**Instituto Tecnológico de Costa Rica**

**Proyecto 2**

**Archivo 1**

CE-1104 Fundamentos de Sistemas Computacionales

Grupo:1  
**Profesor:**

Milton Villegas Lemus  
**Semestre:** I Semestre 2025  
**Estudiantes:** Kendall Camacho Salas

Jason Vargas Arce

**Carnet:**

2025095631

2024240152  
**Fecha de entrega:** 25 de junio de 2025

**1. Introducción**

El presente documento detalla el desarrollo del Proyecto 2 del curso CE-1104, el cual complementa el Proyecto 1 integrando un circuito lógico combinatorio (Exceso 3 con suma circular) y una conexión inalámbrica entre una Raspberry Pi Pico W y una computadora personal. El objetivo principal es ampliar la interactividad del sistema empotrado y facilitar el registro de partidas, perfiles de jugador y rankings mediante una interfaz gráfica en Python.

Este proyecto fortalece los conocimientos en diseño lógico, electrónica digital, programación de microcontroladores y comunicación inalámbrica, consolidando el aprendizaje en áreas clave para la Ingeniería en Computadores.

**Conclusiones**

1. Pudimos aplicar todo lo aprendido en lógica y electrónica de forma práctica.
2. Logramos que el circuito Exceso 3 funcionara con suma circular como se pedía.
3. El éxito del proyecto vino de saber integrar bien la parte física y el software.
4. La Raspberry Pi Pico W fue bastante útil y potente para este tipo de tareas.
5. La suma circular fue todo un reto que nos obligó a pensar más allá de lo básico.
6. Entendimos mejor cómo funcionan los números en binario y hexadecimal.
7. Programar en MicroPython nos facilitó mucho controlar los pines y dispositivos.
8. Los LEDs nos ayudaron a ver rápido cómo estaba funcionando todo.
9. El bit aleatorio le dio ese toque impredecible que hizo el juego más interesante.
10. Validar salidas fue importante para asegurarnos que la lógica estaba bien aplicada.
11. Aprendimos a identificar y arreglar errores en el circuito, como cables mal puestos o polaridades.
12. Con los sockets en Python, logramos que la Raspberry y la compu se hablaran sin problema.
13. La interfaz que hicimos fue bastante amigable para manejar usuarios y puntajes.
14. El proyecto nos obligó a organizarnos bien y dividirnos el trabajo.
15. Usamos varias tecnologías al mismo tiempo: hardware, MicroPython y GUI en Python.
16. Además de lo técnico, mejoramos en comunicación y trabajo en equipo.
17. La lógica del juego quedó bien estructurada y fácil de mantener.
18. Verificar entradas y salidas con LEDs fue súper útil para detectar errores rápido.
19. Documentar el código ayudó mucho a que los dos entendiéramos todo.
20. Pudimos aplicar lo que sabemos sobre sistemas embebidos en algo real.
21. Aprendimos lo valioso que es diseñar antes de conectar cables al azar.
22. Planificar bien nos ahorró un montón de tiempo.
23. El buzzer sirvió perfecto para dar retroalimentación sonora durante el juego.
24. Entendimos cómo se conectan y dependen entre sí los distintos sistemas.
25. Controlar el display de 7 segmentos fue otra parte interesante del proyecto.
26. Implementar un ranking nos hizo usar estructuras de datos reales.
27. Tener código modular ayudó a escalar y modificar sin enredos.
28. Comprobar cosas con casos extremos nos ayudó a encontrar errores antes de tiempo.
29. Combinar hardware con software fue clave para resolver problemas del mundo real.
30. Este proyecto nos dejó habilidades que nos van a servir mucho en otros cursos y en el futuro**.**

**Recomendaciones**

1. Siempre revisar las conexiones con el multímetro antes de encender el circuito.
2. Usar zócalos para los integrados como el 74LS164, así evitás dañarlos con soldadura.
3. Probar primero el circuito Exceso 3 en un simulador antes de armarlo físicamente.
4. Asegurar de que los LEDs estén bien conectados (polaridad correcta).
5. Usar resistencias de 220–330Ω para cuidar los LEDs.
6. Comentar bien el código en MicroPython, eso ayuda un montón después.
7. Poner nombres claros a las variables para que el código no se vuelva un enredo.
8. Dividí bien el código de la interfaz gráfica en partes (módulos).
9. Tener claro cómo se van a comunicar la Pico W y la compu desde el inicio.
10. Dibujar el circuito con un diagrama bonito y bien numerado.
11. Hacer pruebas largas con varias partidas seguidas para ver si todo aguanta.
12. Hacer reuniones constantes para no atrasarse ni duplicar trabajo.
13. Desde el principio, definan quién hace hardware y quién hace software.
14. Si se puede, meter animaciones o efectos visuales para mejorar la experiencia.
15. Subir frecuentemente los avances a GitHub o alguna nube.
16. Probar la interfaz en varias resoluciones ayuda a que se vea bien en cualquier computadora.
17. Poner validaciones para los nombres de usuario y otros datos.
18. Evitar usar muchos sleep innecesarios en MicroPython, eso puede frenar el juego.
19. Preferí estructuras de control simples, son más fáciles de mantener.
20. Separar bien la lógica del juego de la parte que se comunica por WiFi.
21. Usar versiones compatibles de MicroPython y Thonny para evitar errores raros.
22. Medí cuánta corriente consume tu sistema para saber si tu fuente es suficiente.
23. Usá sonidos distintos para cada evento del juego: victoria, error, cambio de nivel.
24. Elegí LEDs potentes si vas a probar el juego en un lugar bien iluminado.
25. Asegurar de entender todo por si te toca explicarlo en la defensa.
26. Si vas a agregar cosas extra, que sea cuando lo principal ya funcione.
27. Probar la red Wi-Fi donde sea que vayas a presentar el proyecto.
28. Tener un botón o algo fácil para reiniciar el juego sin tener que desconectar.
29. Llevá una lista de cambios y pruebas, eso te ayuda a no repetir errores.
30. No dejes para el último día juntar todo

**Análisis de Resultados**

**Tabla de verdad**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **Bit Val** | **X** | **Y** | **Z** |
| **0** | **0** | **0** | **1** | **0** | **1** | **1** |
| **0** | **0** | **1** | **1** | **1** | **0** | **0** |
| **0** | **1** | **0** | **1** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** | **0** |
| **1** | **0** | **0** | **1** | **1** | **1** | **1** |
| **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **0** | **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **1** | **0** | **1** | **0** |

**Simplificación:**

X= A′B′C+A′BC′+A′BC+AB′C′

X= A′[B′C+BC′+BC]+AB′C′

X= A′(B⊕C+BC)+AB′C′

X= A′(B+C)+AB′C′

X= A′B+A′C+AB′C′

Y=A’B’C’+A’BC+AB’C’+ABC

Y=A’(B’⊕C’) + A(B’⊕C’)

Y=B’⊕C’

Z=A’B’C’+A’BC’+AB’C’+ABC’

Z=C’[A’(B’+B)+A(B’+B)]

Z=C’[A’(1)+A(1)]

Z=C’

**Diagrama del circuito**

Usando logic.ly

**Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.**

**Diagrama de módulos del sistema**

|  |
| --- |
| **Interfaz Gráfica (PC)** |
| - Selección/creación de usuario  - Visualización de puntaje (display virtual)  - Comunicación WiFi con Raspberry Pi Pico W  - Ranking de jugadores y almacenamiento en JSON  - Sonidos y animaciones de fondo |

Sockets TCP/IP

|  |
| --- |
| **Raspberry Pi Pico W (MicroPython)** |
| - Lógica del juego físico  - Control de LEDs (matriz 2x2)  - Lectura de botones físicos  - Control de buzzer  - Control del display de 7 segmentos (registro 74LS164)  - Generación de bit aleatorio (habilitación del circuito)  - Obtención del valor en binario desde el display  Envío del puntaje al PC vía socket |

|  |
| --- |
| **Circuito Lógico Exceso 3 Circular** |
| - Entradas: 3 bits LSB del marcador + bit aleatorio  - Operación: Suma binaria +3 en forma circular  - Salidas: 4 LEDs (3 bits + bit de validación)  - Validación de salida en condiciones específicas(1) |

**Descripción**

El sistema se divide en tres grandes módulos: la interfaz gráfica en PC, el microcontrolador Raspberry Pi Pico W, y el circuito lógico Exceso 3. La interfaz en Python permite gestionar jugadores, mostrar puntajes y controlar el flujo de la partida. La Raspberry Pi maneja toda la lógica del juego físico, el control de periféricos y la comunicación con el computador. Finalmente, el circuito Exceso 3 implementa una lógica combinatoria para sumar 3 de forma circular a los 3 bits menos significativos del marcador, incluyendo un bit de validación. La modularidad del diseño permite escalar y mantener el sistema fácilmente.

**Bibliotecas utilizadas**

**• MicroPython**

Es una versión liviana del lenguaje Python, diseñada especialmente para funcionar en microcontroladores como la Raspberry Pi Pico W. Permite controlar directamente el hardware (pines, sensores, actuadores) usando sintaxis muy parecida a Python, lo cual facilita el desarrollo embebido sin necesidad de aprender un lenguaje como C.

**• machine y time**

* machine: Es un módulo específico de MicroPython que permite acceder a funciones de hardware. Por ejemplo:
  + Pin para controlar entradas/salidas digitales.
  + PWM para generar señales de modulación por ancho de pulso.
  + ADC, I2C, SPI, etc., para otras interfaces.
* time: Proporciona funciones como sleep(), ticks\_ms(), y ticks\_diff() que sirven para manejar retardos, tiempos de espera, o controlar la duración de eventos.

**• socket**

Este módulo permite establecer comunicación por red entre dispositivos. En este proyecto se usa para:

* Crear un cliente socket en la Raspberry Pi Pico W que se conecta al servidor en la PC.
* Enviar y recibir mensajes por Wi-Fi, como:
  + "boton1", "acierto", "error", "display:C"…

Esto hace posible la sincronización en tiempo real entre la parte física (la Pico W) y la interfaz gráfica.

**• Python tkinter**

Es la librería estándar de Python para crear interfaces gráficas (GUIs). En este proyecto:

* Se usa en la PC para mostrar una ventana de juego con botones virtuales, puntaje y una simulación del display de 7 segmentos.
* Facilita la interacción del usuario sin necesidad de usar la consola o terminal.

**• json**

Se utiliza para guardar y leer datos persistentes en un formato fácil de usar:

* Almacena información de usuarios, claves y puntajes en archivos .json.
* Permite que el juego recuerde los puntajes y perfiles entre sesiones.
* Es ideal para datos simples estructurados como diccionarios y listas.

**• random**

Este módulo sirve para generar valores aleatorios, lo cual es clave para:

* Encender LEDs al azar durante el juego.
* Generar el bit de habilitación aleatorio en el circuito Exceso-3.
* Aumentar la dificultad e imprevisibilidad del juego.

**• \_thread (o threading)**

Permite que el programa ejecute varias tareas al mismo tiempo (en paralelo). Se usa para:

* Ejecutar la conexión Wi-Fi en un hilo secundario, mientras el juego corre en el principal.
* Escuchar continuamente mensajes de la PC sin detener la lógica del juego.
* En la GUI (tkinter), escuchar mensajes sin congelar la ventana.

Esto es fundamental para mantener una interfaz fluida y reactiva mientras se controla el hardware.

**Diagrama de arquitectura básico**

|  |
| --- |
| **Computadora / PC** |
| - Interfaz gráfica (Tkinter)  - Gestión de perfiles y rankings  - Comunicación Wi-Fi mediante sockets |

|

Comunicación Wi-Fi

|

|  |
| --- |
| **Raspberry Pi Pico W** |
| - Lógica del juego  - Circuito lógico Exceso 3 (suma circular)  - Control de LEDs, buzzer y display 7 segmentos |

|

Control de periféricos

|

|  |
| --- |
| **Periféricos conectados** |
| - Display de 7 segmentos  - LEDs indicativos  - Buzzer |

**Literatura o Fuentes Consultadas**

**OpenAI. (2025). *ChatGPT (versión GPT-4o)*. Recuperado el 18 de junio de 2025 de**

[**https://chat.openai.com**](https://chat.openai.com)

**Logicly. (s.f.). *Logic circuit simulator*. Recuperado el 18 de junio de 2025 de** [**https://logic.ly**](https://logic.ly)

**Raspberry Pi Foundation. (s.f.). *Pico series documentation*. Raspberry Pi. Recuperado el 18 de junio de 2025 de** [**https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html**](https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html)