# Comparativo de software de programación de controladores lógicos programables en la implementación de redes neuronales de la función lógica And.

## Comparative article on programmable logic controller programming software for implementing the logical and function with neural networks.

Héctor Manuel González Cortés* (1).
Universidad Politécnica de Tlaxcala UPTx.
hectormanuel.gonzalez@uptlax.edu.mx.

Alejandro Medina Santiago (2). Instituto Nacional de Astrofísica, Óptica y Electrónica, amedina@inaoep.mx.

Joel Castro Ramírez (3). Universidad Politécnica de Tlaxcala UPTx, joel.castro@uptlax.edu.mx.

Jorge Antonio Orozco Torres (4). TecNM Campus Tuxtla, jorge.ot@tuxtla.tecnm.mx.

Julio Rodríguez González (5). Universidad Politécnica de Tlaxcala UPTx, julio.gonzalez@uptlax.edu.mx.

*corresponding author.

**Artículo recibido en septiembre 16, 2025; aceptado en octubre 26, 2025.**

## Resumen.

*Este trabajo compara distintos softwares de programación para Controladores Lógicos Programables (PLCs) como: U90 Ladder, RS Logix 500, Codesys y Tia Portal, con el fin de explorar la implementación de redes neuronales artificiales (ANN) en entornos industriales. Se utilizó como caso de estudio la función lógica AND, para la cual se entrenó previamente un perceptrón simple en MATLAB, aprovechando que este problema es linealmente separable y puede resolverse con la función de activación hardlim(n). El objetivo central fue analizar cómo cada entorno de programación permite traducir la ecuación del perceptrón a lenguaje de escalera, observando para ello la estructura del programa, la organización de las instrucciones y la variedad de operaciones disponibles. Los resultados demuestran que, si bien estas plataformas no fueron diseñadas originalmente para inteligencia artificial, es posible implementar modelos básicos de RNA utilizando operaciones aritméticas y lógicas inherentes a los PLCs, como sumas, restas, multiplicaciones y comparaciones. Este estudio refleja las posibilidades emergentes de integrar conceptos de IA en la automatización industrial convencional, abriendo camino al uso de redes neuronales en aplicaciones donde el aprendizaje en tiempo real no es un requisito, pero la ejecución confiable y en plataformas robustas sí lo es.*

**Palabras claves:** ANN, comparativo, lenguaje escalera, perceptrón, PLC.

## Abstract.

*Comparison of the following Programmable Logic Controller (PLC) programming software: U90 Ladder, RS Logix 500, Codesys, and Tia Portal, in the implementation of Artificial Intelligence (AI) using Artificial Neural Networks (ANN). This implementation features a non-real-time learning, pre-trained And function with linearly separable datasets created in Matlab. Since the datasets are linearly separable, using a perceptron topology with the hardlim(n)*

*activation function, we will be able to observe the structure of the programming lines in Ladder Logic across the different software platforms and compare the variety of their organizational structures and instructions. This work demonstrates methods for working with neural networks, basic functions, and industrial automation, as it is an emerging application area. It proves that it is possible to implement the concepts of AI and ANN models on industrial platforms that were not originally designed for such purposes. Although these platforms are capable of performing basic operations—such as addition, subtraction, multiplication, division, and comparison—these functions are sufficient for working with basic ANNs.*

**Keywords:** ANN, comparative, ladder logic, perceptron, PLC.

## 1. Introducción.

En el entorno de programación de Controladores Lógicos Programables (PLC) y la implementación de Inteligencia Artificial (IA) en medios industriales el desafío es innegable. En el presente trabajo se busca plantear la posibilidad de que la IA pueda ser parte de la programación de PLCs, en este caso, con la Función Lógica Booleana, como la función And, demostrando que, aunque las líneas de programación en lenguaje de escalera aumentan es posible implementarse. El entrenamiento de la red neuronal se basa en el modelo más simple, como es el perceptrón, que resuelve problemas básicos linealmente separables, como son las funciones que analizamos, limitando la complejidad del entrenamiento en tiempo real que los PLCs no podrían manejar con las instrucciones básicas como la aritmética básica.

El principal objetivo de este artículo es comparar cuatro softwares de programación de PLC diferentes como son: U90 Ladder, RS Logix 500, Codesys y Tia portal en la forma que se puede crear, compilar y correr un programa que contenga la ejecución de la ecuación del perceptrón $\sum \quad wi * pi + b$ y la función de activación hardlim en lenguaje de escalera. La comparación radica en la estructura del programa, instrucciones usadas y la capacidad de los programas para manejar las operaciones del perceptron.

La capacidad de cálculo de los PLCs modernos permite ir más allá de la lógica booleana simple. Existen esfuerzos documentados para implementar estrategias de control sofisticadas directamente en estos dispositivos. Un ejemplo clave es el desarrollo de controladores avanzados, como los basados en lógica difusa o modelos predictivos, que pueden ser programados en lenguajes de PLC estándar (García Jaimes & Arroyave, 2012). Nuestro trabajo se fundamenta en estas premisas, pero se distingue al aplicar por primera vez en este contexto el formalismo de las redes neuronales artificiales para emular una función lógica básica mediante un análisis comparativo entre diferentes plataformas.

## 2. Métodos.

**Procedimiento.**

Diseño de la Red Neuronal y Obtención de Parámetros.

El modelo del algoritmo a implementar fue un perceptrón simple con dos entradas ($p_1$, $p_2$), correspondientes a los operandos de la función lógica, y una salida binaria. La función de activación seleccionada fue hardlim(n), donde la salida es 1 si la entrada neta "$n \geq 0$", y 0 en caso de tener valores negativos a la salida del cuerpo de la neurona artificial. La entrada neta "n" se calcula mediante la ecuación:

$$\sum \quad wi * pi + b; \quad \text{……. Ecuación 1}$$

en caso específico de la función AND podría manejarse como un caso general de dos entradas y una salida con datos linealmente separables es:

$$n = w1 * p1 + w2 * p2 + b \quad \text{…….. Ecuación 2}$$

Los parámetros de la red se obtuvieron mediante un entrenamiento externo en MATLAB. A diferencia de enfoques que utilizan MATLAB para programación de alto nivel en PLCs (Páez-Logaira et al., 2015), este trabajo empleó el software exclusivamente como herramienta de cálculo para el entrenamiento, implementando posteriormente el algoritmo en el lenguaje nativo de cada PLC.

Para la función AND $w_1$=2 y $w_2$=1 son los pesos sinápticos y b=-3 es el bias, tenemos:

$$n = p1 * 2 + p2 * 1 + (-3) \text{ ..... Ecuación 3}$$

Selección de Softwares y Plataforma.

El estudio se llevó a cabo en cuatro entornos de desarrollo para PLCs, seleccionados por su representatividad en el mercado industrial y su diversidad de enfoques:

U90 Ladder: Representante de PLCs de gama baja-micro (Jazz JZ20R10). En la figura 1 se muestra la primera parte de programación donde se tratan las 2 entradas para la función AND con la instrucción ST, Store Direct, le permite escribir un valor constante, MI o SI en otro MI o SI, con ella se logra asignar el valor de alto o bajo voltaje a un valor constante de cero o uno, en la línea 1 y 2 de la red 3 de la fig. 1 se trata la señal de entrada 1 en ambos voltajes 24V y 0V, en la primera linea se usa una memoria "MI" con dirección cero, entrada A, y valor 1, asi mismo en la linea 2 se asigna a la misma memoria "MI" dirección cero el valor 0; lo que permite el uso de instrucciones de aritmética para el cálculo de la ecuación del entrenamiento de la red neuronal; logrando que la instrucción ST asigne un 1 o 0 a la variable MI 0 P1_NUM; asi mismo con la entrada 2 en la red 4.

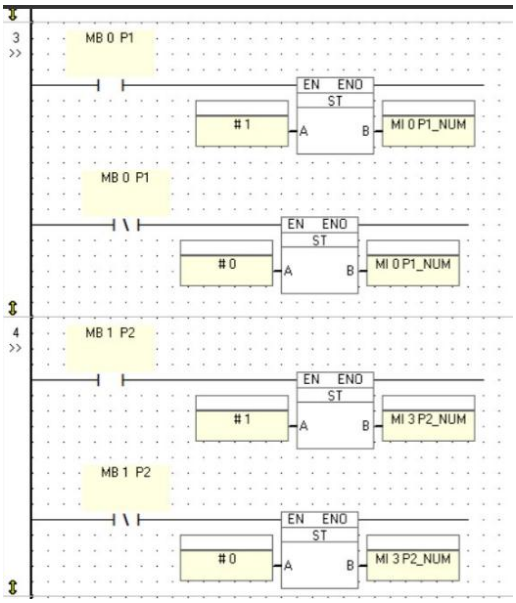



**Figura 1.** Interfaz de desarrollo U90 Sección 1.

En la Figura 2 muestra la red 5, donde se realiza el cálculo de la red neuronal de la función AND por medio de operaciones aritméticas de multiplicación, suma y resta. En la red 6 de la misma figura la instrucción de comparación termina la ejecución.

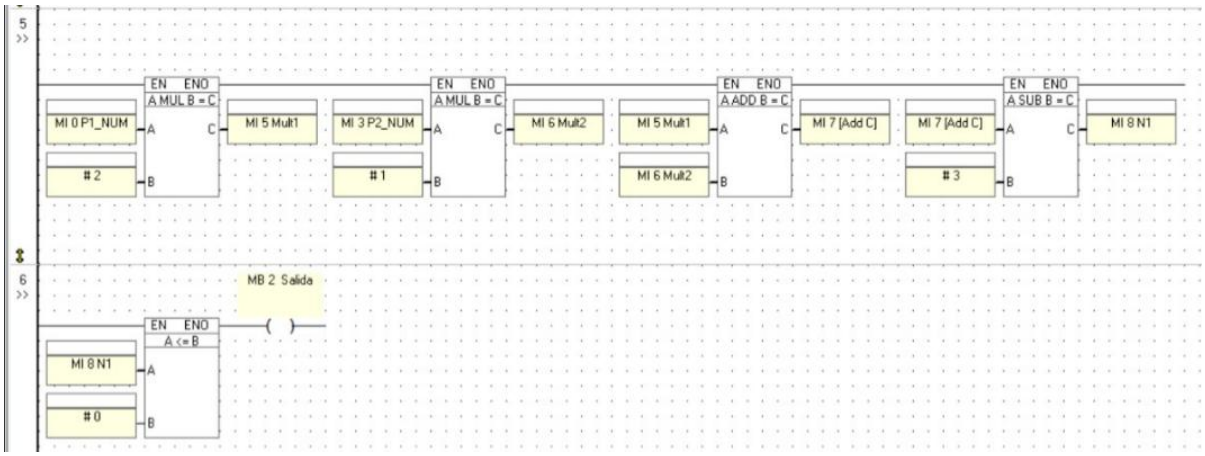



**Figura 2.** Interfaz de desarrollo U90 Sección 2.

RSLogix 500: Entorno clásico para PLCs de gama media (MicroLogix, SLC-500) de Allen-Bradley.

En la figura 3 y 4 esta la primera sección del programa de la misma función AND; donde se procesan las 2 señales de entrada y su conversión a valores constantes con 0 y 1. En la instrucción MOV la dirección de memoria a la que se va a copiar el valor de la fuente o valor constante en nuestro caso, la dirección de destino N7:1 que copia el valor a un entero.


**Figura 3.** Interfaz de desarrollo Rslogix 500 Sección 1.

En la figura 4 en la linea 4 y 5 se ejecutan las multiplicaciones de la fórmula de la red neuronal.


**Figura 4.** Interfaz de desarrollo Rslogix 500 Sección 2.

En la figura 5 en la línea 6 se desarrolla la suma y en la linea 7 la resta o suma del bias; y por último en la linea 8 se realiza la comparación de la función de transferencia hardlim donde el cálculo de la ecuación se compara (mayor o igual que cero), prendiendo o apagando la salida virtual del PLC B3:0.0.


**Figura 5.** Interfaz de desarrollo Rslogix 500 Sección 3.

CODESYS: Entorno independiente del fabricante, conocido por su flexibilidad y programación basada en objetos.

En la imagen 6 estan las líneas de programación para convertir las entradas digitales en valores constantes, en la linea 0001 y 0002 la entrada I0 toma valores de 1 y 0 cuando cambia de alta a bajo usando la instrucción “MOVE”; en las líneas 0003 y 0004 se hace lo conveniente para la entrada I1, los cuales se guardan en las variables entras IN1 y IN2 respectivamente.


**Figura 6.** Interfaz de desarrollo Codesys Sección 1.

En la figura 7 la linea 0005 se desarrolla el calculo de la formula de la red neuronal con las instrucciones de multiplicación y suma, guardando los resultados en las variables MUL1, MUL2, y N1. En la linea 0006 se ejecuta la función de transferencia Hardlim comparando N1 con 0. De esta forma cuando se cumple en la linea 0007 se activa la variable VB cerrando el contacto y energizando la salida OVB. Finalizando la ejecución de la red neuronal.

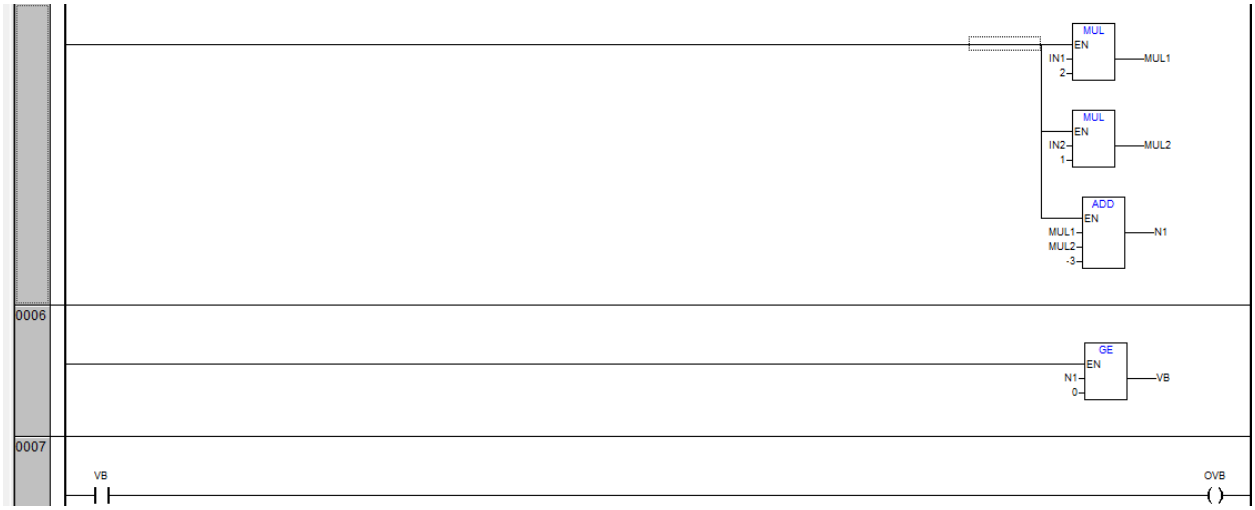



**Figura 7.** Interfaz de desarrollo Codesys Sección 2.

TIA Portal (V17): Entorno integrado moderno para PLCs de gama alta (SIMATIC S7-1200/1500) de Siemens.

En la figura 8 y 9 se encuentran las líneas de programación para el tratamiento de las señales de entrada, se puede ver que las instrucciones usadas tambien es la instrucción “MOVE” al igual que los software Rslogix 500 y Codesys.

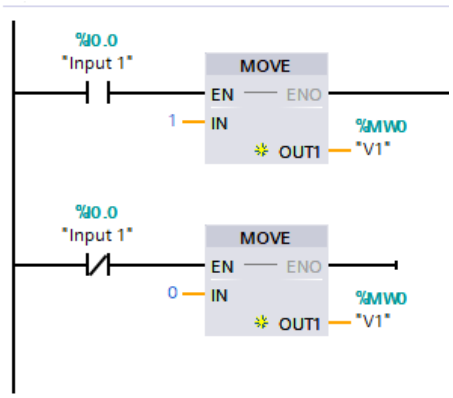



**Figura 8.** Interfaz de desarrollo Tia portal Sección 1

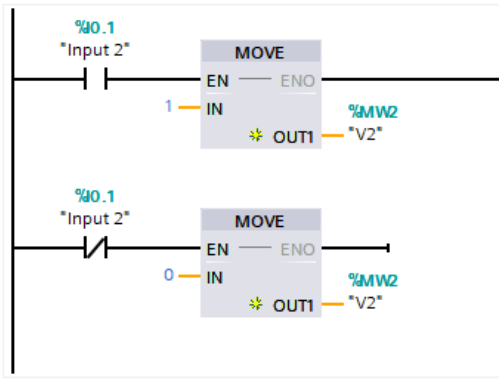



**Figura. 9** Interfaz de desarrollo Tia portal Sección 2

En la figura 10 estan las instrucciones para ejecutar la fórmula de la red neuronal para obtener el valor de N1.

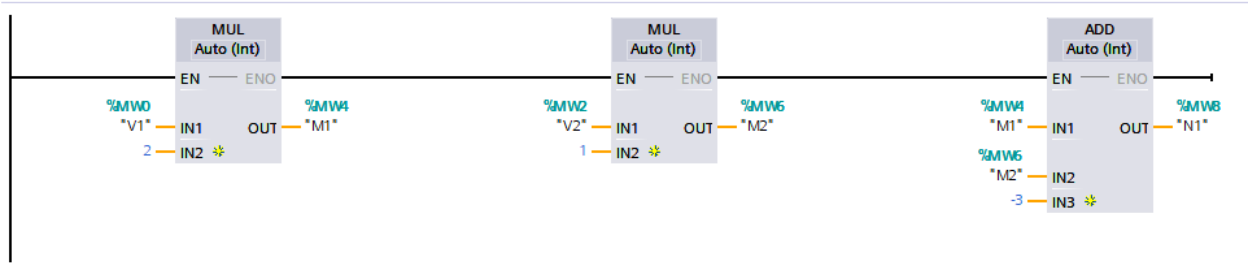



**Figura 10.** Interfaz de desarrollo Tia portal Sección 3.

En la figura 11 finalmente tenemos la linea de programación donde se aplica de función de activación Hardlim realizando la comparación “mayor o igual”.

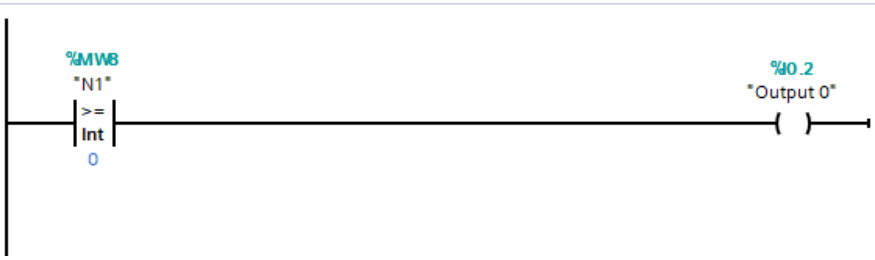



**Figura 11.** Interfaz de desarrollo Tia portal Sección 4.

Para garantizar una comparación justa, el algoritmo se implementó en modelos de PLC dentro de cada software que fueran capaces de ejecutar operaciones aritméticas básicas con números reales (coma flotante).

Procedimiento de Implementación y Prueba.

El procedimiento experimental consistió en las siguientes etapas para cada uno de los cuatro softwares:
Implementación: Se programó la ecuación del perceptrón en lenguaje de escalera.
Esto implicó: Declarar variables para las entradas ($p_1$, $p_2$), la salida, y los parámetros ($w_1$, $w_2$, b).
Utilizar instrucciones aritméticas (MUL, ADD, SUB) para calcular la entrada neta *n*.

Implementar la función hardlim mediante una instrucción de comparación (GEQ, LIM, etc.) que activara la salida coil si $n \geq 0$.

Verificación Funcional: Se sometió cada programa implementado a las cuatro combinaciones posibles de entradas binarias (00, 01, 10, 11). Se verificó que el estado de la salida coincidiera exactamente con la tabla de verdad esperada para la función lógica objetivo (AND), confirmando la corrección de la implementación.

Criterios de Análisis Comparativo.

Tras verificar el funcionamiento correcto en todos los softwares, se realizó un análisis cualitativo comparativo de las soluciones programadas, basado en los siguientes criterios derivados de la observación directa del código:

Complejidad Estructural: Número de rieles (rungs) o segmentos de código necesarios para implementar la lógica completa.

Diversidad de Instrucciones: Tipo y cantidad de instrucciones específicas utilizadas (ej.: bloques matemáticos dedicados vs. instrucciones básicas combinadas).

Elegancia y Claridad del Código: Legibilidad, facilidad de seguimiento lógico y mantenibilidad del programa en escalera.

Requerimientos de Programación: Necesidad de utilizar bloques de función personalizados (FB), funciones especiales o características avanzadas de cada software.

Esta metodología permitió aislar y observar las diferencias inherentes a cada entorno de programación para resolver un mismo desafío algorítmico, más allá de la mera funcionalidad correcta.

De acuerdo a cada software de programación de PLC y tambien al hardware es posible tener acceso a instrucciones de programación con mayor capacidad de procesamiento, tal es el caso de la instrucción CPT del RsLogix, la cual permite simplificar las operaciones y por ende el número de líneas de programación ya que puede hacer operaciones en su interior a partir de una ecuación. En el caso de Tia Portal contamos con la instrucción Math, se usa para realizar operaciones matemáticas como suma, resta, multiplicación, división, exponenciales, trigonometría, etc.

El desarrollo de los programas de este artículo se realizó con las instrucciones básicas de Rslogix y Tia Portal debido a que los softwares de U90Lader y Codesys no cuentan con instrucciones que puedan desarrollar operaciones más complejas como los mencionados anteriormente y poder tener una comprensión de cómo pueden funcionar los diferentes softwares para correr el entrenamiento de redes neuronales en las industrias.

## 3. Desarrollo.

En las figuras 12 a 19 podremos analizar y/o comparar los resultados de correr los programas en los diferentes PLCs de acuerdo a su plataforma, en este caso la referencia es el software de la marca ABB, Codesys, Rslogix y Tia portal. En la figura 12 esta el resultado de la combinación (0,0), de la función AND, de las entradas en posición des energizadas en el renglón 2 y 3 en falso, por tanto, la salida de la red neuronal está en falso debido a que N1 en el renglón 7 tiene un valor de -3, lo que en la aplicación de la función de activación, la comparación de mayor o igual a cero no se cumpla, es menor a “0”, en el último renglón la salida física esta apagada (0,0,0). Asi mismo ocurre en las combinaciones (1,0) y (0,1) en las figuras 13 y 14; donde N1 es -1 y -2 respectivamente. Ahora bien en la figura 15 esta el caso de la combinación (1,1) donde las entradas estan energizadas y la comparación de la función de transferencia resulta ser positiva es decir N1 mayor o igual a “0”. Tambien se presenta los estados de la función AND en Tia Portal en las figuras 16 a 19 de las combinaciones (0,0), (0,1), (1,0) y (1,1) respectivamente.

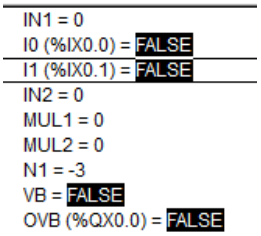



**Figura 12.** Combinación (0,0)

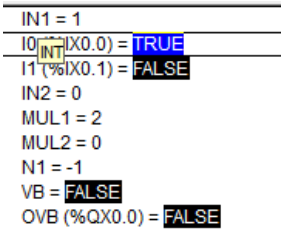



**Figura 13.** Combinación (1,0)

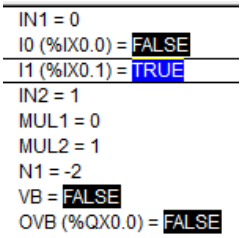



**Figura 14.** Combinación (0,1)

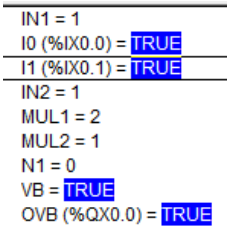



**Figura 15.** Combinación (1,1)

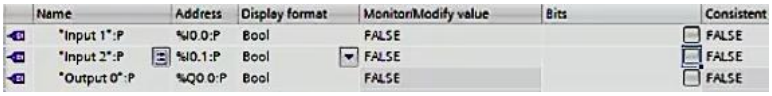



**Figura 16.** Combinación (0,0)

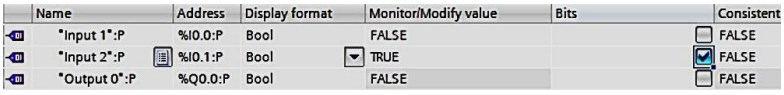
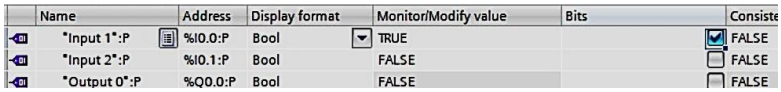
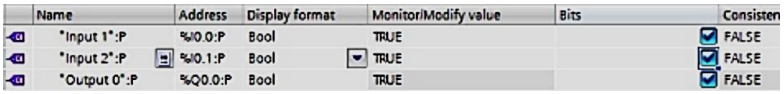



**Figura 17.** Combinación (0,1)



| | Name | Address | Display format | Monitor/Modify value | Bits | Consiste |
|---|---|---|---|---|---|---|
| | "Input 1":P | %I0.0:P | Bool | TRUE | | FALSE |
| | "Input 2":P | %I0.1:P | Bool | FALSE | | FALSE |
| | "Output 0":P | %Q0.0:P | Bool | FALSE | | FALSE |

**Figura 18.** Combinación (1,0)



| | Name | Address | Display format | Monitor/Modify value | Bits | Consisten |
|---|---|---|---|---|---|---|
| | "Input 1":P | %I0.0:P | Bool | TRUE | | FALSE |
| | "Input 2":P | %I0.1:P | Bool | TRUE | | FALSE |
| | "Output 0":P | %Q0.0:P | Bool | TRUE | | FALSE |

**Figura 19.** Combinación (1,1).

## Conclusiones.

Los resultados de esta investigación demuestran que es posible la implementación de la ecuación de redes neuronales artificiales básicas, específicamente un perceptrón para la función lógica AND, en entornos de Controladores Lógicos Programables (PLC) utilizando exclusivamente lenguaje de escalera. Si bien la implementación requirió un mayor número de líneas de código en comparación con entornos de programación tradicionales para IA, se logró validar funcionalmente en los cuatro softwares analizados (U90 Ladder, RSLogix 500, CODESYS y TIA Portal), utilizando operaciones aritméticas básicas y comparaciones.

Una contribución significativa de este trabajo es la evidencia práctica de que plataformas industriales, que no fueron diseñadas originalmente para ejecutar algoritmos de inteligencia artificial, poseen la capacidad de cálculo suficiente para modelos de RNA simples mediante una programación estratégica. Esto abre una puerta a la integración de conceptos de IA en la automatización industrial convencional, incluso en equipos de gama baja y media, sin necesidad de hardware especializado.

Este hallazgo refuerza una idea que ya se vislumbraba en el campo del control industrial: la notable flexibilidad de los PLCs para incorporar paradigmas de control avanzados. Como ya ocurría con otros formalismos, como las Redes de Petri para modelar sistemas de eventos discretos (Murillo-Soto, 2008), ahora demostramos que también es posible adaptar el concepto de redes neuronales para crear sistemas básicos de clasificación y decisión.

Desde el trabajo en forma general observar cómo la elección del software y el hardware del PLC influye directamente en la elegancia y eficiencia del código. Mientras que plataformas como TIA Portal y RSLogix 500 ofrecen instrucciones avanzadas (como CPT o MATH) que permiten condensar la lógica, entornos como U90 Ladder requirieron una implementación más extensa y modular con instrucciones básicas, lo que impacta directamente en la legibilidad y el mantenimiento del programa.

Limitaciones.

El principal factor limitante de este estudio fue la inherente simplicidad del modelo de perceptrón, restringiéndose a problemas linealmente separables. La ausencia de capacidad para un aprendizaje en tiempo real dentro del PLC significa que cualquier ajuste a la red (como cambiar la función lógica o los pesos) requiere un reentrenamiento externo (en este caso, en MATLAB) y una reprogramación manual de los parámetros en el código del PLC.

Al replicar o expandir esta investigación, se recomienda explorar la implementación de redes más complejas, como un perceptrón multicapa, lo que probablemente exigiría el uso de bloques de función (FB) y arrays, funcionalidades no siempre disponibles o fáciles de implementar en todos los entornos, especialmente en PLCs de gama baja. Una limitación de recursos a considerar sería la memoria de programa disponible en PLCs económicos, la cual podría ser insuficiente para RNA más grandes.

Recomendaciones para Trabajo Futuro.

Como continuación de este trabajo, se recomienda investigar la implementación de funciones de activación no lineales (como ReLU o sigmoidal) y la creación de bloques de función reutilizables que empaqueten la lógica de una neurona, facilitando la construcción de redes más grandes. Además, sería valioso desarrollar una metodología para cargar nuevos pesos sinápticos desde una interfaz externa (como una HMI), acercando esta aplicación a un escenario de reconfiguración flexible sin detener la operación.

## Referencias bibliográficas.

**Diego Murillo, L., (2008). Redes de Petri:** *Modelado e implementación de algoritmos para autómatas programables. Tecnología en marcha, 21(4), 102-125.*

**García Jaimes, L. E., & Arroyave Giraldo, M. (2012).** *CONTROLADORES AVANZADOS EN PLC. Revista Politécnica, 8(14), 57-66.*

**Páez-Logreira, H. D., Zamora-Musa, R., & Bohórquez-Pérez, J. (2015).** *Programación de Controladores Lógicos (PLC) mediante Ladder y Lenguaje de Control Estructurado (SCL) en MATLAB. Facultad de Ingeniería, 24(39), 109-119.*

## Información de los autores.

**Héctor Manuel González Cortés,** Ingeniero Electromecánico por el Instituto Tecnológico de Apizaco y Maestro en Ingeniería en Automatización y Control de Procesos por la Universidad Politécnica de Tlaxcala. Cuenta con experiencia en la industria, especializándose en mantenimiento eléctrico y automatización, con dominio de múltiples softwares de programación de PLCs. Su trayectoria se distingue por la capacidad de traducir la teoría en aplicación práctica en proyectos reales. Con una vocación sólida por la enseñanza, ha logrado impactar en la formación tecnológica desde el nivel de educación básica hasta el universitario, vinculando siempre el conocimiento técnico con las necesidades del sector productivo.

**Alejandro Medina Santiago** Investigador del Departamento de Ciencias de la Computación, Instituto Nacional de Astrofísica, Óptica y Electrónica, formación académica es Doctor en Ciencias en Ingeniería Eléctrica por el Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional; Maestro en Ciencias en Ingeniería Eléctrica por el Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional; Ingeniero en Electrónica por el Instituto Tecnológico de Tuxtla Gutiérrez. Mi línea de investigación es procesamiento de señales para automóviles de seguridad, Ciberseguridad a nivel de software y hardware, IoT, Industria 4.0, diseño de circuitos integrados VLSI y diseño de sistemas inteligentes basados en redes neuronales y lógica difusa.

**Joel Castro Ramírez** El Mtro. Joel Castro Ramírez es académico de tiempo completo en ingeniería mecatrónica y automatización en la UP Tlaxcala. Posee grados en Ingeniería Mecatrónica y Maestría en Automatización y Control. Imparte cursos clave en control de robots, aprendizaje automático y visión artificial. Su investigación enfoca modelado de sistemas con ML y automatización de robots manipuladores para eficiencia industrial.

**Jorge Antonio Orozco Torres** (PhD.'18-MC'08-B'94) Profesor del Departamento de Ingeniería Industrial del TecNM/ Instituto Tecnológico de Tuxtla Gutiérrez, Ingeniero Industrial en Producción por el Instituto Tecnológico de Tuxtla Gutiérrez, Maestro en Ciencias de la Computación por el Instituto de Investigación en Tecnologías y Sistemas (CITIS) de la UAEH, Doctor en Desarrollo Tecnológico por la Universidad de Ciencias y Tecnología DESCARTES, Miembro del Sistema Nacional de Investigación (NC), Investigador del Centro de Investigación, Desarrollo e Innovación Tecnológica de la Universidad de Ciencia y Tecnología Descartes UNAM. Profesor de posgrado en la misma Universidad. Intereses de investigación Ciencia de datos, IoT, Industria 4.0, Diseño de sistemas Diseño de Sistemas Inteligentes con Deep Learning, Redes Neuronales y Lógica Difusa.

**Julio Rodríguez González,** ingeniero en electrónica egresado del instituto tecnológico de Orizaba Veracruz, curso la maestría en optoelectrónica en la facultad de físico matemáticas de la BUAP, realizo la especialidad en instrumentación y control de sistemas mecatrónicos por el IPN.