https://repository.uaeh.edu.mx/revistas/index.php/investigium/issue/archive

Boletín Científico INVESTIGIUM de la Escuela Superior de Tizayuca

InvESTigium

ISSN: 2448-4830

Publicación semestral, Vol. 11, No. Especial (2025) 165-173

"Cuarto Congreso Internacional de Investigación e Innovación en Tendencias Globales y Segundo Coloquio de Posgrado en Gestión y Desarrollo de Nuevas Tecnologías"
24 –30 de octubre de 2025

Implementación digital de modelos neuronales acoplados y su aplicación en una estrategia de navegación autónoma para robots móviles.

Digital implementation of coupled neuronal models and its application in an autonomous navigation strategy for mobile robots

Jazmin Virgilio Escamilla [a], Elizabeth Reyes Guerrero [a], Kristian Freyri Maya Gress [a], Juan-Carlos González-Islas [a], America-Berenice Morales Díaz [b], Jonatan Peña Ramírez [c] .*

Abstract:

This paper presents the digital implementation of a pair of Hindmarsh-Rose (HR) neurons in a low-cost microcontroller. The HR model is chosen because it can reproduce most of the behaviors observed in an isolated biological neuron. The neurons are coupled in a bidirectional configuration and the corresponding synaptic current of each neuron is strategically designed such that, in the absence of obstacles, these currents are identical and the neurons achieve synchronization. In contrast, when an obstacle is present, the synaptic currents become different and the neurons lose synchrony. Here, this strategy is implemented by using programming threads for the parallel reading of sensors and numerical methods for the solution of differential equations required in the mathematical model, with an application to collision avoidance in mobile robotics. Specifically, we show that when the neurons are synchronized, the robot moves in a straight line, but when an obstacle appears, the synchrony in the neurons is lost, which ultimately results in having one of the wheels spinning faster than the other to avoid the collision. The obtained experimental results show that the proposed avoidance collision strategy may be useful in a real-world scenario.

Keywords:

Hindmarsh-Rose, Neurons, ESP32, Mobile robotics

Resumen:

Este artículo presenta la implementación digital de un par de neuronas Hindmarsh-Rose (HR) en un microcontrolador de bajo costo. Se eligió el modelo HR porque puede reproducir la mayoría de los comportamientos observados en una neurona biológica aislada. Las neuronas están acopladas en una configuración bidireccional y la corriente sináptica correspondiente de cada neurona está diseñada estratégicamente de tal manera que, en ausencia de obstáculos, estas corrientes son idénticas y las neuronas logran la sincronización. Por el contrario, cuando hay un obstáculo, las corrientes sinápticas se vuelven diferentes y las neuronas pierden la sincronía. En este trabajo, esta estrategia se implementa mediante hilos de programación para la lectura paralela de sensores y métodos numéricos para la solución de ecuaciones diferenciales requeridas en el modelo matemático, con una aplicación a la prevención de colisiones en robótica móvil. Específicamente, demostramos que cuando las neuronas están sincronizadas, el robot se mueve en línea recta, pero cuando aparece un obstáculo, se pierde la sincronía en las neuronas, lo que resulta en que una de las ruedas gire más rápido que la otra para evitar la colisión. Los resultados experimentales obtenidos muestran que la estrategia de evitación de colisiones propuesta puede ser útil en un escenario del mundo real.

***Palabras Clave**:*

Hindmarsh-Rose, Neurona, ESP32, Robots móviles

[a*] Autor de correspondencia, Área Académica de Computación y Electrónica, Universidad Autónoma del Estado de Hidalgo, 42184, Mineral de la Reforma, Hidalgo, México, https:// 0009-0000-6166-9654 , Email: re447553@uaeh.edu.mx

[a] Área Académica de Computación y Electrónica, Universidad Autónoma del Estado de Hidalgo, 42184, Mineral de la Reforma, Hidalgo, México, https://orcid.org/0009-0005-7326-8811, https://orcid.org/0000-0001-8425-0960, https://orcid.org/0000-0002-2190-0660 Email: vi499166@uaeh.edu.mx kristian_maya10493@uaeh.edu.mx, juan_gonzalez7024@uaeh.edu.mx

[b,] Centro de Investigación y de Estudios Avanzados,| Saltillo, Coahuila, México, https://orcid.org/0000-0003-1450-3664, Email: america.morales@cinvestav.edu.mx

[c] Laboratorio de Sistemas Dinámicos, CICESE. Carretera Ensenada-Tijuana 3918, Zona Playitas, C.P. 22860, Ensenada, B.C. México, https://orcid.org/0000-0001-8453-6694, jpena@cicese.mx

Fecha de recepción: 10/09/2025, Fecha de aceptación: 08/10/2025, Fecha de publicación: 12/12/2025

DOI: https://doi.org/10.29057/est.v11iEspecial.15980

I. Introducción

El desarrollo de hardware inspirado en principios biológicos ha impulsado soluciones innovadoras en áreas como el control de robots autónomos, el procesamiento sensorial eficiente y la computación de bajo consumo. En este contexto, la implementación electrónica de modelos neuronales se perfila como una estrategia potente para reproducir comportamientos complejos como adaptación, sincronización y generación de patrones dinámicos [1], [2].

Entre los diversos modelos matemáticos, el de Hindmarsh–Rose (HR) ha adquirido relevancia debido a su capacidad de representar una amplia gama de dinámicas neuronales, incluyendo disparos regulares, patrones de bursting, estados caóticos y reposo. Este modelo mantiene un equilibrio entre realismo biológico y simplicidad computacional, lo que lo convierte en una alternativa atractiva frente a modelos más complejos como el de Hodgkin–Huxley [3], [4].

La implementación del modelo HR en hardware digital ha sido ampliamente explorada en plataformas como FPGA (por sus siglas en inglés que significa Matriz de Puertas Programables en Campo), donde las ecuaciones diferenciales pueden resolverse mediante aproximaciones numéricas optimizadas. Nazari y Jamshidi [5] propusieron un diseño digital eficiente empleando el algoritmo CORDIC, eliminando la necesidad de multiplicadores y reduciendo significativamente los recursos lógicos y el consumo de potencia, a la vez que mantienen la fidelidad del modelo. De manera similar, Haghiri [6] introdujo una implementación basada en polinomios aproximados, lo cual reduce los requerimientos de área y permite ejecutar simulaciones en tiempo real.

Además, se han desarrollado técnicas de aproximación segmentada (uniform-segmented) para representar las funciones no lineales del modelo HR, aprovechando operaciones simples de desplazamiento y suma, con resultados prometedores en síntesis y validación en FPGA [7]. Una visión más amplia de estas técnicas puede encontrarse en revisiones recientes sobre arquitecturas reconfigurables para cómputo neuromórfico, que destacan el papel de FPGAs y ASICs (Circuito Integrado de Aplicación Específica) en la emulación neuronal masivamente paralela [8], [9].

En paralelo, se han explorado implementaciones analógicas y mixtas. Algunos trabajos han diseñado circuitos basados en amplificadores operacionales y elementos pasivos capaces de reproducir comportamientos caóticos del modelo HR, acercándose a un funcionamiento neuromórfico natural [10]. Asimismo, existen propuestas en tecnologías CMOS (Semiconductor complementario de óxido metálico) de bajo voltaje (0.65 V), lo que abre la posibilidad de aplicaciones en sistemas portátiles y de ultra bajo consumo [11].

En el ámbito del cómputo embebido, investigaciones recientes han mostrado que es posible implementar el modelo HR en microcontroladores de bajo costo, generando patrones rítmicos de locomoción en robots móviles con recursos computacionales limitados [12]. También se han estudiado aproximaciones que emplean funciones hiperbólicas como la tangente para simplificar la representación de no linealidades del modelo extendido (eHR), lo cual facilita su integración en firmware optimizado para sistemas embebidos [13].

Por otro lado, procesadores neuromórficos como Intel Loihi incorporan neuronas electrónicas basadas en principios biológicos y ofrecen aprendizaje en línea, baja latencia y eficiencia energética [14]. Estos desarrollos, junto con arquitecturas híbridas digitales-analógicas en FPGA y ASIC, marcan la tendencia hacia sistemas escalables y adaptativos capaces de abordar tareas complejas en tiempo real.

Además de las aproximaciones digitales y de bajo consumo, también se han desarrollado circuitos analógicos que permiten estudiar directamente la dinámica del modelo HR en hardware físico. Un ejemplo notable es el trabajo de Chialvo y colaboradores [15], quienes propusieron una implementación electrónica del modelo HR que reproduce patrones caóticos y de bursting, facilitando la exploración experimental de fenómenos neuronales en tiempo real.

En resumen, la implementación del modelo Hindmarsh–Rose en hardware digital, analógico y embebido representa un área de investigación activa que combina teoría matemática, eficiencia computacional y viabilidad tecnológica. Este enfoque promete aplicaciones en robótica autónoma, procesamiento sensorial y plataformas neuromórficas de próxima generación [2], [5], [8].

A pesar de estos avances, la mayoría de los desarrollos anteriores requieren plataformas costosas o complejas como en [20] donde el costo de la construcción de una sola neurona es elevado, además de tener un ensamble complejo de implementar. En contraste, este trabajo propone la implementación del modelo de Hindmarsh–Rose en un microcontrolador económico, aprovechando programación multihilo para la adquisición paralela de sensores de distancia y métodos numéricos para resolver en tiempo real las ecuaciones diferenciales del sistema. Las neuronas estarán acopladas de forma difusa y su comportamiento modulado en función de las señales sensoriales, permitiendo su aplicación en una estrategia novedosa de evasión de obstáculos para robots móviles. Este enfoque no solo demuestra la viabilidad de aplicar modelos biológicamente realistas en plataformas de bajo costo, sino que también abre el camino hacia sistemas embebidos inteligentes inspirados en redes neuronales biológicas.

La estructura del presente trabajo empieza analizando las herramientas utilizadas para lograr el objetivo de evasión mediante modelos neuronales, posteriormente se describe la implementación de los modelos neuronales en el

microcontrolador, después, muestra la prueba experimental realizada y para terminar se muestran las conclusiones.

II. Modelos neuronales

En esta sección se presentan las características del comportamiento neuronal a modelar, así como algunas herramientas de *hardware* utilizadas para lograr el objetivo de este trabajo.

Una neurona es una célula altamente especializada cuya función principal es recibir estímulos y transmitir impulsos nerviosos. Desde el punto de vista estructural, una neurona típica se compone de tres partes fundamentales: el soma, el axón y las dendritas [16]. La Figura 1 muestra el soma, o cuerpo celular, alberga el núcleo y los principales orgánulos celulares; por su parte, el axón es una prolongación del soma que actúa como el principal canal de conducción, transmitiendo señales eléctricas a lo largo de su recorrido, finalmente, las dendritas son extensiones ramificadas que emergen del soma y cumplen la función de captar señales provenientes de otras neuronas o de receptores sensoriales externos [16][17][18].

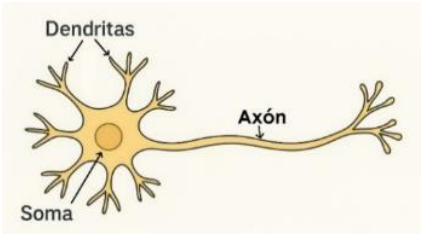


Figura 1. *Elementos básicos una neurona.*

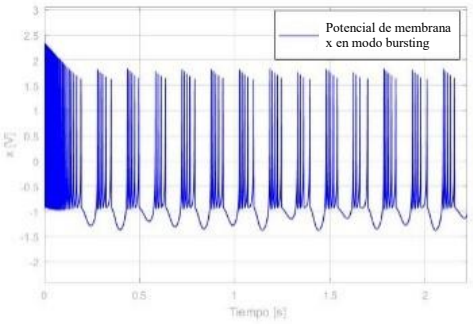
Las neuronas se comunican entre sí a través de conexiones especializadas llamadas sinapsis, donde la transmisión de información ocurre mediante impulsos eléctricos conocidos como potenciales de acción o potenciales de membrana. Estos impulsos generan la liberación de sustancias químicas llamadas neurotransmisores. La membrana citoplasmática de la neurona delimita su entorno interno del medio extracelular, y en ambos lados de esta membrana existe una distribución desigual de cargas eléctricas, lo que da lugar a una diferencia de potencial. Este fenómeno es denominado potencial de membrana, y resulta de la separación entre dos soluciones con distintas concentraciones iónicas a ambos lados de la membrana.

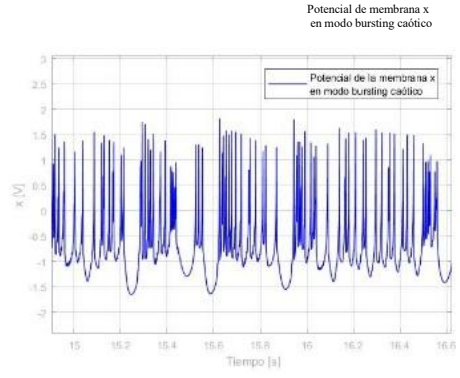
El comportamiento dinámico de una neurona está determinado por la evolución temporal de este potencial [16][17][19]. Cuando no se recibe ningún estímulo, se dice que la neurona está en estado de reposo, y el interior de la célula presenta una carga negativa respecto al exterior. Ante ciertos estímulos, el potencial de membrana puede cambiar: si se vuelve más positivo se habla de despolarización, y si se vuelve más negativo, de hiperpolarización. Cuando la despolarización supera un umbral crítico en un breve periodo de tiempo, se genera una espiga o "*spike*", que forma parte de la señal neuronal.

La alternancia entre estas espigas y los estados de reposo se denomina actividad tipo *spiking* [16][17][18][19], caracterizada por trenes de espigas en el potencial de membrana. Cuando varias espigas ocurren agrupados en un corto intervalo de tiempo, esta actividad se conoce como ráfaga o *burst*. A su vez, cuando se producen varias ráfagas sucesivas que contienen el mismo número de espigas, se presenta el fenómeno llamado *bursting*. Existe también una variación de este comportamiento denominada *bursting* caótico (*chaotic bursting*), en la cual el número de espigas por ráfaga varía de manera impredecible entre una secuencia y otra [16][18][20].

Modelo Hindmarsh-Rose

Este modelo neuronal es una modificación de modelo *FitzHugh Nagumo*, en donde se remplaza una función lineal por una función cuadrática lo cual permite representar los potenciales de una neurona de una manera más realista; se incorpora una tercera ecuación para así obtener un sistema de ecuaciones diferenciales no lineales tridimensionales que producen un comportamiento similar al del modelo *Hodgkin-Huxley* pero con una estructura más simple; de esta manera se tiene los tres comportamientos posibles de la neurona, *spiking*, *bursting* y *bursting caótico* como se ilustra en la Figura 2.





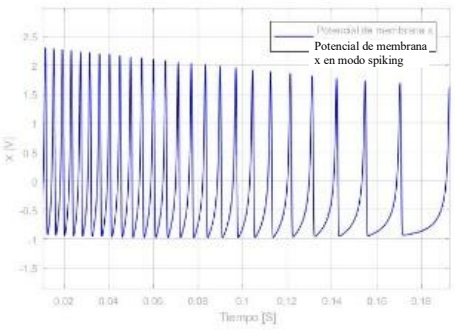


Figura 2. *Los tres estados diferentes de la membrana.*

La estructura del modelo es

$$\begin{aligned} \dot{x}_i &= y_i + bx_i^2 - ax_i^3 - z_i + I_i, \\ \dot{y}_i &= c - dx_i^2 - y, \\ \dot{z}_i &= r[s(x_i - x_0)] - z_i, \end{aligned} \qquad (1)$$

Siendo $x \in R$ el potencial de la membrana, $y \in R$ es la corriente de recuperación que modela el transporte de iones de potasio y sodio a través de canales de iones rápidos, $z \in R$ es la corriente de adaptación que modela el transporte de otros iones a través de canales lentos, mientras que los parámetros constantes $a \in R$ permite conmutar entre el comportamiento de *bursting* y el comportamiento de *spiking*, además de determinar la frecuencia de este último, $r \in R$ determina el comportamiento del *bursting*, $s \in R$ rige la adaptación, si posee valores alrededor de 1 causa un comportamiento de *spiking* rápido, $x_0 \in R$ es el potencial de reposo, también utilizado para potencial de referencia, $b, c, d \in R^+$ son parámetros de ajuste definidos positivos y por último $I \in R$ es la corriente externa, que siendo la entrada del sistema y con una correcta sintonización de los parámetros causa los tres comportamientos posibles y el subíndice i que va desde 1 hasta n que corresponde al número de neuronas en el sistema.

Acoplamiento Neuronal

Dos neuronas iguales pueden ser relacionadas o interconectadas mediante un acoplamiento difuso como se describe en la ecuación (2)

$$\begin{aligned} \dot{x}_1 &= y_1 + bx_1^2 - ax_1^3 - z_1 + I_1 + v_1, \\ \dot{y}_1 &= c - dx_1^2 - y, \\ \dot{z}_1 &= r[s(x_1 - x_0)] - z_1, \end{aligned} \qquad (2)$$

$$\begin{aligned} \dot{x}_2 &= y_2 + bx_2^2 - ax_2^3 - z_2 + I_2 + v_2, \\ \dot{y}_2 &= c - dx_2^2 - y, \\ \dot{z}_2 &= r[s(x_2 - x_0)] - z_2, \end{aligned} \qquad (3)$$

donde

$$\begin{cases} v_1 = \varepsilon(x_2 - x_1), \\ v_2 = \varepsilon(x_1 - x_2). \end{cases} \qquad (4)$$

siendo $\varepsilon > 0$ la fuerza de acoplamiento entre las neuronas HR1 (2) y HR2 (3), que, con un valor considerablemente grande se sincronizan aun teniendo valores ligeramente diferentes en sus corrientes de entrada ($I_1 \approx I_2$), de lo contrario, las neuronas se desincronizan.

En esta sección se analizan las herramientas requeridas para llevar a cabo la implementación de una neurona digital para aplicaciones de evasión de obstáculos.

III. Preliminares

En esta sección se enlistas las herramientas necesarias para la construcción e implementación del sistema de evasión de obstáculos de un vehículo diferencial. Un esquema general se muestra a continuación en la Figura 3.

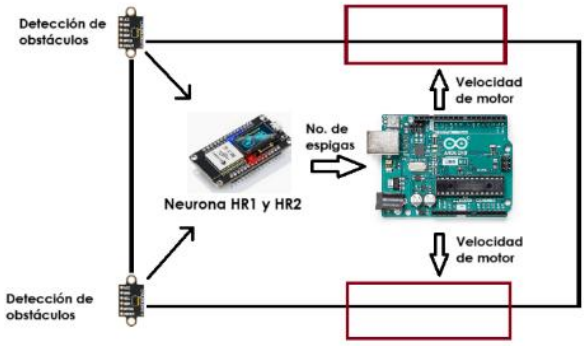


Figura 3. *Esquema general de la plataforma.*

Recursos de hardware

Sensor VL53L0X

El VL53L0X es un módulo de medición de distancias láser de tiempo de vuelo (ToF), alojado en el encapsulado más compacto del mercado actual, (Ver Figura 4(a)), que proporciona una medición precisa de distancias independientemente de la reflectancia del objetivo, a diferencia de las tecnologías convencionales. Puede medir distancias absolutas de hasta 2 m, estableciendo un nuevo referente en el rendimiento de medición de distancias y abriendo la puerta a diversas aplicaciones. El VL53L0X integra un conjunto SPAD (diodos de avalancha de fotón único) de vanguardia e incorpora la tecnología patentada *FlightSense* de segunda generación de ST. El emisor VCSEL (láser de emisión superficial de cavidad vertical) de 940 nm del VL53L0X es totalmente invisible al ojo humano y, junto con filtros infrarrojos físicos internos, permite mayores distancias de medición, mayor inmunidad a la luz ambiental y mayor robustez para cubrir la diafonía óptica del vidrio [25].

Placa ESP32

La ESP32 es un microcontrolador de bajo costo y alto rendimiento desarrollado por *Espressif Systems*, ampliamente utilizado en aplicaciones de Internet de las Cosas (IoT), sistemas embebidos y automatización. Integra un procesador de doble núcleo *Tensilica Xtensa* LX6 que puede operar a frecuencias de hasta 240 MHz, acompañado de una memoria SRAM de alta velocidad y soporte para memoria externa. Entre sus principales características destaca la conectividad inalámbrica integrada, ya que incorpora Wi-Fi (802.11 b/g/n) y Bluetooth clásico/Bluetooth *Low Energy* (BLE), lo que facilita el desarrollo de dispositivos conectados sin necesidad de módulos adicionales (ver Figura 4 (b)). Además, cuenta con una amplia variedad de periféricos: conversores ADC y DAC, interfaces SPI, I²C, I²S, UART, PWM, sensores táctiles capacitivos, temporizadores y controladores para motores [26].

(a) *(b)*

Figura 4. *Hardware utilizado para la implementación de la neurona. (a) sensor VL53L0X. (b) ESP32.*

Se incluye el sensor en esta sección por la relación que tiene la distancia medida en relación con la corriente de entrada en los modelos neuronales.

Programación paralela

En la ESP32, una *Task* (tarea) es una unidad de ejecución independiente dentro del sistema operativo en tiempo real (*FreeRTOS*) que incorpora el chip [21]. Cada tarea funciona como un hilo (*thread*) que puede ejecutarse de forma concurrente con otras, compartiendo recursos del microcontrolador pero con su propio contexto, pila y prioridad [22].

Las *Tasks* permiten dividir un programa en procesos más pequeños y específicos que se ejecutan de manera paralela o pseudo-paralela gracias al planificador de *FreeRTOS*. Por ejemplo, una tarea podría gestionar la lectura de un sensor mientras otra controla la comunicación por *Wi-Fi*, sin que una bloquee a la otra [23].

En la ESP32, se pueden crear tareas usando la función:

```
xTaskCreate(
   function_name,       // Función que ejecutará la tarea
   "task_name",         // Nombre identificativo
   stack_size,          // Tamaño de la pila en palabras
   parameter,           // Parámetro opcional para la función
   priority,            // Prioridad de ejecución
   &task_handle         // Referencia para controlar la tarea
);
```

Cada *Task* corre bajo la administración del planificador de *FreeRTOS*, que asigna el tiempo de CPU disponible entre ellas según sus prioridades y el estado de ejecución (*Ready, Running, Blocked, Suspended*) [22]. En la ESP32, gracias a su arquitectura de doble núcleo (CPU0 y CPU1), es posible incluso fijar tareas a un núcleo específico para optimizar el rendimiento [21].

Esta herramienta es esencial cuando se requiere ejecutar procesos concurrentes, mantener tiempos de respuesta cortos y aprovechar al máximo los recursos del microcontrolador sin bloquear el flujo principal de ejecución [23].

Método de integración de Euler

El método de Euler es un procedimiento numérico para resolver ecuaciones diferenciales ordinarias de manera aproximada. Se fundamenta en avanzar paso a paso utilizando la pendiente de la función en un punto conocido, lo cual permite obtener un valor estimado en el siguiente. Matemáticamente, dado el problema inicial

$$\frac{dy}{dt} = f(t, y), \qquad y(t_0) = y_0,$$

la iteración se define como

$$y_{n+1} = y_n + hf(t_n, y_n),$$

donde h es el tamaño de paso. Aunque es un método simple y de fácil implementación, su precisión es limitada y puede ser inestable si el tamaño de paso no se elige correctamente [27]. Por ejemplo, la Figura 5 compara la solución exacta de la ecuación diferencial $y' = y$, con $y(0) = 1$ contra la aproximación obtenida por Euler para $h = 0.5$ y $h = 0.1$

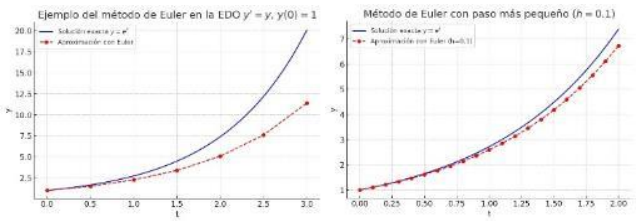


Figura 5. *Método de Euler con paso de h=0.5 y h=0.1.*

Modelo cinemático de robot diferencial

El movimiento de un robot diferencial se rige de acuerdo a su modelo cinemático, el cual está dado por

$$\begin{cases} \dot{x} = v\cos(\theta), \\ \dot{y} = v\sin(\theta), \\ \dot{\theta} = \omega, \end{cases} \qquad (5)$$

Siendo x y y las coordenadas Cartesianas de la posición del robot, θ es la orientación en la cual se encuentra el robot con referencia al eje horizontal, v y ω son las velocidades traslacional y rotacional, respectivamente. Por otro lado, las características físicas del robot a considerar se muestran en la Figura 6

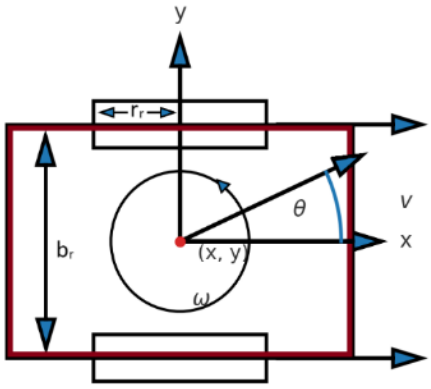


Figura 6. *Esquema de robot diferencial.*

donde b_r es la distancia entre las ruedas, r_r es el radio de las ruedas. Las velocidades de cada rueda están determinadas por

$$\omega_r = \frac{1}{r_r}\left(v + \frac{b_r\omega}{2}\right) \qquad (6)$$
$$\omega_l = \frac{1}{r_r}\left(v - \frac{b_r\omega}{2}\right)$$

de donde se puede deducir también que

$$v = \frac{\omega_r r_r + \omega_l r_r}{2}, \qquad (7)$$
$$\omega = \frac{\omega_r r_r - \omega_l r_r}{b_r}.$$

Esta conversión es necesaria dado que las velocidades rotacionales y traslacionales del robot diferencial estarán regidas por las velocidades de cada llanta y estas a su vez serán controladas por el número de espigas que tenga cada neurona.

IV. Configuración experimental

Para implementar un sistema de modelos neuronales dentro del IDE de la ESP32 se requiere contar con la siguiente conexión mostrada en la Figura 7

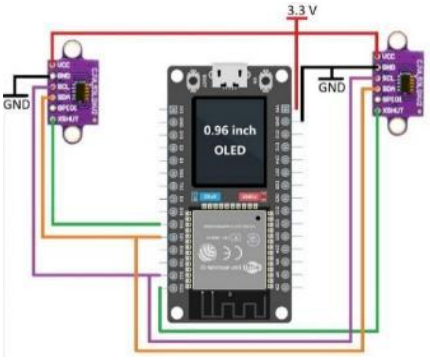


Figura 7. *Esquema de conexión entre sensores VL53L0X y la ESP32.*

Lo que permite conectar las terminales las terminales de los sensores SCL y SDA de manera serial, es el pin XSHUT que se usa para desactivar y activar los sensores individualmente y cambiar su dirección. Es necesario el uso de dos sensores de distancia ya que determinarán el valor de la corriente de cada modelo neuronal; que para probar el acoplamiento planteado en la ecuación (4) es necesario de dos modelos neuronales, y cada uno de los sensores determinará el valor de la corriente de cada modelo.

Acoplamiento de corriente

Se propone el siguiente acoplamiento para la relación entre la corriente de entrada de las neuronas con respecto a la distancia de los obstáculos con los que se pueda encontrar el robot diferencial [24], se plantea que el robot diferencial tenga un sensor de cada lado para la detección de obstáculos, por lo que se propone

$$I_1 = \begin{cases} a_1 I_{min}, & d_{der} \le x_{min} \\ a_1 I_{min}\left(\frac{I_{max}}{I_{min}}\right)^{\frac{d_{der}-x_{min}}{x_{max}-x_{min}}}, & x_{min} \le d_{der} \le x_{max} \\ c_1 I_{max} & d_{der} \ge x_{max} \end{cases} \qquad (8)$$

$$I_2 = \begin{cases} a_1 I_{min}, & d_{izq} \le x_{min} \\ a_1 I_{min}\left(\frac{I_{max}}{I_{min}}\right)^{\frac{d_{izq}-x_{min}}{x_{max}-x_{min}}}, & x_{min} \le d_{izq} \le x_{max} \\ c_1 I_{max} & d_{izq} \ge x_{max} \end{cases} \qquad (9)$$

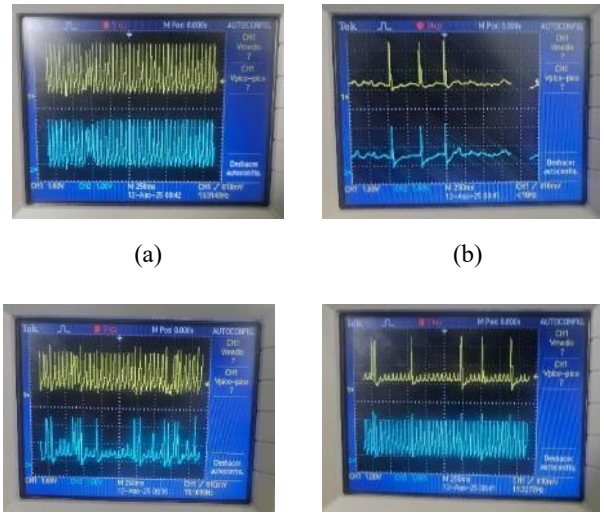
De modo que I_1 es la corriente de entrada de la neurona HR1, con I_2 es la corriente de entrada de la neurona HR2, I_{min} es la corriente de entrada mínima permitida para las neuronas , I_{max} es la corriente de entrada máxima permitida para las neuronas , d_{der} es la distancia del obstáculo al robot detectada por el sensor

derecho, d_{izq} es la distancia del obstáculo al robot detectada por el sensor izquierdo, x_{min} es la distancia mínima que puede detectar el sensor, x_{max} es la distancia máxima que puede detectar el sensor.

La lectura de los sensores y la conversión de distancia a corriente por medio de las ecuaciones (8) y (9) se ejecutan de manera paralela al programa principal de la ESP32, definiendo la *Task* antes de la sección de configuración del programa, posteriormente se definen los pines de lectura de los sensores, así como la configuración de las direcciones de cada uno de ellos para la lectura simultánea. Al final de la sección de configuración se manda a llamar el *Task* de la tarea paralela.

En la sección del ciclo o *loop* se recuperan los valores de las corrientes I_1 e I_2, las cuales son introducidas a las ecuaciones de las neuronas HR1 y HR2 correspondientes a las ecuaciones (2) y (3), y por medio del método numérico de Euler para la integración, se solucionan las ecuaciones diferenciales de dichos modelos, lo que permite tener disponibles los valores de las variables x_1 y x_2 que corresponden a los potenciales de cada neurona.

La Figura 8 muestra el comportamiento de dichos potenciales (a) sin obstáculos en los sensores (gráfica amarilla correspondiente a sensor derecho y gráfica azul a izquierdo), (b) ambos sensores con obstáculos, (c) sensor izquierdo con obstáculo y sensor derecho libre y (d) sensor
derecho con obstáculo y sensor izquierdo libre.



(a) (b)

(c) (d)

Figura 8. *Comportamientos del acoplamiento neuronal.*

Matemáticamente, los potenciales tienen tanto valores negativos como positivos, para lo cual se modifica el *setpoint* de cada potencial para que estos sean totalmente positivos por cuestiones electrónicas, ya que las salidas analógicas de la ESP32 solo van de 0 V a 3.3 V. De cualquier modo, se aprecia la sincronización de las neuronas gracias al acoplamiento propuesto en (4), y la reducción en el número de espigas cuando se acerca un obstáculo al sensor.

V. Análisis y discusión de resultados

Se presentan los resultados de implementación de los modelos neuronales en un robot diferencial para la evasión de obstáculos. Este robot cuenta de dos motores de 5V en cada llanta, un Arduino controla su velocidad por medio de un puente H; los sensores de distancia determinan la corriente en las neuronas contenidas en la esp32, misma que manda al Arduino el número de espigas obtenidas para así determinar la velocidad de cada motor. Los parámetros empleados para las pruebas experimentales se resumen en la Tabla 1

Tabla 1. Parámetros de la neurona

Parámetros del acondicionamiento de la corriente I (eq. 8 y 9)			
$a_1 = 0.95$	$b_1 = 1.35$	$c_1 = 0.85$	$a_1 = 0.95$
$I_{min} = 3.25$	$I_{max} = 10$	$x_{min} = 0.01[m]$	$x_{max} = 0.5[m]$
Parámetros de los modelos neuronales (eq. 2 y 3)			
$a = 1.0$	$b = 3.0$	$c = 1.0$	$d = 5.0$
$s = 4.0$	$x_0 = 1.6$	$r = 0.005$	$\varepsilon = 0.5$
Parámetros del vehículo			
$p = 0.7$	$r_r = 0.03\ [m]$	$b_r = 0.16[m]$	*p=0.7*
Ventana de tiempo para determinación del número de espigas = 100 ms			

La velocidad de cada llanta del robot diferencial está determinada por las ecuaciones

$$\omega_r = pk_1, \qquad (10)$$
$$\omega_l = pk_2.$$

Donde p es un parámetro de ajuste de velocidad del vehículo, k_1 y k_2 son los números de espigas en la ventana de tiempo de cada potencial de membrana x_1 y x_2 respectivamente. Lo cual indica que, si no hay obstáculos, el robot se moverá en línea recta de frente. Si hay obstáculo del lado derecho, la rueda izquierda rodará más lento ya que se asocia el número de espigas menor que se genera con esa detección; por el contrario, si se detecta obstáculo del lado izquierdo. La rueda derecha girará a menor velocidad ya que también se le asocia el menor número de espigas dentro de la ventana de tiempo asociada a esa detección. Se planteó una prueba que consiste en la evasión de dos obstáculos, uno del lado izquierdo y el otro del lado derecho para que el vehículo se dé a la tarea de evadirlos como se plantea en la Figura 9.

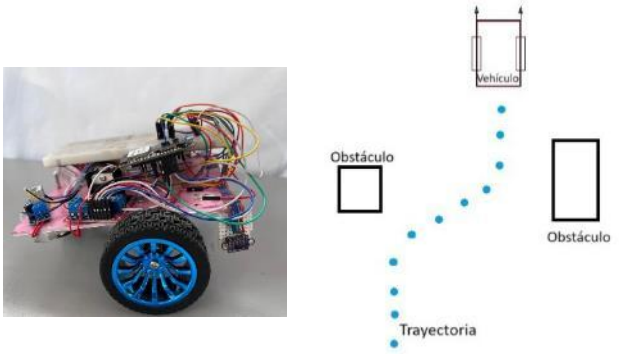


Figura 9. *Robot diferencial y su trayectoria de prueba.*

Esta prueba dio como resultado el comportamiento de los potenciales x_1 y x_2 mostrado en la Figura 10

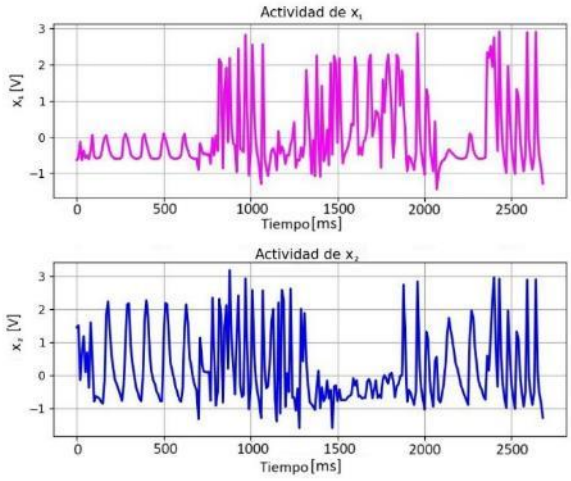


Figura 10. *Comportamiento de los potenciales x_1 y x_2 al momento de detectar obstáculos.*

Cuando se detecta un obstáculo del lado izquierdo, el potencial de x_1 disminuye en el sentido del número de espigas, por otro lado, si se detecta un obstáculo del lado derecho, x_2 reduce el número de espigas. Esto causa que tanto la rueda derecha e izquierda reduzcan la velocidad y así evadir los obstáculos. Al final de la prueba, las ruedas giran a la misma velocidad lo que causa que el vehículo se mueva en dirección recta.

Conclusiones

Se implementaron dos modelos neuronales en un microcontrolador ESP32, en el cual por medio de programación paralela se logró la lectura de dos sensores VL53L0X y la conversión de distancia a corriente sin perder velocidad del proceso, al mismo tiempo a través del modelo Hindmarsh-Rose se generó la señal de los potenciales x_1 y x_2 para obtener el número de espigas que genera cada señal, lo que rige la velocidad de las ruedas del vehículo móvil, los comportamientos de estos potenciales no son constantes, depende del tipo de textura de obstáculo, la iluminación, entre otros factores que presenta la lectura de los sensores, incluso la constante de acoplamiento tiene que ver con el comportamiento de los potenciales. Se han ajustados los parámetros p para tener una velocidad adecuada del vehículo para la lectura de los sensores y la interpretación de las distancias en el programa.

Como trabajos futuros se realizará un control híbrido el cual consistirá en un seguimiento de trayectoria del vehículo, que en cuanto detecte un obstáculo cambie a un control por modelos neuronales semejante al presentado en este trabajo. Posteriormente se incluirán más robots para trabajos colaborativos utilizando estos modelos neuronales.

Referencias

[1] Indiveri, G., & Liu, S. C. (2015). Memory and information processing in neuromorphic systems. Proceedings of the IEEE, 103(8), 1379–1397.

[2] Mead, C. (2020). How we created neuromorphic engineering. Nature Electronics, 3(7), 434–435.

[3] Hindmarsh, J. L., & Rose, R. M. (1984). A model of neuronal bursting using three coupled first order differential equations. Proceedings of the Royal Society of London. Series B. Biological Sciences, 221(1222), 87–102.

[4] Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? IEEE Transactions on Neural Networks, 15(5), 1063–1070.

[5] Nazari, S., & Jamshidi, S. (2024). Efficient digital design of the nonlinear behavior of Hindmarsh–Rose neuron model in large-scale neural population. Scientific Reports, 14, 3833.

[6] Haghiri, S. (2023). FPGA implementation of the Hindmarsh–Rose neuron model using polynomial approximation. International Journal of Circuit Theory and Applications, 51(5), 1783–1797.

[7] Samwi, S., Rostami, M., & Olesen, H. B. (2023). Reconfigurable digital FPGA implementations for neuromorphic computing: A survey on recent advances and future directions. Frontiers in Neuroscience, 17, 1112345.

[8] Furber, S. B. (2016). Large-scale neuromorphic computing systems. Journal of Neural Engineering, 13(5), 051001.

[9] Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., & Plank, J. S. (2017). A survey of neuromorphic computing and neural networks in hardware. arXiv preprint arXiv:1705.06963.

[10] Sinha, S., & Ditto, W. L. (1998). Dynamics based computation. Physical Review Letters, 81(10), 2156–2159.

[11] Ranjan, A., et al. (2021). Low-voltage CMOS implementation of the Hindmarsh–Rose neuron model. Microelectronics Journal, 112, 105012.

[12] Shilnikov, A. L., & Cymbalyuk, G. S. (2005). Transition between tonic spiking and bursting in a neuron model via the blue-sky catastrophe. Physical Review Letters, 94(4), 048101.

[13] Hsiao, C. F., et al. (2022). Efficient nonlinear approximations for embedded neuron models. IEEE Transactions on Biomedical Circuits and Systems, 16(4), 912–922.

[14] Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., Choday, S. H., ... & Flickner, M. D. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro, 38(1), 82–99.

[15] Chialvo, D. R., & Jalife, J. (1987). Nonlinear dynamics of excitable cells: a modeling approach. Chaos, Solitons & Fractals, 1(5), 475–493.

[16] Martínez, J. C. (2016). Modelos de neuronas artificiales en software para su uso en preparaciones de electrofisiología.

[17] Fernández, Á. F. J., Moreno, D. G. J., & Barranco, D. A. L. (2010). Diseño y evaluación de sistemas de control y procesamiento de señales basadas en modelos neuronales pulsantes. Sevilla: Tesis Doctoral. Universidad de Sevilla. Escuela Técnica Superior de Ingeniería Informática.

[18] Y. Marín Mas, Determinación de parámetros de modelos de neuronas. Tesis de Maestría, Universidad de las Islas Baleares, Instituto de Física Interdisciplinar y Sistemas Complejos (IFISC), Departamento de Física, 2018.

[19] Carrillo-Reid, L., & Bargas, J. (2008). Codificación de estados funcionales en redes neuronales biológicas. TIP Revista Especializada en Ciencias Químico-Biológicas, 11(1), 52-59.

[20] Equihua, G. G. V., & Ramirez, J. P. (2018). Synchronization of hindmarsh-rose neurons via huygens-like coupling. IFAC-PapersOnLine, 51(33), 186-191.

[21] Espressif Systems. ESP-IDF Programming Guide — FreeRTOS Tasks. Disponible en: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.htm

[22] FreeRTOS.org. Tasks and Co-routines. Disponible en: https://www.freertos.org/tasks.html

[23] Arduino-ESP32. FreeRTOS on ESP32. Disponible en: https://github.com/espressif/arduino-esp32

[24] Martínez Sánchez, E. (2022). Navegación autónoma de un robot móvil en un ambiente desconocido usando una red de neuronas electrónicas (Master's thesis, Tesis (MC)--Centro de Investigación y de Estudios Avanzados del IPN Departamento de Ingeniería Eléctrica. Sección de Mecatrónica).

[25] STMicroelectronics. (2024, junio). VL53L0X: Time-of-Flight ranging sensor (Datasheet No. DS11555 Rev 6). Recuperado de https://www.st.com/resource/en/datasheet/vl53l0x.pdf

[26] Espressif Systems. (2023). ESP32 series: Datasheet. Espressif Systems. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[27] Burden, R. L., & Faires, J. D. (2011). Análisis numérico (9.ª ed.). México, D.F.: Cengage Learning.