

IA SOBRE EL PROTOCOLO CAN EN ROBÓTICA

Márquez Gutiérrez, Pedro Rafael; Baray Arana, Rogelio Enrique
Tecnológico Nacional de México/Tecnológico de Chihuahua
División de Estudios de Posgrado e Investigación
Av. Tecnológico #2909, Chihuahua, Chih., CP 31200,
Tel.: +52(614)201-200, Fax: +52(614)413-5187
{pedro.mg, rogelio.ba}@chihuahua.tecnm.mx

RESUMEN.

El Controllor Area Network (CAN) es un protocolo de comunicación en serie ampliamente utilizado en la industria automotriz e industrial. En el ámbito de la robótica, CAN ha encontrado aplicaciones cruciales debido a su robustez, fiabilidad, y capacidad para gestionar comunicaciones en tiempo real entre múltiples nodos, como controladores, sensores y actuadores. La aplicación de la inteligencia artificial (IA) en el contexto del CAN representa un avance significativo en el campo de las comunicaciones automotrices e industriales. La integración de la IA en los sistemas CAN aborda la creciente complejidad de las redes modernas. Este artículo plantea la aplicación de la IA al protocolo CAN en robótica, destacando su papel en la coordinación, control y mejoramiento de su eficiencia, fiabilidad y adaptabilidad en sistemas robóticos. Además, presenta un programa completo en Python que utiliza el aprendizaje automático para detectar anomalías en el tráfico CAN. El programa simula una red CAN, genera tráfico normal y anómalo, y utiliza un clasificador de aprendizaje automático para detectar estas anomalías. Palabras Clave: Controllor Area Network (CAN), robótica, machine learning, simulación en Python.

ABSTRACT.

The Controllor Area Network (CAN) is a serial communication protocol widely used in the automotive and industrial industries. In the field of robotics, CAN has found crucial applications due to its robustness, reliability, and ability to manage real-time communications between multiple nodes, such as controllers, sensors, and actuators. The application of artificial intelligence (AI) in the context of CAN represents a significant advance in the field of automotive and industrial communications. Integrating AI into CAN systems addresses the increasing complexity of modern networks. This article discusses the application of AI to the CAN protocol in robotics, highlighting its role in coordinating, controlling and improving its efficiency, reliability and adaptability in robotic systems. In addition, it presents a complete program in Python that uses machine learning to detect anomalies in CAN traffic. The program simulates a CAN network, generates normal and anomalous traffic, and uses a machine learning classifier to detect these anomalies.

Keywords: Controllor Area Network (CAN), robotics, machine learning, simulation in Python.

1. INTRODUCCIÓN

La robótica moderna demanda sistemas de comunicación que sean fiables, rápidos y capaces de operar en entornos ruidosos. El protocolo Controllor Area Network (CAN), desarrollado por

Bosch en la década de 1980, ha emergido como una solución ideal para estas necesidades debido a su capacidad para proporcionar comunicación en tiempo real y con alta integridad de datos. Aunque inicialmente diseñado para la industria automotriz, CAN ha sido adoptado en diversas áreas de la robótica, donde la precisión y la sincronización de las operaciones son críticas [2].

2. APLICACIONES DE CAN EN ROBÓTICA

2.1. Coordinación y Control de Movimientos.

CAN permite que los controladores se comuniquen con sensores y actuadores de manera eficiente, asegurando que las operaciones se sincronicen adecuadamente. Por ejemplo, en un brazo robótico, los motores responsables de mover las articulaciones deben recibir y procesar comandos en tiempo real, basados en la retroalimentación proporcionada por los sensores de posición y velocidad. CAN proporciona una plataforma robusta para esta comunicación, permitiendo que los movimientos sean suaves y precisos [3,4].

2.2. Vehículos Autónomos y Robótica Móvil.

Los vehículos autónomos y robots móviles dependen en gran medida de la integración y comunicación entre múltiples subsistemas, como la navegación, la detección de obstáculos y el control de movimiento. En estos sistemas, CAN es utilizado para asegurar que los datos críticos, como las lecturas de los sensores LIDAR y las señales de control de los motores, se transmitan de manera confiable y en tiempo real. Esto permite que el robot responda rápidamente a los cambios en su entorno, garantizando una operación segura y eficiente [5].

2.3. Sistemas Multi-Robot.

En sistemas que involucran múltiples robots trabajando en conjunto, como en entornos de fabricación automatizada o en misiones de búsqueda y rescate, CAN facilita la comunicación entre los robots. Esto permite la coordinación de tareas, la evitación de colisiones y la optimización del uso de recursos. Los robots pueden compartir información sobre su posición, estado y entorno a través de una red CAN [6,7].

2.4. Integración con Redes de Sensores.

Los robots a menudo dependen de redes de sensores distribuidos para recopilar información sobre su entorno. CAN se utiliza para

interconectar estos sensores con el controlador principal del robot, permitiendo una recopilación de datos eficiente y en tiempo real. Los datos obtenidos de los sensores, como la temperatura, presión, o proximidad, pueden ser utilizados por el robot para tomar decisiones informadas, ajustar su comportamiento, o responder a cambios en su entorno [1].

3. VENTAJAS DEL PROTOCOLO CAN EN ROBÓTICA

El protocolo CAN ofrece varias ventajas en aplicaciones robóticas:

- **Fiabilidad:** CAN proporciona mecanismos robustos para la detección y corrección de errores, lo que es crucial en entornos ruidosos y en aplicaciones donde la integridad de los datos es crítica [2].
- **Bajo Costo y Simplicidad:** La simplicidad del protocolo y su capacidad para reducir el cableado necesario entre los nodos contribuyen a reducir los costos y la complejidad del diseño de sistemas robóticos [4].
- **Escalabilidad:** CAN permite la fácil adición de nuevos nodos a la red, lo que es particularmente útil en sistemas robóticos que deben ser escalados o adaptados para nuevas tareas [7].

La aplicación de protocolos en robótica es un área multifacética que abarca varios métodos de comunicación, estrategias de control y marcos de interacción. Los protocolos sirven como columna vertebral para la comunicación efectiva entre los sistemas robóticos y sus operadores, así como entre múltiples robots en entornos colaborativos [6].

Una de las principales funciones de los protocolos de comunicación en robótica es permitir una interacción eficiente entre los robots y los operadores humanos. Por ejemplo, [28] destaca la implementación de un agente robótico cognitivo consciente diseñado para el trabajo en equipo en misiones de exploración lunar. Este agente utiliza "puntos de coordinación conversacional" para mejorar las interacciones entre el robot y los astronautas humanos, mejorando así los esfuerzos de colaboración en entornos desafiantes. Estos protocolos no solo agilizan la comunicación, sino que también mejoran las capacidades cognitivas de los robots, lo que les permite comprender y responder mejor a las órdenes humanas [10].

En entornos industriales se ha demostrado que la integración de protocolos con herramientas de software como MATLAB facilita el *servoing* visual predictivo en sistemas robóticos. [25] describe cómo las funciones de MATLAB controladas por parámetros pueden interactuar con robots industriales, lo que permite un control y una comunicación personalizados a través de protocolos establecidos. Esta integración es crucial para optimizar las operaciones robóticas en la fabricación y la automatización, donde el control preciso y la retroalimentación en tiempo real son esenciales para la eficiencia y la productividad.

Además, la elección de los protocolos de transporte tiene un impacto significativo en el rendimiento de los sistemas robóticos, especialmente en aplicaciones en tiempo real. [35] analiza el uso del Protocolo de Transporte en Tiempo Real (RTP) para el control remoto de sistemas robóticos a través de Internet, enfatizando sus ventajas sobre protocolos tradicionales como TCP y UDP en términos de latencia y confiabilidad. Esto es particularmente relevante en escenarios donde las respuestas oportunas son críticas, como la teleoperación y el monitoreo remoto de robots [31].

El protocolo XMPP ha surgido como una solución viable para la teleoperación de robots industriales, como lo demuestran Atmoko et al. La investigación [19] indica que XMPP ofrece una comunicación segura e interoperable, abordando los desafíos que plantean los entornos de red heterogéneos. Esta interoperabilidad es vital en la robótica moderna, donde los sistemas a menudo necesitan comunicarse a través de diferentes plataformas y tecnologías.

En el contexto de los sistemas multi-robot, ha utilizado el Protocolo de Red de Servicios de Robots (RSNP) para crear funciones de control remoto para los servicios de visitas al campus que utilizan robots. [26] ilustra cómo la RSNP facilita la comunicación y la coordinación entre múltiples robots, mejorando su capacidad para trabajar juntos de manera efectiva en entornos compartidos. Esta capacidad es cada vez más importante en aplicaciones como la logística automatizada y la fabricación colaborativa, donde varios robots deben operar en conjunto.

La flexibilidad de los protocolos de comunicación se ejemplifica aún más con el trabajo de [31], quien exploró el uso de múltiples protocolos de transporte para controlar sistemas robóticos basados en Internet. Sus hallazgos indican que los diferentes protocolos, incluidos TCP y UDP, tienen fortalezas y debilidades únicas, lo que sugiere que puede ser necesario un enfoque híbrido para optimizar el rendimiento en varios tipos de datos y condiciones operativas. Esta adaptabilidad es crucial para garantizar que los sistemas robóticos puedan funcionar eficazmente en entornos dinámicos e impredecibles.

El protocolo MQTT ha ganado terreno en el desarrollo de sistemas de control multi-robot debido a su ligereza y eficiencia en la transmisión de mensajes. En [21], la investigación destaca cómo MQTT reduce la redundancia de mensajes y simplifica la programación de sistemas multirobot, lo que la convierte en una opción atractiva para los desarrolladores. Esta reducción de la complejidad es esencial para escalar las aplicaciones robóticas, especialmente en escenarios que involucran numerosos agentes autónomos.

La seguridad es otro aspecto crítico de la comunicación robótica, especialmente en operaciones remotas. La implementación de medidas criptográficas en los dispositivos de comunicación basados en LoRa, como se discute en [29], ilustra la importancia de los protocolos seguros para proteger los datos intercambiados entre los vehículos terrestres no tripulados y sus operadores. A medida que la robótica continúa evolucionando, garantizar la

integridad y confidencialidad de la comunicación será primordial.

El desarrollo de protocolos inalámbricos ligeros, como el propuesto por [33], responde a la necesidad de comunicación de baja latencia en los sistemas telerrobóticos. Su implementación en sistemas embebidos demuestra cómo los protocolos especializados pueden mejorar la capacidad de respuesta y la eficiencia del control robótico, particularmente en aplicaciones que requieren retroalimentación en tiempo real. Este enfoque en minimizar los retrasos es crucial para mantener la eficacia operativa en tareas urgentes.

Además, el protocolo de enrutamiento AODV se ha mejorado para mejorar la transmisión de datos entre robots inteligentes en redes móviles ad hoc. [24] enfatiza la importancia de optimizar los protocolos de enrutamiento para reducir la pérdida de paquetes y mejorar el rendimiento en tiempo real, que es esencial para mantener una comunicación fluida en las redes robóticas. Estos avances son vitales para aplicaciones en las que los robots deben navegar por entornos complejos y comunicarse dinámicamente entre sí.

Los protocolos de interacción humano-robot (HRI) también desempeñan un papel importante en la mejora de la usabilidad y la eficacia de los sistemas robóticos. La investigación [23] combina métodos de neurociencia con protocolos HRI para mejorar el diseño de robots y la calidad de la interacción, aumentando así la validez ecológica de la investigación en neurociencia social y cognitiva. Este enfoque interdisciplinario subraya la importancia de comprender el comportamiento y la cognición humanos en el desarrollo de sistemas robóticos efectivos.

La integración de la tecnología de realidad virtual (RV) con IoT para el telefuncionamiento de robots industriales ha sido explorada por [27], quien descubrió que la RV puede mejorar significativamente la intuición y la capacidad de respuesta del control robótico. Esta integración no solo mejora el rendimiento del operador, sino que también abre nuevas vías para la capacitación y la simulación en aplicaciones robóticas.

El control de posicionamiento en células robóticas flexibles es otra área en la que los protocolos de comunicación desempeñan un papel fundamental. [32] demuestra cómo los mensajes de comunicación pueden optimizar la interacción entre los vehículos de guiado automático (AGV) y los componentes de las células robóticas, lo que mejora los tiempos de ciclo y la precisión de posicionamiento. Esta optimización es esencial para mejorar la eficiencia general de los sistemas robóticos en los entornos de fabricación.

El protocolo de comunicación CANopen se ha utilizado eficazmente para el control sincrónico de sistemas multimotor en robótica. [22] destacan sus ventajas para garantizar una transmisión estable y de alta velocidad, lo cual es crucial para aplicaciones que requieren un control preciso del motor. La capacidad de sincronizar múltiples motores es particularmente importante en sistemas robóticos complejos donde es necesario un movimiento coordinado.

En el contexto de la respuesta a desastres, la integración de los protocolos MQTT y CoAP ha sido evaluada por [34], revelando que MQTT ofrece velocidades de transferencia de datos superiores en plataformas de comunicación multi-robot. Este hallazgo es significativo para el desarrollo de estrategias de comunicación sólidas en escenarios de emergencia donde el intercambio rápido de información es crítico.

La síntesis de protocolos de autonomía para el control compartido en robótica, tal como se explora en [20], enfatiza la necesidad de una interacción efectiva entre humanos y robots. Al modelar estas interacciones como procesos de decisión de Markov, el estudio proporciona información sobre la optimización de escenarios de control compartido, que son cada vez más relevantes en la robótica colaborativa.

El uso de la comunicación TCP/IP para el control remoto de robots industriales ha sido demostrado por [30], quien presenta una interfaz fácil de usar para controlar robots sin requerir amplios conocimientos de programación. Esta accesibilidad es vital para expandir la usabilidad de los sistemas robóticos en diversas industrias.

4. ESTRUCTURA Y FUNCIONAMIENTO DE CAN

4.1. Topología y Arquitectura de Red

El protocolo CAN sigue una topología en bus, lo que significa que todos los nodos en la red están conectados a un solo cable que actúa como un canal de comunicación compartido. Esto permite que cada nodo se comunique con todos los demás nodos en la red sin necesidad de un controlador central. La simplicidad de esta topología contribuye a la eficiencia del sistema y a la reducción de costos de cableado.

- **Nodos:** Cada dispositivo o módulo en la red CAN se denomina nodo. Un nodo puede ser un microcontrolador, un sensor, un actuador, o cualquier dispositivo que se comunique en la red. Todos los nodos pueden transmitir y recibir mensajes, lo que permite una comunicación bidireccional.
- **Bus:** El bus CAN es el medio físico a través del cual se transmiten los mensajes. En la mayoría de las implementaciones, se utilizan dos cables (CAN_H y CAN_L) que forman un par trenzado, lo que ayuda a reducir las interferencias electromagnéticas (EMI).

4.2. Comunicación Basada en Mensajes

El protocolo CAN es un sistema de comunicación basado en mensajes, lo que significa que los datos se envían en paquetes denominados "mensajes" o "frames". A diferencia de los sistemas de comunicación punto a punto, donde se establece una conexión directa entre el emisor y el receptor, en CAN cualquier nodo puede enviar un mensaje que será recibido por todos los nodos conectados al bus. Es responsabilidad de cada nodo decidir si el mensaje es relevante y, por lo tanto, procesarlo.

- **Identificadores de Mensaje:** Cada mensaje en CAN contiene un identificador único que determina su

prioridad y contenido. Los identificadores más bajos tienen mayor prioridad, lo que es crucial en sistemas que requieren comunicación en tiempo real.

- **Campo de Control de Acceso:** Antes de enviar un mensaje, el nodo transmisor comprueba si el bus está libre. Si más de un nodo intenta transmitir al mismo tiempo, se lleva a cabo un proceso de arbitraje basado en los identificadores de los mensajes. El mensaje con el identificador más bajo gana el arbitraje y se transmite, mientras que los otros nodos esperan su turno.

4.3. Estructura del Mensaje CAN

Un mensaje CAN está compuesto por varios campos, cada uno con una función específica:

- **Campo de Inicio de Trama (SOF - Start of Frame):** Marca el inicio de un mensaje y permite la sincronización de todos los nodos en la red.
- **Campo de Identificador:** Contiene el identificador del mensaje, que define su prioridad en el bus. Puede ser de 11 bits (CAN estándar) o 29 bits (CAN extendido).
- **Campo de Control:** Indica el tipo de mensaje y la longitud del campo de datos.
- **Campo de Datos:** Contiene los datos a ser transmitidos, con una longitud de hasta 8 bytes.
- **Campo de Chequeo de Redundancia Cíclica (CRC):** Proporciona una verificación de errores para asegurar la integridad del mensaje.
- **Campo de Reconocimiento (ACK):** Permite al receptor confirmar la recepción del mensaje sin errores.
- **Campo de Fin de Trama (EOF - End of Frame):** Marca el final del mensaje.

4.4. Arbitraje y Prioridad de Mensajes

El proceso de arbitraje es uno de los aspectos más importantes del protocolo CAN. Cuando dos o más nodos intentan transmitir simultáneamente, el bus CAN resuelve el conflicto mediante un sistema de prioridad basado en el identificador del mensaje.

- **Arbitraje No Destructivo:** Durante el proceso de arbitraje, cada nodo transmite su identificador bit a bit. Si un nodo detecta que otro nodo está transmitiendo un bit dominante (0) mientras él está transmitiendo un bit recesivo (1), cede el control del bus. Este proceso es no destructivo, ya que el nodo perdedor puede reintentar la transmisión sin la necesidad de retransmitir los bits anteriores.
- **Prioridad:** El identificador del mensaje determina su prioridad. Los mensajes con identificadores más bajos tienen prioridad sobre los mensajes con identificadores más altos. Esto es crucial en sistemas donde algunos mensajes deben ser transmitidos con la menor latencia posible, como en los sistemas de frenos o control de motor en vehículos.

4.5. Detección y Corrección de Errores

CAN incluye mecanismos robustos para la detección y corrección de errores, lo que es esencial para su uso en entornos críticos como la automoción y la medicina [9].

- **Chequeo de Redundancia Cíclica (CRC):** Cada mensaje incluye un campo CRC que permite a los receptores detectar errores de transmisión.
- **Reconocimiento (ACK):** Después de recibir un mensaje sin errores, el receptor envía un bit de reconocimiento al transmisor. Si no se recibe el ACK, el transmisor sabe que debe retransmitir el mensaje.
- **Detección de Errores:** CAN puede detectar varios tipos de errores, incluyendo errores de bit, errores de bit *stuffing*, errores de formato, y errores de ACK. Cuando se detecta un error, el nodo que lo detecta envía una trama de error y todos los nodos desestiman el mensaje erróneo.
- **Corrección de Errores:** Aunque CAN es capaz de detectar errores, no corrige errores en la transmisión. En su lugar, el mensaje con errores se descarta y se solicita una retransmisión.

4.6. Modos de Funcionamiento de CAN

El protocolo CAN puede operar en diferentes modos, adaptándose a las necesidades de las aplicaciones específicas.

- **Modo Normal:** El modo más común donde los nodos pueden transmitir y recibir mensajes.
- **Modo de Escucha Única (Listen-Only):** En este modo, el nodo solo escucha los mensajes del bus y no participa en el arbitraje ni en la transmisión. Es útil para la monitorización de tráfico CAN sin interferir en la red.
- **Modo de Bucle Cerrado (Loopback):** Utilizado para pruebas, este modo permite que un nodo CAN escuche sus propias transmisiones sin interferir en el bus.

5. CRITERIOS PARA SELECCIONAR UN LENGUAJE DE PROGRAMACIÓN

La elección del lenguaje de programación adecuado para implementar el protocolo CAN en aplicaciones robóticas depende de varios factores, incluyendo:

- **Soporte para Comunicación en Tiempo Real:** La capacidad del lenguaje para manejar la comunicación en tiempo real es crucial, dado que los sistemas robóticos dependen de la sincronización precisa de los mensajes entre los nodos.
- **Acceso a Bajo Nivel:** Dado que CAN es un protocolo de comunicación en el nivel de hardware, el lenguaje debe permitir el acceso a bajo nivel para interactuar con los controladores CAN y los buses de comunicación.
- **Rendimiento y Eficiencia:** Los sistemas robóticos a menudo requieren alta eficiencia y rendimiento, por lo

que el lenguaje debe ser capaz de ejecutar tareas críticas en tiempo real sin introducir latencias significativas.

- **Portabilidad y Soporte en Plataformas Integradas:** El lenguaje debe ser compatible con microcontroladores y plataformas integradas, donde CAN es más comúnmente implementado.
- **Facilidad de Desarrollo y Mantenimiento:** La disponibilidad de bibliotecas, herramientas de desarrollo y la facilidad para escribir y mantener el código también son consideraciones importantes. Figura 1.

Lenguaje	Ventajas	Desventajas
C/C++	Acceso a bajo nivel, alto rendimiento, portabilidad, soporte extenso para CAN	Complejidad, manejo manual de memoria
Python	Facilidad de uso, abundancia de bibliotecas, desarrollo rápido	Rendimiento, acceso limitado a bajo nivel
Rust	Seguridad, manejo de concurrencia, alto rendimiento	Curva de aprendizaje, ecosistema menos maduro
Java	Portabilidad, seguridad	Rendimiento, acceso limitado a hardware

Figura 1. Comparación de Lenguajes

5.1. Prueba de Concepto.

Para demostrar la ventaja de integrar la Inteligencia Artificial (IA) con el protocolo CAN, escribimos un programa completo en Python que utiliza el aprendizaje automático para detectar anomalías en el tráfico CAN. El programa simula una red CAN, genera tráfico normal y anómalo, y luego utiliza técnicas de IA (en este caso, un clasificador de aprendizaje automático) para detectar las anomalías.

También incorporamos análisis estadísticos para mostrar el rendimiento del modelo de IA. Procedimiento:

1. Simular datos de tráfico CAN normales y anómalos. Figura 2.
2. Utilizar un algoritmo de aprendizaje automático (por ejemplo, Random Forest) para detectar anomalías. Figura 3.
3. Realizar análisis estadísticos para evaluar el rendimiento del modelo. Figura 4.
4. Graficar los resultados para una mejor interpretación.

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

# Simulate CAN network traffic
def simulate_can_data(num_samples=1000):
    np.random.seed(42)

    # Generate random CAN IDs and data
    can_ids = np.random.randint(0, 2048, size=num_samples) # CAN IDs range from 0 to 2047
    data_bytes = np.random.randint(0, 255, size=(num_samples, 8)) # Each CAN message has 8 bytes

    # Normal traffic: CAN IDs between 0 and 1500, and data between 0 and 200
    normal_traffic = (can_ids < 1500) & (data_bytes.max(axis=1) < 200)

    # Anomalous traffic: CAN IDs > 1500 or data bytes with values > 200
    anomalous_traffic = ~normal_traffic

    labels = np.where(normal_traffic, 0, 1) # 0 for normal, 1 for anomaly

    # Combine into a DataFrame
    data = pd.DataFrame(data_bytes, columns=[f'data_byte_{i}' for i in range(8)])
    data['can_id'] = can_ids
    data['label'] = labels

    return data
```

Figura 2. Simulación de Tráfico

```
# Generate simulated CAN data
data = simulate_can_data()

# Split into features and labels
X = data.drop(columns=['label'])
y = data['label']

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train a Random Forest Classifier to detect anomalies in CAN traffic
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the performance
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy: ", accuracy_score(y_test, y_pred))

# Plot confusion matrix
def plot_confusion_matrix(cm, classes):
    plt.figure(figsize=(6,6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.show()
```

Figura 3. Aprendizaje de Máquina

```
# Generate confusion matrix plot
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, classes=['Normal', 'Anomaly'])

# Statistical results
normal_data = data[data['label'] == 0]
anomalous_data = data[data['label'] == 1]

print("\nStatistical Summary of Normal CAN Traffic:")
print(normal_data.describe())

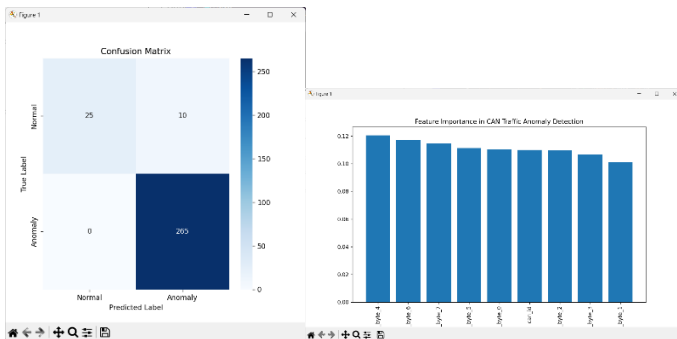
print("\nStatistical Summary of Anomalous CAN Traffic:")
print(anomalous_data.describe())

# Feature importance plot
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]
features = X.columns

plt.figure(figsize=(10,6))
plt.title("Feature Importance in CAN Traffic Anomaly Detection")
plt.bar(range(X.shape[1]), importances[indices], align='center')
plt.xticks(range(X.shape[1]), [features[i] for i in indices], rotation=90)
plt.show()

# Predicting new samples
new_samples = np.array([[1502, 202, 200, 210, 215, 220, 230, 240, 2048], # Anomalous sample
                        [100, 100, 100, 100, 100, 100, 100, 100, 1000]]) # Normal sample
new_predictions = model.predict(new_samples)
print("\nNew Sample Predictions: ", new_predictions)
```

Figura 4. Análisis Estadístico



```
Statistical Summary of Normal CAN Traffic:
data_byte_0 data_byte_1 data_byte_2 ... data_byte_7 can_id label
count 111.000000 111.000000 111.000000 ... 111.000000 111.000000 111.0
mean 97.756757 108.747748 98.009009 ... 96.243243 812.711712 0.0
std 59.819914 54.974948 59.357238 ... 56.210354 426.655245 0.0
min 0.000000 0.000000 2.000000 ... 7.000000 1.000000 0.0
25% 48.500000 62.500000 41.000000 ... 51.500000 412.000000 0.0
50% 92.000000 117.000000 100.000000 ... 93.000000 866.000000 0.0
75% 152.000000 151.000000 151.000000 ... 144.500000 1185.500000 0.0
max 198.000000 199.000000 197.000000 ... 195.000000 1496.000000 0.0

[8 rows x 10 columns]

Statistical Summary of Anomalous CAN Traffic:
data_byte_0 data_byte_1 data_byte_2 ... data_byte_7 can_id label
count 889.000000 889.000000 889.000000 ... 889.000000 889.000000 889.0
mean 129.888639 131.089989 132.268841 ... 129.040495 1085.479190 1.0
std 73.515534 73.508662 74.307333 ... 75.076227 575.206815 0.0
min 1.000000 0.000000 0.000000 ... 0.000000 1.000000 1.0
25% 68.000000 68.000000 69.000000 ... 63.000000 632.000000 1.0
50% 131.000000 137.000000 135.000000 ... 134.000000 1109.000000 1.0
75% 192.000000 196.000000 201.000000 ... 192.000000 1588.000000 1.0
max 254.000000 254.000000 254.000000 ... 254.000000 2047.000000 1.0

[8 rows x 10 columns]

New Sample Predictions: [1 0]
```

```
Confusion Matrix:
[[ 25  10]
 [  0 265]]

Classification Report:

              precision    recall  f1-score   support

     0       1.00      0.71      0.83         35
     1       0.96      1.00      0.98        265

 accuracy          0.97         300
 macro avg          0.98         300
weighted avg          0.97         300

Accuracy: 0.9666666666666667
```

Figura 5. Graficación de Resultados

6. CONCLUSIONES

El protocolo CAN es una tecnología clave en la robótica moderna, proporcionando una comunicación fiable y en tiempo real entre los diversos componentes de un sistema robótico. Su capacidad para manejar la comunicación en entornos ruidosos y su facilidad de integración han hecho que sea ampliamente adoptado en aplicaciones robóticas críticas. A medida que la robótica avanza hacia sistemas más complejos y autónomos, el papel de CAN en asegurar la comunicación efectiva y la operación sincronizada de estos sistemas seguirá siendo esencial. La elección del mejor lenguaje de programación para implementar el protocolo CAN en aplicaciones robóticas depende en gran medida del contexto específico de la aplicación. C/C++ se destaca como la opción más robusta y confiable para la mayoría de las aplicaciones robóticas, gracias a su rendimiento, acceso a bajo nivel, y extensa compatibilidad con plataformas embebidas. Python es ideal para prototipos rápidos y desarrollo de aplicaciones de alto nivel, pero su uso en sistemas críticos de tiempo real está limitado por su rendimiento. Rust ofrece un compromiso interesante con su enfoque en la seguridad y el rendimiento, aunque aún no ha alcanzado la madurez de C/C++. Java puede ser útil en sistemas más complejos que requieren integración con otros servicios, pero es menos adecuado para tareas críticas en tiempo real.

La aplicación de protocolos en robótica es esencial para mejorar la comunicación, el control y la interoperabilidad entre los sistemas robóticos. Desde la automatización industrial hasta la teleoperación y la coordinación de múltiples robots, los medicinas protocolos desempeñan un papel fundamental para garantizar que los robots puedan operar de manera efectiva en diversos entornos. A medida que la tecnología robótica continúa avanzando, el desarrollo y el refinamiento de los protocolos de comunicación seguirán siendo un área de enfoque clave, impulsando la innovación y mejorando las capacidades de los sistemas robóticos.

7. REFERENCIAS.

- [1] Guan, T., Liu, J., Zhang, X., & Li, Y. (2018). A real-time control system based on CAN bus for autonomous mobile robots. *Journal of Intelligent & Robotic Systems*, 91(1), 113-129.
- [2] Hartwich, F., Bosch, R., & Eitzold, T. (1999). CAN protocol implementation and its applications in robotic control. *IEEE Transactions on Robotics and Automation*, 15(5), 812-818.
- [3] Hoffmann, G., Tomizuka, M., & Paden, B. (2005). Distributed control for autonomous robot teams using CAN bus. *Proceedings of the IEEE Conference on Decision and Control*, 4391-4396.
- [4] Laukemann, D., Menzel, T., & Stein, D. (2017). CAN bus-based robotic control system with real-time capabilities. *International Journal of Advanced Robotic Systems*, 14(1), 1-9.
- [5] Li, S., Zhang, X., & Yang, J. (2019). Design and application of CAN-based communication for mobile robots. *IEEE/ASME Transactions on Mechatronics*, 24(4), 1674-1683.
- [6] Nerger, M., Ewert, A., & König, M. (2020). CAN communication in multi-robot systems for cooperative tasks. *Robotics and Autonomous Systems*, 131, 103589.
- [7] Zhang, L., Li, H., & Liu, J. (2016). Real-time communication for multi-agent systems based on CAN protocol. *Procedia Computer Science*, 92, 418-424.
- [8] Alkhatri, M. (2024). The role of artificial intelligence in addressing cybersecurity challenges. *International Journal of Academic Research in Business and Social Sciences*, 14(4). <https://doi.org/10.6007/ijarbss/v14-i4/21312>
- [9] Briganti, G. and Moine, O. (2020). Artificial intelligence in medicine: today and tomorrow. *Frontiers in Medicine*, 7. <https://doi.org/10.3389/fmed.2020.00027>
- [10] Chen, M., Miao, Y., Jian, X., Wang, X., & Humar, I. (2019). Cognitive-lpwan: towards intelligent wireless services in hybrid low power wide area networks. *Ieee Transactions on Green Communications and Networking*, 3(2), 409-417. <https://doi.org/10.1109/tgcn.2018.2873783>
- [11] Garg, A. (2017). Blockchain for artificial intelligence. *International Journal for Research in Applied Science and Engineering Technology*, V(XI), 2414-2417. <https://doi.org/10.22214/ijraset.2017.11336>
- [12] Latah, M. and Toker, L. (2019). Artificial intelligence enabled software-defined networking: a comprehensive overview. *Iet Networks*, 8(2), 79-99. <https://doi.org/10.1049/iet-net.2018.5082>
- [13] Li, Z., Zhao, Y., Li, Y., Liu, M., Zeng, Z., Xin, X., ... & Zhang, J. (2020). Self-optimizing optical network with cloud-edge collaboration: architecture and application. *Ieee Open Journal of the Computer Society*, 1, 220-229. <https://doi.org/10.1109/ojcs.2020.3030957>
- [14] Lin, B. (2021). Toward an ai-enabled o-ran-based and sdn/nfv-driven 5 g & iot network era. *Network and Communication Technologies*, 6(1), 6. <https://doi.org/10.5539/nct.v6n1p6>
- [15] Rafique, D. and Velasco, L. (2018). Machine learning for network automation: overview, architecture, and applications [invited tutorial]. *Journal of Optical Communications and Networking*, 10(10), D126. <https://doi.org/10.1364/jocn.10.00d126>
- [16] Russia, E. (2020). Standardization of artificial intelligence for the development and use of intelligent systems. *Advances in Wireless Communications and Networks*, 6(1), 1. <https://doi.org/10.11648/j.awcn.20200601.11>
- [17] Samdanis, K., Abbou, A., Song, J., & Taleb, T. (2023). Ai/ml service enablers and model maintenance for beyond 5g networks. *Ieee Network*, 37(5), 162-172. <https://doi.org/10.1109/mnet.129.2200417>
- [18] Sun, Y., Peng, M., Zhou, Y., Huang, Y., & Mao, S. (2019). Application of machine learning in wireless networks: key techniques and open issues. *Ieee Communications Surveys & Tutorials*, 21(4), 3072-3108. <https://doi.org/10.1109/comst.2019.2924243>
- [19] Atmoko*, R., Yang, D., Abas, M., Arfianto, A., & Rahim, R. (2019). Teleoperation cloud industrial robot using xmpp protocol. *International Journal of Recent Technology and Engineering*, 8(3), 6280-6284. <https://doi.org/10.35940/ijrte.c5893.098319>
- [20] Cubuktepe, M., Jansen, N., Alshiekh, M., & Topcu, U. (2021). Synthesis of provably correct autonomy protocols for shared control. *Ieee Transactions on Automatic Control*, 66(7), 3251-3258. <https://doi.org/10.1109/tac.2020.3018029>
- [21] Guo, Z. (2023). Design and implementation of multiple mobile robot control system based on mqtt. <https://doi.org/10.3233/atde230507>
- [22] Hung, C., Lee, R., Huang, B., & Yu, S. (2019). Multi-motor synchronous control with canopen. *Journal of Robotics Networking and Artificial Life*, 5(4), 236. <https://doi.org/10.2991/jrnal.k.190220.006>
- [23] Kompatsiari, K., Perez-Osorio, J., Tommaso, D., Metta, G., & Wykowska, A. (2018). Neuroscientifically-grounded research for improved human-robot interaction. <https://doi.org/10.31234/osf.io/98hcx>
- [24] Kong, X., Wang, F., & Cai, X. (2018). Improvement and simulation of aodv routing algorithm based on ns-3. *Destech Transactions on Computer Science and Engineering*, (ceic). <https://doi.org/10.12783/dtcsc/ceic2018/24564>
- [25] Krishnan, M., Vijayan, A., & Ashok, S. (2020). Interfacing an industrial robot and matlab for predictive visual servoing. *Industrial Robot the International Journal of Robotics Research and Application*, 48(1), 110-120. <https://doi.org/10.1108/ir-05-2020-0100>
- [26] Kusu, T., Takahashi, M., Nomoto, Y., Ito, Y., Tsuchiya, Y., Narita, M., ... & Kato, Y. (2013). An rpg-like campus tour service using remote control robots. <https://doi.org/10.1109/iecon.2013.6700516>
- [27] Luu, T. (2024). Effectiveness of iot and vr for real-time teleoperation of industrial robots. <https://doi.org/10.21203/rs.3.rs-3812859/v1>
- [28] Lyell, M. and Drozd, W. (2010). Protocol-aware, enhanced cognition robot agent design for team work effort in lunar exploration missions. <https://doi.org/10.1145/1967112.1967116>
- [29] Manuel, M. and Daimi, K. (2021). Implementing cryptography in lora based communication devices for unmanned ground vehicle applications. *Sn Applied Sciences*, 3(4). <https://doi.org/10.1007/s42452-021-04377-y>
- [30] Oltean, S. (2023). Remote user interface for abb irb120 robot using tcp/ip communication. *Acta Marisiensis Seria Technologica*, 20(2), 49-56. <https://doi.org/10.2478/amset-2023-0018>
- [31] Phung, M., Tran, T., Nguyen, T., & Tran, Q. (2012). Control of internet-based robot systems using multi transport protocols. <https://doi.org/10.1109/iccais.2012.6466605>
- [32] Spirileanu, C., Vladareanu, L., & Deng, M. (2015). Positioning control through command messages inside a flexible robotic cell with agv. <https://doi.org/10.1109/cscs.2015.29>
- [33] Toker, O. (2013). Lightweight wireless protocol based on ieee 802.11 for delay sensitive telerobotic systems. *Turkish Journal of Electrical Engineering & Computer Sciences*. <https://doi.org/10.3906/elk-1110-9>
- [34] Yamin, M., Kuswadi, S., & Sukaridhoto, S. (2018). Real performance evaluation on mqtt and coap protocol in ubiquitous network robot platform (unrpf) for disaster multi-robot communication. *Emitter International Journal of Engineering Technology*, 6(2), 369-385. <https://doi.org/10.24003/emitter.v6i2.305>
- [35] Zuiiev, A. (2024). Research of microprocessor device and software for remote control of a robotic system. *Technology Audit and Production Reserves*, 1(2(75)), 31-37. <https://doi.org/10.15587/2706-5448.2024.297339>