Instituto Tecnológico de Costa Rica

Ingeniería en Computadores

Curso:

Fundamentos de Sistemas Computacionales

Documentación Técnica del proyecto 2

Número de grupo: 01

Profesor:

Milton Villegas Lemus

Estudiante:

Luis Fernando Ortiz Mata

Fecha:

26/11/2025

Introducción

El objetivo de este proyecto fue el de implementar un circuito decrementador en 3 de las salidas de los goles anotados por los jugadores. Para esta implementación se requiere que el número de goles anotados se deba convertir a su representación en binario. A partir del número obtenido, se deben tomar los 3 bits menos significativos. Estos 3 bits serán utilizados como entradas para el circuito decrementador en 3.

La cuarta entrada del circuito es un bit de habilitación, que se generará de manera aleatoria. Cuando este circuito sea cero, significa que el circuito decremento 3 no está habilitado, por lo que no se deberá realizar la resta a los goles anotados.

El circuito decrementador 3 se debe realizar la operación de restar 3 a los bits de entrada, utilizando la operación decremento **circular.** El resultado debe ser mostrado en 4 LEDs adicionales colocados sobre la maqueta del Proyecto 1. El cuarto LED (de habilitación) solo se activará únicamente cuando las otras tres salidas sean válidas.

Para la sección del software se utilizó el IDE de Python 3.15.5 se utilizó Visual Studio Code y también el IDE de Micropython llamado Thonny 4.1.7. En estas dos plataformas se implementó la lógica de programación sobre cómo comunicarse entre ellas sin necesidad de un cable (via WiFi) y sobre cómo convertir de base diez a binario, obtener sus tres dígitos menos significativos y descomponer dichos números en entradas (A, B y C) para los pines de la Raspberry Pi Pico W 2022. En cuanto a sistema operativo se utilizó Windows 11 versión 25H2.

Con esto pasamos a la sección de hardware donde se utilizó un Raspberry Pi Pico W 2022, cuatro LEDs siendo tres de estos rojos y uno verde, 10 metros de cable para protoboard, una protoboard, estaño, cautín de 40 W, dos compuertas MB74LS86, tres compuertas SN74LS04, dos compuertas SN74LS32, dos compuertas SN74LS08 y cuatro resistencias de 330 Ω.

Conclusiones

1. La tabla de verdad del circuito decrementador 3 fue correctamente elaborada con entradas A, B y C.
2. Se aplicaron teoremas y axiomas de álgebra booleana para simplificar la lógica.

3. Se diseñaron circuitos a nivel de dibujo con compuertas básicas (NOT, XOR, OR, AND).
4. Se adquirieron los componentes necesarios antes de la implementación física.
5. El cableado total de 10 m permitió flexibilidad en la protoboard.
6. Se seleccionaron LEDs de distintos colores para diferenciar salidas.
7. Se utilizaron compuertas MB74LS86, SN74LS04, SN74LS32 y SN74LS08 según requerimientos lógicos.
8. Se incluyeron resistencias de 330 Ω para protección de los LEDs.
9. Se consultaron datasheets de cada componente antes de la implementación.
10. Los pines GP27, GP26 y GP22 de la Raspberry Pi Pico W se usaron como entradas A, B y C.
11. Se generaron nodos en la protoboard para entradas negadas y no negadas.
12. Las salidas se construyeron siguiendo las especificaciones de los datasheets.
13. El circuito fue probado en protoboard antes de soldar o integrar.
14. Se desarrolló código en Python 3.13.5 para simulación en PC.
15. Se programó en MicroPython (Thonny 4.1.7) para ejecución en la Raspberry Pi Pico W.
16. Se implementó la conversión de números decimales a binario con bin(abs(n))[2:].
17. Se extrajeron los tres bits menos significativos con b[-3:].
18. Se descompusieron los bits para obtener cada dígito individual.
19. Se habilitó un bit aleatorio en el pin GP21 como señal de control.
20. Se realizaron comprobaciones de funcionamiento contra la tabla de verdad.
21. Se detectó un fallo en la salida Y1 durante las pruebas iniciales.
22. El fallo se debió a la falta de alimentación en la compuerta MB74LS86.
23. Una vez corregida la alimentación, el circuito funcionó correctamente.
24. La integración entre hardware y software se logró de manera satisfactoria.

25. El uso de Visual Studio permitió depuración más avanzada del código Python.
26. El uso de Thonny facilitó la programación directa en la Raspberry Pi Pico W.
27. La lógica del decremento circular se validó con pruebas prácticas.
28. La combinación de teoría (tabla de verdad) y práctica (protoboard) fue efectiva.
29. La documentación de cada paso permitió identificar errores rápidamente.
30. El proyecto alcanzó un nivel funcional completo tras la corrección del error en Y1.

Recomendaciones

1. Elaborar tablas de verdad para cada nuevo circuito antes de implementarlo.
2. Aplicar siempre simplificación booleana para reducir el número de compuertas.
3. Diseñar diagramas previos para visualizar la lógica antes del montaje físico.
4. Comprar componentes con anticipación para evitar retrasos.
5. Usar cables de diferentes colores para identificar señales en la protoboard.
6. Asignar colores de LEDs según función (ej. verde = habilitación, rojo = error).
7. Seleccionar compuertas con bajo consumo energético para optimizar el circuito.
8. Verificar valores de resistencias con multímetro antes de instalarlas.
9. Revisar datasheets siempre antes de conectar componentes.
10. Documentar qué pines de la Raspberry se usan para cada señal.
11. Crear nodos organizados en la protoboard para evitar cruces de cables.
12. Seguir estrictamente las configuraciones de los datasheets en las salidas.
13. Probar primero en protoboard antes de soldar en placa definitiva.

14. Usar entornos de desarrollo robustos como Visual Studio para pruebas iniciales.
15. Mantener Thonny actualizado para compatibilidad con la Raspberry Pi Pico W.
16. Validar funciones de conversión binaria con casos de prueba adicionales.
17. Probar extracción de bits con diferentes longitudes de números binarios.
18. Documentar cada paso de descomposición de bits para futuras referencias.
19. Implementar control de habilitación con más de un bit para mayor robustez.
20. Comparar resultados de pruebas con la tabla de verdad en cada iteración.
21. Revisar conexiones de alimentación antes de iniciar pruebas.
22. Usar fuentes de alimentación estables para evitar fallos en compuertas.
23. Verificar voltajes de operación de cada componente antes de integrarlo.
24. Realizar pruebas conjuntas de hardware y software para validar integración.
25. Aprovechar depuración avanzada de IDEs para detectar errores lógicos.
26. Usar Thonny para pruebas rápidas y Visual Studio para análisis profundo.
27. Validar decremento circular con casos límite (ej. 000 y 111).
28. Combinar teoría y práctica en cada etapa para asegurar comprensión total.
29. Mantener bitácoras detalladas para identificar errores y soluciones.
30. Realizar pruebas finales completas antes de considerar el proyecto terminado.

Análisis de resultados

Tabla de verdad

A	B	C	Y2	Y1	Y0
0	0	0	1	0	1
0	0	1	1	1	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	0	1	1
1	1	1	1	0	0

Tabla de verdad realizada en Excel

Se realizan todas las combinaciones posibles de A, B y C y a cada una de las entradas de estas se ve como un mismo numero en binario de tres dígitos, por ejemplo, si A= 1, B = 0 y C = 1, entonces sería 101 en binario, a este número se le hace la resta de 3 y quedaría en 010 y cada uno de los dígitos se descompone en un valor, en este caso Y2 = 0, Y1 = 1 y Y0 = 0 y así sucesivamente con todas entradas.

Simplificación

Para la simplificación de este circuito se decantó por el uso de mintérminos (Suma de Productos). Para efectos de este documento se representará el complemento como una “ ´ ”, de modo que A negada se denota A ´.

Para Y0 la simplificación fue:

Y0: A ´B ´C ´ + A ´BC ´ + AB ´C ´ + ABC ´

Y0: A ´C ´(B ´ + B) + AB ´C ´ + ABC ´ ; Distributiva

Y0: A ´C ´ + AB ´C ´ + AB ´C ´ ; Complemento e Identidad

Y0: A ´C ´ + AC ´(B ´ + B) ; Distributiva

Y0: A ´C ´ + AC ´ ; Complemento e Identidad

Y0: C´(A´ + A) ; Distributiva

Y0: C´ ; Complemento e Identidad

Para Y1 la simplificación fue:

Y1: **A**´B´C + A´BC´ + AB´C + ABC´

Y1: A´(B´C + BC´) + A(B´C + BC´) ; Doble distributiva

Y1: (B´C + BC´)(A´ + A) ; Distributiva

Y1: B´C + BC´ ; Complemento e Identidad

Se puede ver que Y1 se simplifica a un XOR de B y C

Para Y2 la simplificación fue:

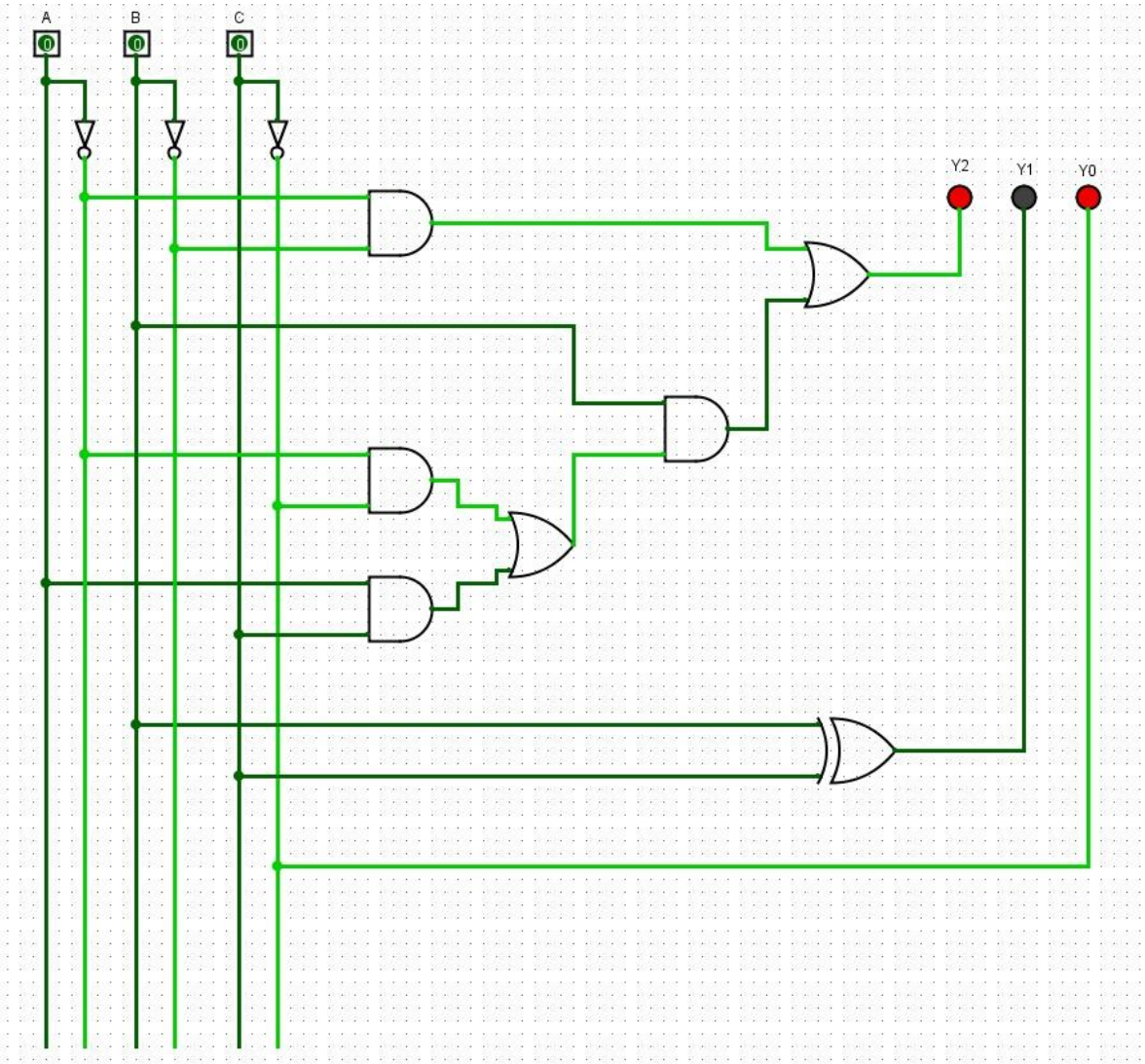
Y2: A´B´C´ + A´B´C + A´ BC´ + ABC

Y2: A´B´(C´ + C) + A´BC´ + ABC ; Distributiva

Y2: A´B´ + A´BC´ + ABC ; Complemento e Identidad

Y2: A´B´ + B(A´C´ + AC) ; Distributiva

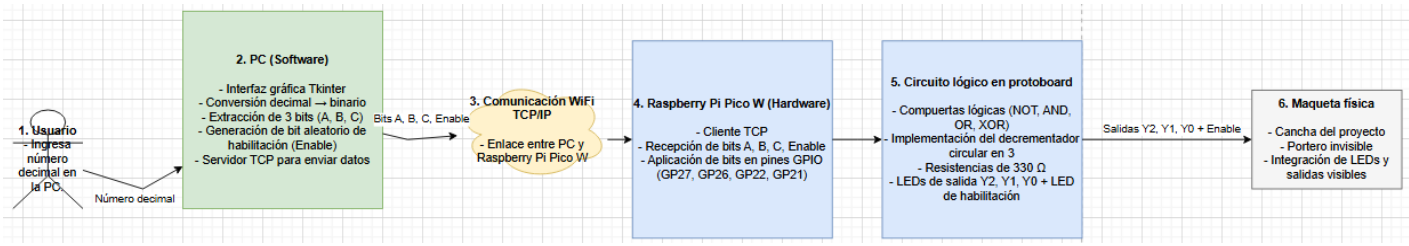
Diagrama del circuito



Para la representación del diagrama del circuito se utilizó *Logisim v4.0.0*

En el diagrama se puede ver cuál circuito es de cada salida y en este caso las entradas son A, B y C donde todas tienen el valor de 0, dando la salida como resultado de Y2 = 1, Y1 = 0 y Y0 = 1 como en la tabla de verdad.

Diagrama de módulos del sistema



Para la elaboración de este diagrama se utilizó draw.io 28.2.5

Literatura o Fuentes

[1]

Build software better, together. (s. f.). GitHub. https://github.com/search?q=network%20micropython&type=repositories

[2]

LuTec Lab. (2025). Proyecto II MILT 2024 [PDF]. Instituto Tecnológico de Costa Rica. Proyecto II Milt 2024 v3.pdf

[3]

www.alldatasheet.com. (s. f.-c). *SN74LS08N datasheet(1/4 Pages) ONSEMI*. LOW POWER SCHOTTKY. https://www.alldatasheet.com/html-pdf/12617/ONSEMI/SN74LS08N/183/1/SN74LS08N.html

[4]

www.alldatasheet.com. (s. f.-d). *SN74LS86N datasheet(1/4 Pages) ONSEMI*. Quad 2-Input Exclusive OR Gate. https://www.alldatasheet.com/html-pdf/177332/ONSEMI/SN74LS86N/218/1/SN74LS86N.html

[5]

www.alldatasheet.com. (s. f.-d). *SN74LS32N datasheet(1/4 Pages) ONSEMI*. LOW POWER SCHOTTKY. https://www.alldatasheet.com/html-pdf/12650/ONSEMI/SN74LS32N/183/1/SN74LS32N.html

[6]

www.alldatasheet.com. (s. f.-c). *SN74LS04N datasheet(1/4 Pages) ONSEMI*. LOW POWER SCHOTTKY. https://www.alldatasheet.com/html-pdf/12615/ONSEMI/SN74LS04N/183/1/SN74LS04N.html

Link del repositorio

https://github.com/LuizTek/SecondProyectFSC.git