Implementación de un tablero de ajedrez basado en microprocesadores y microcontroladores

Jeaustin Calderón Quesada, Gabriel Campos Oviedo, Esteban Chavarría Chaves, Diana Cerdas Vargas, Emmanuel Muñoz Peña
jeauscq@estudiantec.cr gabrielitooviedito@estudiantec.cr echavarriach@estudiantec.cr, diana@estudiantec.cr, emmanuel111232@estudiantec.cr,
Área académica de Ingeniería Mecatrónica
Instituto Tecnológico de Costa Rica

***Resumen*—Un microprocesador es capaz de procesar una gran cantidad de instrucciones y diferentes procesos simultáneos en un solo chip integrado. Por otro lado, los microcontroladores se encargan de una tarea más específica y solo manejan un programa dedicado a la aplicación. Se desarrolló un proyecto en el cual se utiliza un microprocesador y un microcontrolador para la implementación de un tablero de ajedrez físico y una interfaz que permita una asistencia de juego y un modo de juego contra la máquina. Esto se logró mediante programación en Python para el microprocesador, programación en Arduino para el microcontrolador, el protocolo de comunicación UART para la transmisión de datos entre ambos y la paralelización del ciclo principal y el ciclo de comunicación.**

***Palabras clave*—Interrupciones, Raspberry Pi, ESP 32, UART**

I. Introducción

Los microprocesadores y microcontroladores son componentes esenciales en los proyectos, sistemas mecatrónicos y computadoras en general. Los microprocesadores se definen como un circuito integrado capaz de procesar instrucciones y formar una unidad de procesamiento [1]. Por otro lado, los microcontroladores son dispositivos electrónicos completos en un solo chip diseñados específicamente para el control y monitoreo de sistemas en tiempo real y, a diferencia de los microprocesadores, estos manejan un solo programa destinado a esta aplicación [2].

Se tiene que ambos desempeñan un papel esencial en el desarrollo del proyecto, siendo el microprocesador el encargado de realizar el procesamiento de los datos, instrucciones y el análisis del tablero en tiempo real. En contraste, el microcontrolador se encarga del manejo de interrupciones y el control de la matriz de luces LED con los datos recibidos mediante comunicación serial por parte del microprocesador.

Los microprocesadores actuales cuentan con diferentes núcleos de procesamiento que logran la realización de múltiples tareas de forma eficiente y simultánea, esto se logra mediante la paralelización de las diferentes instrucciones y la asignación de estas a los diferentes núcleos o mediante hilos de instrucciones. En el caso del lenguaje de programación Python, se tienen diferentes módulos que dan soporte a la programación basada en hilos y procesos, entre estos *threading* que se basa en la programación mediante hilos. Los hilos, como lo expresa [3] son un flujo de control activo que se puede activar en paralelo con otros subprocesos dentro del mismo proceso, siendo un proceso una instancia de ejecución de un programa.

Por otro lado, para la comunicación entre el microcontrolador y el microprocesador se utilizó el protocolo de comunicación de receptor/transmisor asíncrono universal (UART por sus siglas en inglés), el cual consiste en la transimisión de datos de manera asíncrona a través de dos líneas: la línea de transmisión (TX) y la línea de recepción (RX), con un mecanismo de detección de bits de parada y de inicio para dar una sincronización de datos sin la necesidad de una señal de reloj. [4]

En la solución realizada, el paralelismo de los procesos permitió la elaboración de la comunicación entre el microprocesador y el microcontrolador ya que lograba separar el ciclo infinito de la interfaz y el de comunicación. Asimismo, esta comunicación fue basada en el protocolo UART para establecer una comunicación bidireccional entre ambos dispositivos, en la cual el microprocesador enviaría las instrucciones y movimientos al microcontrolador a través de una línea de transmisión y el microcontrolador recibiría los comandos y ejecutaría los movimientos correspondientes en el tablero de luces LED.

II. DESCRIPCIÓN DE LA SOLUCIÓN

II-A. DISEÑO DE LA INTERFAZ

Para el desarrollo de la interfaz se propuso utilizar la biblioteca pygame, la cual es una librería de código abierto que contiene módulos de lenguaje para el manejo de objetos gráficos 2D sobre una ventana, la carga de ficheros e imágenes y la detección directa de interacción con objetos periféricos como el mouse o el teclado.

Para fines del proyecto, inicialmente se definieron como variables globales lo diferentes colores existentes dentro de tablero(blanco, azul, verde, verde oscuro, amarillo, rojo y negro). Posteriormente, se inicializó el display del tablero con un determinado tamaño de 512x512 pixeles de tamaño; esto es acorde a lo necesario para el tamaño de 8x8 que se desea en el tablero.

Para simplificación del trabajo, se definió que las formas de las piezas fueran imágenes precargardas de tipo png, por lo que es necesario que a la hora de ejecutarse el programa,

exista un archivo que contiene las imágenes definidas en la misma carpeta que contiene la programación. Adicionalmente, a la hora de cargarse las imágenes del juego, se realice una única vez para evitar posibles errores de lag durante la ejecución; las imágenes se escalan a un tamaño adecuado para su visualización y se almacenan en un diccionario global para su posterior uso en otras partes del sistema.
Por otro lado, se define un reloj del sistema con una cantidad máxima de 15 FPS para la animación del juego y actualización del display de la pantalla según las actualizaciones del estado del tablero que ocurran durante la partida.
Una vez se haya generado el display, diferentes funciones cargan la matriz del tablero y recorren por completo cada fila y columna y acorde a ello, se generan cuadros de diferentes colores(el color de una casilla llena o vacía, el color de una casilla para un movimiento inválido, movimiento posible o una casilla que en la que puede comer) y se pintan las diferentes piezas según la posición de la matriz indicada.
Dentro de diferentes partes de código, se implementa el módulo de eventos con periféricos, esperando en cada ejecución a que exista un evento definido. Para la implementación, fueron necesarios los eventos de un click en la pantalla, que detectan lo deseado por el usuario guardando las direcciones de posición dentro del display en dónde se generó el click y para almacenarlo en otras variables para su posterior interpretación en otras partes del código.
Finalmente, mediante la librería pygame también se aplican las alertas existentes en el programa, que corresponde a objetos de tipo font que muestran al usuario mensajes en pantalla que indican una alerta al usuario en caso de movimientos no válidos, que no es el turno o el mensaje que indica la victoria del lado ganador.

II-B. MOVIMIENTO DE PIEZAS

El movimiento de piezas es particular para cada tipo de pieza, por lo que se utilizó POO, de esta manera, al seleccionar la casilla se crea un objeto del tipo de pieza y se usan algoritmos parametrizados que permiten encontrar las casillas a las que se pueden mover o capturar otra casilla. Las posiciones se definen con una matriz de arreglos, por lo que de esta manera se distinguen el color y el tipo de pieza en cada una de las matrices.

II-C. DESARROLLO MODO ASISTENTE

El modo asistente corresponde a una alternativa que permite al usuario poder jugar bajo condiciones controladas. Dentro de sus modalidades, permite observar los movimiento válidos e inválidos de cada una de las piezas con solo presionarlas, permite esto para cualquiera de los grupos de pieza, inclusive si no es el turno de dicha pieza. Dentro de los movimiento contemplados se encuentran los movimientos correspondientes a cada una de las piezas, la comida al paso, el enroque y la promoción, además detecta cualquier movimiento que dejaría en jaque al rey o que no defiende al rey en caso de que ya se encuentre en jaque. Además, en caso de que el usuario seleccione un movimiento inválido, se va a mostrar una alarma que indica la razón por la que el movimiento no es válido. En la figura 2 se muestra el diagrama de flujo detrás de la lógica del modo asistente.

II-D. DESARROLLO MODO JUEGO

El modo de juego, como está ilustrado en la figura 3, cuenta con la diferencia de que turno de por medio se debe hacer un movimiento automático que tenga legalidad en el juego. Para garantizar este requisito, se toma una casilla al azar, se verifica que sea del equipo con el que juega la PC, se consultan los movimientos disponibles y se escoge uno al azar para seguidamente consulta su validez, si es válido entonces realiza dicho movimiento.

II-E. DESARROLLO MODO ASIGNAR TABLERO

Se buscó el desarrollo del modo de asignación de tablero, en dónde el software permite recibir un tablero inicial con la posición, cantidad y tipo de piezas definidas manualmente por el usuario y ser capaz de presentarlo a la hora de iniciar la partida.

Para ello, inicialmente se propuso presentar una nueva ventana previo al inicio de la partida, la cual presenta dos casillas al usuario permite al usuario para seleccionar si quiere asignar por defecto al tablero oficial de ajedrez o si desea realizar la asignación manual. En caso de seleccionar la primera opción, la matriz de tablero se mantiene sin cambios y en caso de marcar la segunda casilla, se vuelve a cargar una nueva ventana cargando una matriz de tablero de dimensión de 10x8. El tablero se presenta con dos principales áreas: área de tablero vacío de 8x8 y área de seleccionador de 2x8 que contiene todas las piezas disponibles separadas por color.

Para la lógica de colocación o traslado de las piezas, se utiliza la misma lógica explicada en la sección II-B. En caso de que se quiera colocar alguna pieza desde el área de selección o se quiera mover una pieza dentro del área de tablero, se espera por un primer evento de click sobre la posición de alguna pieza, lo que guardará la clase de la pieza como el origen que se desea poner/mover y se espera por un segundo evento de click en algún cuadro del tablero vacío y cuando esto ocurra, se actualiza a la matriz del tablero con una pieza coincidente a la de origen en dicha posición.

Se aplicó un control de diferentes condiciones para la colocación de nuevas piezas:

- Se evita una sobreescritura en el caso en que el primer click del usuario se dio en el área del seleccionador y su segundo click intenta sobreescribir alguna de las piezas del seleccionador.
- Se evita una sobreescritura en el caso en que el primer click del usuario se dio en el área del tablero y su segundo click intenta sobreescribir alguna de las piezas del seleccionador
- En el caso en que el primer click del usuario proviene de una pieza colocada en área del tablero y su segundo click en algún cuadro del tablero(esté vacío o no), se sobrepone

borrando lo que había antes y limpia el cuadro en el que la pieza de origen se encontraba.

Por otro lado, se realiza una lógica de control de condiciones iniciales de ciertas piezas que serán útiles para la posterior partida. En este caso, se vigila en el caso en el que el usuario coloque a las torres o reyes (de cualquier color) en sus posiciones iniciales del tablero tradicional (ejemplo, caso en el que intente colocar a una torre blanca en la esquina inferior izquierda). En dicho caso, se muestra ante el usuario una nueva ventana, consultando si desea definir que ésta nueva pieza ya se ha movido o no.

Esto se implementa para respetar las reglas del movimiento especial de enroque, que dicta que tanto el rey como la torre con la que se intente realizar esta jugada, no deben haberse movido en toda partida; por tanto, según lo que defina el usuario, se almacenará la condición del rey/torre, permitiendo que se pueda o no ejecutar la jugada posteriormente en la partida.

Adicionalmente, en caso de que se haya consultado previamente al usuario, pero se genere algún cambio, se define que la pieza NO se encuentra en su posición inicial si se dan los siguientes casos:

- El usuario no ha colocado ninguna pieza en dicha posición.
- El usuario ha colocado otra pieza que no corresponde en dicha posición.
- El usuario ha borrado la pieza en dicha posición.
- El usuario ha movido la pieza a otra posición.

Luego, se aplica un control de la cantidad máxima de piezas que se permite colocar dentro del tablero para cada tipo y color. Esto para respetar las posibilidades lógicas dentro de una partida. Al existir la posibilidad de aplicar la jugada especial de promoción, se debe adecuar la cantidad piezas máximas acorde la cantidad de promociones disponibles; por ejemplo, se puede permitir colocar más de una reina, puesto que se puede inferir que la segunda reina existe debido a una promoción, sin embargo, se limita esta posibilidad si ya hay más promociones realizadas que peones disponibles.

Se determina si el número de promociones que se han realizado, ha superado la cantidad de peones aún disponibles para colocar. En caso de superarse dicho límite, la posibilidad de colocar más promociones se inhabilitará; por ejemplo, si ya se ha colocado un peón y nueve torres (dos por defecto y siete por promoción), se bloquea poder colocar una octava torre o poder colocar un segundo peón.

Finalmente, se define que no es posible colocar más de un rey por color y que no es posible asignar un tablero para jugar una partida que no posea ambos reyes.

La lógica del programa se puede observar en el diagrama de flujo presentado en la figura 1 de los anexos.

II-F. IMPLEMENTACIÓN DE COMUNICACIÓN MICROPROCESADOR - MICROCONTROLADOR

La comunicación entre el microprocesador (Raspberry Pi) y el microcontrolador (ESP-32) se dio mediante el uso del protocolo UART, utilizando el puerto USB como salida de la Raspberry Pi, un cable USB a micro USB para la transmisión de datos y la entrada micro USB del microcontrolador. Se eligió este método debido a que presentaba una forma conveniente para realizar la comunicación de datos mediante el protocolo UART, que era una de las opciones permitidas para el desarrollo de esta parte, ya que evitaba el uso de cables conectados a los puertos GPIO de ambos dispositivos, ya que estos pueden llegar a tener conexiones inestables si no son soldados en el puerto, dando paso a interrupciones en la comunicación o errores en los datos enviados y/o recibidos.

Se dio una transmisión de la matriz de caracteres utilizada en la programación principal para la determinación del estado del tablero. Esta transmisión de datos se dio mediante la conversión de esta matriz en un *string*(Tipo de dato formado por un arreglo de caracteres) terminado en "\n" (Indica el final del dato para el microcontrolador), esto debido a que mediante comunicación serial solo se pueden transmitir datos seriales y no una matriz completa de datos .

Consiguientemente, se realiza la codificación de este dato en UTF-8 la cual es la codificación de caracteres de longitud variable utilizada para la transmisión, logrando así la conversión de los caracteres a bits interpretables por el microcontrolador. Luego de esto se envía el string de datos y se recibe en el ESP-32 el cual se encarga rehacer la matriz mediante la fórmula:

$$board[i][j] = datos.substring((16{*}i{+}2{*}j), (16{*}i{+}2{*}j){+}2) \quad (1)$$

Donde i y j son la fila y columna correspondientes de la matriz y estos van en un rango de 0 a 7. De esta forma se vuelve a tener una matriz de datos que representa el estado actual del tablero y con esto se determina el color correspondiente para cada LED del tablero. Se tiene que las conexiones dadas para la comunicación pueden ser apreciadas en el anexo 4.

II-G. PARALELISMO DEL SISTEMA

Dentro del software, se realizó la implementación del paralelismo mediante hilos y se definieron dos principales:

- **Hilo del ciclo principal**: Toda la ejecución del main principal correspondiente al manejo de la interfaz, movimiento de piezas, la interacción con el usuario, los modos asistente, juego y asignar tablero.
- **Hilo de comunicación procesador - controlador**: Toda la ejecución de la función de comunicación serial con el ESP-32 y todo lo explicado en la sección II-F.

Una vez ya se hayan inicializado ambos hilos, el software ejecuta de manera paralela los ciclos infinitos de comunicación serial y de la ejecución del ciclo principal.

II-H. MONTAJE DEL TABLERO FÍSICO

Para la implementación física del tablero se utilizó una tira LED WS2812B por la posibilidad de ajustar el color del

componente rojo, verde y azul direccionado individualmente para cada led con precisión PWM de 8 bits. Además cada píxel se puede cortar sin dañar el resto de píxeles y volverlos a conectar soldando de extremo a extremo los contactos JST-SM de 3 pines, lo que permite colocarlas en paralelo para generar una matriz 8x16.

Asimismo, se creó un tablero tipo caja de construcción plegable con sujeciones geométricas precisas, fabricado mediante corte láser en material MDF en el que se colocará la matriz de LEDs dentro. La cara superior posee agujeros cuadrados en cada casilla, alineados con cada píxel LED en su interior, de forma que se aprecie la pieza y el bando (blancas o negras) de la misma. Este además posee marcado el tablero de ajedrez clásico y un código de colores para cada pieza.

II-I. MANEJO DE INTERRUPCIÓN

Para el manejo de interrupciones en el software del microcontrolador, se propuso el uso de un botón externo para ejecutar la funcionalidad del salto de turno. Para ello, se conectó un botón a uno de los pines del ESP-32 y se configuró ese pin como entrada de resistencia pull-up, es decir, que el estado del pin se mantiene en alto hasta que se presione el botón.

Posteriormente, una función es adjuntada a la interrupción ocasionada por el pin conectado al botón, y se activa con la condición de flanco negativo, es decir, cuando se presiona el botón. La función adjunta determina que la interrupción es existente y por ende, se imprime de manera serial el string ”saltar_turno”. La impresión serial del string anterior, es enviada para su lectura a través de la comunicación serial explicada en la sección II-F y funciona como ïndicador.[en] el software del microprocesador que ocurrió una interrupción en el microcontrolador, lo que activa las correspondientes variables para ejecutar el salto de turno.

Adicionalmente, en el software del microprocesador, se implementó un pequeño control de tiempo para evitar posibles errores a la hora de presionar el botón o imperfecciones de la capacidad del botón; puesto que al trabajarse con tiempos tan pequeños (unidades de milisegundos) si no se quita el dedo con suficiente velocidad, puede interpretarse que el botón se presiona dos veces existe y se genera un error al ejecutarse la interrupción dos veces cuando se presiona una única vez, por lo que se agrega un breve delay temporal para evitar que las interrupciones se puedan ejecutar de manera casi simultanea o una detrás de otra de manera seguida en un lapso muy corto.

III. ANÁLISIS DEL DISEÑO

Para este proyecto se tenían dos partes, la primera es todo el algoritmo que conlleva el juego y la segunda sería un tablero físico que represente lo que se muestra en el monitor. A continuación se va a hacer un análisis del producto final obtenido.

Primeramente, para el diseño del tablero físico se consideraron diferentes formas para realizarlo. La primera fue utilizar un tablero de ajedrez convencional en el cual las piezas se mueven por medio de un sistema con un principio de funcionamiento similar al de una CNC. También se pensó en hacer un brazo robótico con varios servomotores para permitir un amplio rango de movimiento. Como tercera opción se pensó en utilizar LEDS RGB para representar las fichas.

Esta última era la opción que simplificaba más la solución, ya que no involucra partes móviles, además, los LEDS son componentes electrónicos de fácil acceso y baratos. No obstante, se tenía como obstáculo el hecho de que se necesitaría como mínimo 64 LEDS para poder representar todas las casillas del tablero de ajedrez y cada LED RGB tiene tres patillas (una para cada color), por lo que al final se iba a necesitar un microprocesador con un gran número de patillas de salida.

Luego de investigar opciones se encontraron las cintas de LEDS utilizadas para proyectos de Ambilight. Estas se conectan de forma serial y por medio de una señal de PWM se puede controlar el color de cada uno de los LEDS, reduciendo así la cantidad de pines necesarios a 3 (5V, GND y un pin de PWM). Otra ventaja que tienen estos LEDS es que existen librerías tanto en Arduino como en Python que facilitaron su programación. Lo que se utilizó en este caso, fue un gran Switch Case en el cual se diferenciaba cada pieza y su color (blanca o negra). Luego, con una única línea de código se le asigna al número de LED específico la cantidad de rojo, verde y azul necesaria para crear el color deseado. Además de esto, se optó por añadir un LED extra a cada casilla para poder diferenciar entre si la ficha era blanca o negra.

Por otra parte, para la parte del algoritmo, se optó por utilizar Python como lenguaje de programación, ya que se estaba más familiarizado con su sintaxis en comparación a C++. Ahora, para la parte gráfica del juego se hizo uso del complemento de Python llamado Pygame. Este se seleccionó principalmente porque se encontraron en internet una amplia variedad de ejemplos que sirvieron para ver cómo funcionaba el complemento y diversos comandos que sirvieron de base para comenzar a desarrollar el tablero, mostrar sobre él las diferentes piezas, así como moverlas a diferentes posiciones.

Una vez que se logró generar el display base, se comenzó con la lógica detrás de los movimientos válidos para cada pieza y con el modo asistente. Para esto, lo único que se utilizó como apoyo fueron las reglas generales del ajedrez y el display. Cuando ya se finalizó todo el modo asistente, se le agregaron algunas partes extra para poder realizar el modo juego (contra la PC). Como se mencionó anteriormente, lo que se hace es que la PC selecciona una pieza al azar y la mueve de acuerdo con sus movimientos válidos. Un punto de mejora hubiera sido agregarle algún tipo de inteligencia artificial para que sea más completo, un ejemplo es la librería Stockfish, la cual está especializada en ajedrez y utiliza algoritmos de búsqueda avanzados que le darían un mayor nivel de dificultad a este modo.

Además, después de haber desarrollado la mayor parte del código, se descubrió que existe una librería de código abierto en Python llamada python-chess que genera toda la lógica detrás de un juego de ajedrez. Aunque esta hubiera podido ayudar a simplificar el algoritmo usado, cabe resaltar que no cumplía con todos los requerimientos de diseño planteados, como por ejemplo, mostrar las alarmas de jugadas inválidas,

pintar las casillas válidas, inválidas, donde se puede comer, y las jugadas especiales.

Otra decisión importante para el proyecto fue la selección del microcontrolador, para esto, se analizaron varias opciones. Los principales requerimientos definidos inicialmente fueron: que estuviera disponible en el país, que contara con los protocolos de comunicación I2C, SPI o UART y que tuviera una salida de 5 V, tierra y al menos una salida de PWM. En la tabla I, se muestra un resumen de las principales características de tres microcontroladores diferentes. Como se puede apreciar, los tres superan los tres requisitos principales que se tenían, por lo que la selección final se hizo basándose en el precio y los lenguajes de programación, que eran las únicas diferencias significativas para esta aplicación. Por lo tanto, al final se utilizó el ESP-32, pues costaba $20 y se podía programar con el lenguaje de Arduino y Python, los cuales contienen las librerías para la programación de los LEDS.

Cabe resaltar que inicialmente se tenía planeado programar el microcontrolador con Python, pero investigando se notó que el ESP-32 tiene un mayor soporte para el IDE de Arduino, además de que es más simple de utilizar. Un problema que hubo durante la programación del microcontrolador fue la comunicación, ya que al inicio se intentó utilizar I2C, pero a la hora de implementarlo, la Raspberry Pi no detectaba al microcontrolador, luego se concluyó que era porque el ESP-32 no tiene soporte nativo para ser esclavo, sino que este se hacía por medio de una librería y esta tenía ciertos errores. Otra razón por la que se decidió no utilizar el protocolo I2C fue porque existe una diferencia de tensión entre el ESP23 (opera a 5 V) y la Raspberry Pi (brinda 3.3 V), entonces se necesitaba usar un convertidor DC-DC y se corría el riesgo que con la interrupción o por algún error se quemara la Raspberry Pi. Como solución a esto, se optó por cambiar el protocolo de comunicación a UART.

Durante el desarrollo de la comunicación entre el microprocesador y el microcontrolador se definió que se iba a utilizar polling, ya que se quería que el microcontrolador estuviera refrescando constantemente la cinta del LEDS y que si en algún momento se desactualizaba el tablero, que se pudiera cambiar rápido y que no afectara la experiencia del usuario. Sin embargo, luego se notó que el código principal del juego funcionaba con polling, por lo que se tomó como oportunidad para realizar la paralelización. Entonces, mientras un hilo manejaba toda la parte del juego, la interfaz y demás, el otro manejaba la comunicación y por ende las dos tareas se hacían de forma simultánea sin depender directamente una a la otra.

Por último, durante la revisión del proyecto, se descubrió un Bug en el programa. Se hizo un salto de jugada cuando el rey blanco estaba en jaque, por lo que no se le dio la oportunidad de defenderse y al ser el turno de las negras, un caballo tenía la posibilidad de comerse al rey y cuando se intentó hacerlo, el código falló. La razón de esto puede ser porque en ninguna parte del código se contempló la posibilidad en una jugada suceda lo comentado, ya que en la realidad no se puede hacer un salto de turno, por lo que las blancas deben defenderse obligatoriamente o bien se llega a un jaque mate y el juego termina.

IV. CONCLUSIONES

1. El microcontrolador ESP-32 no posee soporte de slave en I2C nativamente, por lo que se utiliza protocolo tipo RS-232 UART.
2. Es posible dividir tareas utilizando paralelización en subtareas más pequeñas, ejecutándolas simultáneamente para mejorar la eficiencia y el rendimiento de los programas.
3. Las interrupciones permiten una respuesta más rápida y un manejo de recursos más eficiente que la lógica polling ante eventos externos.

V. RECOMENDACIONES

1. Para reducir el trabajo de programación se puede utilizar la librería de ajedrez de Python y añadirle las funciones extra, pues al ser de código abierto es posible modificarla.
2. Se puede añadir la condición de que el rey no esté en jaque para saltar turno con el fin de evitar que ocurra un fallo en el programa, ya que en ese momento será posible comer a la pieza del rey.
3. Se puede utilizar IA con el fin de que la computadora realice movimientos coherentes y con la intención de ganar.

VI. CITAS Y/O REFERENCIAS

REFERENCIAS

[1] S. Harris,, & D. Harris, Digital design and computer architecture. Morgan Kaufmann. 2007

[2] J. R. Tocci, & S. N. Widmer, Sistemas digitales: principios y aplicaciones. Pearson Educación. 2003

[3] G. Zaccone, Python parallel programming cookbook. 2015 Packt Publishing Ltd.

[4] Y. Fang, & X. Chen,Design and Simulation of UART Serial Communication Module Based on VHDL. 2011, 3rd International Workshop on Intelligent Systems and Applications. doi:10.1109/isa..5873448

VII. ANEXOS

Cuadro I: Principales características de los tres mejores microcontroladores encontrados.

Aspecto	Microcontrolador 1 ESP-32	Microcontrolador 2 Teensy 3.2	Microcontrolador 3 Feather M0
Tensión de alimentación	5V o 3.3V	1.71 V a 3.6 V	3.3 V
Número de entradas analógicas	15 con 18 Bits	21 entradas de alta resolución.	10
Número de salidas analógicas	2 con 18 Bits	1 con un ADC de 12 bits	1
Pines GPIO	34	34	20
Costo	$ 20	$ 32.95	$ 42.95
Programación	C, C++, Python, Lua, JavaScript, Arduino IDE, Espressif	C o con el Add-on Teensyduino para usar el IDE de Arduino	Python
Pines de PWM	16	12	8
Protocolos de comunicación	SPI, UART, I2S y I2C.	SPI, I2C, I2S, CAN Bus, IR modulator	Serial, I2C y SPI
Comunicación serial por medio de USB	Sí	Sí	Sí
Memoria RAM	520 KB	64 KB	32 KB
Memoria Flash	4 MB	256KB	256 KB
Clock	160 MHz	72 MHz	900 MHz

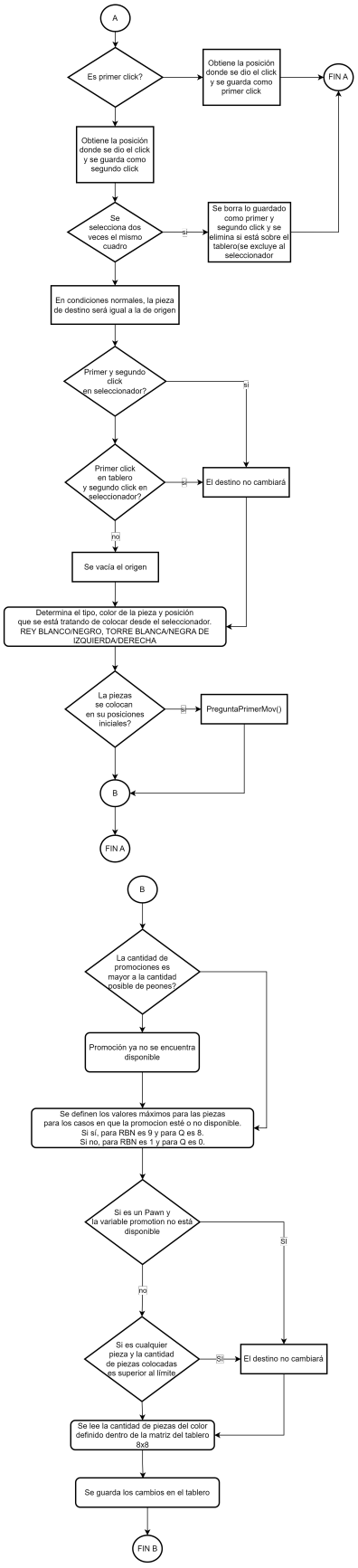

Figura 1: Diagrama de flujo del modo asignar tablero.

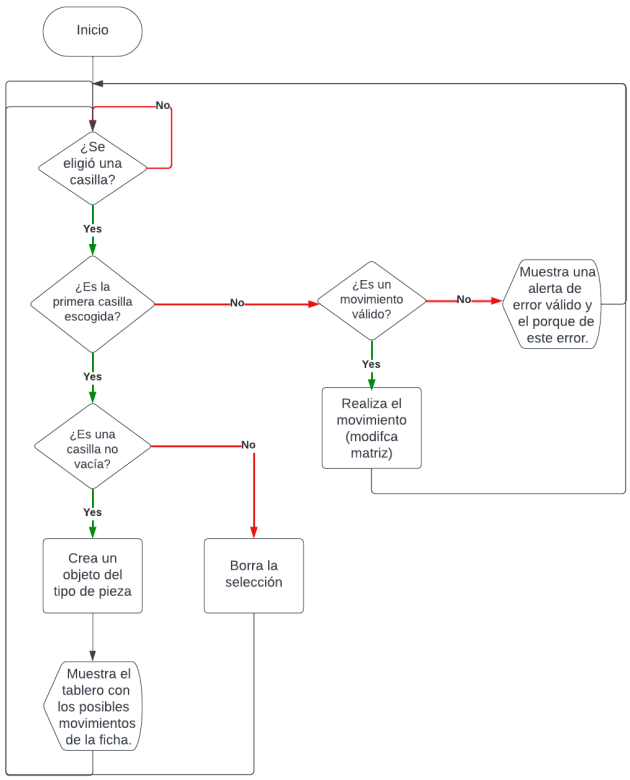

Figura 2: Diagrama de flujo del modo asistente

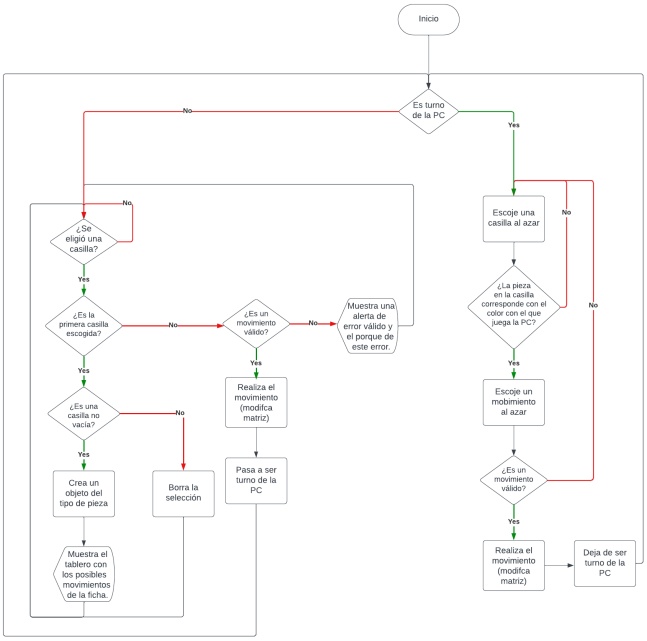

Figura 3: Diagrama de flujo del modo juego

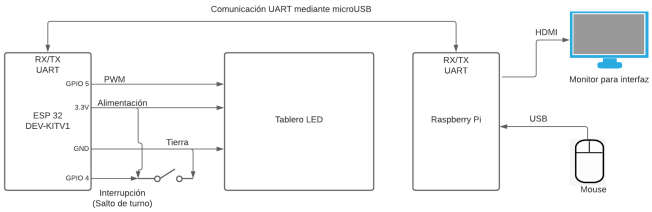

Figura 4: Esquemático de conexiones del proyecto.