos de mensaje para optimizar el tiempo en el que se realiza la secuencia.

III. Pruebas y resultados

A. Movimiento del Brazo Robótico

En esta subsección se presentan las pruebas de funcionamiento realizadas para verificar la correcta ejecución de las secuencias de movimiento del brazo robótico a través de la implementación del protocolo CAN. Se evaluó si el sistema respondió adecuadamente a los comandos enviados y si los nodos ejecutaron las acciones esperadas en el orden correcto.

El robot consta de 3 secuencias de movimiento. En todas las secuencias, el robot tendrá como objetivo recoger, movilizar y asentar una caja desde una ubicación a otra. La diferencia entre las acciones se basará en la velocidad de funcionamiento y tipo de movimiento que realizará, es decir, la variación de los ángulos de movimiento de cada articulación. La primera secuencia (acción 1) observada en la Fig. 18a, 18b, 18c está encargada de movilizar una caja desde una posición a la izquierda de la base del robot a una posición a la derecha del robot a una velocidad baja. La segunda secuencia (acción 2) encontrada en la Fig. 18d, 18e y 18f tiene el objetivo de recoger una caja en la parte central delantera más cercana de la base y colocarla en una ubicación central delantera alejada de la base. La tercera secuencia (acción 3), realiza los mismos movimientos que la primera secuencia (acción 1), pero a una velocidad más elevada, es por esto que se omite la tercera acción ya que es la misma secuencia de movimientos de la acción 1, con la diferencia de una velocidad incrementada.

B. Comunicación CAN

En esta subsección se detalla el proceso de verificación y validación de la comunicación a través del protocolo CAN implementado en el sistema. El objetivo principal fue asegurar que los mensajes transmitidos y recibidos por el bus fueran consistentes, íntegros y que cumplieran con la estructura

definida por el protocolo CAN 2.0. Para ello, se llevaron a cabo distintas pruebas técnicas utilizando herramientas de hardware y software especializadas, que permitieron observar tanto el contenido lógico de los mensajes como su comportamiento físico a nivel de señal.

La primera prueba consistió en la captura de mensajes presentes en el bus CAN utilizando un adaptador USB2CAN conectado a una computadora y el software SavvyCAN. Esta herramienta permitió visualizar en tiempo real los mensajes transmitidos, mostrando información clave como el identificador (ID) del mensaje, la longitud de los datos, la interpretación en formato ASCII y los datos en hexadecimal.

ID	Ext	RTR	Dir	Bus	Len	ASCII	Data
0x018	0	0	Rx	0	7	ACTION1	41 43 54 49 4F 4E 31
0x051	0	0	Rx	0	5	START	53 54 41 52 54
0x021	0	0	Rx	0	5	STEP1	53 54 45 50 31
0x051	0	0	Rx	0	4	PICK	50 49 43 4B
0x061	0	0	Rx	0	6	CLOSED	43 4C 4F 53 45 44
0x051	0	0	Rx	0	6	PICKED	50 49 43 4B 45 44
0x021	0	0	Rx	0	4	DONE	44 4F 4E 45
0x051	0	0	Rx	0	5	PLACE	50 4C 41 43 45
0x061	0	0	Rx	0	4	OPEN	4F 50 45 4E
0x052	0	0	Rx	0	3	FIN	46 49 4E
0x022	0	0	Rx	0	5	RESET	52 45 53 45 54

Fig. 19: Recepción de Mensajes CAN: Action1

En las capturas obtenidas con SavvyCAN en la Fig. 19 se registró el conjunto de mensajes correspondientes a la acción 1. El primer mensaje (“ACTION1”) recibido tiene el identificador 0x18, proveniente del nodo maestro, lo que indica el inicio de la secuencia. A continuación, se recibe un mensaje (“START”) con ID 0x51 enviado por el esclavo 4 (motor lineal), que informa que se encuentra en la posición retraída.

Seguidamente, el esclavo 3 (servomotor R1) envía un mensaje (“STEP1”) con ID 0x21 notificando que ha realizado su primer movimiento, posicionándose a 140° con respecto al eje Y. Luego, el esclavo 4 indica que se ha extendido hasta la posición programada para permitir la recolección con el mensaje “PICK”. Después, el esclavo 5 (garra) envía un mensaje (“CLOSED”) que confirma que se ha cerrado, es decir, que ha recogido la caja correctamente.

Posteriormente, el esclavo 4 informa que el motor lineal se ha retraído completamente mediante el mensaje “PICKED”. Luego, el esclavo 3 envía un mensaje (“DONE”) notificando que ha completado el último movimiento de su secuencia, desplazándose a la posición de 40° con respecto al eje Y. El motor lineal (esclavo 4) vuelve a extenderse, esta vez hasta la posición de colocación de la caja, como se indica en el mensaje “PLACE” y posteriormente la garra (esclavo 5) confirma mediante un mensaje (“OPEN”) que se ha abierto para soltar el objeto.

Finalmente, el esclavo 4 indica que se ha retraído completamente (“FIN”), lo que marca el fin de la secuencia. El último mensaje (“RESET”) proviene del esclavo 3, con el identificador 0x22, y señala que el sistema ha sido reseteado. Este mensaje reactiva la funcionalidad táctil de la pantalla, que permanecía deshabilitada durante la ejecución de la acción para evitar interrupciones no deseadas.

ID	Ext	RTR	Dir	Bus	Len	ASCII	Data
0x012	0	0	Rx	0	7	ACTION2	41 43 54 49 4F 4E 32
0x051	0	0	Rx	0	5	START	53 54 41 52 54
0x021	0	0	Rx	0	5	STEP1	53 54 45 50 31
0x051	0	0	Rx	0	4	PICK	50 49 43 4B
0x061	0	0	Rx	0	6	CLOSED	43 4C 4F 53 45 44
0x051	0	0	Rx	0	6	PICKED	50 49 43 4B 45 44
0x021	0	0	Rx	0	4	DONE	44 4F 4E 45
0x051	0	0	Rx	0	5	PLACE	50 4C 41 43 45
0x061	0	0	Rx	0	4	OPEN	4F 50 45 4E
0x052	0	0	Rx	0	3	FIN	46 49 4E
0x022	0	0	Rx	0	5	RESET	52 45 53 45 54

Fig. 20: Recepción de Mensajes CAN: Action2

La misma secuencia de mensajes ocurre para la acción 2, como se indica en la Fig. 20, con la única diferencia que el primer mensaje que se recibe (“ACTION2”) tiene el identificador 0x12. Este mensaje activa otro estado dentro de la programación de cada nodo, en donde cada mensaje provoca un movimiento distinto.

ID	Ext	RTR	Dir	Bus	Len	ASCII	Data
0x013	0	0	Rx	0	7	ACTION3	41 43 54 49 4F 4E 33
0x051	0	0	Rx	0	5	START	53 54 41 52 54
0x021	0	0	Rx	0	5	STEP1	53 54 45 50 31
0x051	0	0	Rx	0	4	PICK	50 49 43 4B
0x061	0	0	Rx	0	6	CLOSED	43 4C 4F 53 45 44
0x051	0	0	Rx	0	6	PICKED	50 49 43 4B 45 44
0x021	0	0	Rx	0	4	DONE	44 4F 4E 45
0x051	0	0	Rx	0	5	PLACE	50 4C 41 43 45
0x061	0	0	Rx	0	4	OPEN	4F 50 45 4E
0x052	0	0	Rx	0	3	FIN	46 49 4E
0x022	0	0	Rx	0	5	RESET	52 45 53 45 54

Fig. 21: Recepción de Mensajes CAN: Action3

Por último, en los mensajes recibidos del proceso de la acción 3 presentes en la Fig. 21 se tiene que el primer mensaje recibido (“ACTION3”) tiene el identificador 0x13, lo que provoca la activación de otro estado en cada nodo, generando los mismos movimientos que en la acción 2 pero a una velocidad mayor.

Como se puede observar en la recepción de mensajes en cada figura mencionada, los movimientos de los motores son activados por palabras que hacen la función de subprocesos dentro de cada acción. Es por esto que, a pesar que se reciben los mismos mensajes dentro de cada proceso, exceptuando el primer mensaje de comando, los nodos están programados para realizar diferentes movimientos dentro de cada acción.

La segunda prueba fue realizada mediante un osciloscopio, con el cual se midieron directamente las señales eléctricas en el bus CAN. Esta herramienta permitió capturar y analizar la forma de onda completa de cada trama transmitida. Esta medición fue crucial para confirmar el correcto funcionamiento de la comunicación a nivel físico ya que con esto

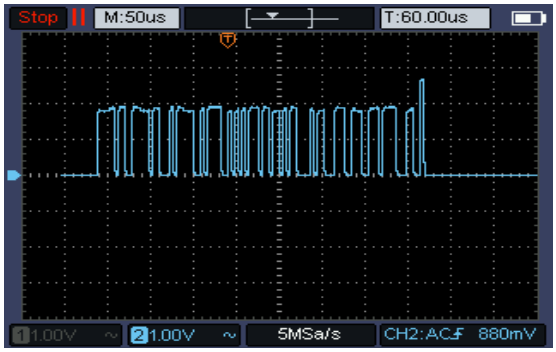


Fig. 22: Medición de la Trama del Mensaje CAN: Comando de acción 1

se pudo comprobar que los niveles de voltaje registrados sean los adecuados. El primer mensaje que se analizó fue el mensaje de comando proveniente del nodo maestro (“ACTION1”) que tiene como identificador 0x18. Como se puede observar en la Fig. 22 los niveles de voltaje varían de forma binaria entre 0V y 2V debido a que la medición se realizó entre las líneas CAN-Alto y CAN-Bajo, obteniendo así el voltaje diferencial. En este tipo de señal, cuando se transmite un bit recesivo, ambas líneas se estabilizan alrededor de 2.5V, resultando en una diferencia cercana a 0V. En cambio, al transmitirse un bit dominante, CAN-Alto sube a aproximadamente 3.5V y CAN-Bajo baja a cerca de 1.5V, generando una diferencia de aproximadamente 2V. En el segundo mensaje capturado en la Fig. 23 se observó la misma variación de voltaje, lo cual indica un bus estable y sin reflexiones de señal significativas. Esto sugiere que la terminación del bus está correctamente implementada y que las impedancias están adecuadamente adaptadas, lo que garantiza la integridad de la señal y una comunicación confiable entre los nodos.

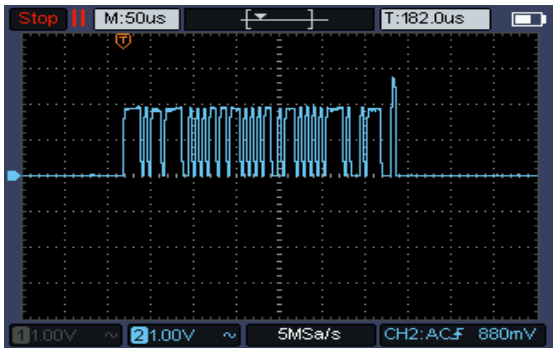


Fig. 23: Medición de la Trama del Mensaje CAN: FIN de la acción 1

C. Visualización del Proceso en las interfaces gráficas

En esta sección, se describe la visualización del proceso de control del brazo robótico mediante las interfaces gráficas de los dispositivos utilizados en el sistema, como se indica en la Fig. 24. Cada nodo esclavo (R1, R2, R3 y L2), exceptuando la garra, está equipado con pantallas OLED que muestran los mensajes enviados y recibidos desde dicho nodo, lo que facilita la monitoreo del flujo de comunicación en la red. Se puede observar un ejemplo en el que el esclavo 3 envía el último mensaje que restablece las condiciones iniciales del sistema como se indica en la Fig. 24c y el resto de esclavos reciben el mensaje al mismo tiempo como indan la Fig. 24a, 24b y 24c

(a) Esclavo 1: Recibe ’RESET’

(b) Esclavo 2: Recibe ’RESET’

(c) Esclavo 3: Envía ’RESET’

(d) Esclavo 4: Recibe ’RESET’

Fig. 24: Mensajes recibidos por los esclavos

Además, el nodo maestro cuenta con una pantalla LCD táctil que presenta tres botones, cada uno correspondiente a una acción específica (acción 1, acción 2, acción 3), como indican en las imágenes de la Fig. 25. Al presionar uno de estos botones, se activa una animación que simula el movimiento de un robot de izquierda a derecha, ubicada a la derecha del botón seleccionado, como una indicación visual de la acción tomada. En la parte inferior de la pantalla, se muestra un contador de mensajes enviados desde el nodo maestro, que se actualiza cada vez que se presiona un botón y tras la finalización de una acción, la pantalla muestra la duración del proceso, como se describe a continuación.

(a) Acción 1

(b) Acción 2

(c) Acción 3

Fig. 25: Duración de cada Acción

Para registrar la duración de cada acción ejecutada por el brazo robótico, se utilizó la función nativa del ESP32 millis(), la cual permite obtener el tiempo transcurrido en milisegundos desde el inicio del programa. Mediante esta función, se almacenó el momento en que se enviaba el comando de inicio de una acción y el instante en que se recibía el último mensaje con identificador 0x22, correspondiente al final de dicha acción. Al calcular la diferencia entre estos dos valores y convertirlos a minutos, se obtuvo la duración total de cada tarea. Esta información se muestra en la pantalla OLED de cada nodo una vez completada la acción, permitiendo una verificación directa del tiempo expresado en minutos con dos cifras decimales, lo que facilita la interpretación rápida y precisa de los resultados. En la Fig. 25a, 25b y 25c se presentan los tiempos registrados: la Acción 1 tuvo una duración de 1.12 minutos, la acción 2 de 0.96 minutos y la acción 3 de 0.99 minutos, respectivamente. La menor duración registrada en la acción 2 se debe a que esta contempla un rango de movimiento más reducido en comparación con las otras acciones. Por otro lado, tanto la acción 1 como la acción 3 presentan la misma secuencia de movimientos, sin embargo, la acción 3 se ejecuta en menor tiempo, lo cual se atribuye a un incremento en la velocidad programada, evidenciando así el impacto de este parámetro sobre el desempeño del sistema.

IV. Conclusiones

La implementación del protocolo CAN-Bus para el control del movimiento de un brazo robótico ha demostrado ser una solución eficaz, robusta y altamente adaptable a entornos distribuidos. Durante el desarrollo del proyecto, se validó la integridad y fiabilidad de la comunicación entre nodos mediante múltiples herramientas, destacando SavvyCAN para el análisis de tramas y un osciloscopio para la verificación de niveles diferenciales de señal. Esto permitió no solo comprobar que los mensajes cumplen con el estándar CAN 2.0 en su estructura y contenido, sino también confirmar que las condiciones eléctricas del bus, como la estabilidad de la señal y la correcta terminación, eran las adecuadas, evitando reflexiones o interferencias.

Los mensajes capturados mostraron una secuencia lógica clara entre el nodo maestro y los nodos esclavos, asignados a actuadores específicos como el motor lineal, los servos y la garra. Se evidenció un uso coherente de identificadores y un orden correcto en la ejecución de tareas, desde la activación inicial hasta el reseteo final que reactiva las funciones táctiles. Este comportamiento demostró una arquitectura distribuida eficiente, donde la comunicación asincrónica y priorizada del CAN-Bus permitió una sincronización y control adecuado.

Se observó que el tiempo de ejecución varía directamente con el rango de movimiento y la velocidad configurada. También se comprobó que el diseño modular basado en CAN no solo reduce significativamente la complejidad del cableado, sino que facilita el mantenimiento y la escalabilidad del sistema.

En conclusión, el protocolo CAN-Bus resultó ser una tecnología adecuada para sistemas robóticos que requieren confiabilidad, sincronización y modularidad, mejorando el rendimiento y estabilidad del brazo robótico. No obstante, se recomienda que en futuros trabajos se preste especial atención al diseño y construcción mecánica del brazo, ya que problemas como holguras y desalineaciones afectaron la precisión y estabilidad del sistema. Una estructura robusta es esencial para garantizar la repetibilidad, eficiencia y seguridad del robot en aplicaciones reales.

V. Futuras Investigaciones

Para futuras investigaciones, se prodría explorar la implementación del protocolo CAN FD (Flexible Data-rate) como una evolución del estándar CAN 2.0 utilizado en el presente proyecto, ya que CAN FD permite transmitir tramas con mayor longitud de datos (hasta 64 bytes) y velocidades más altas, lo que se traduce en una mejora significativa en la eficiencia de la comunicación y en la reducción del tiempo de transmisión. Además, se sugiere realizar un análisis comparativo entre el protocolo CAN (incluyendo su versión FD) y otros protocolos industriales como Profibus o Modbus, evaluando su comportamiento en términos de velocidad de transmisión, inmunidad al ruido, latencia y facilidad de integración. Este tipo de estudio permitiría tomar decisiones más informadas al seleccionar la tecnología de comunicación más adecuada para aplicaciones específicas en automatización y robótica.

Referencias

[1] S. Banzi and E. Mainardi, "A can-based distributed control system for upper limb myoelectric prosthesis," in *2005 ICSC Congress on Computational Intelligence Methods and Applications*, 2005.

[2] Z. Yu, Q. Huang, J. Li, X. Chen, and K. Li, "Computer control system and walking pattern control for a humanoid robot," in *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Jul. 2008.

[3] X. Zhou, Z. Yu, M. Wang, D. Chen, and X. Ye, "Design of control system for lower limb exoskeleton robot," in *2022 8th International Conference on Control, Automation and Robotics (ICCAR)*, 2022, pp. 122–126.

[4] C. A. H. Martínez, C. C. Morales, and O. A. Martínez, "Automation of a robotic arm by analyzing the can protocol," in *Proceedings of the 14th International Conference on Electronics, Communications and Computers (CONIELECOMP'04)*, 2004, pp. 1–6.

[5] O. Barker, R. Beranek, and M. Ahmadi, "Design of a 13 degree-of-freedom biped robot with a can-based distributed digital control system," in *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2010, pp. 836–841.

[6] P. Zhang, X. Gao, M. Miao, and P. Zhao, "Design and control of a lower limb rehabilitation robot based on human motion intention recognition with multi-source sensor information," *Machines*, vol. 10, no. 1125, pp. 1–19, Nov. 2022.

[7] R.-J. Wang, H.-P. Huang, P.-T. Lee, and H.-F. Liao, "A vertically intersected dual-axis modularized reconfigurable actuator: Design and application for a six-axis humanoid robot arm," *Journal of the Chinese Institute of Engineers*, vol. 36, no. 4, pp. 530–541, Jun. 2013.

[8] Y. Wang and Y. Ge, "The distributed control system of a fruit and vegetable picking robot based on can bus," in *Department of Electrical Engineering and Automation, Luoyang Institute of Science and Technology*, 2011.

[9] R. Chen, B. Liu, M. Pan, and H. Zhou, "Design of distributed control system for the pick-up robot based on can bus," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2019, pp. 102–107.

[10] X. Gao, M. Wu, Z. Fu, Y. Zhao, and S. Chen, "The control system design of a climbing welding robot based on can bus," in *Robotic Welding, Intelligence and Automation*, T.-J. Tarn, S.-B. Chen, and G. Fang, Eds. Springer Berlin Heidelberg, 2011, pp. 429–434.

[11] K. Xiang, Z. Sun, H. Dai, Q. Chen, and J. Liu, "Can-bus based distributed control system for hydraulic turbine blade repairing robot," in *Intelligent Robotics and Applications*, H. Liu, H. Ding, Z. Xiong, and X. Zhu, Eds. Springer Berlin Heidelberg, 2010, pp. 695–704.

[12] X. Yang, "The design of gluing robot control system based on can bus," *Advanced Materials Research*, vol. 1006–1007, pp. 627–630, Aug. 2014.

[13] D. Pan, L. Zhang, H. Peng, T. Liu, C. Xu, and X. Yue, "Design and test of a distributed control system of weeding robot based on multi-stm32 and can bus," *Journal of Physics: Conference Series*, vol. 2203, no. 1, p. 012019, 2022.

[14] D. Gomez-Ibanez, E. Stump, B. Grocholsky, V. Kumar, and C. J. Taylor, "The robotics bus: A local communications bus for robots," in *Proc. of SPIE Vol. 5609, Mobile Robots XVII*, D. W. Gage, Ed. SPIE, 2004, pp. 155–163.

[15] O. Nezamuddin, R. Bagwe, and E. D. Santos, "A real-time field bus architecture for multi-smart-motor servo system," *Scientific Reports*, vol. 14, no. 3918, 2024.

[16] A. Wendt and T. Schüppstuhl, "A solution to the generalized ros hardware io problem – a generic modbus/tcp device driver for plcs, sensors and actuators," in *2021 IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1–8.

[17] M. Migliavacca, A. Bonarini, and M. Matteucci, "Rtcan: A real-time can-bus protocol for robotic applications," Dipartimento di Elettronica, Informazione e Bioingegneria - DEIB, Politecnico di Milano, Milano, Italy, Tech. Rep., 2013.

[18] M. D. Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. New York, NY, USA: Springer, 2012.

[19] Y. Lv, W. Tian, and S. Yin, "Design and confirmation of a can-bus controller model with simple user interface," in *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control*, Beijing, China, 2015, pp. 640–644.

[20] T. S. Vamsi and G. R. L. V. N. S. Raju, "Arm-based industrial circuitous like robotic implementation using can bus for surveillance," in *Lecture Notes in Electrical Engineering*, 2018, vol. 434, pp. 263–273.

[21] W. Voss, *A comprehensible guide to controller area network*. Copperhill Media, 2008.

[22] T. Instruments, *Introduction to the Controller Area Network (CAN)*, SLOA101B, Aug. 2002; revised May 2016, 2002, online, Available: www.ti.com. [Online]. Available: https://www.ti.com

[23] A. M. Devices, *Actuonix L16 Datasheet*, Rev. E, November 2019, 2019, online, Available: https://bit.ly/42pBui4.

[24] F. Auto, *Feedback Rod Linear Actuator Specifications*, 2024, online, Available: https://www.firgelliauto.com/products/feedback-rod-actuator.

[25] *MG996R High Torque Metal Gear Dual Ball Bearing Servo*, 2024, online, Available: https://bit.ly/3DZXGWL.