Algoritmos de escape implementados en hardware.

Ian P. Larrieu Lacoste y Demian N. Mozo
Depto. de Electrónica y Computación, Facultad de Ingeniería - UNMDP
Mar del Plata, 7600, Argentina. *Julio 2025*
{ianlarrieu, dmozo}@alumnos.fi.mdp.edu.ar

***Resumen*—En este trabajo se presenta la implementación de un sistema de control digital sobre FPGA para un robot autónomo capaz de detectar obstáculos, ajustar su trayectoria y resolver un laberinto desconocido. La navegación se basa en un algoritmo de llenado (flood fill), que permite construir un mapa del entorno y encontrar el camino más corto hacia la salida. La arquitectura implementada aprovecha la paralelización que ofrece el hardware configurable, logrando un comportamiento eficiente en tiempo real.**

***Palabras clave*—FPGA, VHDL, Robot, Maze-solver, Flood-fill.**

I. Introducción

Este proyecto se desarrolló en la asignatura Técnicas y Dispositivos Digitales II (3er año de Ingeniería Electrónica). En esta materia se abordan temas de diseño digital, una introducción a los dispositivos lógicos programables, entre ellos FPGAs.

La resolución de laberintos es un problema clásico en robótica y teoría de grafos, que consiste en encontrar una ruta eficiente desde una posición inicial hasta un destino dentro de un entorno complejo. El robot debe evitar obstáculos y seleccionar una trayectoria óptima según criterios como distancia o tiempo. En escenarios dinámicos, también debe adaptarse en tiempo real ante cambios en el entorno. Este problema es relevante en aplicaciones como navegación autónoma y logística en almacenes. Muchos algoritmos de navegación se desarrollan sobre microcontroladores o microprocesadores. Un ejemplo común son las placas Arduino basadas en microcontroladores de la familia ATmega, utilizadas en [1], [2] y [3]. En este último, se emplean dos variantes del algoritmo —el básico (FFA) y una versión modificada (MFFA)— junto con sensores ultrasónicos para navegación autónoma en entornos desconocidos.

La implementación de estos algoritmos en hardware, particularmente en FPGAs, permitiría una ejecución más rápida y eficiente, especialmente en aquellos casos donde puede aprovecharse su naturaleza concurrente. A pesar de las ventajas que ofrecen las soluciones basadas en lógica reconfigurable —como el paralelismo, la alta velocidad y la eficiencia energética—, hasta donde alcanza nuestro conocimiento no se han reportado implementaciones de este tipo. Esto puede atribuirse a los desafíos que implica el diseño de arquitecturas distribuidas y el uso de lenguajes de descripción de hardware.

En este proyecto se diseña e implementa el algoritmo de llenado (flood-fill) mediante una arquitectura completamente paralela sobre FPGA, en la que el laberinto es modelado como una matriz de celdas interconectadas. Esta propuesta permite resolver el laberinto en tiempo real, y a la vez explora las ventajas del procesamiento distribuido aplicado a sistemas embebidos de navegación.

I-A. *Algoritmo de llenado*

El algoritmo de llenado utilizado se basa en una estrategia exploratoria, en la que el robot parte del supuesto de un entorno libre de obstáculos. A medida que avanza, actualiza dinámicamente un mapa interno, asignando a cada celda un valor que representa la distancia al objetivo y marcando como inaccesibles aquellas celdas adyacentes a los muros detectados por sus sensores. Esta información se emplea para recalcular rutas óptimas, permitiendo que el robot se adapte progresivamente al entorno real y construya su propio mapa del laberinto [3].

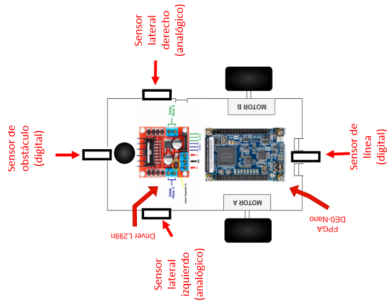

Figura 1. Esquema funcional del robot implementado.

II. Implementación

II-A. *Robot*

El robot de la Fig.1 incorpora dos motores de corriente continua controlados por un driver L298N [4]. Además, dispone de un sensor frontal digital, un sensor digital para detección de línea, y dos sensores laterales infrarrojos analógicos.

El control y procesamiento de los datos de los sensores, así como el accionamiento de los motores, se gestionan en una DE0-Nano Field Programmable

Gate Arrays (FPGAs) de Altera [5]. En esta FPGA se implementa el algoritmo de navegación, el cual regula los movimientos del robot en función de las lecturas de los sensores. Los sensores infrarrojos laterales están conectados a dos canales de un conversor Analógico-Digital (ADC) integrado en la placa de desarrollo. Este ADC es gestionado por la FPGA para recibir en tiempo real la información de distancia de los laterales del robot, permitiendo así un desplazamiento recto y estable.

II-B. Laberinto

El laberinto de la Fig.2 está construido en madera, con columnas impresas en 3D, y diseñado con cuadrículas delimitadas por líneas negras detectables por el robot. Además, cada cuadrícula está configurada para permitir la incorporación de muros, lo que permite ajustar y variar la complejidad del laberinto para diferentes escenarios de navegación.

Figura 2. Laberinto implementado con paredes rebatibles.

III. Control del Robot

El algoritmo de llenado fue implementado en una FPGA con el objetivo de resolver un laberinto de 4×4. Para modelar el entorno, se diseñó una arquitectura modular basada en celdas, donde cada celda representa una posición del laberinto. Cada celda fue implementada como una unidad funcional en VHDL (Fig. 3), con capacidad para interactuar con sus celdas vecinas. Cada bloque de celda recibe como entradas:

- La información sobre muros en las cuatro direcciones (N, S, E, O).
- Los valores de peso (distancia estimada al objetivo) provenientes de las celdas adyacentes.

Con esta información, la celda calcula su propio peso siguiendo una lógica simple: adopta el menor peso de sus vecinos accesibles (es decir, aquellos sin muro en común) y le suma una unidad. Luego, transmite su peso actualizado y su configuración de muros hacia las celdas vecinas, propagando la información a lo largo de toda la red.

Al interconectar las 16 celdas en una matriz, se forma un laberinto completo en el que cada celda

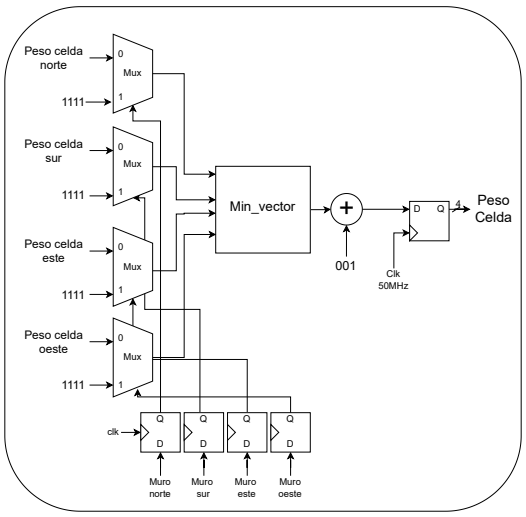

Figura 3. Esquemático combinacional de una celda.

actúa localmente, pero contribuye al comportamiento global del sistema. Esta estructura distribuida permite que el algoritmo sea altamente paralelizable y se ejecute eficientemente en hardware. Durante la exploración física del entorno, el robot actualiza dinámicamente la información de muros en las celdas correspondientes, lo que fuerza una nueva propagación de pesos. De esta forma, el sistema converge progresivamente hacia una representación real del laberinto, y el robot puede decidir su próximo movimiento basándose en los pesos más bajos disponibles en las celdas accesibles.

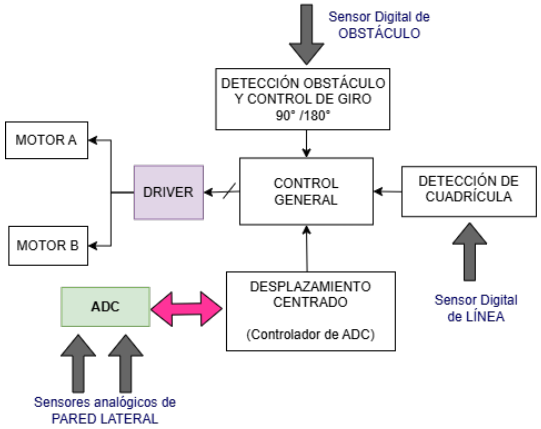

Figura 4. Esquema del control implementado.

Una vez propagados los pesos, el robot evalúa las celdas adyacentes y selecciona aquella con el menor peso, siempre que no haya un muro que impida el paso. Tras avanzar a una nueva celda, actualiza la información local detectando la presencia de muros mediante el sensor frontal. Estos datos se almacenan en la celda correspondiente del laberinto y se comparten con las demás a través de las interconexiones. Si se detectan cambios en la

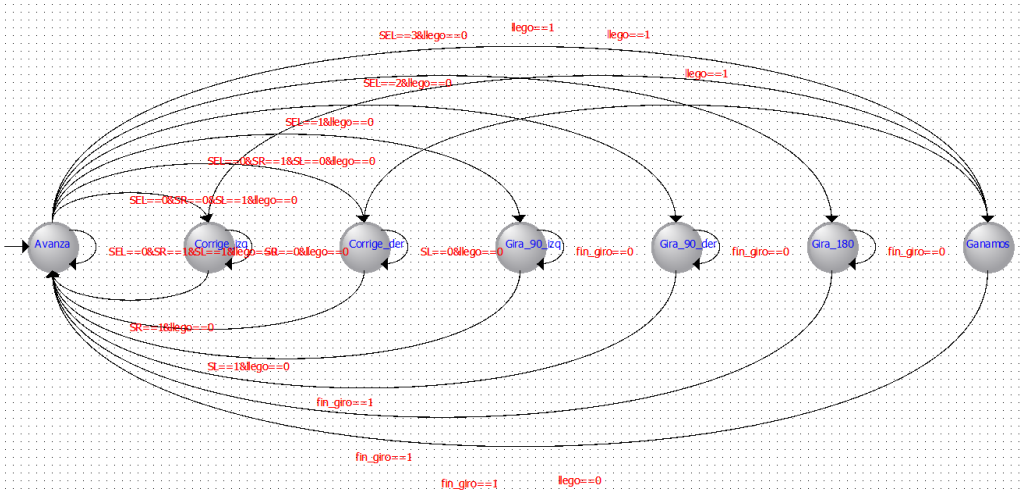

Figura 5. Diagrama de estados: control de motores.

topología, se desencadena una nueva propagación. El sistema de movimiento permite giros de $\pm 90°$ o $180°$ según sea necesario para alinearse con la celda elegida. Esta lógica le permite adaptarse de forma eficiente a las condiciones reales del entorno.

Cuando el robot alcanza la celda objetivo, se detiene y queda a la espera de nuevas instrucciones. El sistema incorpora un mecanismo de reinicio parcial mediante un pulsador externo: al activarse, se reinicia únicamente la posición actual del robot, conservando la memoria de los muros previamente detectados. Así, en ejecuciones sucesivas, el robot navega hacia el objetivo sin necesidad de volver a explorar. La Fig. 4 muestra un esquema del control implementado en la FPGA, destacando las funcionalidades necesarias para el desplazamiento: detección de obstáculos y giro, desplazamiento centrado, control de motores y conteo de celdas atravesadas.

Funciones del sistema de control

- **Desplazamiento centrado y control de motores:** se implementa mediante una máquina de estados que comanda los motores en tiempo real (Fig. 5). Por ejemplo, el robot avanza en línea recta utilizando los sensores laterales, si el valor entregado por los sensores (entradas al ADC) es menor a un umbral, se ajusta la dirección para alejarse de la pared. Por otro lado, si el control general decide que se debe realizar un giro para cambiar de casilla, este bloque ejecuta la acción.
- **Detección de obstáculos y giro:** el robot emplea el sensor frontal para detectar muros. Si un muro es detectado, el sistema acciona los motores y ejecuta giros de $90°$ o $180°$.
- **Detección de cuadrícula:** mediante un sensor de línea, se detecta el cruce entre celdas. Esta señal actualiza la posición actual del robot.
- **Control general y lógica de decisión:** este módulo selecciona la estrategia de navegación y decide el próximo movimiento en función de la información disponible.

La implementación completa de este trabajo, incluyendo el diseño en VHDL y la documentación correspondiente, se encuentra disponible en [6].

IV. Conclusiones y trabajo futuro

Se integraron con éxito los sensores y motores del robot controlados por FPGA, permitiendo una respuesta en tiempo real clave para la navegación en el laberinto. A futuro, se evaluará la escalabilidad del sistema para entornos más complejos y aplicaciones industriales, como automatización en almacenes o inspección en ambientes peligrosos, requiriendo pruebas en condiciones reales.

Además, la FPGA ofrece una ventaja significativa al permitir la ejecución paralela de tareas como detección de muros, actualización del mapa y control de motores, evitando interferencias y retrasos típicos de microprocesadores secuenciales, que pueden bloquear tareas y provocar pérdidas durante la navegación.

Referencias

[1] I. Elshamarka and A. B. S. Saman, "Design and implementation of a robot for maze-solving using flood-fill algorithm," *International Journal of Computer Applications*, vol. 56, no. 5, 2012.

[2] S. Tjiharjadi, "Design and implementation of flood fill and pledge algorithm for maze robot," *International Journal of Mechanical Engineering and Robotics Research*, 2019.

[3] Nadour and Cherroun, "Using flood-fill algorithms for an autonomous mobile robot maze navigation," *International Journal of System Assurance Engineering and Management*, 2022.

[4] STMicroelectronics, "L298N Dual Full Bridge Driver - Datasheet," https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf, 2000, accessed: 2025-06-15.

[5] *DE0-Nano User Manual*, Terasic Inc., 2016.

[6] D. N. Mozo, "Proyecto escape de laberinto - robot autónomo en fpga," https://github.com/demianmozo/proyecto_laberinto, 2024.