# SISTEMA EMBEBIDO PARA EL JUEGO DE TRES EN RAYA CON INTELIGENCIA ARTIFICIAL POR MEDIO DEL ALGORITMO MINIMAX

## *EMBEDDED SYSTEM FOR THE TIC-TAC-TOE GAME WITH ARTIFICIAL INTELLIGENCE USING THE MINIMAX ALGORITHM*

***Felipe Santiago Espinosa***
Universidad Tecnológica de la Mixteca, México
*fsantiag@mixteco.utm.mx*

***Fermín Hugo Ramírez Leyva***
Universidad Tecnológica de la Mixteca, México
*hugo@mixteco.utm.mx*

***Edgardo Yescas Mendoza***
Universidad Tecnológica de la Mixteca, México
*yescas@mixteco.utm.mx*

**Recepción:** 18/noviembre/2024 **Aceptación:** 19/mayo/2025

## Resumen

Tres en raya es un juego clásico entre dos contendientes, quienes deben colocar fichas en un tablero de 3x3 localidades, gana el jugador que alinea tres fichas en forma horizontal, vertical o diagonal. En este trabajo se describe un sistema embebido para el juego de tres en raya, implementado en la tarjeta DK-TM4C129X. El sistema presenta tres variantes: jugador vs jugador, jugador vs sistema y sistema vs jugador, las últimas dos difieren en la ejecución del primer turno. En los turnos del sistema, este genera sus decisiones con el algoritmo de Minimax, ampliamente documentado en la Inteligencia Artificial y la Teoría de Juegos. El propósito de este trabajo es demostrar la viabilidad de aplicar una técnica de IA en un entorno limitado de recursos, como el microcontrolador Tiva TM4C129X incluido en la tarjeta DK-TM4C129X. La tarjeta también tiene una pantalla táctil, que es fundamental para visualizar y ejecutar el juego.

**Palabras Clave:** Inteligencia Artificial, Microcontrolador TM4C129X, Minimax, Sistema embebido.

## *Abstract*

*Tic-Tac-Toe is a classic game between two players, who must place pieces on a 3x3 board. The player who aligns three pieces horizontally, vertically or diagonally wins. This work describes an embedded system for the Tic-Tac-Toe game, implemented on the DK-TM4C129X board. The system has three variants: player vs player, player vs system and system vs player, the last two variants are different in the execution of the first turn. In the system's turns, it generates its decisions with the Minimax algorithm, widely documented in Artificial Intelligence and Game Theory. The purpose of this work is to demonstrate the feasibility of applying an AI technique in a resource-limited environment, such as the Tiva TM4C129X microcontroller included in the DK-TM4C129X board. The board also has a touch screen, which is essential for viewing and running the game.*

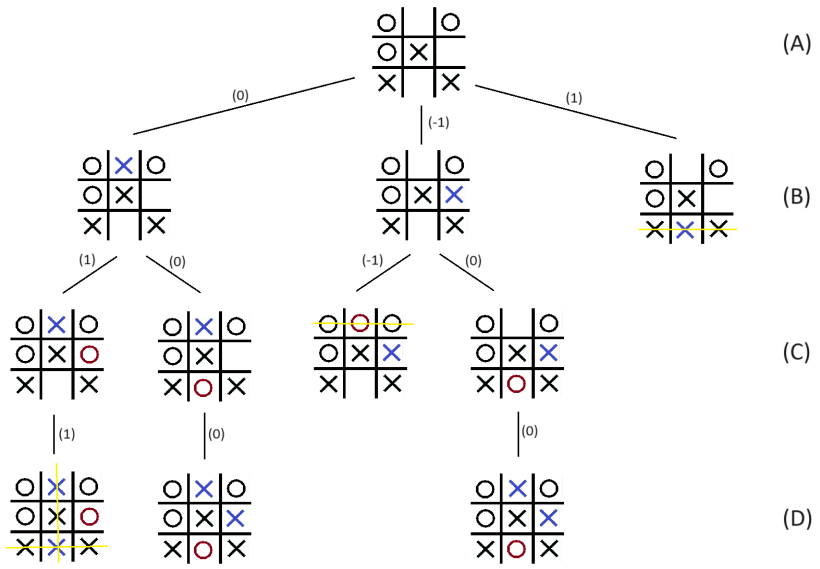
***Keywords:*** *Artificial Intelligence, Embedded system, Minimax, TM4C129X microcontroller.*

## 1. Introducción

El algoritmo Minimax tiene un lugar distinguido en la historia del pensamiento humano, presentando una estrategia para la toma de decisiones en juegos entre dos contendientes, como ajedrez, go y tres en raya (también conocido como juego del gato). La prueba del Teorema Minimax fue publicada por John von Neumann en 1928, en su obra *Theory of Games and Economic Behavior*, en la que compartió créditos con Oskar Morgensterm [Pineda, 2021].

El objetivo principal del algoritmo Minimax es determinar la mejor alternativa de tiro para un jugador. De los dos contendientes, uno es considerado como maximizador, mientras que el otro es el minimizador. Asumiendo que ambos juegan para ganar, el maximizador busca un resultado máximo en cada tirada mientras que el minimizador busca el resultado mínimo. Durante un turno se crea el árbol del juego con todos los movimientos posibles, el árbol puede crecer en varios niveles, alternando los del maximizador con los del minimizador. El árbol se debe recorrer en profundidad para llevar el aporte de un tiro de los niveles más profundos a niveles superiores [Russell ,2004], [Winston, 1992].

En juegos donde las piezas tienen valores diferentes, como el ajedrez, el algoritmo Minimax maneja números enteros positivos o negativos, dependiendo de las piezas involucradas y del nivel explorado. El juego de tres en raya solo tiene dos fichas: X y O, por ello, el algoritmo Minimax utiliza los valores 1 (victoria para X), -1 (victoria para O) y 0 (empate). En la Figura 1 se muestra una situación que refleja el comportamiento del algoritmo, el jugador con la ficha X debe tomar la decisión en el nivel A, que es maximizador, por lo que de los 3 valores recibidos del nivel B va a optar por el de la derecha, que le conduce a la victoria, las otras dos posibilidades llevan a un empate y una derrota. El nivel B es minimizador, a esto se debe la elección de los valores que genera el nivel C. El nivel C es maximizador, pero no tiene que decidir porque las opciones que genera el nivel D son únicas.




*Fuente: elaboración propia*

Figura 1 Comportamiento del algoritmo Minimax en el juego de tres en raya.

De manera práctica, la creación y evaluación del árbol se hace mediante una función recursiva, el fin de la recursión se establece cuando no hay casillas libres en una rama, no obstante, en juegos con mayor complejidad se puede definir un límite para el número de niveles. No es necesario almacenar la información de todos los posibles caminos en la memoria del sistema, es suficiente con mantener la mejor opción, actualizando la información durante el recorrido, por ejemplo, para el nivel

A de la Figura 1, la primera revisión regresa el valor 0, que significa un empate, este valor se conserva en la segunda opción que corresponde a una derrota, pero se reemplaza con la última revisión, ya que lleva a una victoria.

La inteligencia artificial ha tomado una gran relevancia en los últimos años porque ha transformado la forma de hacer las cosas en diferentes áreas, como medicina, industria y educación. En consecuencia, también ha despertado el interés en los desarrolladores de sistemas embebidos, principiantes o expertos, sobre la posibilidad de implementar técnicas de IA en microcontroladores [TRBL, 2023]. Por ello, en el Instituto de Electrónica y Mecatrónica de la Universidad Tecnológica de la Mixteca, se implementó un sistema embebido para el juego de tres en raya con tres modos de operación, estos son: jugador vs jugador, jugador vs sistema y sistema vs jugador, los últimos dos modos difieren en la ejecución del primer turno y las decisiones del sistema se toman con el algoritmo Minimax.

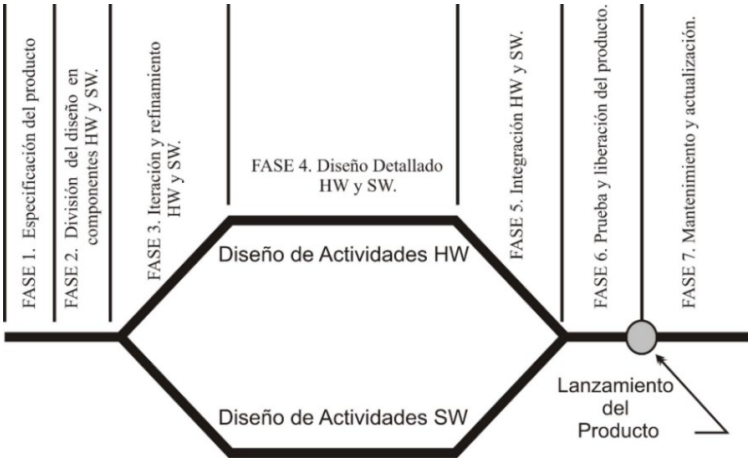
El sistema embebido se implementó en la tarjeta de desarrollo DK-TM4C129X, la cual está soportada por el microcontrolador Tiva TM4C129X, que tiene un núcleo ARM® Cortex™-M4 de $32\ bits$ [TI_UG, 2013]. Entre otras características, la tarjeta cuenta con una pantalla QVGA a color con sensor táctil resistivo, este periférico es fundamental porque permite una interfaz visual para seleccionar el modo de juego y dar seguimiento a su ejecución, además de hacer muy intuitivo su uso.

Una revisión del estado del arte demuestra que el algoritmo Minimax es pionero en la teoría de juegos, por lo que en la literatura se pueden encontrar diferentes descripciones, algunos ejemplos los encontramos en [Russell, 2004], [Winston, 1992] y [Rich, 1994]. En [TRBL, 2023] se exponen los retos que representa dotar de IA a los sistemas embebidos, para las redes neuronales sugieren el uso de un sistema con mayores prestaciones en las fases de entrenamiento, sin embargo, también hacen referencia a dos marcos de IA para aprendizaje automático enfocados a sistemas embebidos, estos son: TFLM (*TensorFlow Lite for Microcontrollers*) y AIfES (*Artificial Intelligence for Embedded Systems Framework*). En [Wulfert, 2024] se hace una descripción de AIfES y una comparativa con TFLM, en un sistema en chip (SoC) basado en un ARM Cortex-M4, utilizando redes neuronales completamente conectadas (FCNN) y redes neuronales

convolucionales (CNN). El sitio oficial sobre AIfES está disponible en [AIfES, 2024]. El libro escrito por Gian Marco Iodice, titulado *TinyML Cookbook: Combine artificial intelligence and ultra-low-power embedded devices to make the world smarter*, es un trabajo que conviene revisar para incorporar IA en sistemas embebidos, ya que describe diferentes aplicaciones de aprendizaje automático para redes neuronales en plataformas como Arduino y Raspberry Pi Pico [Iodice, 2022].

## 2. Métodos

El sistema embebido para el juego del gato o tres en raya se desarrolló con la metodología propuesta por Arnold S. Berger [Berger, 2002], quien organiza el ciclo de desarrollo en 7 fases, como se muestra en la Figura 2. La fase 7 se descarta porque queda fuera del alcance del trabajo.




*Fuente: [Berger, 2002]*

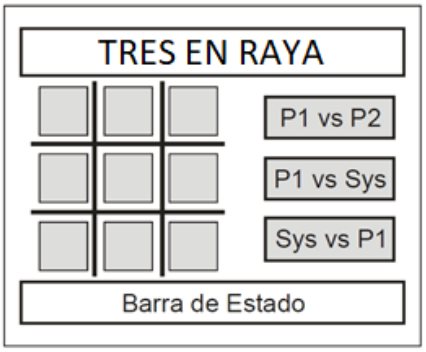
Figura 2 Ciclo de desarrollo de un sistema empotrado.

### Especificación del producto

El juego de tres en raya requiere de un tablero con 9 casillas, formadas con dos líneas horizontales y dos verticales cruzadas, formando el símbolo #. Los jugadores tradicionalmente usan la X y la O como fichas del juego. El objetivo para cada jugador consiste en la alineación horizontal, vertical o diagonal de tres de sus fichas, gana quien alcanza el objetivo o el juego se empata si las casillas se agotan sin alinear tres fichas. Durante una tirada, el jugador evalúa si puede conseguir una

alineación ganadora o si es mejor bloquear al contrincante para que no lo haga. El sistema embebido se desarrolló con las siguientes especificaciones:

- Se consideró una pantalla táctil como interfaz de usuario, en la cual se colocaron 3 botones para la selección del modo de juego, el tablero y una barra para mostrar el estado. La Figura 3 muestra la interfaz gráfica requerida por el sistema, se han colocado en gris las secciones que producirán eventos en la zona táctil.
- El sistema debe validar que el usuario no intente tirar en una posición ocupada.
- Después de cada tirada, el sistema debe revisar si hay un ganador o si ya no hay casillas libres, para indicar que ocurrió un empate.
- En la barra de estado se indica a quien le corresponde el turno, quien fue el ganador o si hubo un empate.



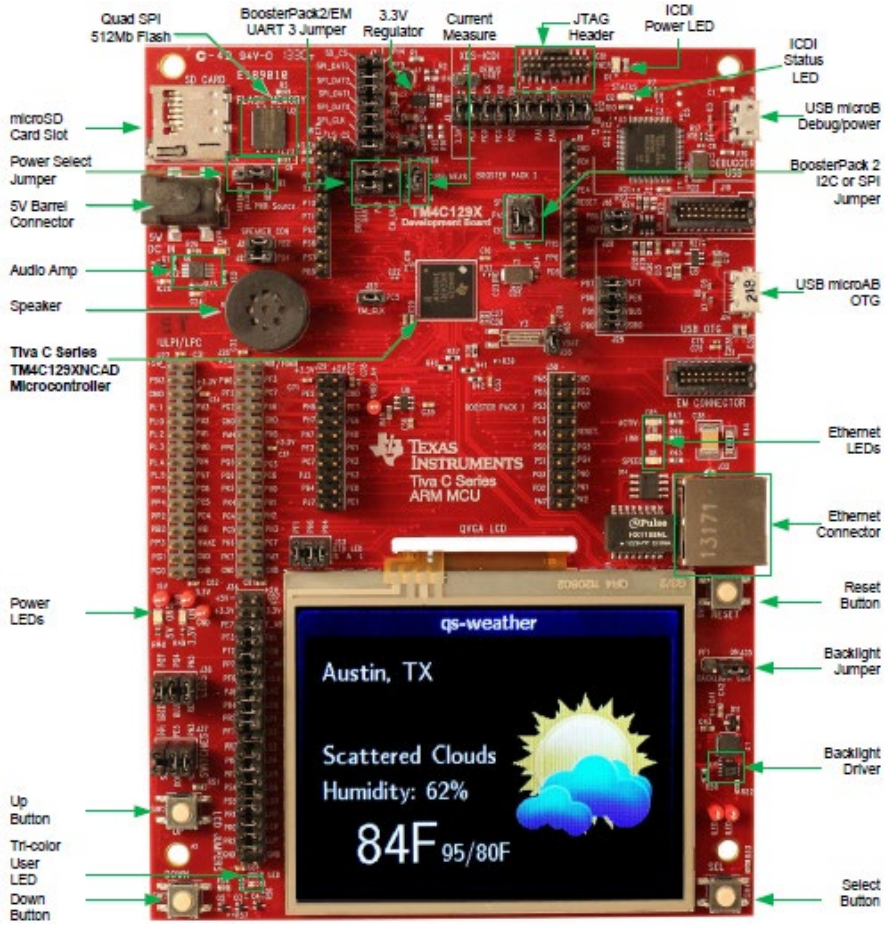

*Fuente: elaboración propia*

Figura 3 Especificación del producto: juego de tres en raya.

**División del diseño en componentes de hardware y software**

En cuanto a hardware, se utilizó la tarjeta de desarrollo DK-TM4C129X, la cual está soportada por el microcontrolador Tiva TM4C129X, que tiene un núcleo ARM® Cortex™-M4 de $32\,bits$ [TI_UG, 2013]. La Figura 4 muestra la tarjeta empleada, destaca la pantalla táctil, como elemento principal de interacción con el usuario para cumplir con las especificaciones del producto. El elemento principal de la tarjeta es el microcontrolador TM4C129X, el cual tiene un núcleo ARM® Cortex™-M4 que puede operar a $120\,MHz$. El MCU tiene un mapa lineal de memoria con diferentes

espacios: $1024\ KB$ de Flash para código, $256\ KB$ de RAM y $6\ KB$ de EEPROM para datos, así como una ROM interna con rutinas optimizadas para el manejo de periféricos. El TM4C129X cuenta con un extenso número de recursos, incluyendo puertos de entrada o salida (GPIO's), temporizadores, UART's, interfaces I2C, interfaces SPI y ADC's de $12\ bits$ [TI_DS, 2014].




*Fuente: [TI_UG, 2013]*

Figura 4 Tarjeta de desarrollo DK-TM4C129X.

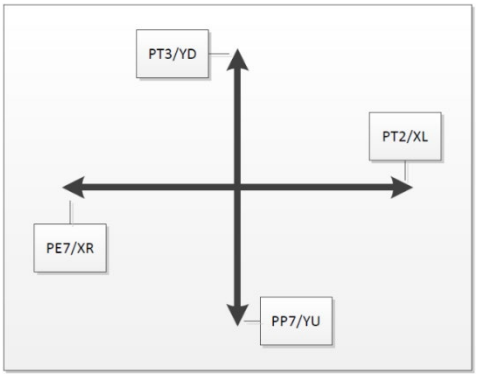
Respecto al software, el sistema requiere de tres componentes principales: El primero es la Biblioteca de Controladores Periféricos (*TivaWare Peripheral Driver Library*), que incluye funciones para la inicialización y el acceso a los recursos internos del MCU [TI_DRLib, 2014]. El segundo es la Biblioteca de Gráficos (*TivaWare Graphics Library*), con las funciones necesarias para el manejo de la pantalla, así como para la gestión de los eventos producidos en la zona táctil

[TI_GRLib, 2013]. El tercer componente de software se refiere a los controladores de la pantalla táctil, son dos bibliotecas desarrolladas por Kentec Electronics Limited, fabricante de la pantalla, la biblioteca Kentec320x240x16_ssd2119_8bit contiene funciones para el manejo del LCD y la biblioteca touch con funciones para la inicialización de la zona táctil [Kentec, 2024].

**Diseño detallado del hardware**

El DK-TM4C129X está equipado con una pantalla táctil resistiva Kentec K350QVG-V2-F, RGB de $320x240$, que es manejada por el controlador LCD del MCU TM4C129X, usando el controlador de pantalla de interfaz LCD (LIDD) en modo de $8\ bits$ [TI_UG, 2013].

La zona táctil funciona leyendo un eje a la vez, para ello se destinan 4 terminales del MCU y un ADC. En la Figura 5 se pueden identificar las terminales involucradas en el manejo de la pantalla, estas son PE7 (XR), PT2 (XL), PT3 (YD) y PP7 (YU). Para leer al eje X se debe configurar PE7 en alto, PT2 en bajo y PT3 en alta impedancia, el valor analógico se obtiene de PP7, los eventos que se generan más a la izquierda corresponden a voltajes más altos. El eje Y se lee configurando a PT3 en alto, PP7 en bajo y PT2 en alta impedancia, el valor analógico se obtiene en PE7 y los voltajes altos se obtienen en la parte superior de la zona táctil.




*Fuente: [TI_UG, 2013]*

Figura 5 Pantalla táctil resistiva.

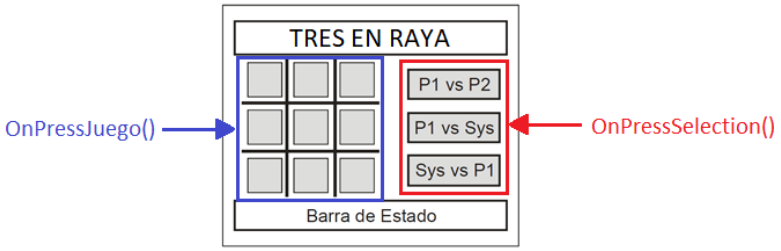
La generación de las salidas en la pantalla y la obtención de eventos se realiza con el apoyo de bibliotecas de funciones, lo que simplifica el manejo del hardware.

Durante la creación de los objetos que aparecerán en pantalla se indica su ubicación y la función que atenderá a los eventos que ocurran en esa zona. Por medio de interrupciones se hace el sondeo de la zona táctil y en un búfer de memoria se almacenan los eventos ocurridos, en el programa principal se debe invocar una función que analice la existencia de eventos para darles atención.

**Diseño detallado del software**

El estado del juego se almacena en cinco variables globales, un arreglo de 9 elementos llamado *board* con la información del tablero, y cuatro enteros, para almacenar el *modo* de juego, el *turno* del jugador, en número de *tiros* y una bandera para indicar si hay *ganador*. Como ya se ha mencionado, el sistema tiene 3 modos de juego y este se indica en la variable *modo* con los valores 1, 2 y 3, el valor 0 es una indicación de que no hay un juego en proceso.

El programa se puede dividir en 3 secciones, dos funciones para la atención de los eventos en la zona táctil y el programa principal. En la Figura 6 se indica a qué sección de la zona táctil atiende cada función, manejando arreglos con botones en cada caso.



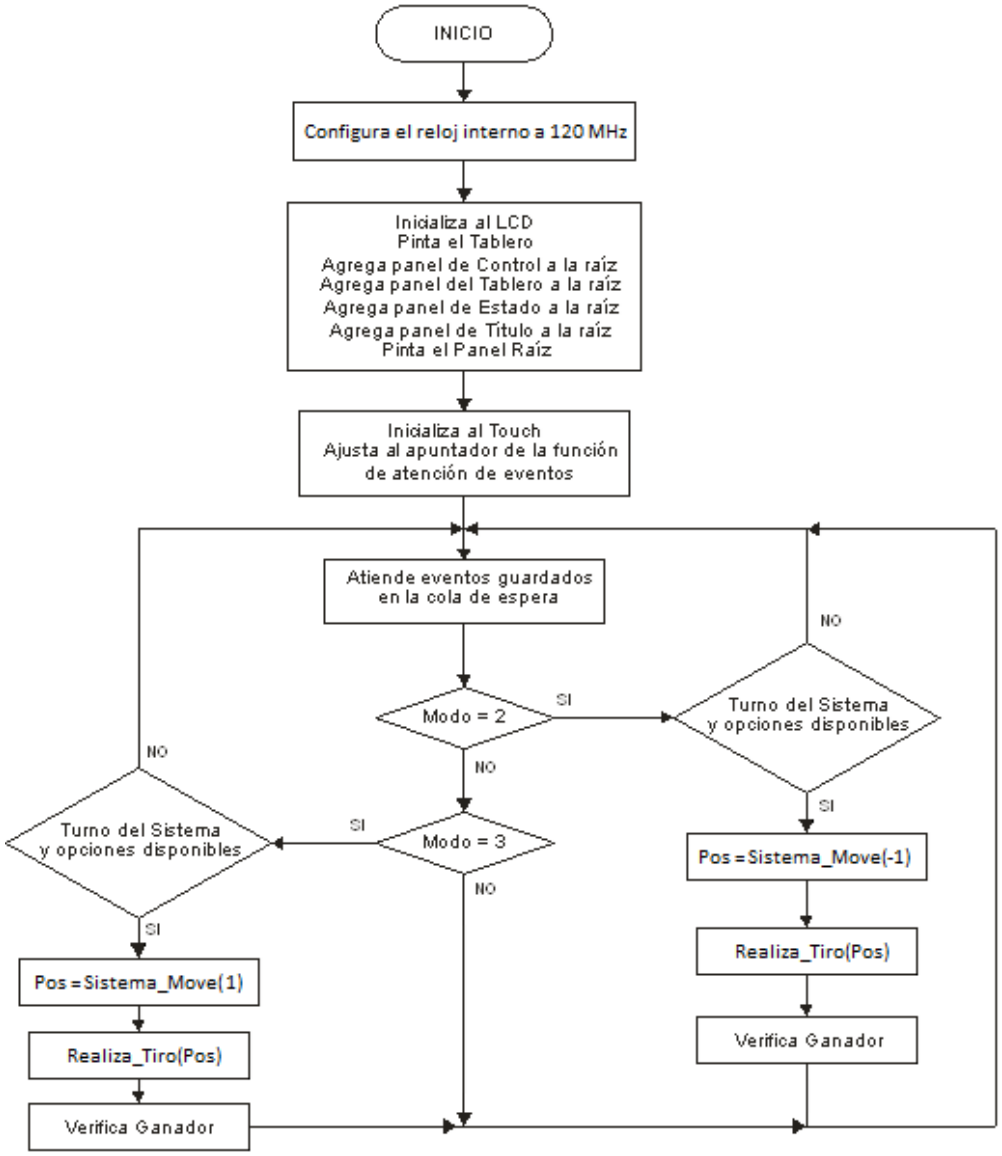

*Fuente: elaboración propia*

Figura 6 Funciones que atienden eventos según la zona de ocurrencia.

La función *OnPressSelection()* inicializa el juego, colocando ceros en el arreglo *board*, ajustando el *modo* y *turno* según el botón presionado, limpiando el número de *tiros* y la bandera *ganador*. La función *OnPressJuego()* evalúa si hay un juego activo (*modo* != 0) y si aún hay casillas para jugar (*tiros* < 9). Dependiendo del modo, revisa a quien le corresponde el turno para colocar la ficha, validando que el espacio seleccionado esté disponible. Si ya han ocurrido 5 o más turnos se analiza si hay

un ganador para ajustar el estado del juego. El modo 1 (jugador vs jugador) es desarrollado completamente por eventos. En los modos 2 y 3 (jugador vs sistema y sistema vs jugador) los eventos se manejan para los turnos del jugador, mientras que los turnos del sistema se realizan desde el programa principal. Si se selecciona el modo 3 con la función *OnPressSelection()*, el primer turno del sistema es generado automáticamente de manera aleatoria.

En la Figura 7 se presenta un diagrama de flujo con el comportamiento del programa principal.




*Fuente: elaboración propia*

Figura 7 Comportamiento del programa principal del sistema.

Los movimientos del sistema se determinan con la función *Sistema_Move()* que a su vez se apoya en la función *MiniMax()* para seleccionar la mejor opción de un tiro. En el modo 2 el sistema es minimizador por ser el segundo en turno, por ello, sus movimientos son solicitados con -1, mientras que en el modo 3 el sistema es maximizador y sus movimientos se solicitan con 1. Cuando no es el turno del sistema o no hay opciones disponibles en el tablero, el programa se centra en revisar los eventos de la zona táctil pendientes por ser atendidos. El cambio de turno se hará en la función *OnPressJuego()*.

La función *Sistema_Move()* coloca el valor del jugador en turno en una casilla libre y llama a la función *MiniMax()* con el jugador contrario para continuar con el juego, esto lo hace con todas las casillas disponibles y al final de la revisión regresa la mejor opción. En la Figura 8 se muestra el código de la función *Sistema_Move()*.

```
// Regresa el tiro del sistema
int Sistema_Move(int player) {

    int move = -1;
    int score = -2;
    int k;
    int tempScore;

    procesando = 1;
    for(k = 0; k < 9; k++) {
        if(board[k] == 0) {                    // Si hay casilla vacía
            board[k] = player;                 // Tira
            tempScore = -minimax(-1*player);   // Obtiene el efecto del tiro
            board[k] = 0;                      // Quita el tiro
            if(tempScore > score) {
                score = tempScore;             // Actualiza la posición con
                move = k;                      // un mayor beneficio
            }
        }
    }
    procesando = 0;

    return  move;                              // Regresa la posición
}
```

*Fuente: elaboración propia*

Figura 8 Función que obtiene la mejor posición para el tiro del sistema.

La inteligencia artificial del sistema está en la función *MiniMax()*, función recursiva que genera un árbol para evaluar todos los posibles tiros, cada rama termina cuando se encuentra un ganador o cuando ya no hay opciones disponibles en el tablero. En la Figura 9 se muestra el código de la función *MiniMax()*. La función *hay_ganador()*

regresa 0 si no encuentra combinaciones ganadores; o bien, -1 o 1, dependiendo del jugador que consigue el triunfo.

```
//Implementacion del algoritmo recursivo de MiniMax
int MiniMax(int player) {

int move = -1;
int score = -2;
int winner = hay_ganador();
int k;
int thisScore;

    if(winner != 0) return winner*player;       // Termina si hay ganador

    for(k = 0; k < 9; k++) {                    // Evalúa todas las opciones
        if(board[k] == 0){
            board[k] = player;
            thisScore = -minimax(-1*player);    // Llamada recursiva
                if(thisScore > score) {
                    score = thisScore;
                    move = k;
                }
            board[k] = 0;
        }
    }

    if(move == -1)                              // Empate
        return 0;

    return score;                               // Ganador
}
```

*Fuente: elaboración propia*

Figura 9 Función que implementa el algoritmo Minimax.

El producto *winner*player* será 1 si el jugador que hizo el tiro obtiene el triunfo y -1 si el triunfo es del contrario. El argumento en cada llamada de *MiniMax()* se cambia de signo por la alternancia entre los dos jugadores para cada rama del árbol, por el mismo motivo, el retorno de la función se conmuta para que corresponda con el nivel actual, considerando que un nivel es maximizador y el siguiente minimizador. La variable *score* inicia con -2, por lo que el triunfo de cualquier jugador o un empate dará un resultado mayor. La variable *move* inicia con -1 y mantiene su valor cuando no hay casillas libres para continuar con el juego, en esa situación la función *MiniMax()* regresa 0.

El entorno de desarrollo proporciona una biblioteca de gráficos que facilita colocar en la pantalla elementos primitivos como líneas y rectángulos, así como elementos interactivos, como los botones [TI_GRLib, 2013]. Con el apoyo de esta biblioteca se implementó la interfaz de usuario, que complementa al software del sistema.

**Integración HW y SW**

El programa para el microcontrolador se realizó en el entorno Code Composer Studio de Texas Instruments, se utilizó la versión 6.0 porque es la incluida con la tarjeta DK-TM4C129X. La integración del hardware con el software se realizó en forma satisfactoria, como lo demuestra la sección de resultados. El entorno permite hacer un análisis estático de la memoria utilizada por el proyecto y los resultados obtenidos se presentan en la Tabla 1, en donde se puede ver que hay espacio suficiente para aplicaciones con mayores requerimientos de procesamiento.

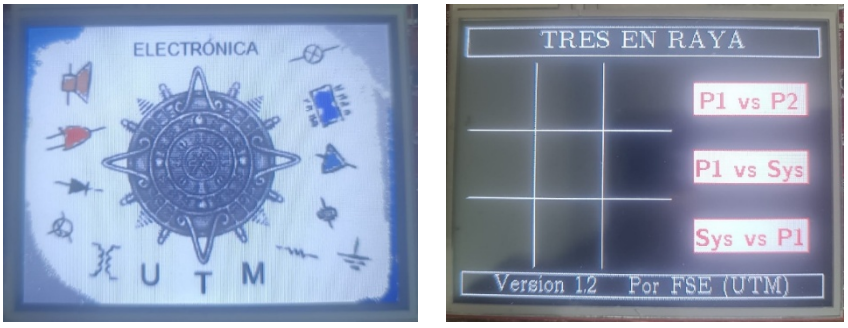
Tabla 1 Porcentaje en el uso de la memoria.

| Memoria | Disponibilidad | Uso | Porcentaje |
|---|---|---|---|
| FLASH | $1024\ Kbyte$ | $53{,}596\ bytes$ | 5% |
| SRAM | $256\ Kbyte$ | $2{,}588\ bytes$ | 1% |

*Fuente: elaboración propia*

## 3. Resultados

La Figura 10 muestra 2 pantallas iniciales, la primera es una imagen de bienvenida que se presenta durante $3\ s$. En la segunda pantalla se puede ver el tablero vacío y los botones de selección, en la barra de estado se indica la versión y autor del sistema. El área del juego será ignorada mientras no se seleccione el modo de competencia.



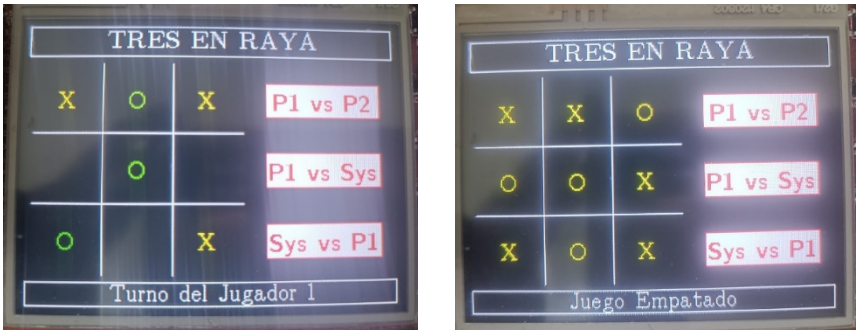

*a) Imagen de bienvenida al juego.* *b) Tablero vacío.*

*Fuente: elaboración propia*

Figura 10 Pantallas iniciales del juego.

La barra de estado va a presentar información diversa para dar seguimiento al juego, dos ejemplos se presentan en la Figura 11, en el primero se muestra la

indicación de que le corresponde el turno al jugador 1 y en el segundo se indica un empate entre los jugadores.




*a) Indicación del siguiente turno.* *b) Situación de empate.*

*Fuente: elaboración propia*

Figura 11 Pantallas iniciales del juego.

La Figura 12 presenta dos situaciones de triunfo, la primera corresponde al triunfo del sistema y la segunda al triunfo del jugador 1. En la barra de estado se indica quien es el ganador. El triunfo de un jugador solo se consigue en el modo Jugador vs Jugador, ya que en los otros dos modos resulta imposible ganarle al sistema.


*a) Triunfo del sistema.* *b) Triunfo del jugador 1.*

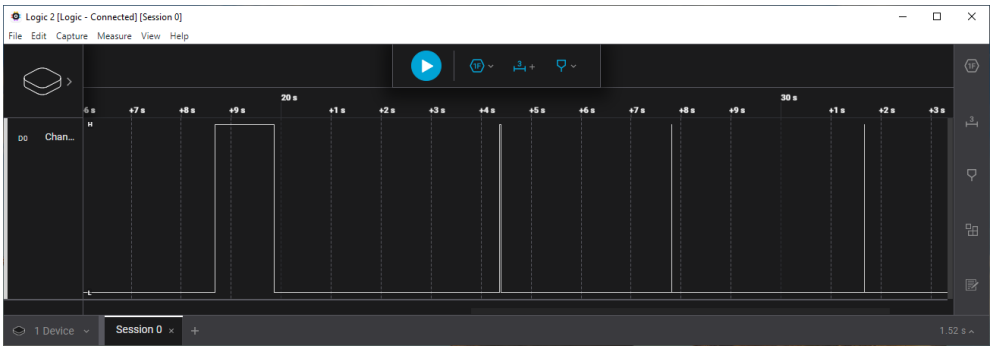
*Fuente: elaboración propia*

Figura 12 Dos situaciones de triunfo.

Una vez que el juego concluye, sin importar quién fue el ganador o si ocurrió un empate, se debe presionar uno de los botones de selección para iniciar con un nuevo juego.

Un factor importante por medir es el tiempo de respuesta del sistema, la peor situación se presenta en el modo jugador vs sistema, ya que después del tiro del

jugador quedan 8 posibilidades a analizar por el sistema, estas se irán reduciendo conforme se avanza en el juego por lo que la respuesta será más rápida. Para medir el tiempo de respuesta se dispuso de una salida digital que se pone en alto en los turnos del sistema, el comportamiento de esta señal se observó con un analizador lógico y en la Figura 13 se presenta un ejemplo de ello.




*Fuente: elaboración propia*

Figura 13 Comportamiento del tiempo de respuesta.

El analizador lógico permite ampliar el eje del tiempo para conocer con precisión su duración, con esta información se construyó la Tabla 2, en donde se puede ver que el sistema mejora significativamente la velocidad de respuesta en cada uno de sus turnos. Los turnos se muestran por pares porque los impares corresponden al jugador humano. La Tabla 2 demuestra que el sistema tiene una respuesta muy buena, aun en el peor de los casos, dado que el microcontrolador está trabajando a $120\ MHz$, genera y evalúa el árbol del juego a una velocidad muy alta en cada uno de los tiros.

Tabla 2 Tiempo que tarda el sistema en dar respuesta.

| Turno | Duración | Casillas que se revisan |
|---|---|---|
| *2* | 573.2923 $ms$ | *8* |
| *4* | 9.7335 $ms$ | *6* |
| *6* | 441.625 $\mu s$ | *4* |
| *8* | 45.375 $\mu s$ | *2* |

*Fuente: elaboración propia*

En el modo sistema vs jugador la primera tirada es inmediata porque se define con la generación de un número aleatorio para proporcionar versatilidad al inicio del

juego. Después debe tirar el jugador y en la próxima tirada del sistema solo tiene 6 posibilidades por explorar, con esta cantidad de niveles el tiempo de respuesta promedio de 5 mediciones fue de $71.107\ ms$.

Otro resultado a destacar es la evaluación del desempeño del sistema con diferentes usuarios. El sistema fue expuesto en ferias de promoción de la carrera de ingeniería en electrónica de la Universidad Tecnológica de la Mixteca y fue jugado por más de 50 estudiantes de nivel medio superior con los siguientes resultados: De las veces en que el sistema inició el juego, en el $70\%$ de los casos obtuvo la victoria y en el $30\%$ restante el resultado fue un empate. Cuando el usuario inició el juego, en el $80\%$ de los casos se obtuvo un empate y en el $20\%$ restante el sistema obtuvo la victoria.

## 4. Discusión

Este trabajo se enfocó en la descripción detallada del algoritmo de Minimax para aplicarlo en el juego de tres en raya e implementarlo en un microcontrolador, esto con la finalidad de integrar inteligencia artificial en un sistema embebido. Los resultados que se obtuvieron fueron satisfactorios, el sistema se utilizó como medio de promoción académica y su funcionalidad resultó muy atractiva para alumnos de diferentes niveles, despertó un amplio interés en los usuarios, quienes se sorprendían al no poder vencer al sistema y solo lograr un empate como su mejor resultado, aun siendo ellos quienes ejecutaban el primer tiro.

El algoritmo de Minimax aparenta ser muy simple, sin embargo, requiere del conocimiento de árboles y recorridos, esto dificulta su comprensión inmediata. El juego de tres en raya puede ser un buen ejemplo para un curso de estructuras de datos y, en cursos posteriores, el juego de tres en raya se puede implementar en un microcontrolador.

Como ya se ha mencionado, durante la creación y recorrido del árbol se van alternando los niveles máximizador y minimizador, en la literatura e internet se pueden encontrar versiones que agregan una variable como argumento adicional al algoritmo para indicar si se está o no haciendo una maximización, de esta manera se obtiene un código más simple pero más extenso. En el código desarrollado no

se agregó esta variable para crear un programa compacto, la alternancia entre maximizador y minimizador se consigue con el cambio de signo en cada llamada a la función Minimax.

## 5. Conclusiones

La implementación de un sistema embebido para el juego de tres en raya demostró que es posible incorporar un algoritmo de IA en un microcontrolador. A pesar de ser un algoritmo muy antiguo, el algoritmo de Minimax puede ser un punto de partida para entender como dotar de inteligencia a los sistemas, aspecto que actualmente es interesante por las múltiples herramientas disponibles para IA.

La tarjeta de desarrollo DK-TM4C129X resultó muy adecuada para la implementación del sistema, el microcontrolador fue configurado para trabajar a $120\,MHz$, que es su velocidad máxima, y en los resultados se puede ver que el tiempo de respuesta es muy corto. Solo en el peor escenario la espera es superior a medio segundo, en los demás aparenta ser inmediata.

La metodología empleada ayudó a desarrollar el proyecto de manera ordenada, fue aplicable a pesar de que la tarjeta DK-TM4C129X fue el único elemento de hardware requerido.

La revisión del estado del arte permitió conocer la existencia de otras técnicas de Inteligencia Artificial que se están enfocando a los sistemas embebidos y el desarrollo de este sistema demostró que la tarjeta empleada tiene una alta capacidad de procesamiento, estos dos aspectos abren un abanico de posibles trabajos futuros, ya que, con la misma tarjeta de desarrollo se podrían aplicar otras técnicas de IA para el juego de tres en raya u otras aplicaciones, también se podría buscar emplear el algoritmo Minimax para resolver problemas diferentes, Otra idea puede ser la expansión del juego de tres en raya a un nivel de 3D, conmutando entre pantallas con el apoyo de uno de los botones incluidos en la tarjeta.

## 6. Bibliografía y Referencias

[1] AIfES, (2024). Fraunhofer Institute for Microelectronic Circuits and Systems, Artificial Intelligence for Embedded Systems,

https://www.ims.fraunhofer.de/en/Business-Unit/Industry/Industrial-AI/Artificial-Intelligence-for-Embedded-Systems-AIfES.html, [Último acceso: Agosto de 2024].

[2] Berger A. S., Embedded Systems Design: An Introduction to Processes, Tools, and Techniques. CMP Books, Lawrence, Kansas, 2002.

[3] Kentec Display, (2024). http://www.kentecdisplay.com/. [Último acceso: agosto de 2024].

[4] Iodice G., M., TinyML Cookbook: Combine artificial intelligence and ultra-low-power embedded devices to make the world smarter, Packt Publishing, 2022.

[5] Pineda C., Luis Alberto, Racionalidad Computacional. Academia Mexicana de Computación, 2021.

[6] Rich E., Knight K., Inteligencia Artificial, Segunda edición, McGraw Hill, 1994.

[7] Russell, S. J., Norvig, P., Inteligencia Artificial: Un Enfoque Moderno, Segunda edición, Pearson Educación, 2004.

[8] TI_DRLib (Texas Instruments, Driver Library), TivaWare™ Peripheral Driver Library. User's Guide, 2014.

[9] TI_DS (Texas Instruments, Data Sheet), Tiva TM4C129XNCZAD Microcontroller, 2014.

[10] TI_GRLib (Texas Instruments, Graphics Library), TivaWare™ Graphics Library. User's Guide, 2013.

[11] TI_UG (Texas Instruments, User's Guide), Tiva™ TM4C129X Development Board, Literature Number: SPMU360, October 2013.

[12] TRBL Services, (2023). ¿Se puede aplicar Inteligencia Artificial en sistemas embebidos?, https://www.linkedin.com/pulse/se-puede-aplicar-inteligencia-artificial-en-sistemas/, [Último acceso: agosto de 2023].

[13] Winston, P. H., Inteligencia Artificial, Tercera edición, Addison-Wesley Iberoamericana, 1992.

[14] Wulfert L., Kühnel J., Krupp L., Viga J., Wiede C., Gembaczka P., Grabmaier A., AIfES: A Next-Generation Edge AI Framework, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 46, No. 6, June 2024.