Memorias del Congreso Científico Tecnológico de las carreras de
Ingeniería Mecánica Eléctrica, Industrial y Telecomunicaciones, sistemas y electrónica

AÑO 3. No. 3. ISSN-2448-7236. SEPTIEMBRE 2018 – AGOSTO 2019. IC-06, pág.: 1 a la 6.



# Implementación de un vehículo robotizado que soluciona esquemas laberínticos, por medio de nodos dentro de un árbol de decisión

***Santiago Roa Martinez**, David Tinoco Varela****

## RESUMEN

Los arboles de búsqueda, son estructuras de datos que se comportan de manera recursiva, principalmente utilizadas para el rastreo, modificación e inserción de nuevos datos dentro de un conjunto de información. Mayoritariamente son utilizados en ambientes informáticos y abstractos.

Existen una gran cantidad de comportamientos o metodologías de resolución de problemas que tienen una visualización recursiva, tales como la modelación de juegos, solución de factoriales, solución de series infinitas, entre otros.

En este proyecto, se presenta el diseño e implementación de un vehículo robotizado que utiliza los nodos de los árboles de búsqueda como memorias de posición, para poder resolver esquemas laberinticos. El vehículo ha sido probado en laberintos con más de dos bifurcaciones por nodo, logrando recordar las posiciones ya recorridas, y por lo tanto, encuentra la salida de un laberinto utilizando cómputo recursivo.

El sistema robotizado ha sido implementado utilizando un sensor de línea QTR-8A para identificar las marcas del laberinto y la placa Arduino UNO como elemento central de procesamiento.

## ABSTRACT

Search trees are data structures that behave recursively, mainly used for the tracking, modification, and insertion of new data within a set of information. Mostly they are used in computer and abstract environments.

There are a lot of methodologies and problem-solving behaviors which have a recursive visualization, such as game modeling, factorial solution, and endless series solution, among others.

In this project, we present the design and implementation of a robotic vehicle that uses the nodes of the search trees as position memories, in order to solve maze schemes. The vehicle has been tested in labyrinths with more than two branches per node, being able to remember the positions already traversed, and therefore, it finds the exit of a labyrinth using recursive computation.

The robotic system has been implemented using a QTR-8A line sensor to identify the maze marks, and the Arduino UNO board as a central processing element.

**Palabras claves**: Resolución de laberintos, arboles de búsqueda, Arduino, Recursividad.

## INTRODUCCIÓN

Hoy en día, Según *Rahnama*, *Elci* y *Metani* [1], la resolución de laberintos es uno de los problemas que han adquirido gran relevancia en los últimos años. Esto debido principalmente al hecho de que las aplicaciones de un sistema robotizado que pueda partir de un punto de origen y llegar a un punto destino, evitando diferentes obstáculos, pueden ser muy variadas, tales como la generación de robots de rescate autónomos, robots exploradores, robots de limpieza y máquinas de identificación de objetos en un entorno abierto.

## PRELIMINARES

En esta sección se describirán los conceptos básicos relacionados al proyecto realizado.

### Recursividad

El proceso por el cual una función se llama a sí misma para resolver un problema es llamado recursión, este concepto puede verse como un proceso que puede repetirse hasta el infinito. Ciertos problemas grandes, tienen la característica de que pueden ser representados como la descomposición de problemas más pequeñas, siendo estos problemas pequeños una réplica del problema principal. Algunos problemas matemáticos que pueden ser resueltos por este tipo de métodos son los factoriales, la *serie de Fibonacci*, las *torres de Hanoi*, y los fractales, entre otros. En la figura 1, podemos ver la representación del fractal llamado "El copo de nieve de Koch", como podemos visualizar, la generación del copo parte de un triángulo, al que se le superpone otro triangulo invertido, a cada nuevo triangulo se le superpone nuevamente un triángulo de su mismo tamaño, generando una superposición infinita en un proceso netamente recursivo.

* Universidad Nacional Autónoma de México, Facultad de Estudios Superiores Cuautitlán, ITSE, exttremo@hotmail.com
** Universidad Nacional Autónoma de México, Facultad de Estudios Superiores Cuautitlán, Departamento de Ingeniería, dativa19@hotmail.com

AÑO 3. No. 3. ISSN-2448-7236. SEPTIEMBRE 2018 – AGOSTO 2019. IC-06, pág.: 1 a la 6.

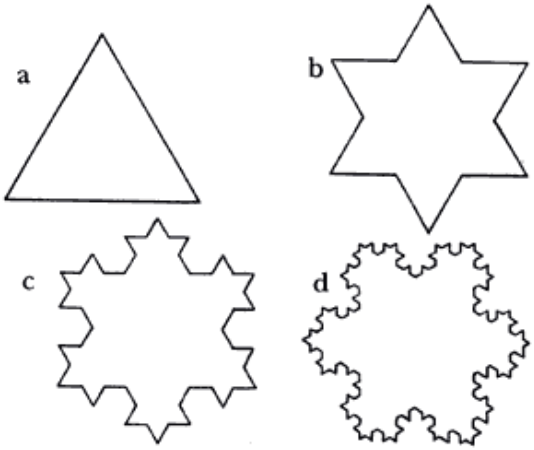


**Fig. 1. El copo de nieve de Koch, fractal de generación recursiva.**

## Árboles binarios

Los árboles binarios (AB) son estructuras de datos abstractas y no lineales. Los AB están compuestos de un conjunto finito de elementos (Nodos), cada uno de estos nodos almacena un dato y a su vez puede apuntar hacia dos nodos más, el sub árbol izquierdo y el sub árbol derecho. En la figura 1, podemos observar la representación de un árbol binario.

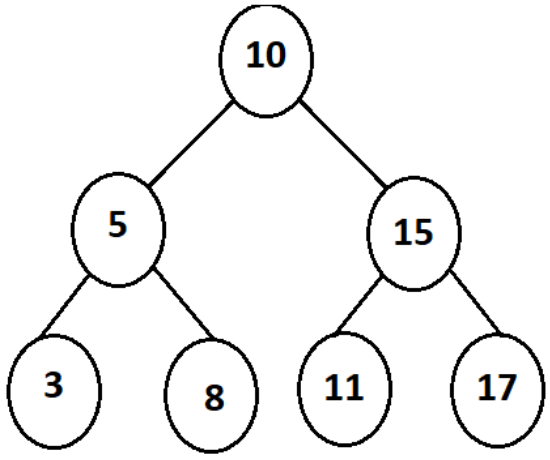


**Fig 2.- Representación de un árbol binario.**

Como se puede apreciar en la figura 2, los AB son estructuras inherentemente recursivas, ya que cada sub nodo presenta el mismo comportamiento que el nodo anterior.

Los AB pueden ser recorridos de diferentes formas (*Preorden*, *Inorden*, *Postorden*) para poder localizar algún dato definido que se encuentre almacenado en él.

Una característica importante de estas estructuras, es que los datos que están almacenados en ellas, pueden ser eliminados, modificados, o incluso, se pueden agregar nuevos datos. Por los motivos descritos, estas estructuras pueden ser utilizadas en una gran cantidad de aplicaciones computacionales, entre las que se encuentra la gestión de grandes cantidades de datos [2] y el desarrollo de juegos [3].

## Arduino

Según la página oficial del proyecto *Arduino* (https://www.arduino.cc/), Arduino es una plataforma electrónica de código abierto basada en hardware y software de fácil manejo. Las placas Arduino pueden leer entradas por medio de sensores y botoneras, procesar estas entradas, y generar controles de diferentes tipos de actuadores en sus salidas.

Esta tarjeta de operaciones, se ha convertido en el proceso central de miles de proyectos, que van de proyectos muy simples a proyectos científicos sumamente complejos.

Arduino fue desarrollado en el *Ivrea Interaction Design Institute* como una herramienta que pudiera ser usada fácilmente por cualquier persona sin experiencia en electrónica y programación para la generación de prototipos tecnológicos. La palca más utilizada se denomina Arduino UNO, y podemos verla en la figura 3.

**Fig. 3. Placa de desarrollo Arduino UNO.**

## ESTADO DEL ARTE

Como ya se mención, los robots que solucionan redes laberínticas pueden ser de gran utilidad tanto tecnológica como científicamente. Existen una gran cantidad de métodos que pueden ser implementados en la búsqueda de robots que cumplan este cometido en la forma más eficiente posible.

En esta sección describiremos solo algunas de las propuestas que se han hecho con respecto al diseño de robo que logran salir de un laberinto.

Una de las metodologías más lógicas es la resolución de laberintos por medio de visión artificial. Cerezo-Sánchez y otros autores [4] presentaron un sistema de visión artificial para el análisis de la trayectoria de un objeto, con su propuesto, ellos lograron resolver un laberinto robotizado analizando las rutas de un objeto móvil en el espacio. Para la identificación de posiciones, los autores mapearon un sistema coordenado, Su sistema cuenta con una interface de usuario realizada en *LabVIEW* desde donde el usuario

AÑO 3. No. 3. ISSN-2448-7236. SEPTIEMBRE 2018 – AGOSTO 2019. IC-06, pág.: 1 a la 6.

puede proporcionar parámetros y visualizar el proceso. Su canal de comunicación fue por medio de una PC a un PIC. Por otro lado, *Rahnama, Elçi* y *Metani* [1], mostraron un algoritmo que se basa en el procesamiento de imágenes y el algoritmo de ruta más corta. Según los autores, el algoritmo propuesto, es eficiente debido al uso de un celular que toma las muestras, procesa los datos, y envía la ruta más corta al robot. Es decir, este enfoque le dará el tiempo necesario de procesar toda la información, antes de comenzar la trayectoria.

Otras técnicas empleadas para el diseño de este tipo de sistemas, implican el uso de conceptos relacionados a inteligencia artificial, tal es el caso de la propuesta hecha en [5], en es te trabajo los autores presentan el diseño e implementación de un sistema de control a través del uso de redes neuronales artificiales (RNA) implementado en un FPGA **(***Field programmable gate array* **)** *Spartan 6*. El esquema que ellos proponen lleva a cabo una etapa de exploración, en la que el robot recoge la información del entorno a través de sus sensores, enviando los datos adquiridos a una computadora que utiliza comunicación inalámbrica, para llevar a cabo el entrenamiento de la RNA.

Existe una gran cantidad de algoritmos que pueden ser utilizados para la solución de esquemas laberinticos, tales como el *Flood-Fill Algorithm* [6], el algoritmo presentado en [7], e inclusive técnicas teóricas que sirven para resolver laberintos a partir de teoría de grafos [8]. Sin embargo, muchas de estas implementaciones son costosas, ya que requieren licencias de software de un gran precio comercial, por lo que la búsqueda de un esquema que sea económico, y que utilice pocos recursos de procesamiento y de memoria, es de suma importancia.

En este trabajo se busca la generación de un vehículo robotizado que no solamente resuelva el problema planteado del laberinto, sino que también lo haga de una manera económica, reducida y que utilice pocos recursos computacionales para su ejecución,

## DESARROLLO DEL PROYECTO

En esta sección se presenta el desarrollo de proyecto tanto la descripción del software como del hardware.

### Estructura de funcionamiento

El programa implementado dentro del Arduino, es un programa recursivo que va generando niveles de recursividad de acuerdo al número de nodos que vaya recorriendo, es decir, cada que llega a un nodo, el algoritmo genera un nuevo nivel, y se vuelve a llamar a sí mismo. En la implementación, cada nodo está dado por una bifurcación dentro del laberinto, en el momento en el que el sensor detecta la bifurcación, se genera un nodo, y con este se guarda la posición en la que este nodo se generó. El vehículo después de generar el nodo, continua su recorrido a través de uno de los caminos existentes en la bifurcación, hasta llegar nuevamente o al fin del camino (sin salida) o a otra nueve bifurcación, repitiendo el paso anterior. Si llega al fin del camino sin encontrar a salida, el vehículo recordara exactamente el punto del que partió, es decir, regresa al nodo anterior, o dicho de otra forma, regresa a la dirección bifurcada de manera directa. Al llegar a ese punto de bifurcación, el robot marca como "sin camino" a la rama ya recorrida, y continua con el recorrido del siguiente camino. Repite estos pasos hasta llegar a la salida. Es importante mencionar, que el vehículo al llegar a una bifurcación, escoge aleatoriamente el camino o rama a seguir, sin embargo, cuando regresa al nodo del que salió, el recorrido lo hace sin detenerse y sin verificar nuevos caminos, ya que se dirige directamente al punto de partida o nodo del árbol de búsqueda generado.

En la figura 4, podemos observar el ejemplo de generación de nodos, vemos que partimos de un nodo A, este nodo solo tiene un camino, hasta el punto B, al llegar a B se encuentra con una bifurcación, es decir, genera un nodo. La bifurcación B tiene dos posibles caminos, llegar a C, o llegar a D. Si escoge ir a C, encontrará que es un punto sin salida, por lo que regresa de orma directa a B, considera el camino hacía C como ya recorrido y escoge ir hacía el punto D, donde se tiene nuevamente una bifurcación y nuevamente se genera un nodo, el proceso se repite hasta llegar a J, que es la salida del laberinto.

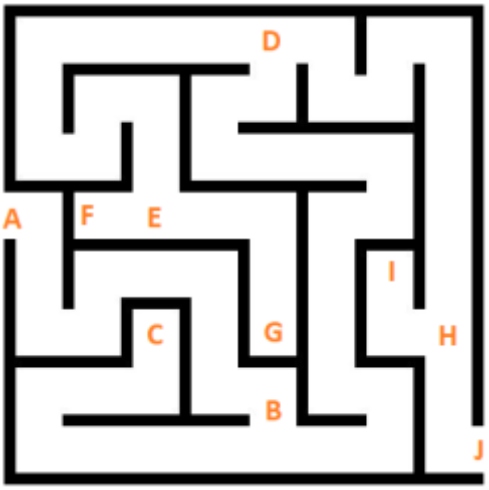


**Fig. 4. Esquema de un laberinto, en donde el vehículo robotizado irá marcando los puntos a seguir.**

En cada paso que da, el vehículo va generando las ramificaciones de un árbol de búsqueda, para así poder regresar rápidamente al nodo anterior en caso de no tener éxito. La figura 5, muestra el árbol de búsqueda que generó el vehículo al realizar el recorrido del laberinto de la figura 4.

AÑO 3. No. 3. ISSN-2448-7236. SEPTIEMBRE 2018 – AGOSTO 2019. IC-06, pág.: 1 a la 6.

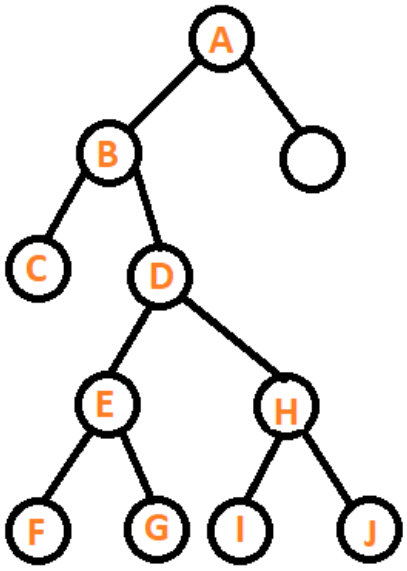


**Fig. 5. Árbol de búsqueda que representa al laberinto de la figura 4.**

## Diseño del hardware

Se generó un vehículo robotizado que tiene como gestor de recursos, entradas y salidas, la placa Arduino UNO, el prototipo puede verse en la figura 6.

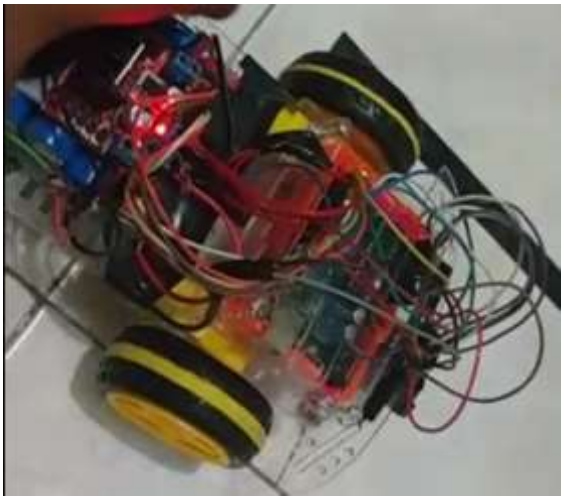

**Fig. 6. Prototipo del vehículo que resuelve laberintos.**

Para el sistema de potencia que controla a las llantas, se utilizo un Puente H L298N, controlando el grado de potencia por medio de *Pulse Width Modulation* (PWM).

Como ya se mencionó anteriormente, la forma en la que el vehículo recorre los laberintos es por medio de un sensor sigue líneas, el sensor utilizado fue el QTR-8A, que es un arreglo de 8 sensores reflectivos, lo podemos visualizar en la figura 7.

**Fig. 7. Sensor QTR-8A.**

La finalidad del QTR-8A, es identificar las líneas con las que el laberinto esta compuesto por medio de los sensores reflectivos centrales. También sirve para identificar los indicadores de bifurcación y de final del laberinto, esto peor medio de los sensores reflectivos de los costados (ver figura 8). Para utilizar este sensor con Arduino, es necesario trabaja con la librería QTRSensors.

Estos sensores son la base primordial de la interacción entre hardware y software del sistema completo, ya que estos sensores son los que enviaran las señales a la Placa Arduino para generar nuevos nodos dentro de nuestro árbol de búsqueda.

Todos los sensores, las baterías y la placa Arduino fueron montados sobre un chasis de bajo costo fabricado en acrílico, tal como se ve en la figura 9.

**Fig. 8. Señales de bifurcación que el vehículo puede detectar por medio de los sensores reflectivos externos.**

AÑO 3. No. 3. ISSN-2448-7236. SEPTIEMBRE 2018 – AGOSTO 2019. IC-06, pág.: 1 a la 6.

**Fig. 9. Chasis de bajo costo, fabricado en acrilico de 3mm.**

## COSTO DE DESARROLLO

La realización de este proyecto se llevó a cabo con costos reducidos de elaboración. En la parte de *software* no se realizó ningún tipo de gasto económico, ya que solo han sido utilizadas herramientas de software libre y de acceso gratuito, sin embargo, en la parte de *hardware* si ha sido necesaria la adquisición de diferentes dispositivos electrónicos, aunque se ha buscado que dichos dispositivos sean de bajo costo.

Los costos de los dispositivos electrónicos utilizados suelen ser variables, en la tabla 1 podemos observar los rangos promedio de los costos representativos de los dispositivos utilizados.

**Tabla 1.- Costos promedio de los dispositivos electrónicos utilizados para la generación del proyecto.**

| **Dispositivo electrónico** | **Precio promedio en pesos** |
|---|---|
| Arduino UNO | $ 200 |
| Sensor QTR-8A | $ 160 |
| Chasis y llantas | $ 250 |
| Puente H | $ 50 |
| **Costo promedio total** | **$ 660** |

## PRUEBAS Y ANALISIS

El vehículo se puso en marcha sobre un laberinto trazado con líneas negras, se colocaron marcas que le indicaban al algoritmo la posición final, y bifurcaciones, tal como se ve en la figura 10. El robot logró salir en el 80% de las ocasiones en que se sometió a prueba. El vehículo puede salir de dos maneas diferentes, la primera es en un recorrido ciego, es decir, realiza el árbol binario y lo va generando hasta el punto de salida. Y el otro es el recorrido informado, en es te caso el vehículo ya guardo en su memoria el árbol de búsqueda que representa al laberinto, y busca llegar a la salida de forma directa, dicho de otra forma, ya no pierde en tiempo recorriendo caminos que no llevan al punto de salida.

Las ventajas que tiene esta propuesta con respecto a otras existentes, es que este esquema no requiere costosas paqueterías de software, ya que toda la programación, ejecución, y procesamiento de la información se llevan a cabo dentro de la placa de desarrollo Arduino. Lo que implica que tampoco requiere de elementos externos costosos como una PC, para monitorizar y procesar su información. Los elementos usados, son dispositivos y sensores no solamente económicos, sino también de fácil acceso, cualquier individuo puede tener la posibilidad de adquirirlos.

En el caso de la propuesta dada en [1], se menciona la necesidad de un celular que toma las imágenes para poder procesar la información antes de que el robot comience la búsqueda de la salida, en nuestro caso, no se requieren este tipo de dispositivos adicionales para su funcionamiento, ya que el software del vehículo presentado irá generando una imagen en memoria de los puntos del laberinto. Como desventaja con respecto a otras implementaciones, es que presenta una resolución del problema lenta, ya que es necesario recorrer varios caminos antes de encontrar el camino correcto. Sin embargo, no es necesario recorrer todos los caminos existentes, como es el caso del algoritmo de la mano derecha.

**Fig. 10. Vehículo realizando el recorrido en un laberinto aleatorio.**

## CONCLUSIONES

En este trabajo se ha presentado el diseño e implementación de un vehículo robotizado que es capaz de encontrar a ruta de salida de un esquema laberintico. La implementación en software fue

AÑO 3. No. 3. ISSN-2448-7236. SEPTIEMBRE 2018 – AGOSTO 2019. IC-06, pág.: 1 a la 6.

realizada por medio de programación recursiva, generando un árbol de búsqueda en la que cada bifurcación del laberinto generaba un nuevo nodo, si el nodo tiene una ruta a seguir el algoritmo guarda en un nivel anterior de recursividad los datos del nodo anterior, es decir, guarda los datos de la última bifurcación recorrida. Logrando que cuando no encuentre la salida en una dirección tomada, pueda regresar automáticamente hacia la bifurcación más cercana hacía atrás recordada.

Una de las ventajas de la propuesta, es que se realizó de una manera económica, ya que no es necesario utilizar computadoras externas al robot, ni paqueterías de software costosas, todo se realiza dentro de la placa Arduino, la cual es sumamente barata y fácil de conseguir. Tampoco fueron necesarios aditamentos como *web cam*, o dispositivos de adquisición de imagen, ya que el reconocimiento del laberinto se realizó por medio del sensor e línea QTR 8A, el cual es relativamente barato. En el caso de software, el software se realizó de una manera sencilla, sin necesidad de algoritmos que requieran muchos requerimientos de procesamiento y almacenamiento.

## REFERENCIAS Y BIBLIOGRAFÍA

[1] Rahnama, B., Elçi, A., & Metani, S. (2012, July). An image processing approach to solve Labyrinth Discovery robotics problem. In *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual* (pp. 631-636). IEEE.

[2] Powers, F. A., & Zanarotti, S. R. (1995). *U.S. Patent No. 5,442,784*. Washington, DC: U.S. Patent and Trademark Office.

[3] Adelson-Velskiy, G. M., Arlazarov, V. L., & Donskoy, M. V. (1975). Some methods of controlling the tree search in chess programs. *Artificial Intelligence*, *6*(4), 361-371.

[4] Cerezo-Sánchez, J., Saldaña-González, G., Bustillo-Díaz, M. M., & Ata-Pérez, A. Sistema de planificación de trayectorias utilizando visión artificial. *Advances in Intelligent Technologies and its Applications*, 141.

[5] Ivan, C., Juan, P., Juan, R., Ferney, H., & Fabio, D. (2013, October). Implementación de una red neuronal artificial tipo SOM en una FPGA para la resolución de trayectorias tipo laberinto. In *Engineering Mechatronics and Automation (CIIMA), 2013 II International Congress of* (pp. 1-6). IEEE.

[6] Elshamarka, I., & Saman, A. B. S. (2012). Design and implementation of a robot for maze-solving using flood-fill algorithm. *International Journal of Computer Applications*, *56*(5).

[7] Singh, A., & Sekhon, G. S. (2011). A New Shortest Path Finding Algorithm For a Maze Solving Robot With Simulator. *Interna-tional Journal of Computer Science and Communication*, *2*(2), 445-449.

[8] Sadik, A. M., Dhali, M. A., Farid, H. M., Rashid, T. U., & Syeed, A. (2010, October). A comprehensive and comparative study of maze-solving techniques by implementing graph theory. In *Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on* (Vol. 1, pp. 52-56). IEEE.

## INFORMACIÓN ACADÉMICA

**Santiago Roa:** Estudiante de la licenciatura "Ingenieria en Telecomunicaciones, Sistemas y Electrónica", impartida en la FES Cuautitlán, Campo 4.

**David Tinoco Varela**: Ingeniero Mecánico Electricista egresado de la Facultad de Estudios Superiores Cuautitlán de la UNAM, Maestría y doctorado en Ciencias por la UNAM.