UNIVERSIDAD DE LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA ELÉCTRICA

Diseño e implementación de un algoritmo que permita mapeo y ubicación simultánea en un laberinto desconocido, basado en robot 3pi.

Br. Ángel Luis Coa Millán.

Mérida, Abril, 2018

UNIVERSIDAD DE LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA ELÉCTRICA

Diseño e implementación de un algoritmo que permita mapeo y ubicación simultánea en un laberinto desconocido, basado en robot 3pi.

Trabajo de Grado presentado como requisito parcial para optar al título de Ingeniero Electricista

Br. Ángel Luis Coa Millán

Tutor: Francisco J. Viloria M.

Mérida, Abril, 2018

UNIVERSIDAD DE LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA ELÉCTRICA

Diseño e implementación de un algoritmo que permita mapeo y ubicación simultánea en un laberinto desconocido, basado en robot 3pi.

Br. Ángel Luis Coa Millán

Trabajo de Grado, presentado en cumplimiento parcial de los requisitos exigidos para optar al título de Ingeniero Electricista, aprobado en nombre de la Universidad de Los Andes por el siguiente Jurado.

_____ _____

Prof. Marco Molina Prof. Oscar Blanco

_____

Prof. Francisco J. Viloria M.

DEDICATORIA

A **mis padres, Juan Manuel y Janett De Jesus,** que han sido mi pilar principal en este largo camino de formación académica y personal.

A **mis hermanos, Maria Eugenia y Luis Manuel,** por ser ese apoyo incondicional con el que siempre pude contar.

A **mi tía, Petra Del Valle,** por sus valiosos consejos y apoyo espiritual.

Ángel Luis Coa Millán

AGRADECIMIENTOS

A la ilustre **Universidad de los Andes** por abrirme sus puertas como casa de estudios y así convertirse en mi alma máter.

A los **profesores de la Escuela de Ingeniería Eléctrica de la Universidad de los Andes**, por el tiempo que se tomaron en expandir mis conocimientos con pasión y dedicación.

A mi tutor, el **Profesor Francisco J. Viloria M.**, por brindarme su apoyo durante la elaboración de este trabajo de grado. Su paciencia, consejos y anécdotas fueron clave en la elaboración de este proyecto. Estoy eternamente agradecido por ayudarme a encontrar mi pasión por la robótica

A mi **familia,** que sin ellos no hubiese comprendido la importancia que tienen los estudios superiores en el desarrollo personal humano. Sus consejos y apoyo a lo largo de esta experiencia universitaria me ayudaron a continuar con espíritu fuerte.

A mis **amigos,** tanto a los que tuve la dicha de conocer en la Facultad, como a los que estuvieron a mi lado antes y durante toda mi experiencia universitaria. Les agradezco todas las experiencias vividas y los momentos compartidos; hicieron inolvidable esta etapa de mi vida.

Ángel Luis Coa Millán. Diseño e implementación de un algoritmo que permita mapeo y ubicación simultánea en un laberinto desconocido, basado en robot 3pi. Universidad de los Andes. Tutor. Prof. Francisco J. Viloria M. **Abril** 2018.

RESUMEN

El diseño e implementación de algoritmo que permita mapeo y ubicación simultánea en un laberinto desconocido, basado en robot 3pi, desarrollado en este trabajo, es un sistema completo que permite la exploración y mapeo de laberintos complejos, de una manera sencilla y clara. El sistema, además de contar con un robot autónomo, está conformado por un servidor que corre en Linux, capaz de procesar los datos de la exploración para realizar el mapeo y ubicación simultanea del robot dentro del laberinto, aplicando la última tendencia de la tecnología; el internet de las cosas. De esta manera, el usuario puede observar en tiempo real el laberinto a medida que se explora, desde la comodidad de un explorador de internet. Haciendo referencia al problema clásico de encontrar un objetivo dentro de un laberinto, la página web creada brinda la posibilidad de indicar un punto destino, hacia el cual el usuario desea que el robot se dirija. Su construcción se realizó en base a la plataforma robótica 3pi, el módulo WiFi ESP8266 y una laptop con sistema operativo Ubuntu en el cual corre un servidor Apache.

Descriptores: Arduino, internet de las cosas, inteligencia artificial, laberinto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La habilidad de un robot para moverse y conocer un espacio desconocido es un punto que ha ido agarrando importancia en los últimos años, junto a investigaciones de carros autónomos y los reconocidos robots desarrollados por la Boston Dynamics. Esta habilidad supone una gran ventaja a la hora de explorar áreas en los cuales un ser humano no podría entrar de manera segura o eficiente.

Además, el internet de las cosas es una tecnología con mucho auge actualmente, inclusive jugando un papel principal en las nuevas tendencias económicas de la nueva revolución industrial. La obtención de datos a distancia a través de la plataforma que brinda el internet, brinda una infinidad de posibilidades al desarrollo de la vida humana en el futuro.

Es por esto que se ha tomado la decisión de realizar esta investigación, incluyendo las últimas tendencias tecnológicas y, de esta manera, aprender sobre ellas, buscando aclarar las posibilidades y alcances que brindan.

Se comienza haciendo el planteamiento del 3pi como plataforma robótica adecuada para los fines de esta investigación, así como la instrumentación utilizada en la construcción del robot; luego se muestra el trabajo de acondicionamiento del 3pi y los sensores usados, es decir, diseño de placa, calibración de instrumentos y establecimiento de la comunicación.

Expuesto lo anterior, se prosigue a plantear los métodos y conceptos utilizados en el desarrollo de la solución, como lo son las técnicas de búsqueda de inteligencia artificial, manejo de base de datos, algoritmo base de la exploración y el importante rol que juega la comunicación utilizando IoT. Se continúa mostrando los algoritmos desarrollados para todas las partes (robot, servidor web y módulo de comunicación WiFi) en el proceso de exploración, así como también, al momento de calcular y seguir la ruta. Finalmente se exponen las pruebas y resultados obtenidos.

CAPÍTULO I
JUSTIFICACIÓN

En el siguiente capítulo se exponen la justificación, objetivos, metodología y alcance de esta investigación.

1.1 JUSTIFICACIÓN

El desenvolvimiento de un robot autónomo en un área desconocida, es un tema de investigación muy importante en los últimos años. Desde aplicaciones para el hogar, como en las industrias y hasta en las calles, se pueden observar cada vez más entes autónomos que deben movilizarse en espacios desconocidos. Esto supone un desarrollo importante de algoritmos y métodos que permitan realizar esta tarea de manera efectiva, por ende, toda investigación orientada hacia esta dirección brinda una nueva perspectiva sobre esta problemática.

1.2 OBJETIVOS

1.2.1 General

Desarrollar un algoritmo avanzado, que permita el mapeo y ubicación simultánea, de un robot 3pi, en un laberinto desconocido de dificultad avanzada; así como, calcular el camino más corto a un punto dentro del laberinto, indicado por el usuario.

1.2.2 Específicos

- Diseñar e implementar del algoritmo de exploración en laberintos de dificultad avanzada.
- Diseñar e implementar el algoritmo de ubicación.
- Diseñar e implementar el algoritmo de cálculo de distancia.
- Establecer comunicación inalámbrica entre el robot y una computadora.
- Diseñar e implementar el algoritmo de mapeo en una computadora.

1.3 METODOLOGÍA

A partir de la caracterización teórica del problema y, tomando en consideración el análisis del mismo, se definirán las técnicas a ser utilizadas. Se contará con módulos que permitan la elaboración de distintos laberintos reales. Al ensamblar todos los componentes se realizarán las pruebas correspondientes para validar el prototipo.

1.4 ALCANCE

Diseño y construcción de un robot autónomo capaz de explorar un laberinto desconocido que permita:

- Explorar cualquier laberinto sin importar su complejidad.
- Recoger los datos necesarios que permitan crear un mapa del laberinto en exploración.
- Conocer su ubicación dentro del laberinto a medida que es explorado
- Calcular la ruta más corta hacia un punto aleatorio dentro del laberinto.

CAPÍTULO II
PLANTEAMIENTO DEL 3PI COMO PLATAFORMA ROBÓTICA ADECUADA

En este capítulo se plantea un esquema a través del cual se expone claramente la forma en que el robot móvil 3pi está constituido; es decir, se describen todas las bondades que éste brinda y porque ha sido seleccionado como plataforma de desarrollo.

2.1 DESCRIPCIÓN DEL ROBOT 3PI

El robot 3pi es una plataforma móvil desarrollado por Pololu Corporation [1]. Las figuras 2.1 y 2.2 muestran las vistas superior e inferior de la placa que forma el robot.

Como es de esperar, el 3pi es gobernado por un microcontrolador ATmega328P (ver figura 2.2); éste proporciona los puertos de entrada y salida que permitirán actuar sobre los demás elementos que conforman al 3pi. Además de esto, cuenta con dos motores (cada uno con su tren de engranes y su respectiva rueda) junto a una rueda multidireccional (rueda loca) que le brinda la posibilidad de avanzar, retroceder, y realizar giros con tan solo 2 motores; lo cual simplifica considerablemente la manera en que se ejecutan los movimientos del robot. Todo esto forma parte del sistema que le proporciona movilidad al robot 3pi.

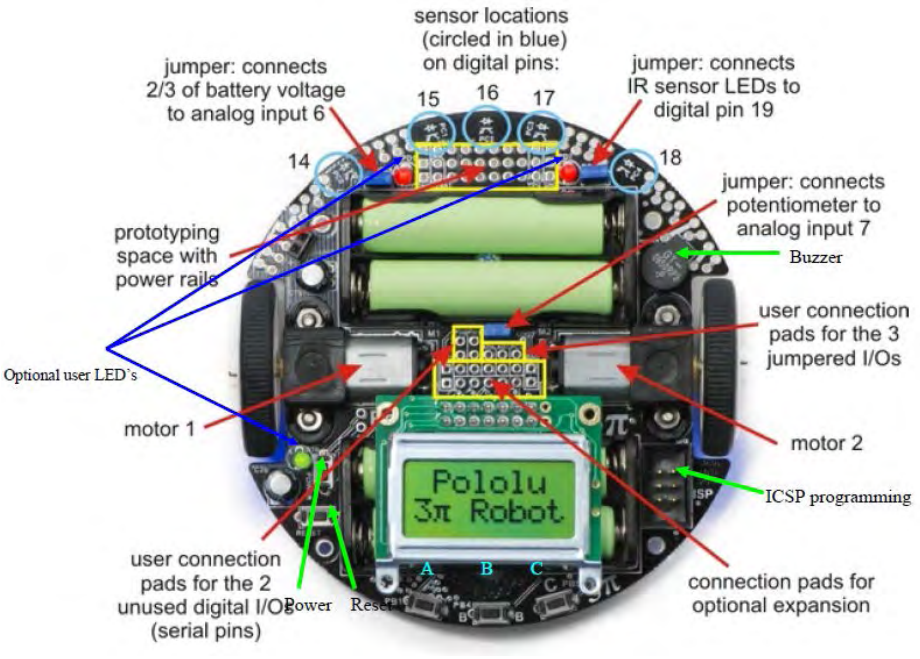

Figura 2.1 Vista superior del robot móvil 3pi [1]

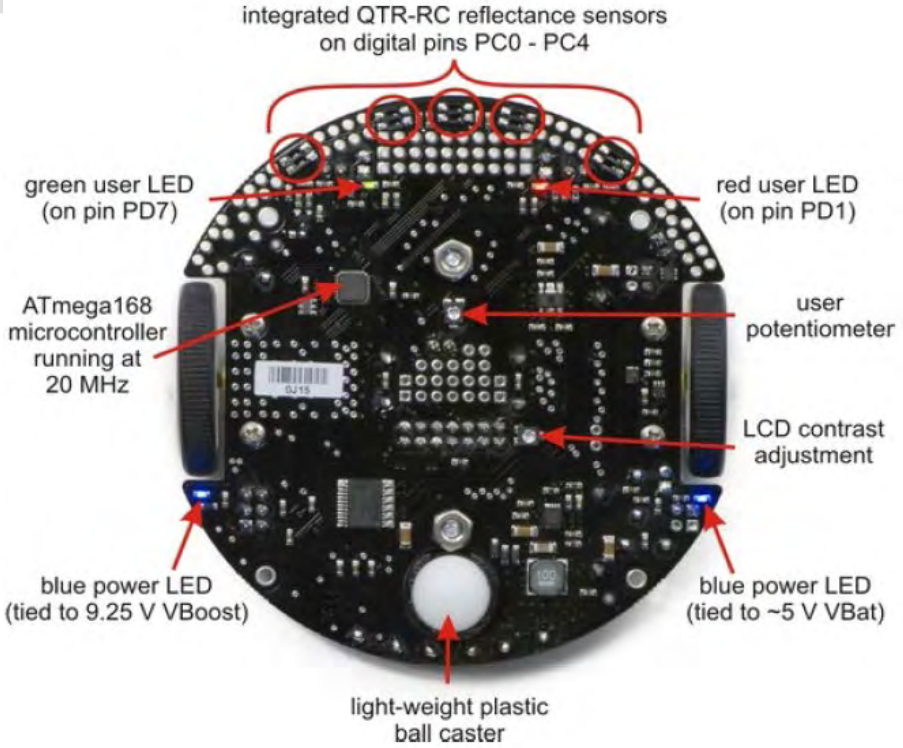

Figura 2.2 Vista inferior del robot móvil 3pi [1]

Adicionalmente cuenta con conectores que permiten expandir las funciones del robot móvil 3pi, al añadirle sensores o módulos de interés por parte del usuario.

Es importante destacar que para descargar las instrucciones en el microcontrolador bajo código en lenguaje C, es necesario hacer uso de un programador externo como el programador Pololu Orangután USB o la serie AVRISP de Atmel [1], mediante un cable ISP que conecta el micro al dispositivo programador; además, se requiere de un cable USB que conecte el programador con el PC).

2.2 SISTEMA DE ALIMENTACIÓN

Es importante que se comience por desglosar el funcionamiento de la gestión de energía en el 3pi, dado que proporcionará un primer acercamiento sobre cómo éste está constituido.

Toda batería con el tiempo de uso va disminuyendo el voltaje que es capaz de proporcionar, lo cual afecta el comportamiento de los componentes eléctricos alimentados por ella. Por esta razón es de gran importancia proporcionar la mayor estabilidad posible en el voltaje de alimentación de cada elemento, y es aquí donde el sistema de alimentación juega un rol clave en el rendimiento del robot.

Con este propósito, dado que al alimentar directamente de las baterías los elementos del sistema es muy inestable y, además, se suelen requerir distintos niveles de voltaje para algunos componentes, se utilizan reguladores de voltaje que mantengan durante el mayor tiempo posible la tensión deseada en cada uno de ellos.

2.2.1 Reguladores lineales

Estos reguladores utilizan un circuito retroalimentado con un transistor que controla la tensión de salida al ajustar continuamente la caída de voltaje a través del transistor [2]; el transistor se encuentra conectado en serie entre la entrada no regulada y la carga, por esta razón también son llamados reguladores en serie.

Este tipo de regulador es muy sencillo de usar, aunque resulta muy ineficiente debido a su alto consumo de potencia, ya que son utilizados para producir una disminución del voltaje de entrada a un valor determinado, y el resto del potencial se pierde. Esta pérdida se transforma en calor, el cual puede ser demasiado para ser controlado por disipadores; por esta razón los reguladores lineales son utilizados en aplicaciones de baja potencia.

2.2.2 Reguladores conmutados (switching)

Estos reguladores utilizan un transistor como conmutador de alta frecuencia que transfiere la energía de la entrada a la carga en paquetes discretos [2]. Luego de esto la señal se convierte en corriente continua a través de un filtro capacitivo-inductivo; este último siendo un elemento clave en este tipo de regulador, debido a que es el encargado de almacenar la energía. Estos dispositivos son utilizados para elevar voltajes y mantenerlos regulados.

Dado que el transistor opera como conmutador, consume menos potencia que en su zona lineal, resultando en un regulador mucho más eficiente (hasta el 80 %) que el regulador lineal. A pesar de esta gran ventaja, presentan un rizado más pronunciado que su contraparte lineal.

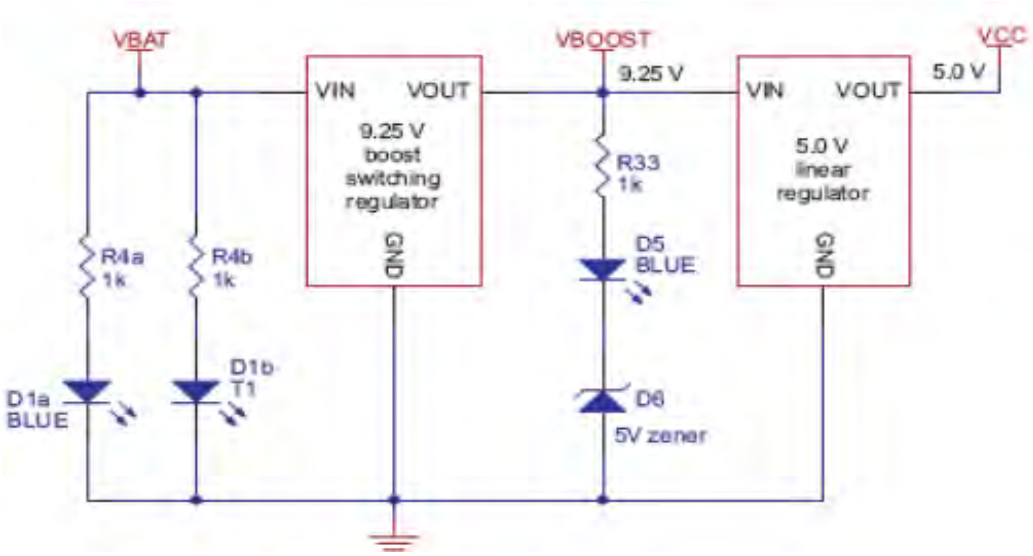

Figura 2.3 Sistema de potencia del 3pi [1]

En la figura 2.3 se muestra el esquema circuital de la gestión de energía en el 3pi. Se observa que se hace uso de ambos tipos de reguladores; el de conmutación para elevar el voltaje de la batería (que puede variar entre 3.5 V y 6 V) a 9.25 V, y el lineal para reducir el voltaje de salida del conmutado a 5 V.

El voltaje regulado a 9.25 V es utilizado para los motores y los leds IR, a la vez que el voltaje reducido de 5 V es reservado para el microcontrolador y las señales digitales.

Al tener los motores alimentados con un voltaje fijo, se garantiza que trabajen a velocidad constante aunque varíe la tensión en los terminales de la batería; esto permite realizar la calibración precisa de los giros a 90º.

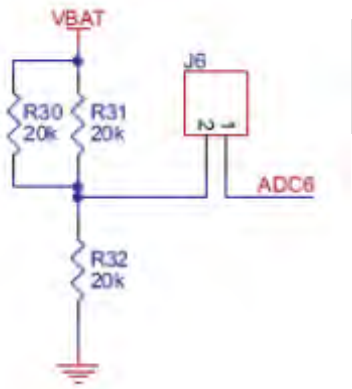

Figura 2.4 Circuito de monitorización de la batería [1]

La figura 2.4 muestra el esquema del circuito utilizado en el 3pi que permite medir el estado de la batería. Debido a que el 3pi funciona a máximo rendimiento hasta que de repente se detiene (en vez de agotarse progresivamente), y que en la guía de usuario del 3pi se hace mucho énfasis en el peligro de programar el 3pi con las baterías descargadas o que éstas

pierdan toda su energía, es de suma importancia hacer uso de dicho circuito de monitorización de baterías.

> "No intentes programar el 3pi con las baterías descargadas, puedes inutilizarlo de forma permanente. Si le compras baterías recargables, comprueba que estén cargadas." [1, P.2]. "Para cualquier batería en funcionamiento, el voltaje suministrado se reduce con el tiempo bajando hasta perder toda la energía (procurar no llegar a ello, puede producir cortocircuito y quedar inutilizada para siempre)." [1, P.6].

El circuito de monitoreo de la figura 2.4 consiste en un sencillo divisor de tensión cuya salida proporciona 2/3 del voltaje en los terminales de la batería a uno de los conversores analógico-digital del microcontrolador. Además de esto la librería AVR Pololu [1] contiene una función que permite devolver el valor de tensión de las baterías en mV.

2.3 MICROCONTROLADOR ATMEGA328P

Un componente clave en el diseño de cualquier robot es el "cerebro" del mismo, el cual brindará las ventajas y limitaciones principales y dará pie a la expansión de nuevos elementos. El microcontrolador con el que cuenta el robot móvil 3pi, es el modelo ATmega328P de la marca Atmel [3] (ver figura 2.5).

Figura 2.5 Microcontrolador Atmega328P [4]

Tal como se observa en la tabla 2.1, el microcontrolador ATmega328P cuenta con 3 puertos de entrada y salida; el puerto B maneja señales digitales, el puerto C tanto señales

digitales como analógicas y el D en su mayoría digitales, pero también algunas pocas analógicas; todo esto además de los pines destinados a alimentación, tierra, voltaje de referencia y 2 convertidores A/D [3].

Con respecto al consumo se observa claramente que éste presenta valores muy bajos (una máxima de 0.2 mA), precisando de una vez que dichos valores son dados por el fabricante para el voltaje mínimo de alimentación de 1.8 V. En el sistema de alimentación del robot móvil 3pi se especifica que el microcontrolador es alimentado con 5 V. "…VCC es para el microcontrolador y las señales digitales." [1, P.7]. (Ver figura 2.3).

Tabla 2.1 Principales características del Atmega328P

Puertos	**3 (B,C,D)**
Memoria de programa	**32 KBytes**
Memoria RAM	**2 KBytes**
Reloj Max.	**20 MHz**
Analógico	**Convertidor A/D de 10 bits y 8 canales**
Pines	**32**
Alimentación	**5 V (5 V en programación)**
Temporizadores	**3 módulos**
Módulos Captura/Compara/PWM	**6 Canales**
Comunicación	**SPI, I2C, USART**
Consumo (1 MHz de Fosc, 1.8 V, 25 °C)	**0.2 mA en modo activo** **0.1 µA en modo apagado** **0.75 µAen modo ahorro de energía**

> “El 3pi tiene un conector estándar de 6 pines que permite la programación directa del micro mediante un cable ISP que se conecta al dispositivo programador. También necesitas un cable USB que conecte el programador con el PC.” [1, P.3].

La afirmación anterior indica que dicha conexión al PC permite descargar y almacenar el código a ser ejecutado por el microcontrolador a través del bootloader que se encuentra en la memoria de arranque del ATmega328P. Haciendo uso de un compilador y software de programación en computador, se codificarán las instrucciones a ser procesadas por el microcontrolador.

Por último, respecto a las funcionalidades que posee el ATmega328P respecto al manejo de temporizadores; es de suma importancia cuando se desea controlar la velocidad de motores, el cual se explica en el siguiente punto.

2.4 MOTORES Y ENGRANAJES

El motor DC de escobillas se usa bastante en el mundo de la robótica, y no es de extrañar que forme parte del diseño del robot móvil 3pi. Este tipo de motor está constituido por imanes permanentes en el exterior y bobinas electromagnéticas montadas en el eje del motor (rotor).

Las escobillas son piezas de carbón que se encuentran fijas a la carcasa del motor, y mediante resortes se mantienen presionadas a las delgas del rotor de la máquina, proporcionando el flujo de corriente de una parte del bobinado a la otra y produciendo una serie de pulsos magnéticos que provocan el giro del eje en la misma dirección.

Existen dos características de interés en los motores DC: Velocidad (rpm) y el par de fuerza o torque (oz.in). “Cada motor tiene una velocidad máxima (sin resistencia aplicada) y un par máximo (cuando el motor está completamente parado).” [1, P.8].

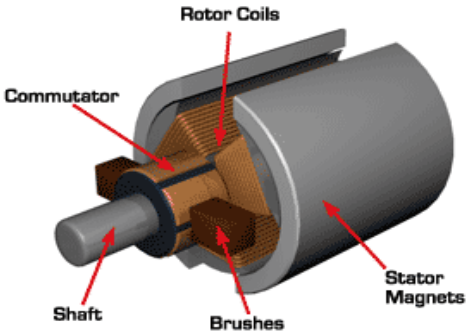

Figura 2.6 Motor DC [5]

Según se ha citado, estos 2 estados del motor son denominados funcionamiento a velocidad libre y par de parada, respectivamente. Como es de imaginar, el motor consume un mínimo de corriente a velocidad libre, y al aplicar fuerzas o resistencias dicho consumo va aumentando hasta que se alcanza el par de parada, en el cual se encuentra el consumo máximo de corriente (ver figura 2.7). El consumo de corriente se ve afectado por las pérdidas mecánicas internas del motor, debido al roce de las piezas como lo son los engranajes.

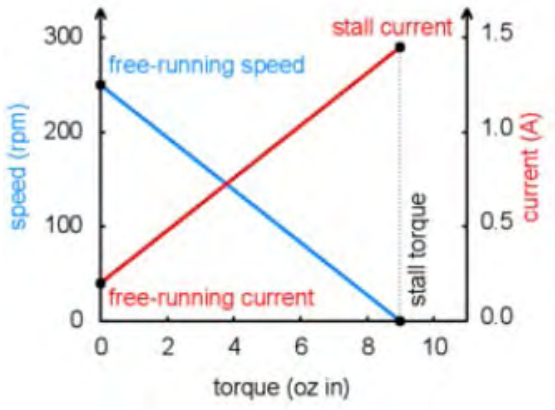

Figura 2.7 Dependencia de velocidad, torque y corriente [1]

La función que existe entre la velocidad y el torque del motor depende de la relación del engranaje de la máquina, el cual es de vital importancia debido a que sin este sistema de engranajes, el motor DC presentaría una velocidad muy elevada, lo cual es muy alto para el desplazamiento del robot; además, de esta manera se le puede otorgar mayor fuerza al motor.

Tabla 2.2 Parámetros de motor DC en el 3pi

Engranaje	**30:1**
Velocidad libre	**700 RPM**
Consumo mínimo	**60 mA**
Par máximo	**6 oz.in**
Consumo máximo	**540 mA**

En la tabla 2.2 se muestran las características del motor DC que forma parte del robot móvil 3pi. Se expresa que la relación del engranaje es 30:1, lo cual indica que por cada 30 vuelvas del motor la rueda gira una vez. Como suele suceder, el consumo del motor DC es alto respecto al resto de la circuitería; en este caso es de 540 mA por cada motor, resultando en 1.08 A de consumo total por los dos motores [1].

Otro parámetro importante que brinda la tabla 2.2 es el par máximo producido por el motor; se indica que éste es de 6 oz.in por cada motor, lo cual significa que se cuenta con un total de 12 oz.in, y dado que cada rueda presenta un radio de 0.67 in, al aplicar la ecuación 2.1 se obtiene una fuerza de aproximadamente 18 oz [1].

$$F = \frac{N° \ de \ motores * torque}{radio} \qquad (2.1)$$

En relación con esto último el 3pi pesa 7 oz con las baterías insertadas, lo que significa que estos motores presentan suficiente fuerza para mover al robot móvil 3pi. “El rendimiento está limitado por la fricción de las gomas: podemos deducir que puede trabajar con pendientes de entre 30º a 40º.” [1, P.8].

Por último, en el sistema de potencia del 3pi se indica que los motores DC son alimentados a 9.25 V. “…Vboost sirve para los motores…” [1, P.7]. (Ver figura 2.3).

2.4.1 Control de dirección de giro del motor

Hechas las consideraciones anteriores, el siguiente punto de importancia hace referencia al control de los motores DC. Para realizar cambios en la dirección de ellos, es necesario cambiar la polaridad del voltaje aplicado; esto se hace posible de manera práctica al hacer uso de los llamados puentes H (ver figura 2.8).

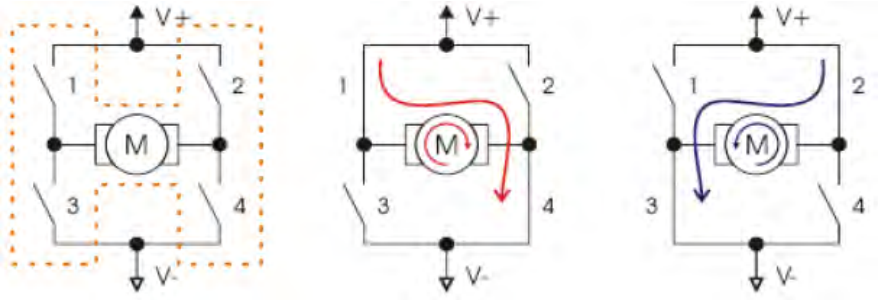

Figura 2.8 Funcionamiento del puente H [1]

En la figura 2.8 se muestra un diagrama del funcionamiento del puente H. En éste se puede apreciar la manera de cambiar el sentido de la corriente que circula a través del motor, al igual que la polaridad del voltaje aplicado, provocando una alteración en el sentido de giro del motor. Como se observa en dicho diagrama esto es posible mediante el uso de interruptores que conectan y desconectan los terminales del motor DC; los interruptores en este caso son los transistores.

El circuito integrado utilizado en el robot 3pi es el TB6612FNG, el cual es conectado a los siguientes pines del microcontrolador configurados como salidas: PD5 y PD6 para el motor M1; PD3 y PB3 para el motor M2 [1].

Tabla 2.3 Tabla de la verdad correspondiente a los motores DC

PD5	PD6	1	2	3	4	M1	PD3	PB3	1	2	3	4	M2
0	0	Off	Off	Off	Off	Parada	0	0	Off	Off	Off	Off	Parada
0	1	Off	On	On	Off	Adelante	0	1	Off	On	On	Off	Adelante
1	0	On	Off	Off	On	Reversa	1	0	On	Off	Off	On	Reversa
1	1	Off	Off	On	On	Frenado	1	1	Off	Off	On	On	Frenado

Se observa claramente en la tabla 2.3 todas las posibles combinaciones de las salidas del Atmega328P destinadas a controlar ambos motores, así como el resultado que éstas tienen sobre los interruptores de los puentes H y por consiguiente en cada motor.

2.4.2 Modulación por ancho de pulso (PWM)

Esta es una técnica que consiste en variar el ciclo de servicio de una señal cuadrada, manteniendo constante su periodo. De esta manera se controla el porcentaje de tiempo en el cual la señal permanece en un “1” lógico (5 V).

Si la señal permanece ¼ del tiempo en alto (25 % del ciclo de trabajo) el motor girará al 25 % de su velocidad máxima; de la misma manera si la señal permanece ¾ del tiempo en alto el motor girará al 75 % de la velocidad máxima. De esta manera se puede controlar de manera fiable la velocidad de cada motor.

Con este propósito el 3pi consigue generar las señales PWM a través de los temporizadores TIMER0 y TIMER2, los cuales brindan la posibilidad de establecer el ciclo de servicio que se desea obtener en las señales de ambos motores; estas señales continuarán siendo generadas en segundo plano sin importar lo que se esté ejecutando según el resto del código.

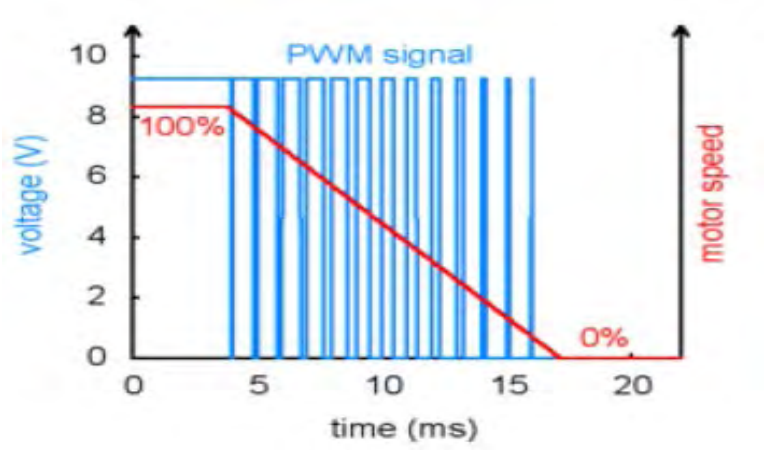

Figura 2.9 Decremento progresivo en la velocidad de un motor mediante PWM [1]

En la figura 2.9 se muestra el efecto en la velocidad de un motor al disminuir progresivamente el ciclo de servicio de la señal que lo alimenta. Se aprecia que a medida que dicho ciclo de servicio disminuye, el voltaje eficaz aplicado al motor, y por ende la velocidad, se degradan en la misma proporción hasta que eventualmente se detiene el motor.

2.4.3 Giro con conducción diferencial

El robot 3pi al contar con dos motores independientes a los lados, permite aplicar un método de conducción denominado conducción diferencial, o también conocido como "conducción de tanques". El método consiste en rodar cada motor a diferentes velocidades; el lado que se mueva con mayor lentitud es el lado hacia el cual el robot tiende a girar. A mayor diferencia rueden los motores, más brusco será el giro realizado (ver figura 1.10).

"La función set_motors() de la librería AVR Pololu(ver sección 6.a) crea el ciclo de trabajo usando una precisión de 8 bits por lo que un valor de 255 corresponderá al 100%. Por ejemplo para una velocidad del 67 % en el M1 y otra del 33 % en el M2 llamaremos a la función de la siguiente forma: set_motors(171,84)." [1, P.9].

Figura 2.10 Conducción diferencial [1]

2.5 PUERTOS DISPONIBLES PARA EXPANSIONES

Después de lo anteriormente expuesto resulta oportuno mencionar que de las funciones integradas en el 3pi, ya han sido nombradas las de interés; así mismo, el resto de las funciones necesarias deben ser añadidas al 3pi. En el orden de las ideas anteriores, se hace evidente que el robot móvil 3pi debe permitir expandir sus funcionalidades, y para que esto sea posible es necesario que se proporcionen pines de entrada y salida disponibles para la debida expansión.

En efecto, el 3pi permite la expansión de sus capacidades mediante puertos libres y ciertas modificaciones que se pueden realizar sin problemas. Resulta oportuno resaltar que el robot móvil 3pi presenta características muy flexibles y dinámicas debido a su alta capacidad de ser adaptado a las necesidades del usuario.

En el marco de las observaciones anteriores, la guía de usuario del 3pi especifica que los pines PD0 y PD1 son líneas digitales I/O disponibles que también pueden ser utilizadas para comunicación serial (RX y TX respectivamente); el pin PC5 puede ser liberado al remover el jumper que lo conecta al hardware del 3pi.

Además de éstos, al quitar la pantalla LCD se liberan los pines PB9, PD2, PD4, PB1, PD7, PB4 y PB5 (los dos últimos son usados como líneas de programación y se debe tener cuidado con que lo que se conecte a estos pines no entren en conflicto).

Todo lo anterior resulta en un máximo de 10 pines digitales I/O accesibles por el usuario para expandir las funciones del 3pi, acorde a lo requerido por él mismo (Ver figura 2.11). Cabe destacar que existen dos pines analógicos disponibles, pero éstos no son necesarios para el propósito de este trabajo.

En la figura 2.11 se aprecian claramente todos los pines disponibles al ser removida la pantalla LCD. El 3pi proporciona un kit de expansión que permite la conexión apropiada de estos puertos a una placa de expansión apropiadamente diseñada para estos casos (ver figura 2.12 y 2.13)

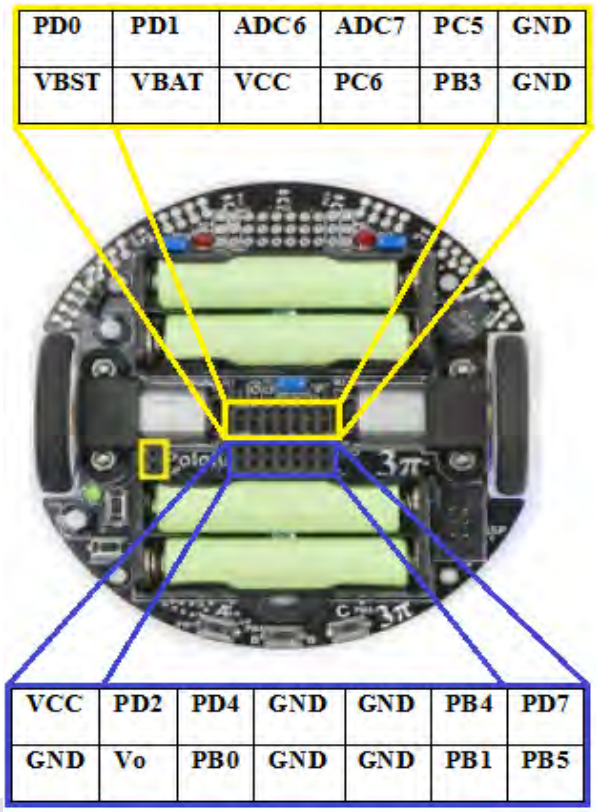

Figura 2.11 Pines de expansión en el 3pi

Figura 2.12 Tornillos, tuercas, espadines y conectores del kit de expansión [1]

En la figura 2.13 se observan los puntos en la placa de expansión a los cuales van los puertos expuestos en la figura 2.11.

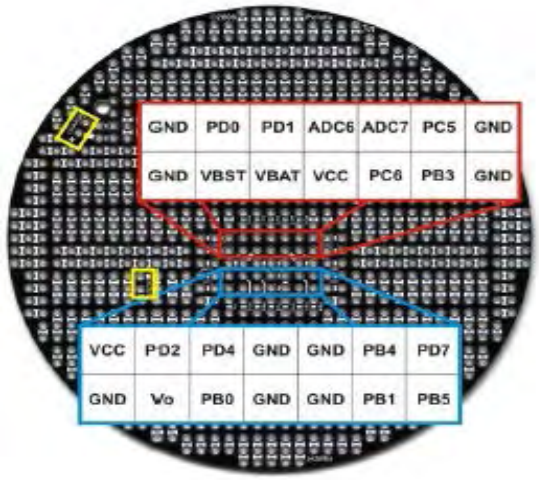

Figura 2.13 Placa de expansión [1]

Figura 2.14 3pi expandido con espacio para la pantalla LCD [1]

El 3pi brinda dos posibilidades al momento de expandir sus funcionalidades: con la pantalla LCD, para el cual se tiene una placa de expansión con espacio para dicha pantalla (ver figura 2.14) y otra placa de expansión completa que es la que se muestra en la figura 2.13 (la necesaria para este caso).

En consecuencia, esto es de gran ventaja debido a que es necesario hacer uso de sensores de distancia (los cuales no forman parte del 3pi) que permitan detectar la presencia o no, de paredes a través de los laberintos.

De la misma manera se desea mantener una comunicación inalámbrica con el robot y para ello se requiere utilizar el módulo de comunicación WiFi ESP8266, así como también, el magnetómetro que forma parte del módulo MPU-9250.

2.6 SENSOR DE DISTANCIA ULTRASÓNICO HC-SR04

A razón de que se requiere detectar presencia de paredes a lo largo de los laberintos, se torna imperativo el uso de un sensor de distancia como lo es el HC-SR04 [6] (Ver figura 2.15).

Este sensor, como su nombre lo indica, funciona emitiendo señales ultrasónicas (por encima de los 20 kHz) y registrando el tiempo que éstas toman en regresar.

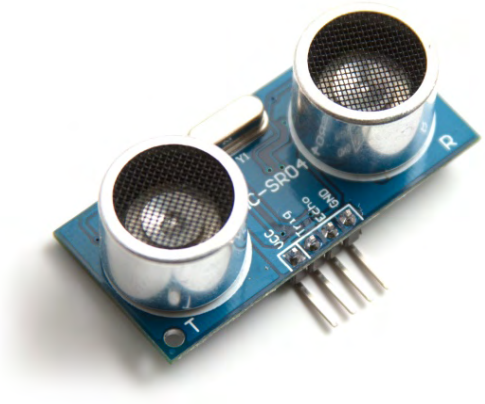

Figura 2.15 Sensor de distancia HC-SR04 [7]

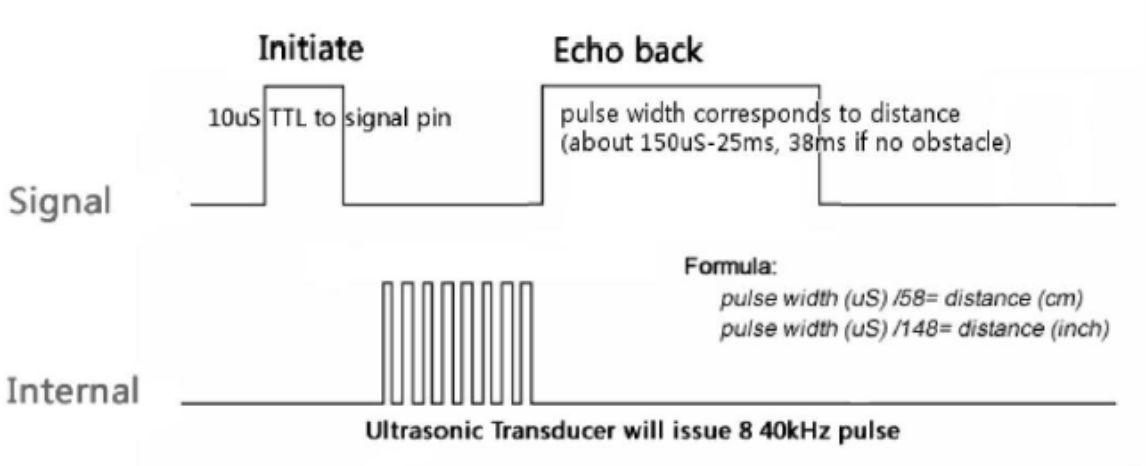

Figura 2.16 Diagrama de tiempos del HC-SR04 [6]

La manera en que funciona el registro de tiempos que permite la medición de distancia en el HC-SR04, es la que se muestra en la figura 2.16. El pin denominado “trigger” es una entrada digital, la cual es activada mediante una señal de nivel TTL (5 V) con duración mínima de 10 uS, esto ocasiona que el HC-SR04 emita una serie de 8 ciclos ultrasónicos de 40 kHz, y a la vez coloca en alto (“1” lógico) el pin de salida llamado “echo” hasta que dicha señal de 40 kHz rebota sobre una superficie y regresa al HC-SR04. De esta manera el tiempo queda registrado en la duración del pin “echo” en estado lógico “1”. Los nombres de los pines pueden ser observados en la figura 2.15.

Finalmente, la distancia a la cual se encuentra el HC-SR04 del objeto se calcula mediante la ecuación 2.2.

$$D = \frac{tr}{58}\ [cm] \qquad (2.2)$$

Siendo tr el tiempo en microsegundos de la señal en alto del pin “echo”. Esta fórmula toma en cuenta la velocidad del sonido en el aire (340 m/s).

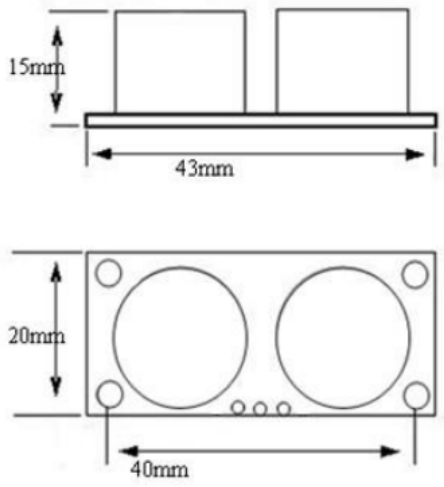

Figura 2.17 Dimensiones del HC-SR04 [6]

Tabla 2.4 Parámetros del HC-SR04

Voltaje de operación	**5 V**
Consumo de corriente	**15 mA**
Frecuencia de operación	**40 kHz**
Rango de medición	**(2 a 400) cm**
Ángulo de medición máximo	**15 °**

En la tabla 2.4 se resumen algunos parámetros ya nombrados, de una manera más clara. Además de esto, se observan algunas características de gran importancia: distancia mínima y máxima que puede ser medida por el HC-SR04, el ángulo de medición máximo que permite el mismo y el consumo de corriente. En la figura 2.17 se muestran las dimensiones que presenta

el HC-SR04. De esta manera se puede concluir que el sensor de distancia por ultrasonido HC-SR04 es apto para las necesidades que se tienen, dado que su rango de medición es más que suficiente para detectar las paredes del laberinto. No obstante, se debe tener extremo cuidado de no sobrepasar el ángulo de medición máximo.

2.7 MÓDULO WIFI ESP8266

Debido a que se requiere establecer una comunicación inalámbrica entre el robot y la PC, se hace uso del módulo WiFi ESP8266 [8] (ver figura 2.18), el cual permite una conexión a internet a través de los protocolos TCP/IP y 802.11 b/g/n; utilizando la respectiva banda de (2.4 a 2.5) GHz. En resumidas cuentas este módulo brinda una conexión total a internet.

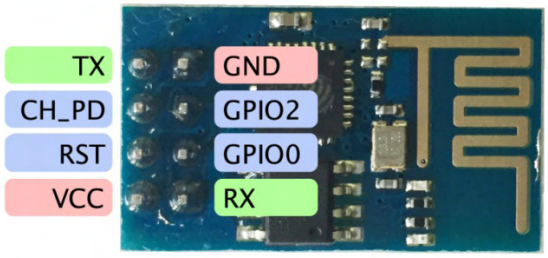

Figura 2.18 Módulo WiFi ESP8266 [9]

Como ya se mencionó, este módulo permite una conexión total a internet debido a la banda en la que opera, así como los protocolos WiFi y de manejo de paquetes de datos bajo los que se rige; estas características son clave ya que son las mismas que tiene cualquier equipo que se utiliza para conectarse a internet en el día a día; además de esto proporciona los estándar de

seguridad más altos en el mercado: Acceso Protegido WiFi 2 (WPA2) y el Estándar de Seguridad Avanzado (AES) [8].

De acuerdo con lo mostrado en la figura 2.18 se puede notar que el ESP8266, además de sus pines de alimentación (VCC Y GND), posee dos pines de entrada/salida de uso general (GPIO). Esto es debido a que el módulo ESP8266 está construido en base a un microcontrolador de 32 bit, lo cual quiere decir que puede ser programado ante las necesidades del usuario, y hacer uso de algunas de sus capacidades y pines de entrada/salida. Por esta razón el fabricante permite el acceso a dos de estos pines de propósito general. Cabe mencionar que el ESP8266 viene con un pin de reset (RST), así como también un pin de habilitación de canal (CH_PD).

Tabla 2.5 Opciones de arranque del ESP8266

GPIO0	GPIO1	Modo	Comentario
H	H	Flash	Bootear desde SPI (modo normal)
L	H	UART	Programar via UART (TX/RX)

Dichos pines de uso general sirven también para decirle al ESP8266 el modo en el que se desea trabajar, como se muestra en la tabla 2.5. Se observa que para programar el ESP8266 a las necesidades del usuario, se debe conectar a tierra el pin GPIO0 y en alto el GPIO1.

Tabla 2.6 Características del módulo ESP8266

Voltaje de operación	**(3.0 a 3.6) V**
Consumo de corriente	**80 mA en modo activo** **<10 uA en modo ahorro de energía** **<5 uA de corriente de filtrado apagado**
Rango de frecuencia	**(2.4 a 2.5) GHz**
Bus periférico	**UART/SDIO/SPI/I2C/I2S/IR Remote Control** **GPIO/PWM**
Protocolo WiFi	**802.11 b/g/n**
Protocolos de red	**IPv4, TCP/UDP/HTTP/FTP**
Seguridad	**WPA/WPA2**
Encriptación	**WEP/TKIP/AES**

En la tabla 2.6 se muestran algunas de las características más importantes del módulo ESP8266. El voltaje de operación de 3.3 V es una de las características que más resalta, debido a que estos niveles no suelen ser tan comunes a pesar de ser la tendencia en cuanto a componentes electrónicos.

Es importante recordar que el ESP8266 trabaja con lógica de 3.3 V, y por ende se deben tomar las respectivas consideraciones al trabajar con niveles lógicos de 5 V como la que maneja el ATmega328P (Esto se plantea con detalle en el capítulo III).

Como se puede observar tanto en la figura 2.18 como en la tabla 2.6, el ESP8266 tiene la capacidad de transmitir y recibir datos vía comunicación serial a través de los pines TX y RX, lo que permite recibir instrucciones del ATmega328P usando comandos AT; el cual es un estándar de comandos para configurar módems.

Dentro del marco de las ideas anteriores, se desea conectar el robot a internet aplicando internet de las cosas (IoT); un sistema interrelacionado de objetos físicos con la capacidad de conectarse a internet y ser controlados o monitoreados a través de esta. De esta manera se pretende establecer la comunicación entre el robot móvil 3pi y la PC, con la finalidad de obtener datos sobre la forma del laberinto explorado, así como la posición del 3pi dentro de él.

2.8 MODULO MPU-9250

A raíz de la necesidad que se presenta de corregir la orientación del robot, se decide hacer uso del módulo MPU-9250 [10], el cual es una unidad de medición inercial que combina los chips MPU6050 que contiene el acelerómetro y giroscopio de 3 ejes cada uno, y el AK8963 que contiene el magnetómetro de 3 ejes. Debido a que solamente es de interés la rotación alrededor del eje Z, se trabaja sólo con el magnetómetro en los ejes X y Y.

Figura 2.19 MPU-9250 [11]

El sistema microelectromecánico de 3 ejes que conforma al magnetómetro contenido en el MPU-9250 es un sensor monolítico de efecto hall con concentrador magnético y se ha escogido debido a su pequeño tamaño, bajo consumo y cantidad de ejemplos disponibles, lo cual facilita su uso. En la tabla 2.7 se observan las características más importantes del MPU-9250.

Tabla 2.7 Parámetros del MPU-9250

Voltaje de operación	**3 V**
Consumo de corriente	**280 uA**
Frecuencia de operación	**8 Hz**
Rango de medición a escala completa	**+-4800uT**
Resolución	**14 bit (0.6uT/LSB)**
Protocolo de comunicación	**I2C**

Como ya se mencionó anteriormente, el MPU-9250 contiene tres sensores: acelerómetro, giroscopio y magnetómetro, y como se muestra en la tabla 2.7, su comunicación es a través del protocolo I2C, el cual trabaja en base a direcciones de dispositivos a comunicar.

En el marco de estas ideas, es importante mencionar que cada sensor tiene su propia dirección, y por ende, se obtienen los datos de cada sensor por separado a través de I2C.

Como se muestra en la figura 2.20, el MPU-9250 no cuenta solamente con las líneas de comunicación SPI/I2C, sino también con un bus auxiliar de comunicación I2C con el

propósito de comunicarse con otros sensores externos que se puedan tener. En esta figura se expone de manera clara la separación entre el MPU6050 (acelerómetro y giroscopio) y el AK8963 (magnetómetro), los cuales comparten el mismo bus de comunicación.

Debido a que ambos chips comparten el mismo bus, el diseño del MPU-9250 cuenta con un multiplexor para acceder a la información del magnetómetro a través de las líneas I2C, al configurar el modo bypass.

Dicho modo permite acceder a los datos del acelerómetro, giroscopio y magnetómetro de manera segura al evitar conflicto de datos provenientes de ambos chips. El MPU-9250 también cuenta con un modo continuo para el magnetómetro.

El protocolo I2C es una comunicación serial de 9 bits en donde cada dispositivo presente en la comunicación tiene su propia dirección, y a través de ésta es que se conoce a cual dispositivo va dirigida la información, o a cual se le solicita información.

Como se muestra en la figura 2.21, primero se envía la dirección de 7 bits del dispositivo con el cual se desea comunicar, luego se indica con el 8vo bit si se va a escribir o leer información (0 si el maestro envía información, 1 si el maestro solicita información), y finalmente el 9no bit que indica si el maestro reconoció o no la información; éste último también se utiliza para terminar la comunicación entre maestro-esclavo. Luego se envían 8 bits de información, más el bit de reconocimiento.

Cada bit es enviado con cada pulso en alto del reloj (SCL), la trama empieza cuando dicho reloj pasa de alto a bajo, y se detiene cuando ocurre lo contrario. De esta manera se mantiene la sincronía maestro y esclavo en el protocolo I2C.

Se concluye que el robot móvil 3pi es una plataforma robótica totalmente adecuada para los fines de este trabajo; explorar un laberinto desconocido de cualquier tamaño y configuración, enviar datos de dicha exploración a una PC, aprenderse el laberinto y ubicarse simultáneamente dentro de él.

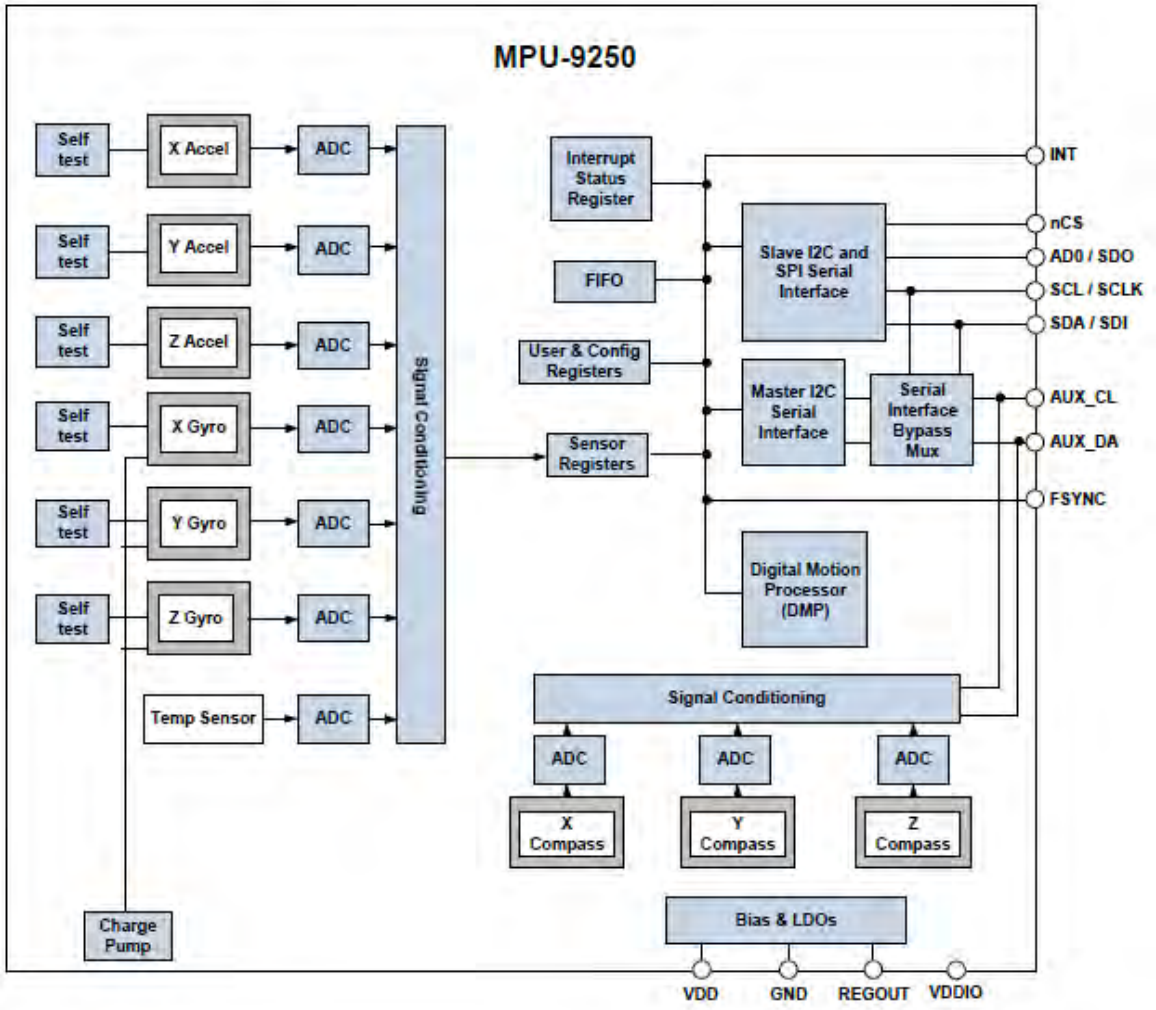

Figura 2.20 Diagrama general del MPU-9250 [10]

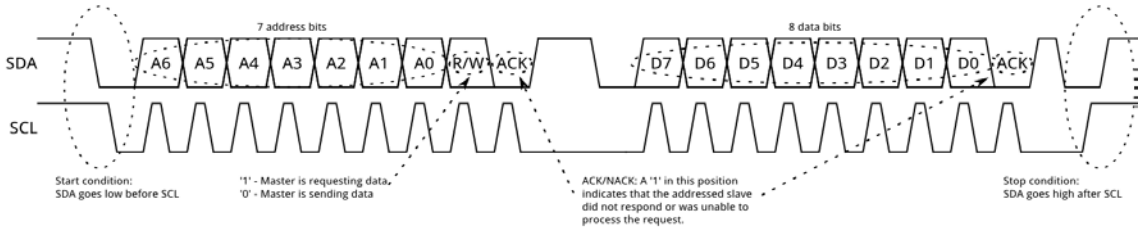

Figura 2.21 Señales del protocolo I2C [12]

CAPÍTULO III
RECONFIGURACIÓN DEL ROBOT 3PI PARA EXPLORACIÓN DEL LABERINTO

Como se explicó en el capítulo anterior el robot móvil 3pi será usado como plataforma robótica base para el desarrollo del robot final que se desea obtener. Bajo el marco de estas ideas, se debe realizar cierto trabajo de acondicionamiento previo; como lo es el diseño de una placa de expansión que permita agregar los módulos necesarios, así como también, pruebas de instrumentación y control.

3.1 DISEÑO DE LA PLACA DE EXPANSIÓN

Debido a la necesidad que se tiene de usar sensores de distancia HC-SR04, módulo WiFi ESP8266 y el módulo MPU-9250, es imperativo crear una placa que permita la conexión de estos elementos. De esta manera se decide utilizar el programa Proteus 8 como software de diseño del circuito impreso (PCB), cuyo resultado se muestra en las figuras 3.1 y 3.2.

Dicha placa se conecta al 3pi a través de cables, haciendo uso de los pines mostrados en la tabla 3.1, los cuales corresponden a los puertos de expansión que proporciona el robot 3pi.

Debido al alto consumo de corriente que presenta el módulo WiFi, se decidió que éste tuviese su propio sistema de alimentación independiente; no como los demás elementos que son energizados a partir de los 5 V que proporciona el sistema de alimentación del 3pi.

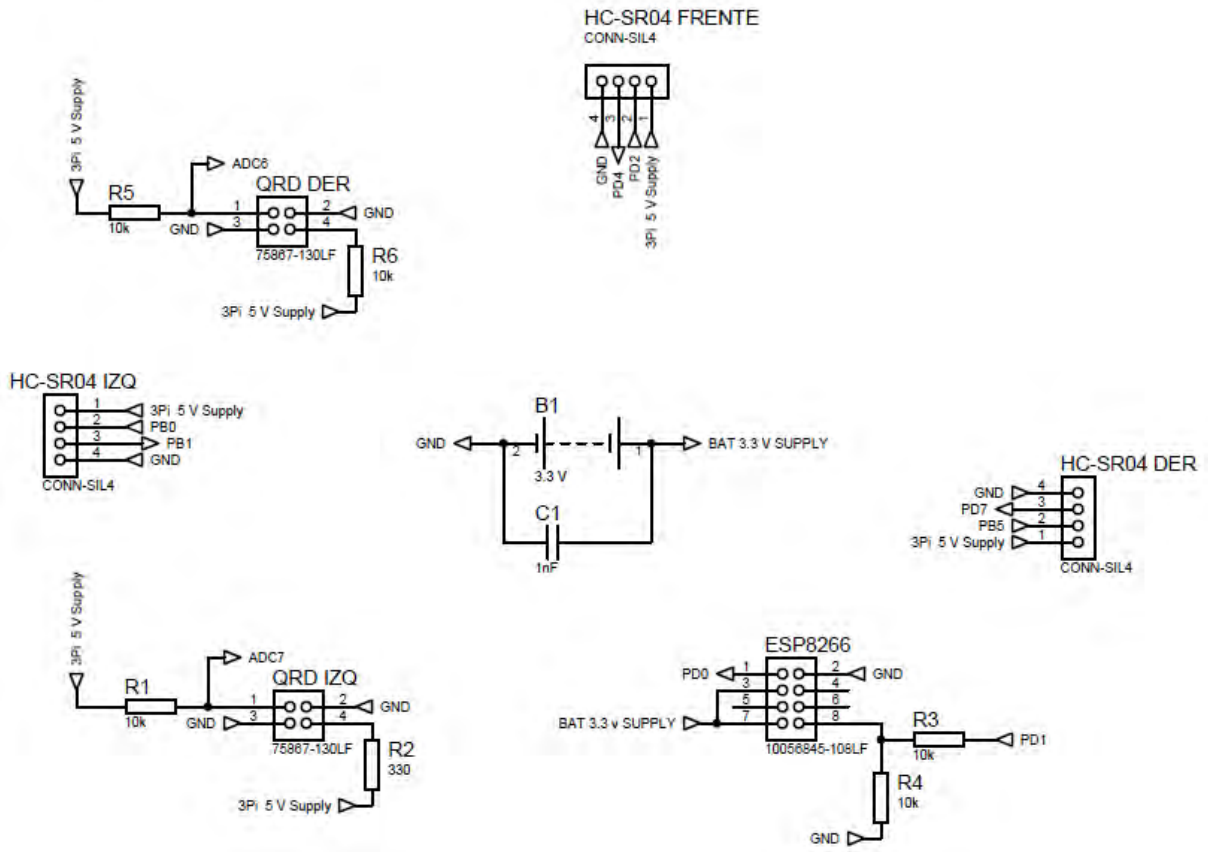

Figura 3.1 Esquema de conexión de la placa de expansión

En la figura 3.2 se puede apreciar el espacio central reservado para la batería que alimentará al ESP8266 y al MPU-9250, así como la distribución de los demás elementos. Se escogió una batería de ión-lithio de 3.8 V y 800 mA como sistema de alimentación del módulo WiFi. Experimentalmente se comprobó que la duración de la batería es de aproximadamente una hora al usar el MPU-9250 y el ESP8266 a la vez.

Inicialmente se consideró el posible diseño y uso de encoders que permitan calcular la distancia recorrida por parte del robot, y por esta razón la placa tiene reservado un espacio para colocar dos QRD1111. A medida que se fue desarrollando el proyecto se tomó la decisión de no hacer uso de estos, y calcular la distancia recorrida con el sensor de ultrasonido delantero.

Luego del proceso de quemado de la placa, se comprobó el correcto funcionamiento de la misma al verificar que no hubiese presencia de cortos circuitos o circuitos abiertos, así como la debida continuidad.

Luego de perforar la placa y soldar los componentes, se añadieron unos *ramplug* con la finalidad de servir como soporte entre el 3pi y la placa de expansión. En las figura 3.3, 3.4 y 3.5 se muestra el resultado final.

Tabla 3.1 Pines usados correspondientes al ATmega328P

Puerto	**Pin**	**Función**
B	0	Trigger HC-SR04 izquierda
B	1	Echo HC-SR04 izquierda
B	4	SDA
B	5	Trigger HC-SR04 derecha
C	5	SCL
D	0	Rx
D	1	Tx
D	2	Trigger HC-SR04 adelante
D	4	Echo HC-SR04 adelante
D	7	Echo HC-SR04 derecha

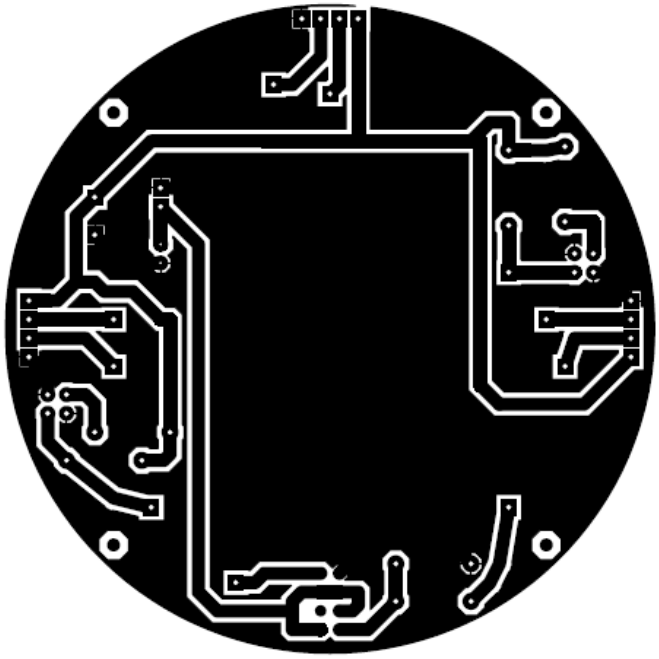

Figura 3.2 Diseño de la placa de expansión.

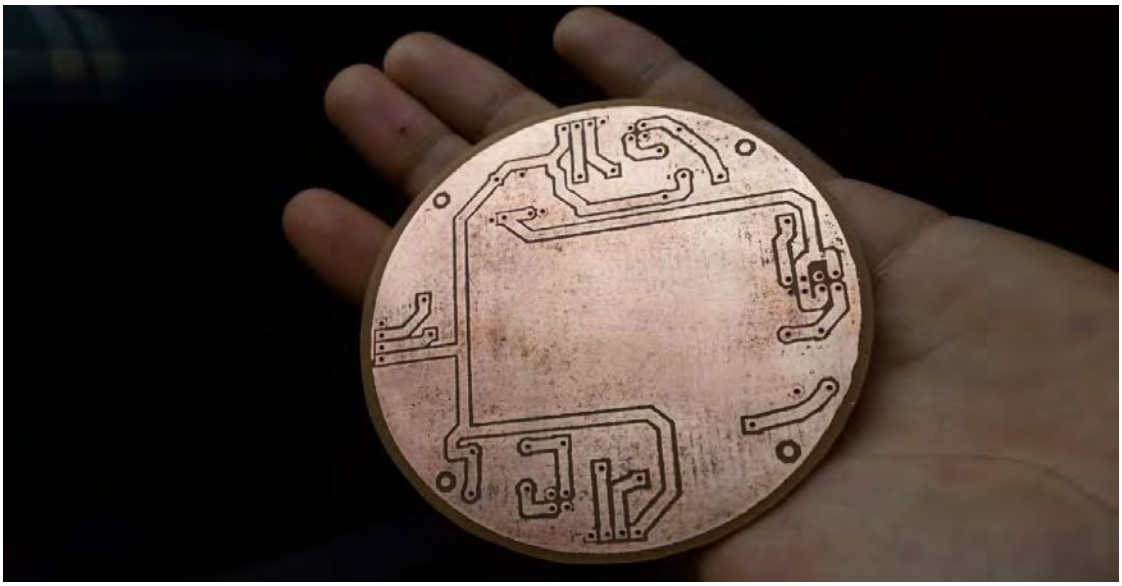

Figura 3.3 Vista inferior de la placa de expansión.

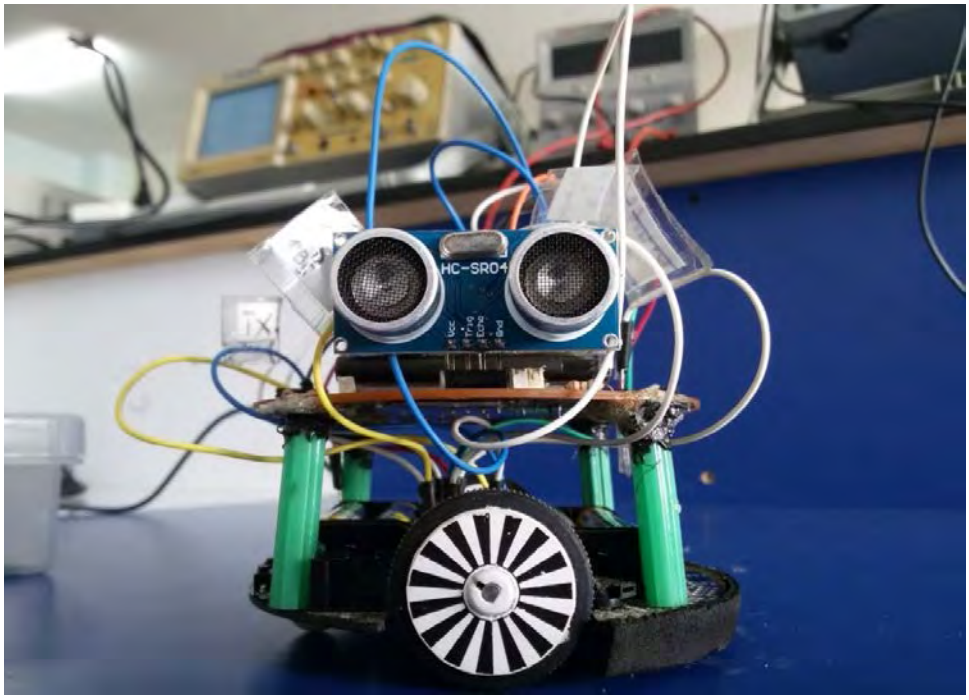

Figura 3.4 Vista lateral del robot.

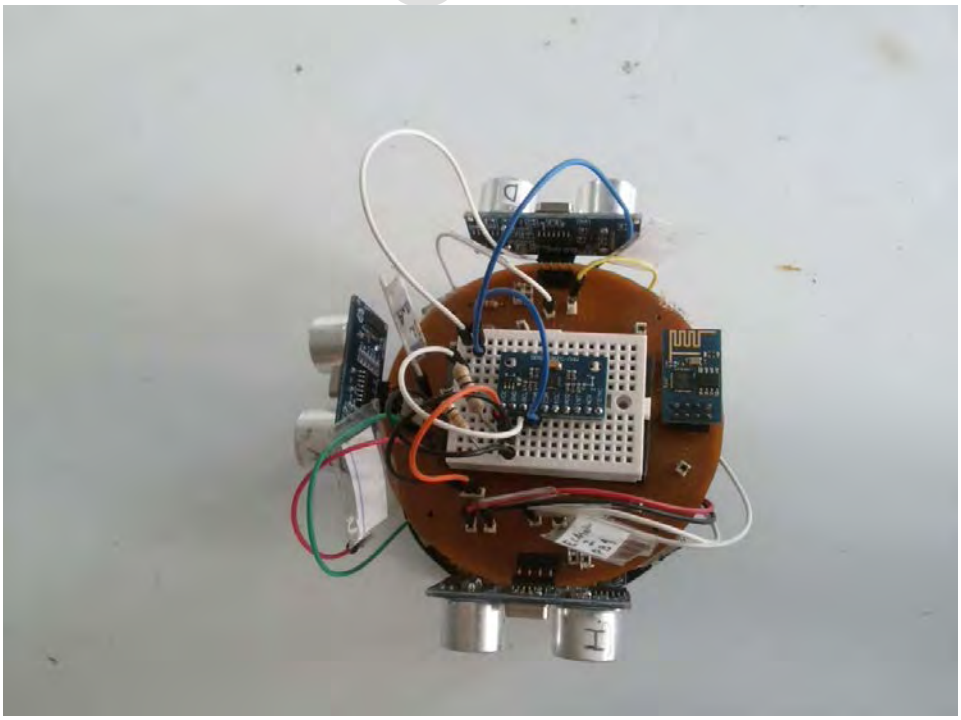

Figura 3.5 Vista superior del robot.

3.2 ESTABLECIMIENTO DE LA COMUNICACIÓN ENTRE ROBOT Y COMPUTADOR

Con la finalidad de mantener una comunicación entre el robot y el computador, se tomó la decisión de aplicar IoT, al interconectar el robot a una laptop, a través del WiFi, en la cual se montó un servidor apache y, de esta manera, hacer uso del módulo WiFi ESP8266 antes mencionado.

Debido a que las necesidades presentadas en este trabajo son muy particulares, las páginas web que ofrecen servicios para proyectos IoT no satisfacen los requerimientos necesarios para alcanzar los objetivos planteados.

Por esta razón se optó por montar el sistema de infraestructura de internet LAMP (Linux, Apache, MySQL, PHP) en un computador con sistema operativo Ubuntu 16.04 LTS. Esto hizo necesario el aprendizaje de desarrollo web, con el objetivo de formar una página web que fuese capaz de recibir y enviar datos, desde y hacia el ESP8266.

Inicialmente lo más importante es que el ESP8266 logre conectarse al WiFi y a través de la red de área local enviar los datos al servidor, en donde dicho servidor recibe y procesa esta información.

Bajo el marco de estas ideas, se planteó la página web observada en la figura 3.7, en donde se pueden visualizar los datos recibidos y procesados por la misma página. Esto se logra al proporcionar en el código del ESP8266 los datos necesarios para que el mismo se conecte al WiFi (SSID y clave de acceso), así como la dirección IP del computador en el cual corre el servidor.

Para lograr que al servidor le llegue la información, no sólo hace falta codificar en el ESP8266 lo antes mencionado, sino también la forma correcta de la solicitud HTTP.

Existen dos tipos de métodos para solicitudes HTTP comúnmente utilizados en el envío de datos entre cliente y servidor, o incluso entre páginas web: GET y POST [13].

3.2.1 Solicitudes GET

Solicita datos de una fuente en específico. Algunas de sus características son:

- Pueden ser almacenadas en la caché.
- Se mantienen en el historial del navegador.
- Pueden ser marcadas.
- No deben ser utilizadas para manejar información sensible.
- Tienen límite de tamaño.
- No se deben utilizar sólo para solicitar datos.
- Los datos son visibles en la URL.

3.2.2 Solicitudes POST

Entrega datos a ser procesados por una fuente en específico. Se nombran algunas de sus características:

- No pueden ser almacenadas en caché
- No se mantienen en el historial del navegador
- No pueden ser marcadas
- No tienen restricción de tamaño
- Los datos no son visibles en la URL

Las solicitudes HTTP mantienen una forma general [14], la cual consiste en:

- Una línea de inicio terminada en un retorno de línea.
- Opcionalmente una o más líneas de cabecera terminadas en retorno de línea.
- Línea en blanco (retorno de línea).
- Opcionalmente un mensaje.

La línea de inicio contiene 3 elementos:

- El método utilizado para la solicitud.
- El objeto sobre el cual se opera
- El nombre de protocolo y su versión.

Las líneas de cabecera son de la forma “PALABRA_CLAVE=VALOR” y son utilizadas para definir distintos parámetros de la solicitud. En el marco de estas ideas, la trama enviada por el ESP8266 presenta la forma que se muestra en la figura 3.6.

Es importante destacar que el código presente en el ESP8266 está en C++, y editado utilizando el ambiente de desarrollo Arduino IDE, junto a las librerías que ofrece para manejar este módulo WiFi.

```
String url = "GET /index.php?item=" + String(data) + " HTTP/1.1";
client.println(url);
client.println("Host: " + (String)host);
client.println("Connection: close");
client.println();
client.println();
```

Figura 3.6 Envío de trama al servidor por parte del ESP8266.

Siendo el nombre de la página (ver imagen 3.7) “index.php”, el cual es el nombre por defecto de la página principal en el servidor web y, por ende, la que el navegador abre por defecto; es decir, no hace falta colocar “host/index.php”. Con sólo la IP del servidor es suficiente.

Debido a la facilidad para encontrar ejemplos de comunicación del ESP8266 utilizando el método GET, y la ausencia de datos sensibles en la comunicación, se decidió hacer uso del método GET.

Una vez que se tiene al ESP8266 conectado al WiFi, se conoce la dirección IP del servidor y el método a través del cual se enviará la información, se prosiguió a comprobar el correcto funcionamiento del envío de datos por parte del ESP8266, así como el procesamiento de dicha información por parte del servidor.

Del lado del servidor los datos son recibidos haciendo uso del conocimiento previo de que estos le llegan usando el método GET. Para esto se utiliza el lenguaje de programación PHP, el cual es un lenguaje que se ejecuta sólo del lado del servidor. Dichos datos son luego almacenados en una base de datos en formato texto, y organizados de forma matricial en PHP, para luego ser mostrados en la página web (ver figura 3.7).

Se cuenta con 6 bases de datos: 2 para los datos de las coordenadas, de las cuales una se encarga de recibir los datos de distancia recorrida entre nodos y la orientación, mientras la otra contiene las coordenadas calculadas por el servidor; 2 para los datos de los nodos, de las cuales una recibe los datos enviados por el robot, correspondientes a los nodos, mientras la otra contiene los mismos datos pero organizados según el nombre del nodo; 1 que mantiene el registro de los nodos por los cuales se ha transitado y, por ende contiene la última posición; 1 para registrar las distancias obtenidas al entrar en cada nodo.

En la figura 3.7 se muestra a la izquierda los datos de desplazamiento enviados por el robot (ORIENTACION,DISTANCIA_RECORRIDA), y a la izquierda las coordenadas registradas (X,Y). El nodo raíz siempre tiene coordenadas 0,0 y las demás coordenadas son calculadas a partir de esta. El orden de los datos es de izquierda a derecha.

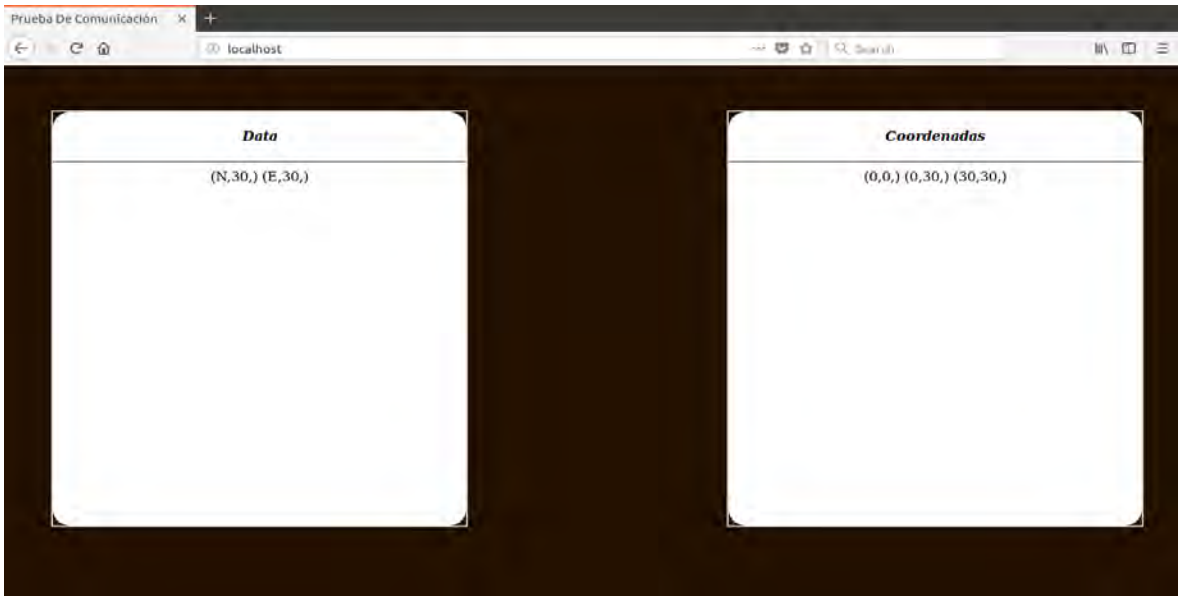

Figura 3.7 Presentación de los datos procesados por parte del servidor

El ESP8266 se programó utilizando el entorno Arduino IDE. Como se mencionó anteriormente el servidor está programado en PHP. El código del servidor contiene también código en Javascript, el cual es ejecutado en el lado del cliente.

A pesar de que en los archivos que definen la página web que corre en el servidor se encuentra código PHP mezclado con Javascript, estos son ejecutados en momentos distintos. PHP se ejecuta sólo en el momento en el que el servidor recibe una petición, es decir que se ejecuta una sola vez por petición; en cambio, el código en Javascript se ejecuta el resto del tiempo que el cliente tenga abierto el navegador, y se ejecuta sólo en dicho navegador.

3.3 CARACTERIZACIÓN DE LOS MODULOS HC-SR04

Con la finalidad de proporcionar una gráfica que permita la clara identificación del rendimiento del módulo de ultrasonido HC-SR04, y a la vez comprobar que se está controlando el módulo para obtener de manera correcta la medición de distancia, es necesario realizar lo que se conoce como la caracterización de la instrumentación.

Dicha caracterización consiste en obtener los datos de medición del instrumento, compararlos con una referencia confiable, y de esta manera conocer el error que presenta el instrumento.

En primer lugar, se desarrollaron las funciones que controlan a cada uno de los 3 módulos HC-SR04. Como ya se explicó en la sección 2.6, el módulo espera una señal que le indica cuando debe disparar las ondas de ultrasonido, a la vez que éste coloca en alto su pin echo mientras la onda de ultrasonido va, rebota y regresa al módulo, momento en el cual el pin echo regresa a un nivel bajo.

La problemática se encuentra en que realmente el módulo HC-SR04 no se activa con un solo pulso en alto de duración 10 uS. En realidad se requiere de un tren de pulsos de más de 10uS cada uno. La cantidad y duración de cada pulso varía entre cada módulo, debido a pequeñas variaciones entre ellos.

Se prosiguió a encontrar de manera experimental la combinación adecuada de número de pulsos y duración de cada uno de estos para cada uno de los módulos utilizados. Se utilizó una regla para conocer la distancia que había entre el módulo y el objeto, y así tener una referencia confiable del resultado que debería obtenerse de cada módulo.

Al obtener una combinación que cumpliera con la medida correcta, la distancia entre módulo y obstáculo se varió con la finalidad de comprobar que éste daba la medición correcta de nuevo. En la tabla 3.2 se pueden observar los resultados.

Tabla 3.2 Configuración de módulos HC-SR04

Módulo HC-SR04	**Nª de pulsos**	**Duración de cada pulso (uS)**	**Distancia entre pulsos (uS)**
Adelante	15	20	10
Derecha	15	15	15
Izquierda	17	14	14

La hoja de datos del HC-SR04 indica que el rango de medición es de (2 – 400) cm, no obstante la caracterización se realizó de (0 – 20) cm. Los resultados se muestran en las figuras 3.8, 3.9, 3.10.

Se puede apreciar de las figuras anteriores que la resolución de los 3 módulos HC-SR04 es de 1 cm, y el mayor error presentado es del 2%, por debajo o por encima de la recta ideal. Tomando en consideración que la resolución es de 1 cm se considera que el error en los módulos de ultrasonido es bastante bajo.

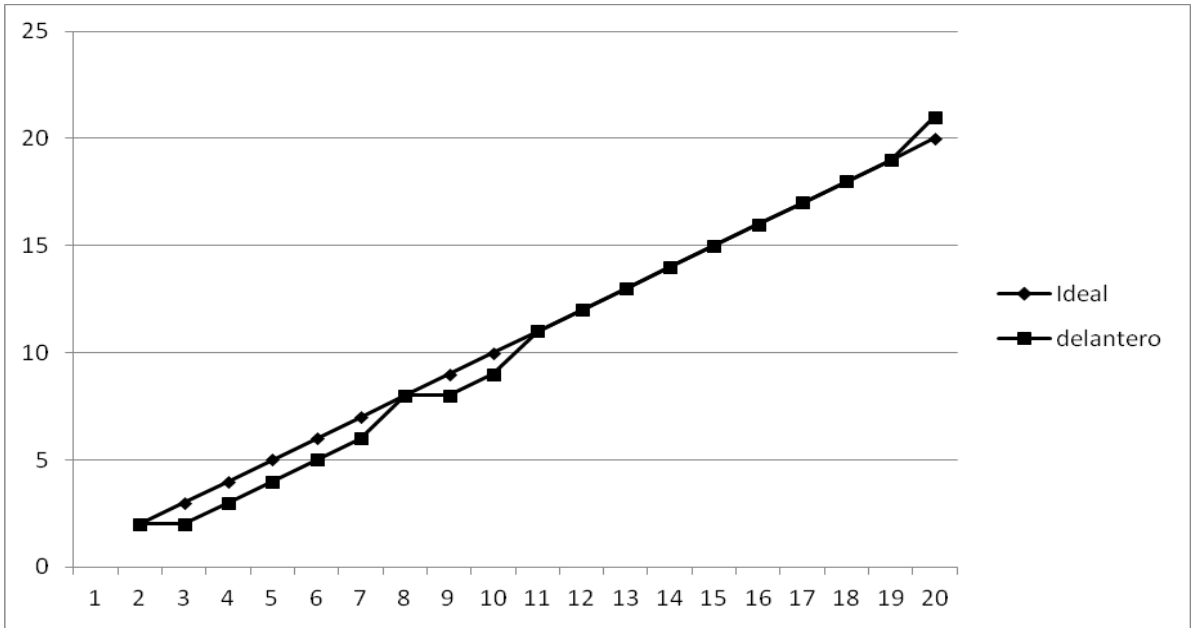

Figura 3.8 Caracterización del módulo HC-SR04 delantero.

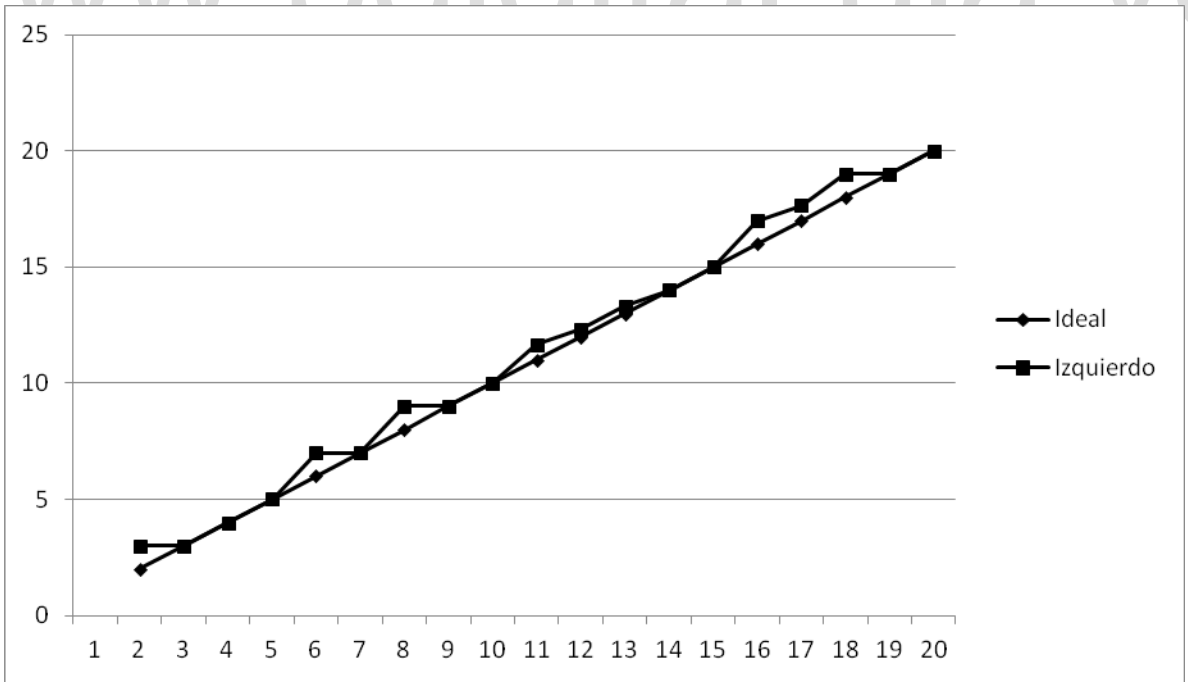

Figura 3.9 Caracterización del HC-SR04 izquierdo.

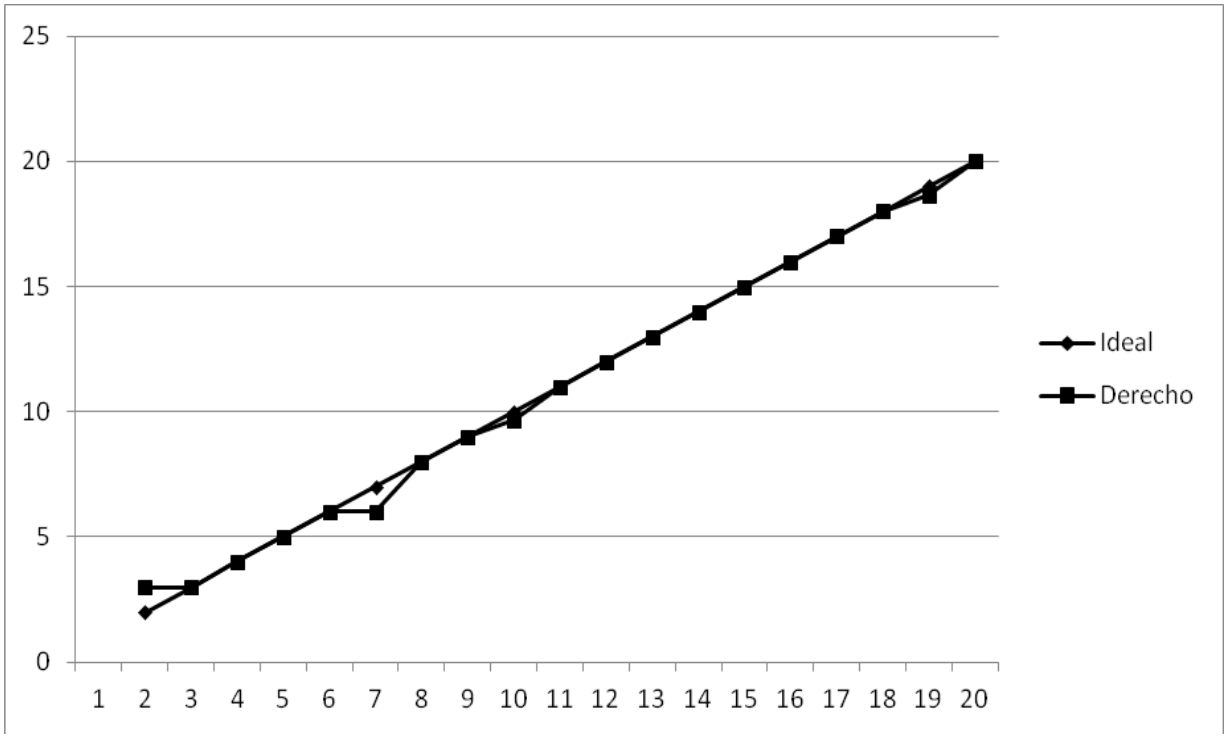

Figura 3.10 Caracterización del HC-SR04 derecho.

3.4 CONTROL PROPORCIONAL INTEGRAL DERIVATIVO

Debido a que el control de los motores no es ideal no se puede contar sólo con estos para asegurar que el robot no choque con las paredes laterales. Es evidente entonces que se hace necesario algún tipo de control que mantenga al robot centrado en la pista.

Ante la situación planteada se decidió aplicar un control proporcional, derivativo, integral (PID), con la finalidad de mantener una distancia fija de 10 cm a la izquierda.

En primer lugar se intentó aplicar el método Ziegler y Nichols, no obstante éste no pudo ser aplicado debido al ancho de la pista, el cual no era lo suficientemente ancho como para permitir la debida oscilación del robot sin antes chocar. En la figura 3.11 se puede observar al robot dentro de la recta utilizada para calibrar el controlador PID.

Figura 3.11 Tramo recto utilizado en la calibración del controlador PID.

Como consecuencia de esto último, se optó por ajustar los valores de las constantes de manera experimental, lo cual requirió de innumerables pruebas hasta alcanzar la mejor estabilidad posible.

Uno de los problemas más grandes, aparte del ancho de la pista, fueron las uniones entre las tablas que forman el laberinto. Dichas uniones alteran en gran medida la onda de ultrasonido, resultando en un error sumamente grande al momento de medir las distancias. Como

consecuencia de esto la estabilidad del controlador PID se veía bastante afectada y, además, el robot lo confundía con un cruce hacia la izquierda o la derecha, dependiendo del módulo que lo detectara primero.

Debido a este error de medición de distancia a causa de las uniones entre tablas del laberinto, se decidió aplicar un filtro de mediana al código del robot, con la finalidad de filtrar aquellos valores que se salieran de la tendencia general de las mediciones. El funcionamiento de este filtro consiste en almacenar en un buffer las últimas 7 mediciones obtenidas por los módulos HC-SR04, organizarlas de menor a mayor y retornar el valor de la posición central (el valor número 4); es decir, retorna la mediana de las últimas 7 mediciones de distancia obtenidas en cada HC-SR04. Es importante que el tamaño del buffer sea un número impar para que el filtro funcione correctamente.

A medida que el tamaño del buffer es mayor, el filtro suaviza más la respuesta y tiene mayor capacidad de filtrar valores indeseados, pero a la vez resulta en una respuesta más lenta, debido al tiempo de respuesta del filtro.

3.5 ORIENTACIÓN DEL ROBOT

A pesar del uso de un controlador PID que mantiene una distancia de 10 cm hacia la izquierda del robot, éste presentaba problemas al momento de realizar los cruces hacia la izquierda o derecha, así como los retornos, debido a que el robot estaba en constante cambio de orientación al ir corrigiendo su trayectoria según la distancia hacia su izquierda, por ende, ocasionalmente al llegar a un cruce o calle sin salida, el robot no se encontraba viendo hacia adelante, si no con una pequeña inclinación dependiendo de su última tendencia de velocidad de motores (la cual está gobernada por el controlador PID).

Esto significa que si los cruces a la izquierda, derecha o de retorno se codifican de manera fija a 90º, -90º o 180º, respectivamente, al estar el robot con cierta desviación hacia la izquierda o derecha causa que el giro no se haga realmente de manera precisa, resultando en una entrada a un nuevo tramo, o retorno de tramo, con un error angular de orientación que

puede ser lo suficientemente grande como para que el robot pierda el control y choque, así como se muestra en la figura 3.12

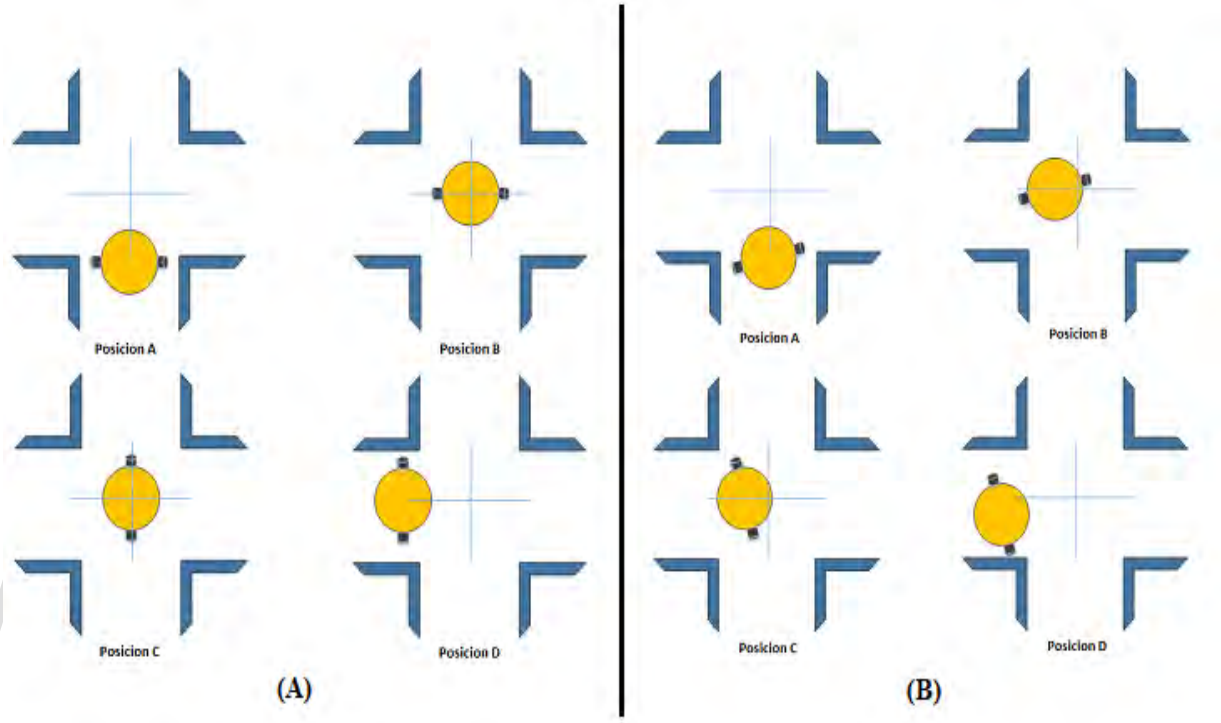

Figura 3.12 (A) Cruce correcto (B) Cruce incorrecto.

El robot al detectar que finaliza la pared detecta que se encuentra en una intersección, se dejan activos los motores para pasar de la posición A hacia la posición B, así como también de la C a la D. Esto se debe a que si el robot realiza el cruce en la posición A, chocaría con la esquina del laberinto. En el caso de la posición C, si se continúa con el algoritmo de exploración el robot detectaría inmediatamente que ya tiene otro cruce, confundiendo de esta manera el tramo del que acaba de salir con un cruce.

Es oportuno resaltar el hecho de que el robot al no estar bien alineado resulta en una medición no perpendicular a la pared del laberinto, causando que la medición obtenida sea mayor que la real. Esto puede causar que el robot esté a menos de 10 cm, en realidad mide una distancia oblicua hacia la pared resultando en una distancia mayor a 10 cm.

Tomando en consideración lo antes planteado, se decidió hacer uso del módulo MPU-9250, específicamente el magnetómetro incluido en él, ya que las lecturas del giroscopio y el acelerómetro no son de interés para la corrección angular en el plano XY.

3.5.1 Caracterización del MPU-9250

Al igual que con el módulo de ultrasonido HC-SR04, es muy importante realizar la caracterización del MPU-9250, en especial porque éste es muy susceptible a ruido, presenta mayor complejidad al momento de ser usado y debe ser calibrado.

Todas estas pruebas se realizaron utilizando un Arduino UNO, haciendo pivotear el MPU-9250 sobre un mismo eje haciendo uso de una plataforma que fue movido de forma manual. Se hizo de esta manera debido a la facilidad de uso del terminal del Arduino IDE.

Con esto en mente lo primero que se debe hacer es obtener los datos en crudo del magnetómetro, tal y como se muestran en la figura 3.13. En esta figura se logra apreciar claramente el desplazamiento que presenta la respuesta del magnetómetro al no estar centrada en el origen del sistema de coordenadas. Así mismo se puede ver en las figuras 3.14 y 3.15 las respuestas aisladas en X y Y, respectivamente. Es claro que en el eje X el offset es mucho mayor que en el eje Y. No obstante, el eje Y presenta también una pequeña desviación.

Este error se debe al campo magnético constante del circuito del propio MPU-9250, que es detectado por el magnetómetro AK8963 y, por ende, resulta agregado a todas las mediciones.

Para cancelar dicho ruido se prosiguió a calcular el promedio de los valores en ambos ejes, y luego se anuló el respectivo promedio de cada eje. Para el caso del eje X la corrección fue de 180, y en el eje Y fue de -22.

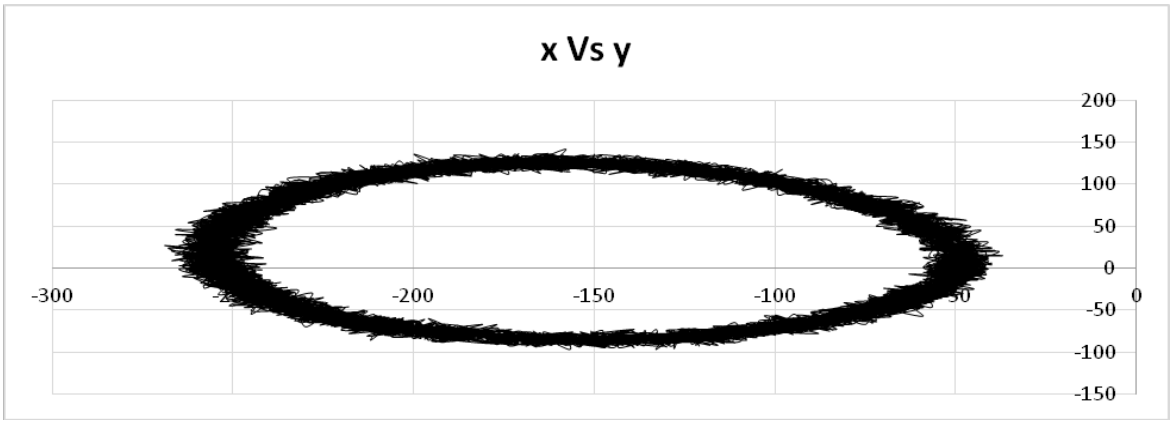

Figura 3.13 Medición del magnetómetro en el plano XY.

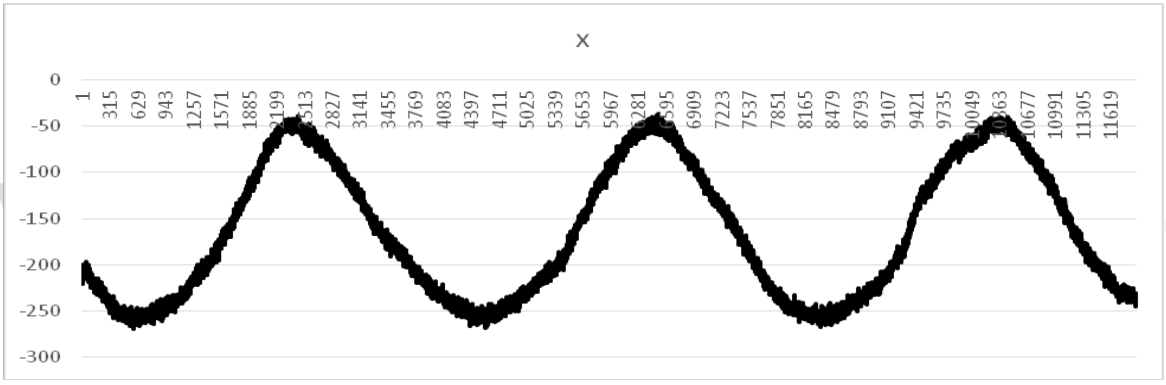

Figura 3.14 Medición del magnetómetro en el eje X.

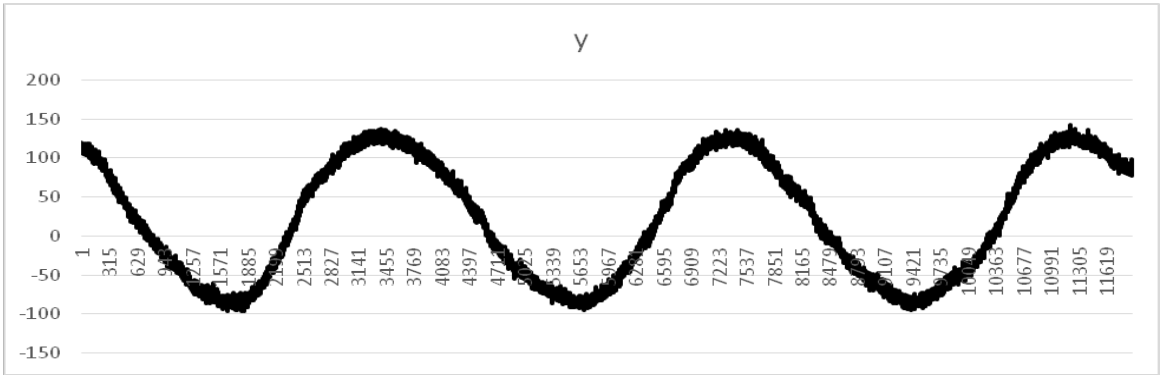

Figura 3.15 Medición del magnetómetro en el eje Y.

Se hace interesante resaltar el comportamiento cíclico de las lecturas en cada eje con cada vuelta que se le dio al MPU-9250, demostrando el correcto funcionamiento del magnetómetro. Al ser corregidas las lecturas de cada eje, se registraron las nuevas mediciones, las cuales pueden ser observadas en las figuras 3.16, 3.17 y 3.18.

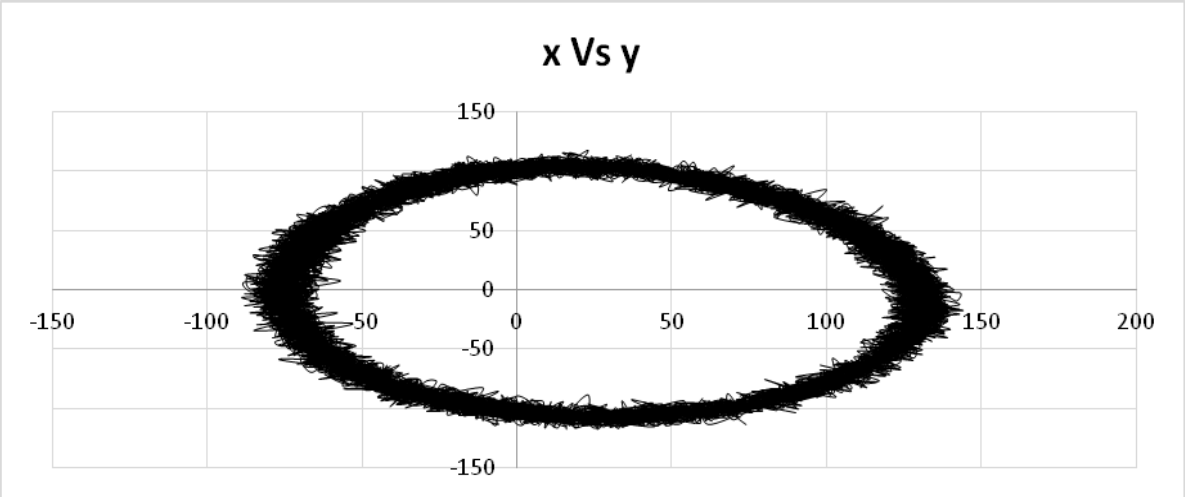

Figura 3.16 Medición calibrada del magnetómetro en el plano XY.

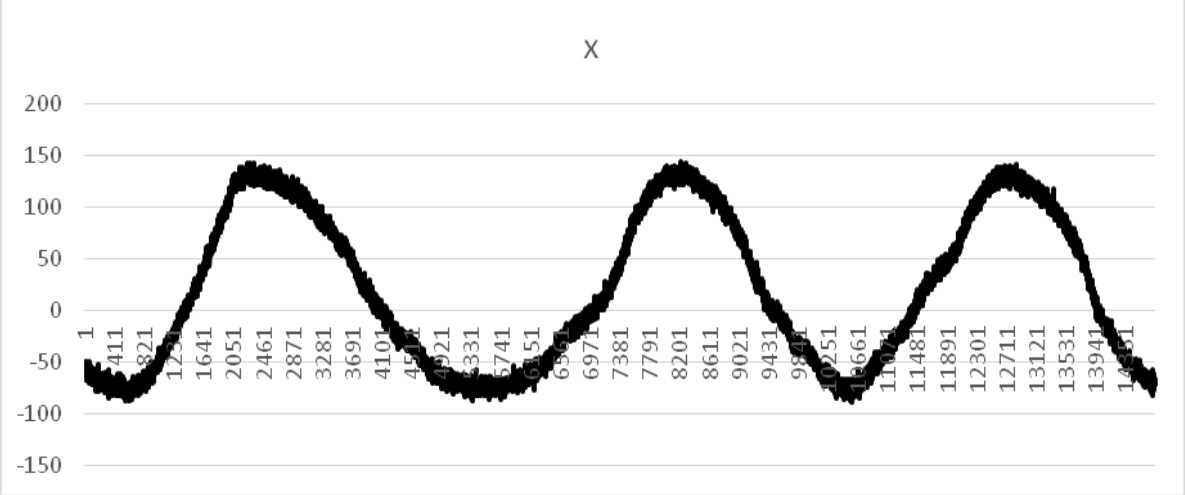

Figura 3.17 Medición calibrada del magnetómetro en el eje X.

Se observa claramente que luego de haber sido calibrado el AK8963 el comportamiento cíclico ocurre alrededor del origen. Esto indica que la calibración se realizó de manera correcta.

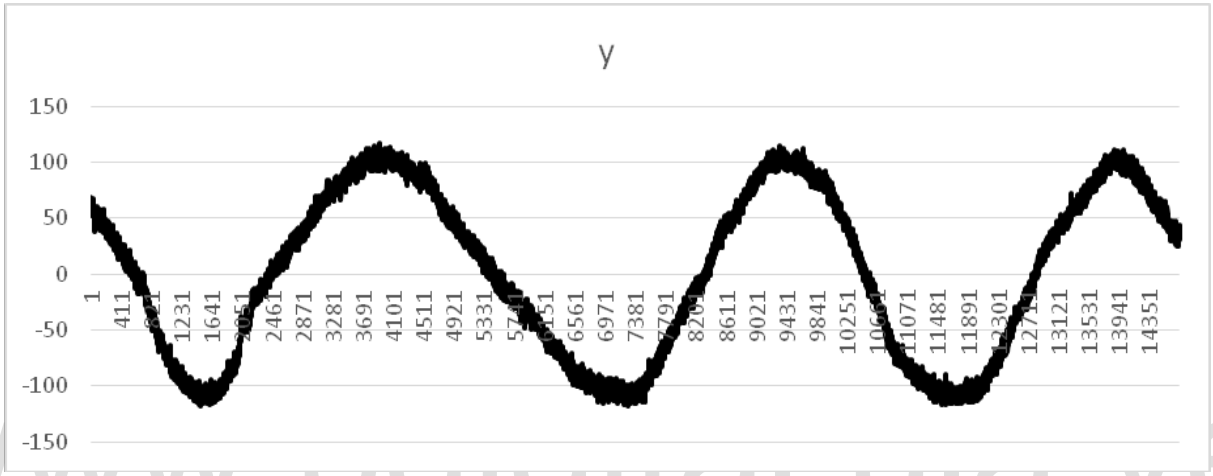

Figura 3.18 Medición calibrada del magnetómetro en el eje Y.

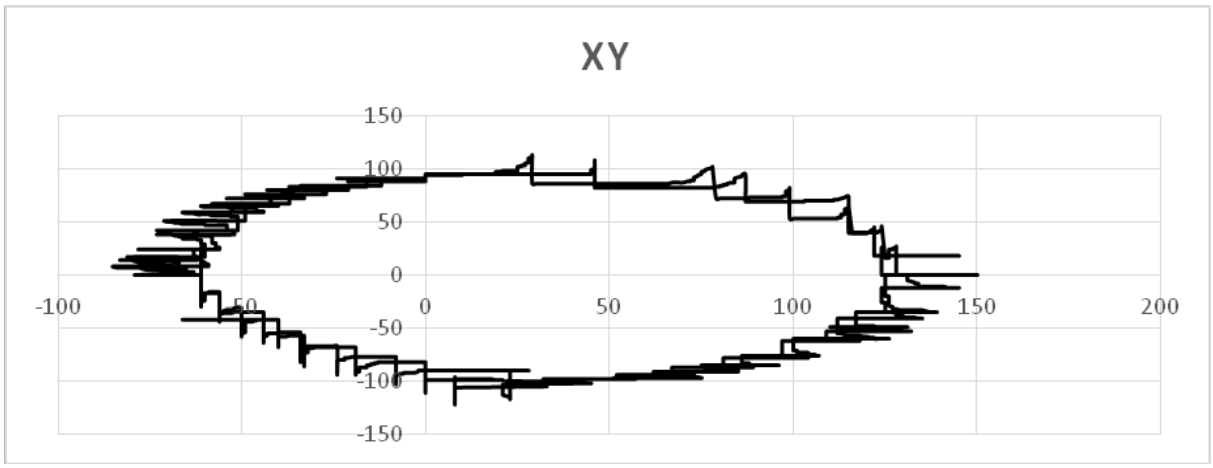

Figura 3.19 Medición del magnetómetro con filtro paso bajo en el plano XY.

Lo siguiente fue disminuir el ruido en la medición, para lo cual se utilizó un filtro paso bajo con la finalidad de suavizar los cambios bruscos en la lectura del magnetómetro. Dicho filtro fue aplicado en el código que gobierna todo lo referente al MPU-9250. El resultado de las lecturas del magnetómetro aplicando el filtro paso bajo se observan en las figuras 3.19, 3.20 y 3.21.

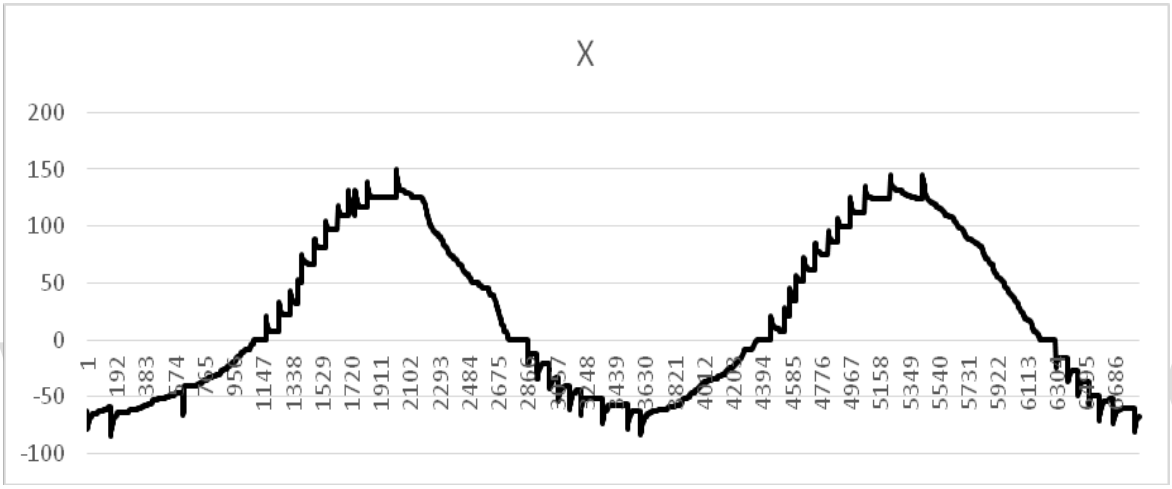

Figura 3.20 Medición del magnetómetro con filtro paso bajo en el eje X.

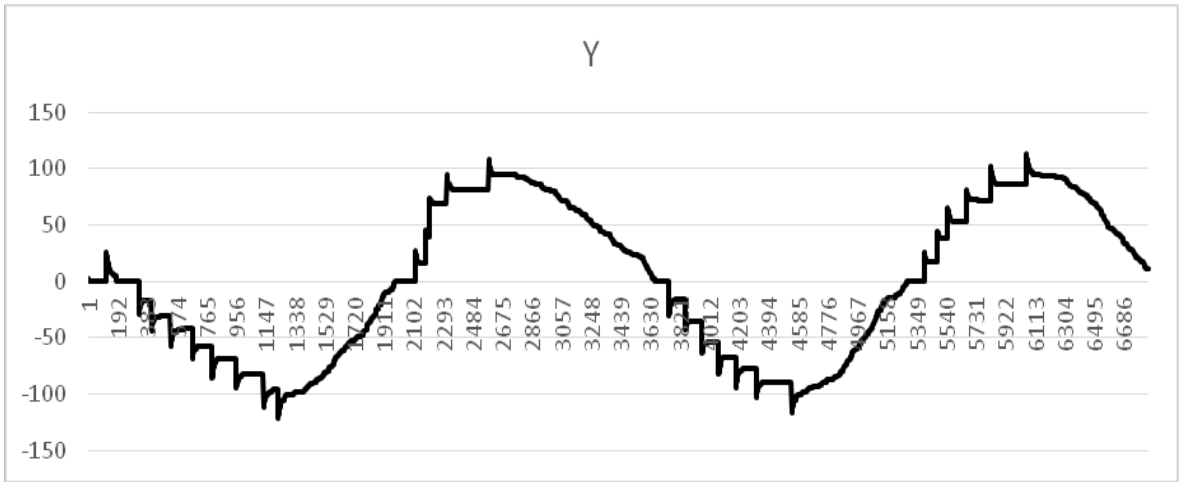

Figura 3.21 Medición del magnetómetro con filtro paso bajo en el eje Y.

Se hace clara la diferencia entre los resultados de las lecturas del magnetómetro con filtro y sin filtro, pero aún cuando el ruido se ha disminuido en gran medida, se sigue observando ciertas perturbaciones en la tendencia general de los valores.

Por esta razón se decidió pasar estas mediciones por un segundo filtro: el anteriormente mencionado filtro de mediana, utilizado en los módulos de ultrasonido HC-SR04. El resultado se muestra en las figuras 3.22, 3.23 y 3.24.

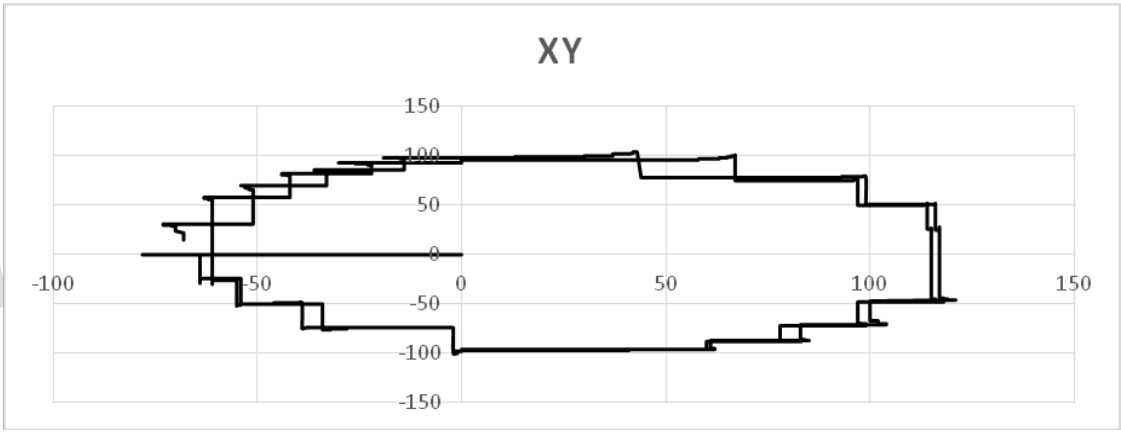

Figura 3.22 Medición del magnetómetro con filtro de mediana en el plano XY.

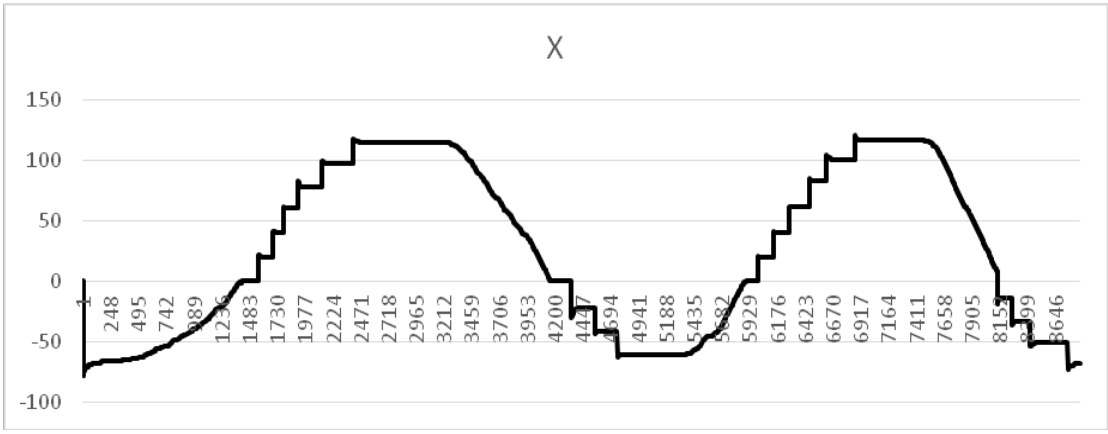

Figura 3.23 Medición del magnetómetro con filtro de mediana en el plano X.

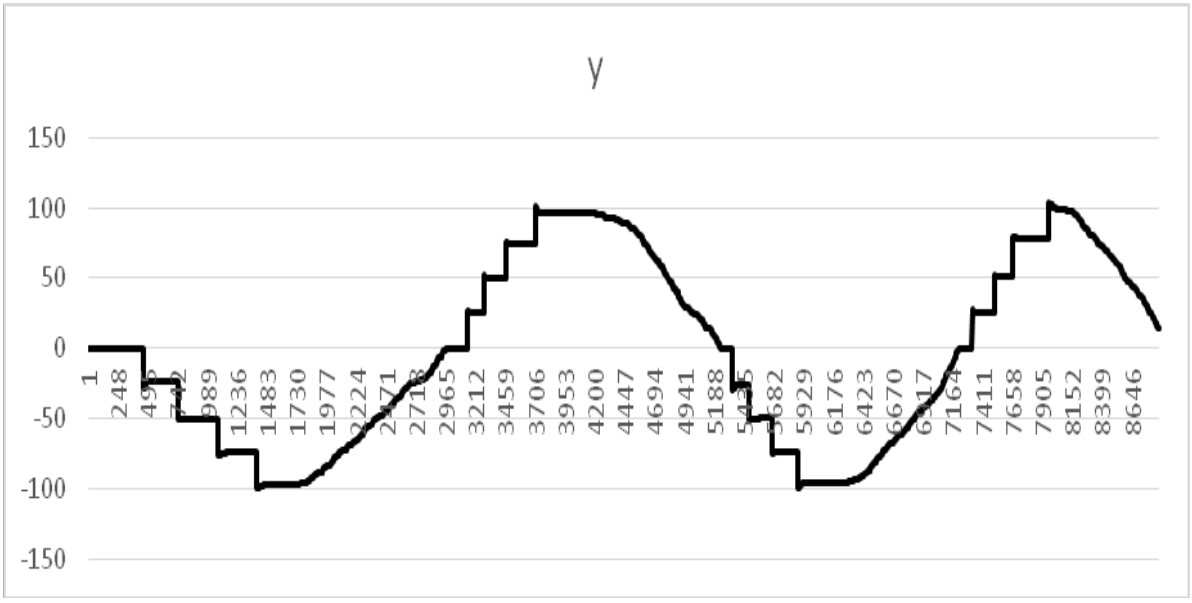

Figura 3.24 Medición del magnetómetro con filtro de mediana en el plano Y.

Aún cuando continúa la presencia de ruido en la lectura, se observa una disminución del mismo. Lo último que se hizo fue pasar estas lecturas a radianes, al calcular el arco tangente de la relación que existe entre X y Y. De esta manera se obtiene cada lectura con un valor entre 0 y 2π.

Es importante resaltar el hecho de que la respuesta en el plano XY es de forma ovalada, debido a que el campo magnético se ve afectado por los materiales que conforman los componentes del propio MPU-9250, causando una alteración en la magnitud del campo magnético detectado proveniente de la tierra.

No obstante, como ya se mencionó anteriormente, tanto el circuito de la placa de expansión, como el del 3pi, los módulos de ultrasonido, cables y motores alteran el campo magnético aledaño al MPU-9250. Incluso la presencia de celulares y otros equipos afecta la medición obtenida del MPU-9250.

3.5.2 Corrección de orientación

La manera en que se alcanzó este objetivo fue al iniciar el programa del robot registrar la medición obtenida del magnetómetro al apuntar en las 4 direcciones virtuales (Norte, Este, Sur, Oeste), siendo el norte la dirección en la cual se encuentra el robot al encender y el resto de las orientaciones se consiguen al girarlo 90º sobre su propio eje hasta registrar estos 4 valores de interés.

Dichos giros de 90º se lograron al probar distintas combinaciones de velocidad de motores junto a retardos, hasta observar que los giros eran lo suficientemente precisos; es decir, el robot calibra el valor de cada orientación de manera automática y autónoma al encender.

Más adelante, al encontrar un cruce el robot gira en la dirección adecuada dependiendo de la situación, hasta encontrar el valor que tiene registrado hacia esa orientación a la cual debe ir. Este proceso lo realiza antes de entrar en un nodo (pasar de la posición A hacia la B), así como también cuando se sale de un nodo (pasar de la posición C a la D) (ver figura 3.12), y al entrar a un nuevo tramo; lo que quiere decir que no sólo se aplica este procedimiento al cruzar, sino que también se corrige la orientación que lleva actualmente.

De esta manera el robot realiza los giros con una exactitud considerable sin importar el efecto que tiene el controlador PID en la orientación. No obstante, debido a la sensibilidad del magnetómetro éste se ve afectado por el ruido magnético que se encuentra a su alrededor.

Dicho ruido se debe tanto al circuito del 3pi, como al circuito de la placa de expansión, cables, y también a los objetos cercanos.

Por lo tanto, a pesar de que la problemática antes expuesta se vio reducida, este giro sigue presentando un error que no sucede todo el tiempo, causando ocasionalmente que el robot choque y no sea capaz de continuar con su exploración al 100%.

CAPÍTULO IV
PLANTEAMIENTO DE SOLUCIÓN

Al tener lista la placa de expansión, la correcta obtención de mediciones provenientes de la instrumentación, así como su caracterización, y el uso de dicha instrumentación para aplicar los métodos que permiten al robot moverse y ubicarse en el laberinto. Se prosigue a establecer la base datos, algoritmos de exploración y demás planteamientos que permitan solventar la problemática planteada.

4.1 ALGORITMO DE EXPLORACIÓN

En primer lugar es importante establecer el algoritmo a través del cual se explorará el laberinto, así como sus ventajas y desventajas. En este sentido se decide utilizar el algoritmo de exploración general de laberintos: la regla de la mano izquierda (9).

Dicha regla dicta que siempre que se mantenga una mano pegada a la pared, se puede salir de casi cualquier laberinto; se ha decidido utilizar la mano izquierda para la exploración.

En el caso del robot esto significa que los cruces hacia la izquierda tiene la prioridad, en segundo lugar está la decisión de seguir hacia adelante y por último girar a la derecha. Para ilustrar mejor este concepto se puede observar el laberinto de la figura 4.1.

Como se muestra en la figura 4.1, el robot, comenzando desde la celda amarilla logra conseguir la salida del laberinto de manera sencilla al aplicar la regla de la mano izquierda a lo largo del trayecto mostrado.

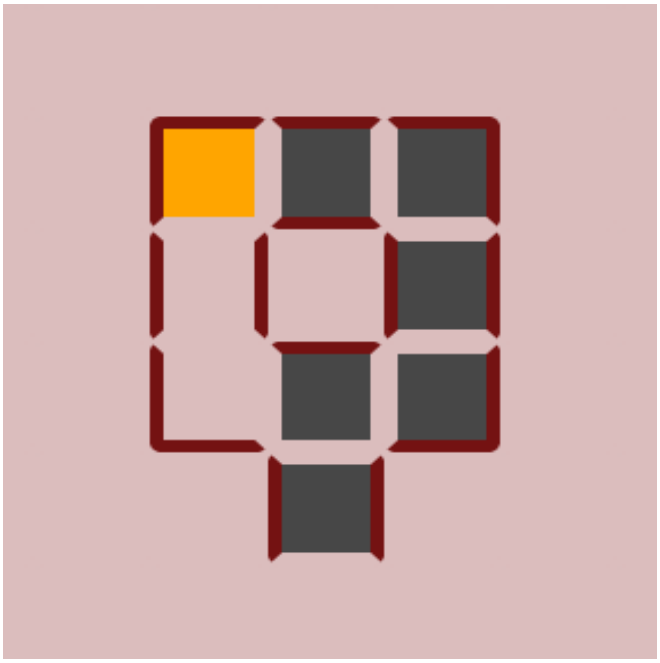

Figura 4.1 Solución de un laberinto aplicando regla de la mano izquierda

No obstante, como se mencionó anteriormente, la regla de la mano izquierda puede resolver casi cualquier laberinto, pero no todos. El problema se da cuando el laberinto tiene paredes independientes, los cuales crean circuitos cerrados o lazos, así como el laberinto de la figura 4.1, pero depende del sentido y posición del robot en el laberinto.

Aplicando un pequeño cambio en la posición del robot se puede apreciar como el mismo algoritmo falla, tal y como se muestra en la figura 4.2; apuntando inicialmente hacia la zona superior de dicha imagen.

Tal y como muestra en la figura 4.2, el robot se queda de manera infinita dando vueltas alrededor de la pared independiente buscando la salida sin llegar a conseguirla, debido a que cuando pasa cerca de ella, esta le queda hacia su derecha, pero según la regla de la mano izquierda la decisión de ir hacia adelante tiene prioridad sobre los giros a la derecha.

Se puede pensar que cambiando la prioridad de las decisiones se puede arreglar este problema, y para este caso en específico es así. De hecho, la regla de la mano derecha es aún

más común que la de la izquierda, pero de nuevo, dependiendo de la posición y orientación del robot esta también fallará. Por esta razón se decidió apegarse a la regla de la mano izquierda, y solventar este problema creando un algoritmo adicional.

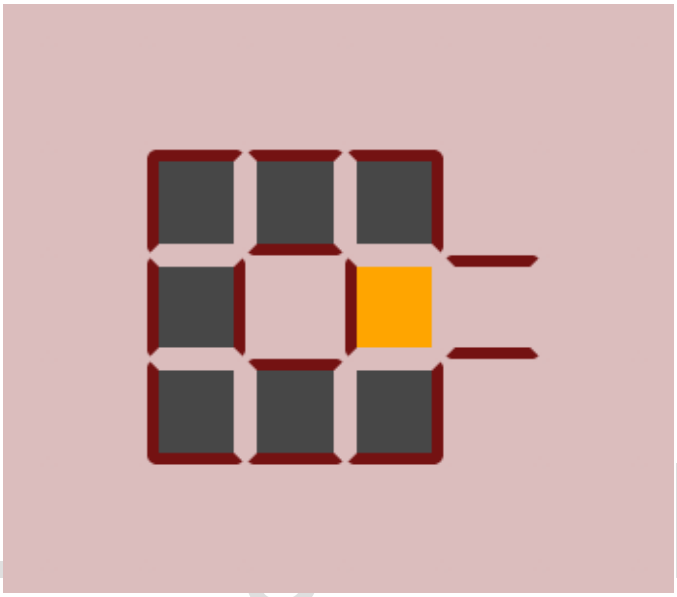

Figura 4.2 Falla en la búsqueda de una salida aplicando regla de la mano izquierda.

La presencia de paredes independientes y circuitos cerrados es lo que hace que un laberinto pase de ser simple a un laberinto complejo de resolver. Más adelante se plantea la solución y sus resultados, pero primero es importante establecer la estructura de la base de datos.

4.2 BASE DE DATOS

Tomando en cuenta que no se conoce de antemano el tamaño del laberinto, así como su forma, posiciones relativas, y que debido a la necesidad de calcular rutas se deben utilizar algoritmos de búsqueda de inteligencia artificial (los cuales serán explicados más adelante), se llegó a la conclusión de que una estructura de datos de tipo árbol es lo más adecuado.

Antes de explicar en qué consiste la estructura de datos tipo árbol, es importante aclarar el concepto de nodo. Se considera nodo en un laberinto a todo aquel punto en donde el robot deba de tomar la decisión de cambiar su dirección; giro hacia la izquierda, derecha, o en retorno.

4.2.1 Estructura tipo árbol

Éste tipo de estructura consiste en un conjunto de nodos unidos entre sí de tal manera que parece formar un árbol, por ende su nombre. Cada nodo puede apuntar a 1 o más nodos y presenta características recursivas al poder contener varios árboles dentro de un árbol, tal y como se muestra en la figura 4.3.

Se escogió este tipo de estructura para la base datos ya que permite ser creada de manera dinámica a medida que el árbol vaya creciendo, por lo tanto no hace falta conocer con anterioridad el tamaño del laberinto, y se utiliza la memoria estrictamente necesaria.

Hacer uso de una base de datos en forma de matriz significa mayor complejidad al momento de relacionar un nodo con otros, y no se garantiza un buen rendimiento de la memoria, además, se hace natural pensar que es mejor almacenar los datos de los nodos del laberinto en una estructura de árbol.

En este tipo de estructura existen nodos padres e hijos; los padres son aquellos de los cuales se genera otro nodo, y los hijos son los que provienen de algún nodo. De la misma manera se puede hablar de nodos hermanos, los cuales comparten un mismo nodo padre.

Para ilustrar mejor este concepto se puede observar la figura 4.3, en la que se puede decir que el nodo 1 es padre de los nodos 3, 4 y 5, los cuales son todos hermanos, y así mismo el nodo 3 es padre de los nodos 8 y 9, a diferencia del nodo 5 que no tiene hijos. De la misma manera los nodos 14 y 15 son hijos del nodo 12 y hermanos entre sí.

Cada uno de los nodos del árbol contiene información sobre cada nodo del laberinto, como lo son el nombre de cada nodo, así como la relación que presentan con los demás nodos; Dicha relación se organiza en orientaciones norte, este, sur y oeste. De esta manera la relación o

“flecha” que une un nodo con otro lleva el nombre de la orientación hacia la cual se encuentra un nodo respecto a otro.

Esto se logró al crear una estructura de nodo como la observada en la figura 4.4, donde se almacena su nombre como una variable entera de 1 byte, un campo para verificar si el nodo ha sido visitado o no (Se usa para el retroseguimiento explicado en la sección 4.3.1) y 4 apuntadores del mismo tipo de estructura del nodo, que llevan como nombre: norte, este, sur y oeste. Aquí se puede observar claramente que cada nodo es recursivo, al permitir contener 4 apuntadores a otros nodos de su mismo tipo.

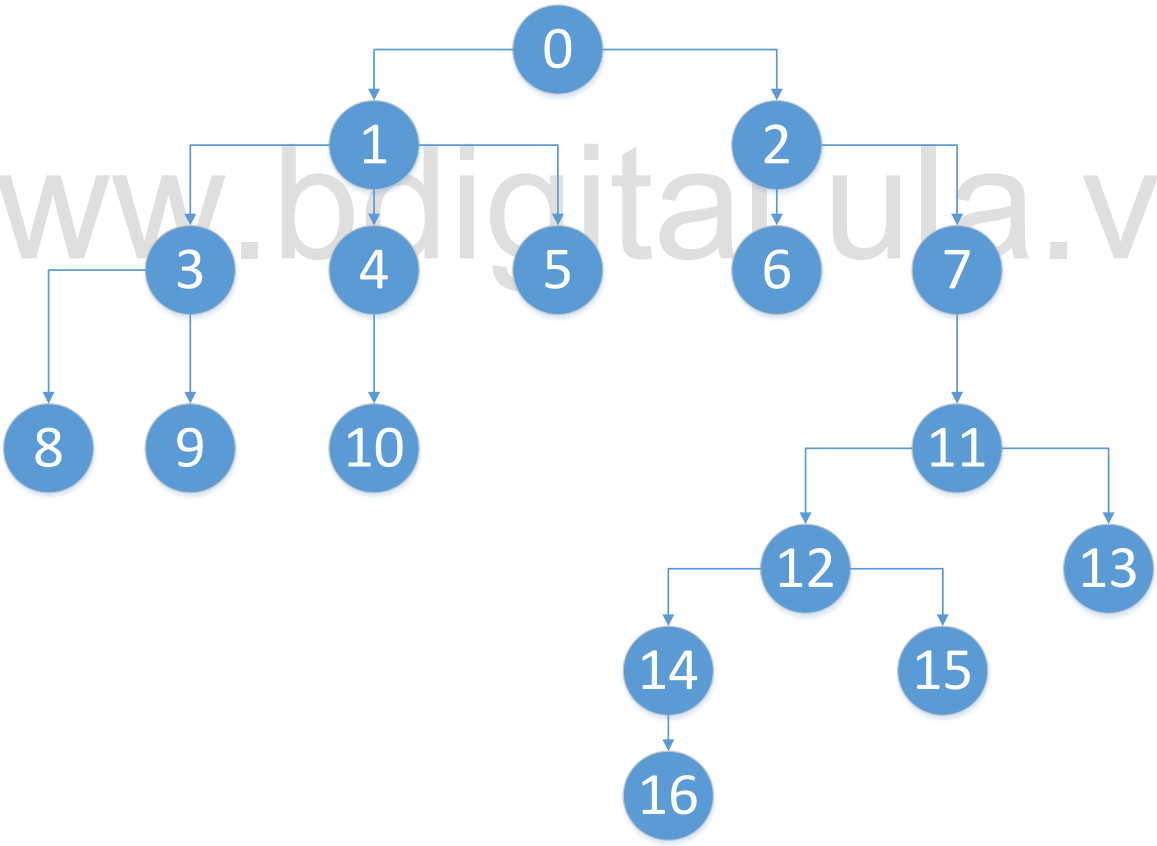

Figura 4.3 Estructura de árbol.

```
typedef struct nodo {
        uint8_t nombre;
        uint8_t visitado;
        struct nodo *norte, *sur, *este, *oeste;
}nodo_t;
```

Figura 4.4 Estructura de dato de cada nodo.

El nombre asignado a cada nodo depende del orden en que hayan sido encontrados por el robot. Es importante destacar que debido a la complejidad de los laberintos que se desean explorar y resolver, la estructura de árbol resultaría en ramas del árbol que se unen, o inclusive nodos o ramas que llegan al nodo raíz, lo cual rompe con la definición de una estructura de árbol. En estos casos la estructura deja de ser del tipo árbol, y pasaría a ser un grafo, el cual si permite sin ningún problema circuitos cerrados entre sus nodos.

Los grafos son muy parecidos a la estructura de árbol, pero en éstos el concepto de padre e hijo se desvirtúa. Incluso cuando el objetivo es resolver laberintos complejos, éste mismo algoritmo podría resolver laberintos más sencillos, en los cuales la estructura de la base de datos quería del tipo árbol; la estructura depende del laberinto a explorar.

Cabe destacar que debido al hecho de que el robot debe ser capaz de ir a un punto indicado por el usuario al finalizar su exploración, los laberintos planteados no cuentan con una entrada y una salida como los laberintos convencionales. Esto es porque en los laberintos convencionales el objetivo es conseguir la salida y ese objetivo es estático, en cambio, para este trabajo se quiso trabajar con objetivos dinámicos.

4.3 TÉCNICAS DE BÚSQUEDA

Debido a la necesidad de buscar la ruta más corta entre un nodo y otro, se hace menester aplicar algún algoritmo de exploración. Con esto en mente se deciden analizar algunas de ellas.

La búsqueda de soluciones es un tema fundamental en la inteligencia artificial, debido a que en muchas ocasiones la capacidad de resolver un problema permite medir el nivel de inteligencia de un hombre o una máquina.

Se puede llegar a pensar que al ser una máquina se debería conseguir la solución óptima a cualquier problema todo el tiempo, pero esto requiere de lo que se conoce como búsqueda exhaustiva; sin embargo, este tipo de búsqueda consume demasiados recursos debido a que la cantidad de combinaciones en un grupo de N elementos es igual a N!.

Esto significa que a medida que el número de elementos aumenta, la cantidad de posibles combinaciones aumenta en mayor medida, resultando en un crecimiento exponencial del tamaño del espacio de búsqueda. En la inteligencia artificial esto se conoce como explosión combinatoria, y para ilustrarlo de mejor manera se muestra en la figura 4.5 dicho comportamiento.

> "Mientras que la técnica exhaustiva o "fuerza bruta", teóricamente siempre funciona, normalmente no es práctica porque consume o demasiado tiempo o demasiados recursos de computación, e incluso ambos. Por esta razón, otras técnicas de búsqueda han ido desarrollándose." [15, P.18].

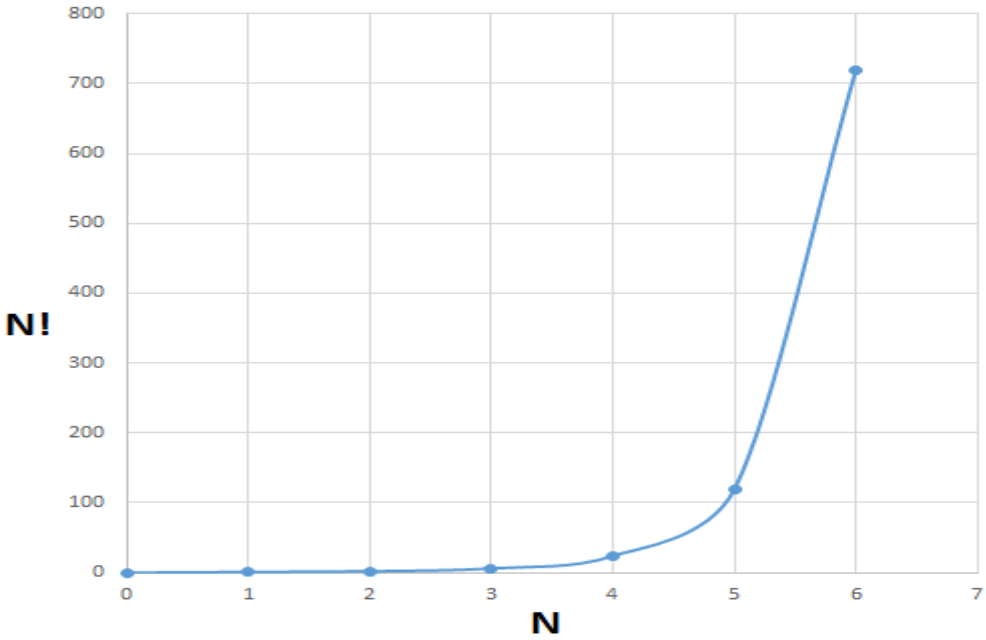

Figura 4.5 Explosión combinatoria.

4.3.1 Búsqueda del primero en profundidad

En el marco de estas ideas se decide trabajar con la técnica de búsqueda del primero en profundidad, el cual explora cada rama hasta encontrar la solución; de no conseguir la solución en una rama, pasa a la siguiente y así hasta conseguir la solución deseada.

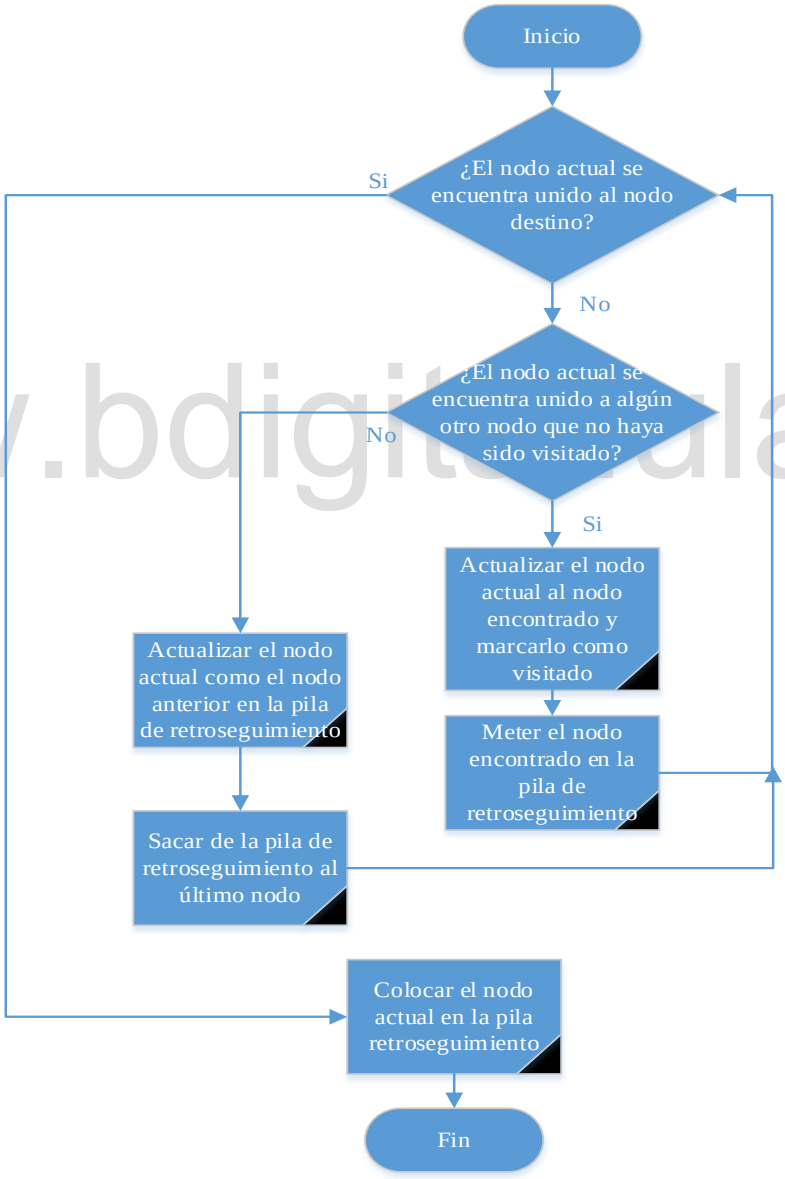

Figura 4.6 Diagrama de flujo búsqueda del primero en profundidad.

Para ilustrar mejor esta técnica, se plantea el caso de que en la figura 4.3 se parte del nodo 0 y se desea llegar la nodo 10. En este caso el orden de búsqueda sería: 1,3,8,3,9,3,1,4,10; con lo que la solución calculada por esta técnica, sería la ruta 1,4,10. Es importante mencionar el uso del retroseguimiento, el cual es un mecanismo de control que permite regresar a través del camino ya transitado con la finalidad de revisar otras sub-ramas, o regresar al inicio de la rama actual para pasar a la siguiente, y sacar de la solución los nodos sobrantes. En la figura 4.6 se puede observar el diagrama de flujo de dicha búsqueda.

La efectividad de la técnica de búsqueda del primero en profundidad depende bastante de la solución que se desea encontrar, así como la organización de los datos dentro de la base de datos. Dicho esto, se hace claro que esta técnica no asegura la solución óptima del problema, no obstante, sí se puede asegurar que la solución correcta será encontrada; puede que sea la solución correcta, pero en un tiempo que quizás no sea el mejor, o en el caso de tener la base de datos en estructura de grafos, no consigue la ruta más corta o lógica desde la perspectiva humana. Por esta razón se han desarrollado distintas técnicas de búsqueda en el campo de la inteligencia artificial, así como métodos para solventar estos problemas de optimización.

Dicho esto, se hace evidente que existen muchas posibilidades para solventar un problema de búsqueda, sin embargo, ninguno de estos asegura encontrar la respuesta más óptima para todos los casos. Por esta razón se decide utilizar la búsqueda del primero en profundidad, y en base a ésta aplicar los demás métodos y características para optimizarla.

4.3.2 Heurística

Debido a que la inteligencia artificial busca optimizar el rendimiento al agregar características que indican la presencia de inteligencia, se hace necesario dotar al programa de criterios que lo ayude a tomar decisiones informadas y, por ende, menos ciegas. Por esta razón, en el campo de la inteligencia artificial se hace común la adición de capacidades heurísticas. “La heurística son reglas simples que habilitan la posibilidad de que una búsqueda proceda en una dirección correcta” [15, P.34].

Tomando esto en cuenta, se decide hacer uso de la técnica de búsqueda del menor-coste, el cual busca conseguir el camino de menor distancia; a diferencia de su contra parte, la técnica de búsqueda de la escalada en colina, la cual busca la ruta que contenga menor cantidad de nodos, basándose en el hecho de que a mayor distancia esté el nodo, menos nodos quedan para llegar al objetivo.

4.3.3 Técnica de búsqueda del menor-coste

Como se mencionó anteriormente, la heurística consiste en brindarle criterios al programa con la finalidad de tomar decisiones informadas y, por ende, el único cambio a realizar en esta técnica respecto a la búsqueda del primero en profundidad, es que la función encargada de encontrar un nodo unido al actual, que no haya sido visitado antes, además, retorne el nodo de menor distancia al nodo actual. De esta manera, se asume que al avanzar hacia el nodo que esté más cercano al actual, se recorrerá una distancia menor hacia el nodo objetivo.

4.3.4 Múltiples soluciones

Debido a que ninguna de estas técnicas asegura la obtención del mejor resultado, se pueden calcular distintas respuestas para que así el programa escoja cual es la mejor de ellas y, de esta manera, aumentar las posibilidades de obtener el mejor resultado.

Con esto en mente, se decide aplicar el método de eliminación de caminos, el cual consiste en mantener como visitados todos los nodos del camino encontrado, y buscar otra solución. De esta manera si se consigue otra posible ruta, ésta es comparada con la anterior y se escoge la mejor opción. Esto luego se repite hasta que ya no se consigan más opciones.

Cabe destacar que este método no consigue todas las posibles rutas al destino, ya que esto significaría realizar una búsqueda exhaustiva. Al dejar los nodos de la ruta como visitados, estos nodos no vuelven a ser tomados en cuenta y por ende se evita redundancia en las respuestas encontradas.

4.4 SISTEMA DE COMUNICACIÓN ENTRE ROBOT Y COMPUTADOR

Como ya se ha expuesto en los capítulos anteriores, la comunicación entre robot y computador se realizó aplicando el internet de las cosas y, por ende, a través del establecimiento de la comunicación entre robot y el servidor Apache, que ya ha sido probado en la sección 3.2.

Con esto en mente, así como los conocimientos que se han expuesto hasta ahora, se puede hacer una distinción entre los 3 módulos (de manera general) considerados en este proyecto; el código ejecutado en el robot, ESP8266 y servidor Apache, son 3 procesos distintos que interactúan entre sí, con la finalidad de alcanzar los objetivos propuestos.

El robot en cuestión es la entidad física que explora el área designada, según las reglas de exploración expuestas en la sección 3.1, así como la recolección de los datos que se requieren del laberinto; el módulo WiFi ESP8266 es el encargado de establecer la comunicación TCP/IP entre robot y servidor, tomando en cuenta el filtrado de ruido, gestión de variables y respuestas provenientes del servidor; el servidor maneja las bases de datos, vista general del laberinto y mapeo del mismo, así como la interacción con el usuario.

Respecto a la inteligencia artificial, ésta reside tanto en el robot como en la página web; en el robot se aplica una búsqueda del primero en profundidad, mientras que en la página web se calculan múltiples soluciones aplicando la búsqueda del menor-coste. Esto se debe a que las distancias entre los nodos sólo se almacenan en la página web y no en el robot, por lo tanto, en este último no se cuenta con información heurística. De esta manera se hace menester una correcta comunicación entre todos estos elementos, dado que es clave para el funcionamiento general de todo el sistema.

4.4.1 Robot móvil 3pi

Desde el punto de vista del robot, la comunicación con el servidor marca la diferencia entre poder explorar de manera exitosa un laberinto complejo. Esto se debe a que cada vez que el robot detecta un nodo, éste le envía al servidor, orientación y distancia recorrida a partir del nodo anterior.

Esto resulta ser clave, ya que de esta manera el servidor puede calcular las coordenadas de cada nodo, y de esta manera detectar cuando el robot se encuentra en un nodo que ya ha explorado anteriormente; es decir, puede reconocer cuando esté pasado por un sitio en el cual ha estado antes, y de esta manera salir de un circuito cerrado, evitando quedarse de manera indefinida en él.

Además de los datos de desplazamiento, el robot le envía al servidor los datos de cada nodo explorado; es decir, el servidor mantiene una base de datos con la misma estructura de los nodos que el robot. De manera adicional, el robot envía las distancias registradas en sus 3 sensores de ultrasonido, con la finalidad de mantener un registro de ellos en caso de fallos. Cabe destacar que a esta obtención de datos también se le llama telemetría.

Con cada uno de los datos que se envía, el robot espera una respuesta en específico y, de esta manera, verifica la correcta recepción por parte del servidor. De esta manera si el robot no recibe la respuesta esperada, envía de nuevo los datos, ya que esto significa que hubo algún error en la comunicación y la información se vio comprometida en el camino, resultando en una recepción errónea de los datos enviados. De la misma manera, si transcurren 2 s y el robot no ha recibido ninguna respuesta, éste envía de nuevo los datos.

El envío y recepción se realizó utilizando los registros de comunicación serial que presenta el ATmega328P; desde la configuración para la velocidad de comunicación serial, así como la detección de datos en el bus de recepción o transmisión y obtención de datos recibidos. Toda la comunicación se estableció a 9600 bps y un byte a la vez, como es de esperar en la comunicación serial.

4.4.2 Sistema de infraestructura LAMP

Como se ha mencionado en la sección 3.2, el sistema de infraestructura LAMP cuenta con un servidor Apache capaz de ejecutar código PHP, cuenta con una base de datos MySQL y, además, corre en un sistema operativo Linux. El papel que juega el servidor Apache es el de albergar la página web que procesa y muestra algunos de los datos de la comunicación entre él y el robot. Como ya se observó en la figura 3.7, la página principal muestra los datos de

desplazamiento y las coordenadas de los nodos. Sin embargo, estos no son todos los datos que maneja la página principal.

Como ya se conoce, el servidor también recibe los datos de los nodos, queriendo decir que mantiene la misma base de datos de nodos que el robot; además, mantiene un registro de las coordenadas de cada nodo, lo cual es muy importante para el mapeo del laberinto en exploración.

A partir de los datos de desplazamiento y coordenadas, el servidor mantiene un historial de las posiciones en las que ha estado el robot; es decir, el orden en el que el robot se ha desplazado de nodo en nodo. Esto con el propósito de conocer claramente la posición del robot en el laberinto.

Dicha posición es utilizada también al momento de darle respuesta al robot, y es un punto clave en el funcionamiento del sistema. Esto se debe a que cuando el servidor detecta que el robot está en un nodo ya explorado, le debe contestar con el nombre del nodo en el cual se encuentra y, de esta manera, el robot pueda actualizar su posición, en vez de crear un nuevo nodo.

Es evidente entonces, que el servidor no contiene sólo la página mostrada en la figura 2.6, si no también, una segunda página en la cual se mapea el laberinto en cuestión.

En dicha página, llamada “prueba.php”, es donde corre el código en javascript que permite la actualización dinámica de la misma, con el fin de mostrar lo que se ha explorado del laberinto, a medida que el robot lo va descubriendo. Además, sobre ella el usuario interactúa con el objetivo de indicarle al robot a donde ir luego de haber explorado todo el laberinto.

En este propósito, claramente esta página hace uso de la información recibida y organizada por la página principal y, de esta manera, obtener los datos necesarios para el mapeo del laberinto. Por tal razón, esta página mantiene una comunicación constante con la base de datos en espera de los datos que se necesitan para realizar esta tarea, lo cual es una de las características más útiles de Javascript (y la razón por la cual se utiliza en este proyecto); la actualización dinámica de una página web sin necesidad que el usuario realice alguna acción.

Dicha actualización dinámica se logra al usar lo que se conoce en el mundo del desarrollo web como AJAX, el cual es un acrónimo para “Síncrono Asíncrono Javascript XML”, por sus siglas en inglés. Éste es un conjunto de técnicas que permite mantener una comunicación continua entre la página web y el servidor, con el objetivo de actualizar la información de manera automática sin necesidad de refrescar la página. De esta manera se va mostrando la información que se obtiene del laberinto, a medida que el robot la envía.

4.4.3 Módulo WiFi ESP8266

Finalmente, el ESP8266 se encarga de que toda la información intercambiada entre servidor y robot sea transmitida correctamente. No solamente en lo que respecta al protocolo TCP/IP, sino también, al protocolo interno del sistema.

Este protocolo interno consiste en las señas que lleva cada trama de información, con la finalidad de detectar qué tipo de información lleva, y de esta manera ser clasificada. Es evidente que los datos de desplazamientos se trabajan de una manera distinta a los datos de los nodos o, incluso, a la telemetría de los sensores de ultrasonido.

Por tal razón, el ESP8266 está codificado para detectar dichas señas y acomodar la información de manera adecuada para ser procesada y entendida por el servidor. De la misma manera, el ESP8266 lleva la tarea de acomodar la información a ser recibida por el robot, es decir, las respuestas del servidor.

Sin embargo, servir de traductor y puente de información entre robot y servidor no es el único papel del ESP8266, sino también filtrar todo aquel ruido que pueda existir en la comunicación serial entre él y el robot. De esta manera se evita el envío de información basura al servidor. Esto último siendo clave para el funcionamiento del sistema, debido a que si el servidor recibe datos erróneos a causa del ruido, éste igualmente le responde al robot que recibió los datos, causando que continúe la exploración y, por ende, corrompiendo la base de datos del servidor. Esto tiene como consecuencia errores en el cálculo de la ruta más corta, enlaces rotos entre nodos y detección falsa de nodos repetidos.

CAPÍTULO V
ALGORITMOS PROPUESTOS

Planteadas las técnicas a implementar en la solución final del sistema, se continúa con el algoritmo de dicha solución al problema. En términos generales hay dos grandes acciones que realiza el sistema como un todo: Explorar el laberinto y encontrar la ruta hacia un nodo aleatorio especificado por el usuario. Con esto en mente, lo primero a resolver es la exploración, dado que a partir de esta, tanto el robot como el servidor, logran obtener los conocimientos del espacio en el cual se desenvuelve y, de esta manera, lograr el cálculo y ejecución de dicha ruta hacia el objetivo dinámico.

5.1 EXPLORACIÓN DEL LABERINTO

En primer lugar se detallan los algoritmos usados tanto en el robot, como en el servidor y el módulo WiFi ESP8266, mientras el laberinto es explorado.

5.1.1 Algoritmo del 3pi

En primer lugar, el robot aplica la ya mencionada regla de la mano izquierda como base para la exploración del laberinto, y a medida que consigue nuevos nodos, los almacena y establece la relación que existe entre ellos y los ya registrados.

A medida que el robot descubre nuevos nodos, éste le envía a la página principal del servidor la información correspondiente a cada nodo descubierto, y así, el servidor va obteniendo y procesando la información necesaria a ser almacenada en las bases de datos del servidor, para luego ser interpretadas por la página en donde ocurre el mapeo simultáneo del laberinto. Lo cual significa que el laberinto se va mostrando en la página web a medida que el robot lo va descubriendo.

Es importante mencionar que cuando el robot se encuentra en una intersección, éste claramente sólo puede tomar uno de los 3 posibles caminos (izquierda, adelante o derecha), significando que al aplicar la regla de la mano izquierda, ésta lleva la prioridad más alta y, por ende, cruza hacia este sentido. Pero con el objetivo de optimizar y hacer más lógica la exploración, el robot marca como direcciones pendientes aquellas a las que no pudo ir en algún nodo dado.

De esta manera, el robot continúa con su exploración básica aplicando la regla de la mano izquierda, manteniendo en memoria que hay 1 o 2 caminos no explorados todavía, en 1 o más nodos anteriormente descubiertos. Es aquí donde la detección de nodos repetidos, así como las técnicas de búsqueda ya mencionas, juegan un papel crucial en la exploración del laberinto.

Dado que el robot al recibir la respuesta del servidor, indicándole que se encuentra en un nodo ya explorado, el robot procede a actualizar su posición actual a partir de la información recibida (nombre del nodo en el cual se encuentra). Por esta razón se considera que la detección de repetición de nodos así como las técnicas de búsqueda son el conjunto de técnicas más importante al momento de la exploración.

Es necesario que el robot aplique la técnica de búsqueda, debido a la falta de una variable para cada nodo. Esto es porque se cuenta con tan sólo 3 variables para almacenar nodos; una variable estática que contiene la información del nodo raíz, un apuntador que almacena la dirección del nodo actual y otro apuntador para el nodo anterior. Como se mencionó en la sección 4.2.1, la base de datos en árbol permite la creación dinámica de memoria de manera óptima. Esto significa que la memoria necesaria para almacenar los datos de cada nodo, se reserva para tal fin al momento de ser necesitada y, por lo tanto, lo que se maneja es

directamente las direcciones de memoria del ATmega328P en donde se encuentra la información de cada nodo, según la estructura mostrada en la figura 4.4.

Por lo tanto, no existe una variable que pueda ser accedida de manera simple y directa, como se haría en cualquier otro caso. Por esta razón, cuando el servidor le envía al robot el nombre del nodo en cual se encuentra, se debe ubicar la dirección de memoria del ATmega328P, que forma parte de la base de datos del laberinto, en el cual está almacenado dicho nombre de nodo y, por ende, el resto de la información de ese nodo; es decir, un problema de búsqueda.

Dicha tarea se realiza partiendo del nodo raíz, o nodo cero, el cual es el único que es almacenado en su propia variable. El resto de los nodos son almacenados en direcciones de memorias reservadas dinámicamente al momento de crear un nuevo nodo; por esta razón se accede al resto de los nodos a través de apuntadores.

De esta manera el robot al recibir una respuesta positiva a un nodo repetido, con la información del nombre del nodo aplica una búsqueda del primero en profundidad a partir del nodo raíz, en busca de la dirección de memoria que almacena dicho nombre, y así actualizar su posición actual para continuar con la exploración.

Conociendo el robot su posición actual, se procede a aplicar nuevamente la búsqueda del primero en profundidad, con la finalidad de investigar sobre la existencia de un nodo pendiente. Al ser detectada la presencia de al menos un nodo pendiente, se procede una búsqueda del primero en profundidad, pero esta vez calculando la ruta que el robot debe tomar para llegar al nodo en el cual quedó una ruta pendiente.

De esta forma se evita seguir explorando nodos ya conocidos, y logra salir de algún posible circuito cerrado como el de la figura 4.2. El robot sigue la ruta calculada por el algoritmo de búsqueda, hasta llegar al nodo ya explorado en el cual quedó 1 o 2 caminos pendientes.

Esta rutina termina cuando el robot gira en dirección del nodo pendiente y avanza hacia él, momento en el cual continúa con el algoritmo básico de exploración hasta encontrarse de nuevo con un nodo repetido.

Así mismo continúa el robot hasta que se tope con un nodo repetido y ya no se detecten nodos pendientes, lo cual significa que todos los nodos han sido descubiertos, la exploración del laberinto ha sido finalizado y debe esperar las instrucciones del usuario sobre el nodo al cual se desea ir.

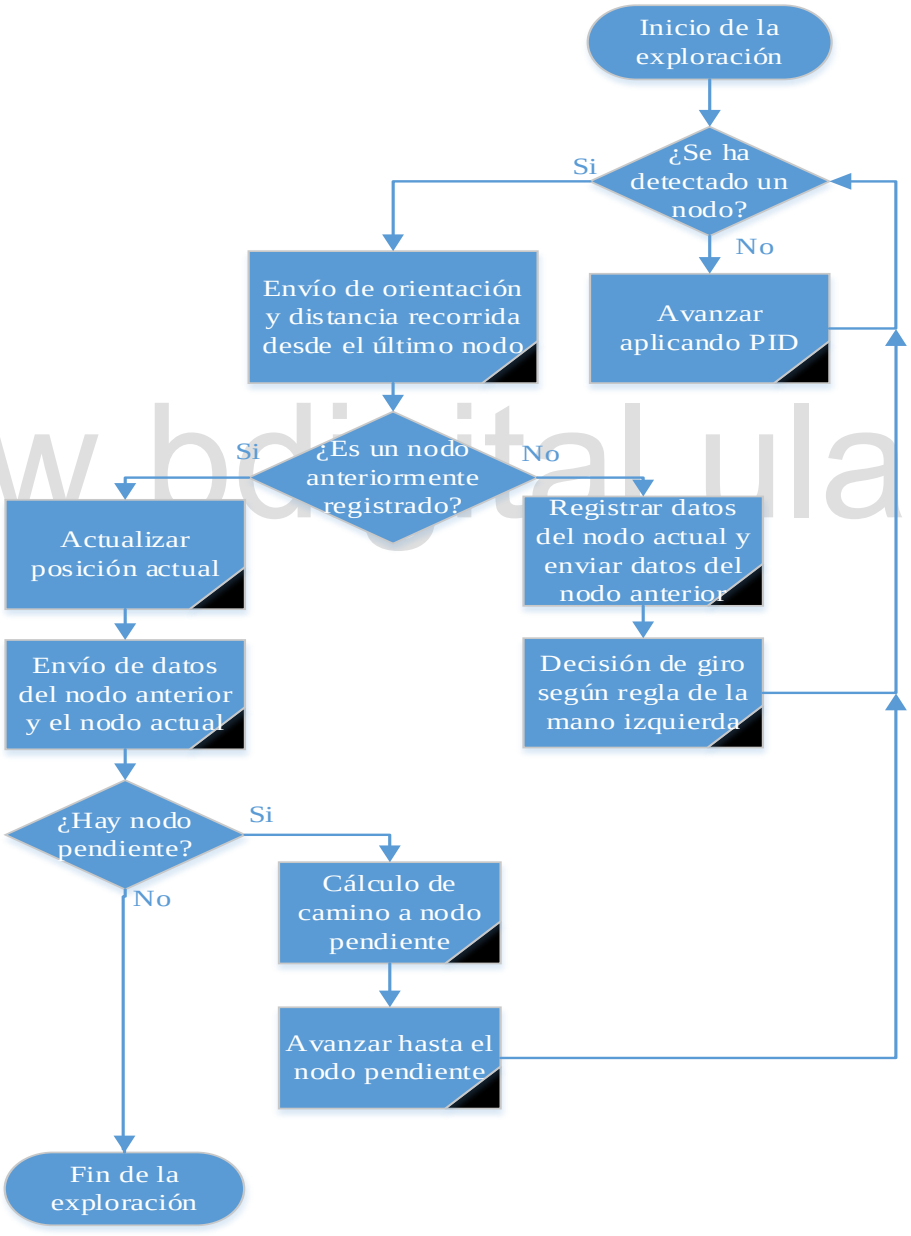

Figura 5.1 Diagrama de flujo de exploración del 3pi

El robot ejecuta el algoritmo para la búsqueda del primero en profundidad, debido a que él no almacena datos sobre las coordenadas o distancias que existen entre los nodos y, por lo tanto, no cuenta con una heurística que le permita optimizar su búsqueda. Esto es así con el propósito de utilizar lo menos posible la memoria del robot, y de esta manera evitar que se agote su memoria para los datos de los nodos.

Cabe destacar que el robot al seguir la ruta indicada hacia el nodo pendiente, le sigue enviando al servidor la información correspondiente a su desplazamiento, con la finalidad de que el servidor conozca la posición del robot en todo momento. Finalmente se muestra en la figura 5.1 el diagrama de flujo del programa propuesto.

5.1.2 Página web principal del servidor

La página principal está construida con HTML, CSS y PHP, lo que quiere decir, como ya se ha mencionado antes, el código de programación en el servidor solamente se ejecuta cuando ésta recibe una petición.

Esto se debe a que PHP es un lenguaje de programación del lado del servidor; es decir, corre solamente en el servidor, y el servidor sólo responde cuando recibe una solicitud. Por lo tanto, el código en PHP se ejecuta una sola vez por cada solicitud enviada desde el ESP8266.

Debido a que esta página es la encargada de procesar y organizar los datos que recibe del 3pi, lo primero que realiza es verificar cuál de las posibles variables es la que ha recibido, para luego procesarla correctamente y responderle al 3pi de manera adecuada.

En primer lugar se encuentran los datos correspondientes al desplazamiento del robot, los cuales son de suma importancia para el reconocimiento de nodos repetidos, así como para el mapeo del laberinto. La trama de información de los datos de desplazamiento contiene primero la orientación actual del robot y luego la distancia recorrida a partir del nodo anterior. Específicamente el servidor espera recibir a través de una solicitud GET, una trama que sea de forma: “item=orientacion,distanciaRecorrida _”; el separador de campos es la coma y el final de trama es un guión bajo.

De esta manera, el servidor al saber la orientación en la cual se movió el robot, conoce el eje equivalente. Para ser más claros, si el robot se mueve en dirección norte, eso es equivalente a desplazarse en el eje Y positivo; así mismo, si el traslado del robot fue en dirección Oeste, quiere decir que se movió en el eje X negativo. Conociendo el eje en el cual se desplazó el robot y la distancia recorrida, se calcula fácilmente las coordenadas del robot en todo momento, y así, verificar si dichas coordenadas ya están registradas o no. Dependiendo de esto el servidor responde de manera positiva o negativa a la repetición de un nodo.

Otro conjunto de datos importantes es la de los nodos, ya que con esta se completa la información necesitada para mapear el laberinto. Esta trama es un poco más larga, conteniendo en primer lugar el nombre del nodo, y luego la información de lo que existe en cada una de las 4 direcciones, siempre en el sentido de las agujas del reloj a partir del Norte (Norte, Este, Sur, Oeste). Específicamente el servidor espera recibir a través de una solicitud GET, una trama que sea de forma: “nodo=nombreDeNodo,nombreDeNodoAlNorte,nombreDeNodoAlEste,nombreDeNodoAlSur ,nombreDeNodoAlOeste _”. Al igual que la trama del desplazamiento, el separador de campos es la coma y el final de trama es el guion bajo.

Los datos obtenidos de estas dos tramas le permiten a la página principal, agregar los datos completos a la base de datos del mapeo del laberinto, la cual cuenta con la información de cada nodo, además de las coordenadas; es decir, se junta la información obtenida en ambas tramas, en el caso de que se descubra un nuevo nodo. Toda la información correspondiente a las coordenadas y los nodos, es almacenada en orden según el nombre de cada nodo. De esta manera se mantiene una estructura organizada en la que la información de cada nodo corresponde con su posición en la base de datos.

En lo que respecta a la telemetría aplicada a los datos de los sensores de ultrasonido, el servidor simplemente almacena (Sólo cuando verifica que lo que recibió fue una trama de distancia) estos datos en otra base de datos. Específicamente el servidor espera recibir a través de una solicitud GET, una trama que sea de forma: “distancia=distanciaIzquierda,distanciaAdelante,DistanciaDerecha_”. De nuevo se observa que el separador de campos es la coma y el final de trama es un guión bajo. El diagrama de flujo de la página web principal se muestra en la figura 5.2

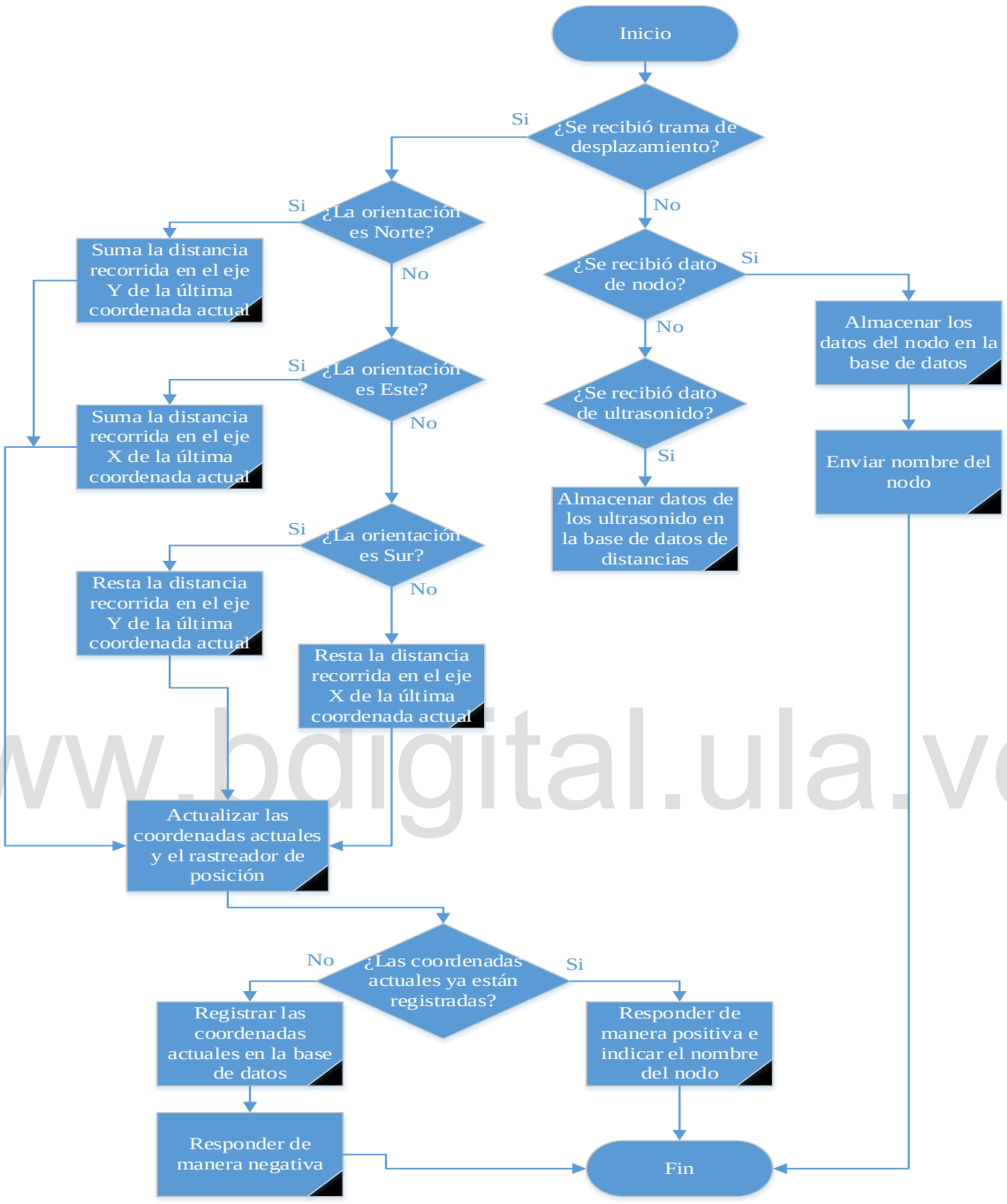

Figura 5.2 Diagrama de flujo de exploración de la página web principal

5.1.3 Página web de mapeo en el servidor

A diferencia de la página principal, esta página de mapeo está codificada en Javascript, debido a la necesidad que tiene de ser dinámica; esto es porque debe actualizar la información

mostrada al detectar un cambio en la base de datos, así como, reconocer el nodo al cual se le hace click una vez esté explorado el laberinto. Esto es de gran ventaja porque se desea que el laberinto vaya apareciendo en esta página a medida que el robot envía los datos de la exploración y, de esta manera, realizar el mapeo y ubicación simultanea dentro del laberinto. Esto último se logra al aplicar AJAX en el código Javascript, el cual es un conjunto de tecnologías que permite:

- Leer información de un servidor después de que la página web haya cargado.
- Actualizar una página web sin refrescar la página.
- Enviar información al servidor sin que el usuario lo note.

A diferencia de HTML, que se encarga de definir la estructura de la página, CSS es la responsable del estilo; esto quiere decir que todos los colores, tamaños, márgenes, etc. vienen definidos el lenguaje CSS. Es importante resaltar que estos dos no son lenguajes de programación, ya que no cuentan con estructuras de decisión; PHP y Javascript sí cuentan con estructuras de decisión. Javascript cuenta con la propiedad de modificar los estilos impuestos por CSS, lo cual significa que puede alterar los colores de los elementos dinámicamente a raíz de un evento dado. Esto es clave a la hora de mostrar el laberinto a medida que se va descubriendo.

La página cuenta con una cuadrícula del mismo color que el fondo de la página, con la finalidad de que la cuadrícula no sea visible. Cada una de las celdas en la cuadrícula tiene una etiqueta de identificación con las coordenadas que representa dicha cuadrícula, lo que permite modificar el estilo de cada una de ellas. Dependiendo de las coordenadas que se registren para cada nodo, el algoritmo de la página selecciona la casilla correspondiente y cambia su color, con el objetivo de hacer visibles aquellas orientaciones donde se detecta una pared, y dejando invisibles aquellas donde exista un tramo al cual dirigirse.

De esta manera, la página web encargada del mapeo del laberinto va obteniendo los datos de la exploración a través de una comunicación continua con el servidor, aplicando AJAX (sin que el usuario lo note), y a medida que el robot va enviando dichos datos, la página se actualiza de manera dinámica para mostrar lo que se ha descubierto del laberinto. Para

visualizar de manera más clara el funcionamiento de la página web encargada del mapeo, se muestra el diagrama de flujo mostrado en la figura 5.3.

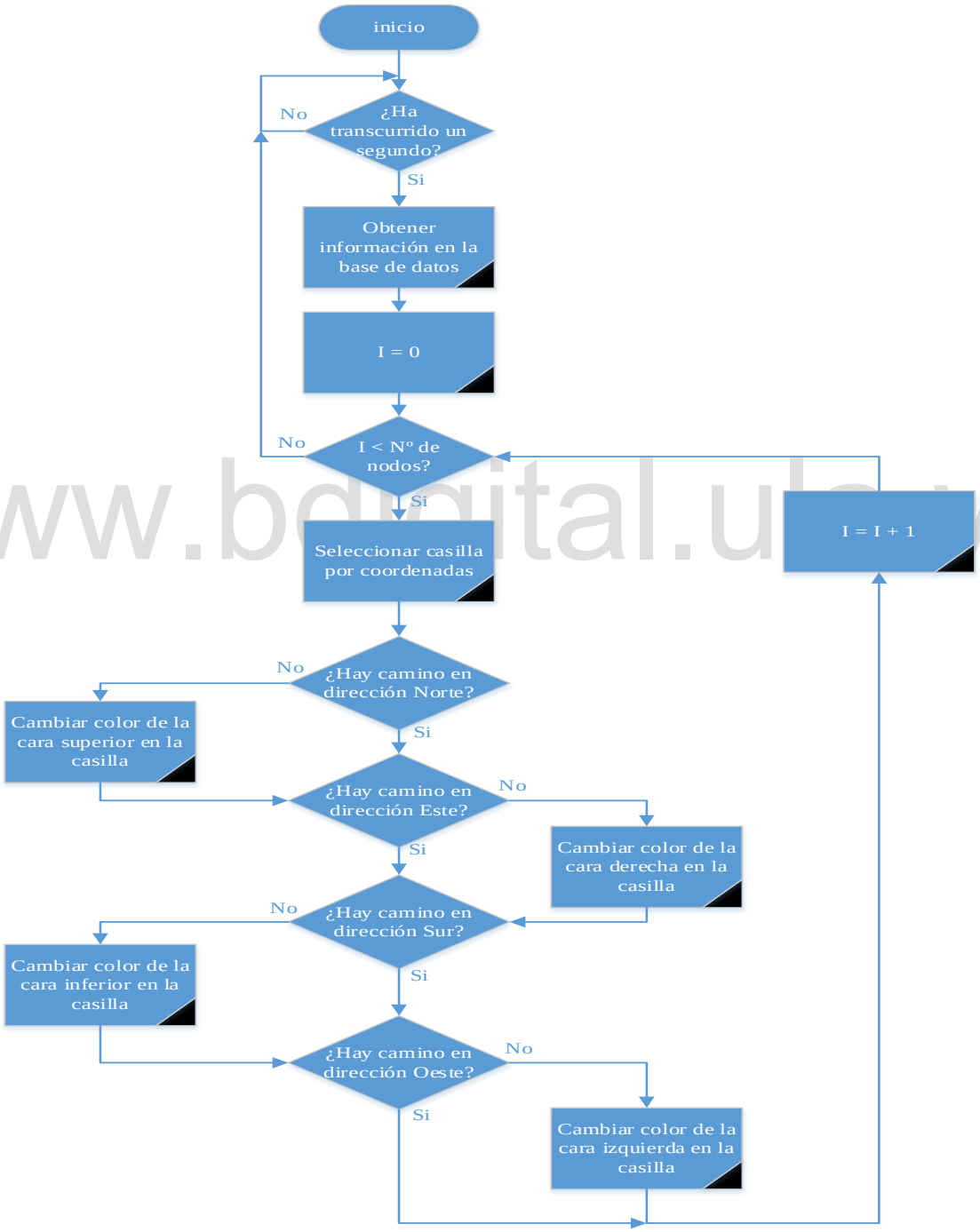

Figura 5.3 Diagrama de flujo de la página web encargada del mapeo del laberinto

5.1.4 Módulo WiFi ESP8266

De la misma manera en que un usuario accede a una página web al colocar la dirección URL en el navegador, se comunica el ESP8266 con el servidor. Al colocar una URL en un navegador, éste se encarga de todo el protocolo y envía una solicitud HTTP al servidor, a lo que el servidor responde con la respectiva información, y el navegador la muestra en la pantalla del usuario. Lo mismo ocurre cuando el ESP8266 se conecta al servidor y le envía la trama GET, tal y como se mostró en la figura 3.6.

La diferencia está en el hecho de que el ESP8266 no está capacitado para interpretar el formato HTML, las instrucciones de estilo del CSS y el código Javascript, por lo tanto, el ESP8266 interpreta todas las respuestas en formato texto y las envía por comunicación serial.

Debido a que la página responde con todo su contenido, el ESP8266 recibe toda la estructura HTML más el texto añadido y el mensaje enviado por la página se ve envuelta en mucha información que no es de interés para el robot.

Debido a esto, la respuesta del servidor al 3pi, contiene una etiqueta utilizada sólo para las respuestas hacia el 3pi, lo cual permite al ESP8266 detectar en qué parte de toda la trama recibida por el servidor se encuentra la respuesta deseada. Esta respuesta es extraída y enviada al 3pi. Dichas etiquetas son “<beginwally>” y “<endwally>”, indicando el inicio y final, respectivamente, de dicha respuesta.

Las respuestas del servidor dependen del tipo de variable que éste haya recibido. Si se trata de un dato de desplazamiento, la respuesta del servidor es una “N” en caso de estar el robot en un nuevo nodo, en cambio, si las coordenadas del robot coinciden con las de un nodo ya registrado, la respuesta del servidor toma la forma de “SnombreDeNodo”. Nótese que no existe un separador de campo, ni final de trama; el ESP8266 gestiona y envía esta respuesta como un vector de caracteres y, por lo tanto, la primera posición del vector corresponde a la “S”, indicando que es un nodo repetido, mientras que la segunda posición indica el nombre del nodo.

Con respecto a los datos de nodos, el servidor responde con el nombre del nodo recibido. De esta manera el robot corrobora que los datos se han procesado correctamente. Finalmente

la trama de respuesta cuando el robot envía los datos de las distancias registradas por los HC-SR04, es una “K”, y así mismo verifica que no ha ocurrido un error en la comunicación.

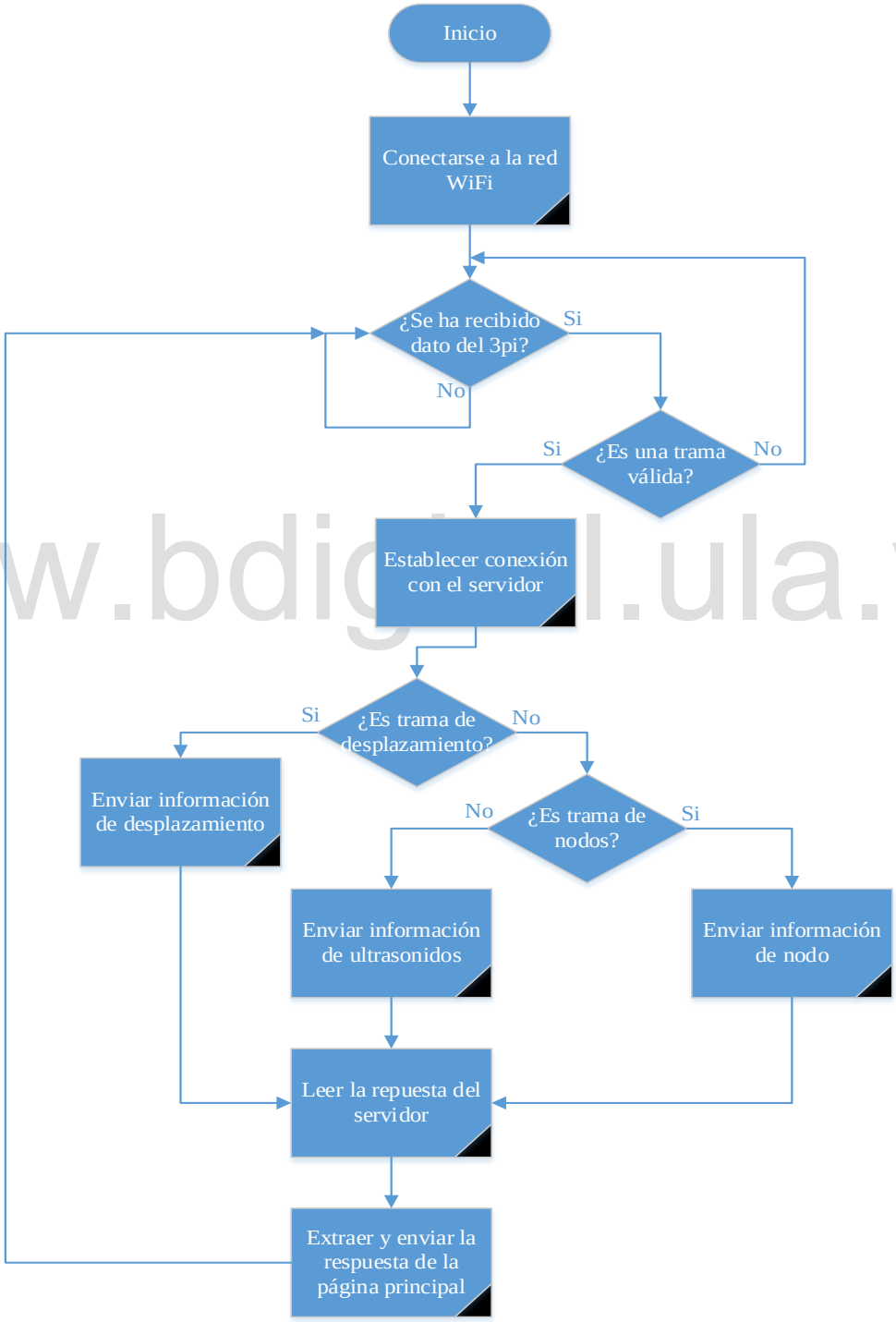

Figura 5.4 Diagrama de flujo del ESP8266

A diferencia de los mensajes enviados por el servidor, los enviados por el 3pi contienen estrictamente lo que se quiere enviar. La diferencia reside en que el 3pi envía distintos tipos de trama y el ESP8266 debe detectar qué tipo de dato es, para enviarlo como la variable GET adecuada a ser interpretada por el servidor. De la misma manera debe discriminar aquellos datos que no coincidan con ninguno de los otros y, de esta manera, evitar el envío de ruido al servidor. Por esta razón, el 3pi al final de cada trama envía un carácter que es específico para cada tipo de dato. De esta forma el ESP8266 detecta qué tipo de información se desea enviar al servidor, y organiza la URL de manera apropiada.

Bajo el marco de estas ideas y, con la finalidad de aclararlas, el final de trama para los datos de desplazamiento es un punto y coma; en cambio, el final de trama para los datos de nodos y distancias, son dos puntos y un guion, respectivamente. De esta manera se logra que el ESP8266 discrimine ruido y reconozca que tipo de trama el 3pi quiere enviar al servidor. En la figura 5.4 se muestra claramente el algoritmo del ESP8266.

5.2 RUTA HACIA EL OBJETIVO DINÁMICO

Una vez explorado el laberinto, el robot espera las instrucciones del servidor sobre a cuál nodo dirigirse, en este sentido, de aquí en adelante el robot funciona más como un esclavo del servidor.

5.2.1 Cálculo del camino más corto

Como se mencionó anteriormente, las búsquedas aplicadas por el 3pi son del primero en profundidad, debido a que en él no se almacenan las coordenadas de los nodos y, por ende, no puede calcular las distancias entre nodos. Por otra parte, el servidor sí cuenta con los datos necesarios para conocer las distancias entre nodos y, por ende, en éste es que se aplica la búsqueda del menor-coste y cálculo de múltiples caminos; es decir, aquellos que requieren de heurística. Debido a que la página durante toda la exploración mantiene un registro de los nodos por los cuales ha estado el robot, su última posición está almacenada en la base de datos

del rastreador de posición. De estos datos, la página encargada del mapeo extrae la posición actual del robot y, de esta manera, conoce el punto de partida para el cálculo de la ruta.

Javascript presenta la característica de detectar eventos sobre elementos HTML, permitiendo conocer cuando se hace *click* sobre uno de estos. De esta manera se logra conocer el nodo destino al cual el usuario desea que el robot vaya. Obtenidos el punto de inicio y destino, se procede a ejecutar el algoritmo de búsqueda de menor-coste más eliminación de caminos, para así obtener la ruta más corta entre los dos puntos.

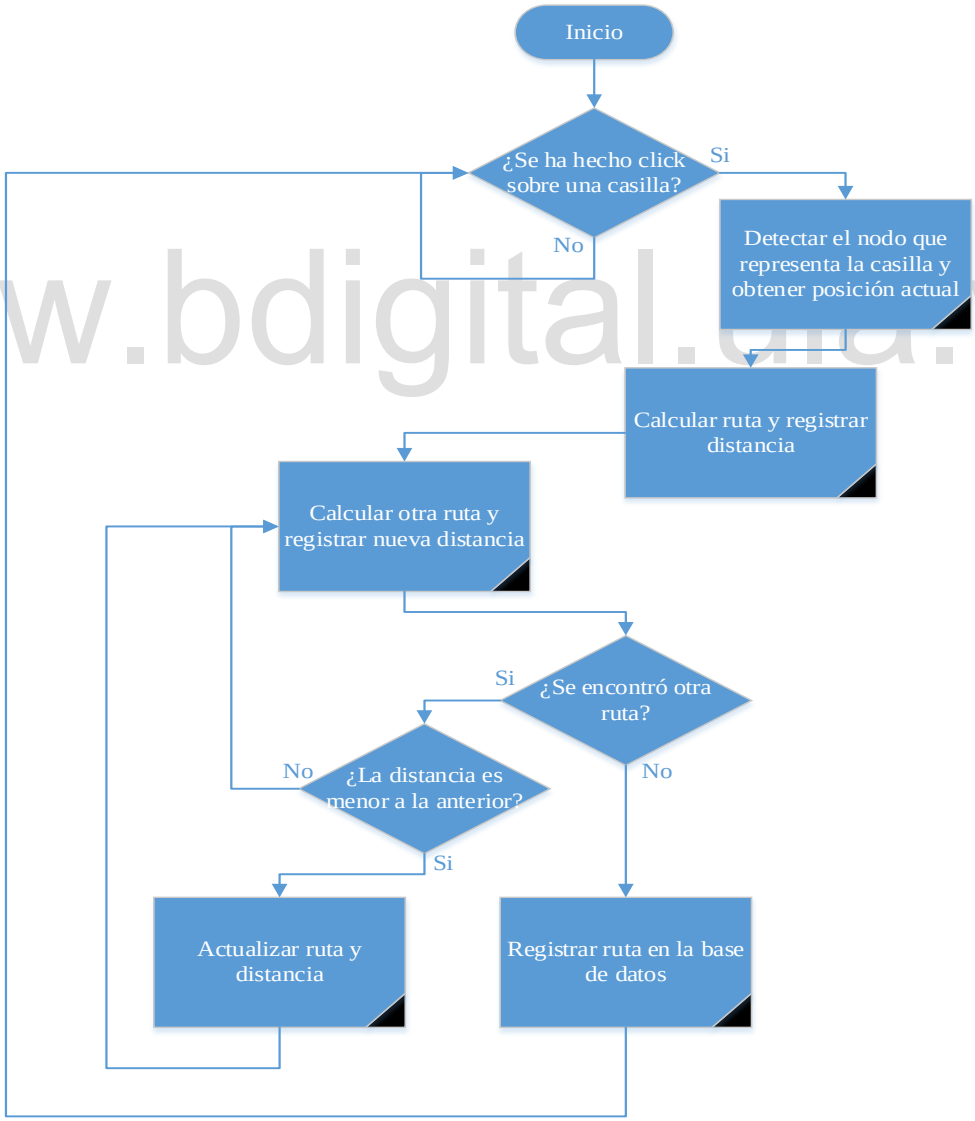

Figura 5.5 Diagrama de flujo del cálculo de ruta

Una vez calculado el camino, éste es almacenado en la base de datos de la ruta, para que el robot la solicite y le sean enviados los nodos. En este sentido, el robot le envía al servidor una “Q” cada dos segundos después de haber finalizado la exploración.

Si en la base de datos de la ruta existen datos, el servidor le envía los nombres de los nodos que conforman la ruta calculada. En caso de no haber una ruta en la base de datos, el servidor responde con una “N”, a lo que el robot espera un segundo antes de volver a solicitar datos de la ruta.

Con los datos de la ruta, la página del mapeo le indica al usuario cuál es el camino a seguir por el robot, aprovechando así las ventajas dinámicas que presenta Javascript. Esto se logra al cambiar de color el relleno de las casillas que forman parte de la ruta y el punto inicial, para el cual se utiliza un color distinto para diferenciarlo de la ruta.

5.2.2 Seguimiento de la ruta asignada

Al momento de finalizar la exploración, el robot entra en la rutina de esperar que el usuario indique el objetivo. Debido a que el servidor sólo responde cuando el 3pi envía una solicitud, éste envía constantemente una señal de que está listo para recibir la instrucción que le llevará al objetivo.

En el momento en que se almacena una ruta en la base de datos, la página principal tiene con que responderle al 3pi, y éste recibe el nombre del nodo al cual debe dirigirse. Se puede pensar que la ruta completa se le envía al robot pero, en realidad, se le envía el siguiente nodo al cual debe ir; es decir, se le va enviando un nodo a la vez, y sólo cuando el robot haya llegado al nodo, se le envía el siguiente hasta que finalmente llega al objetivo.

Al llegar a su destino, ya no hay más ruta disponible en la base de datos, por lo cual, el robot se queda de nuevo preguntando por un nuevo nodo al cual ir. Ahí se quedará hasta que el usuario designe un nuevo punto destino, y se registre una nueva ruta.

De esta manera el usuario puede darle cuantos destino quiera, significando que no solamente el objetivo es dinámico, sino también, el punto de partida. Cuando el robot recibe el siguiente nodo al cual debe dirigirse, éste calcula el sentido en el que debe girar para dirigirse al nodo indicado.

Una vez haya llegado, el robot le envía al servidor el nombre del nodo en el cual se encuentra y, de esta manera, el servidor actualiza la posición en la base de datos; lo cual permite que a medida que el robot avanza, dicha posición se muestra en la página del laberinto.

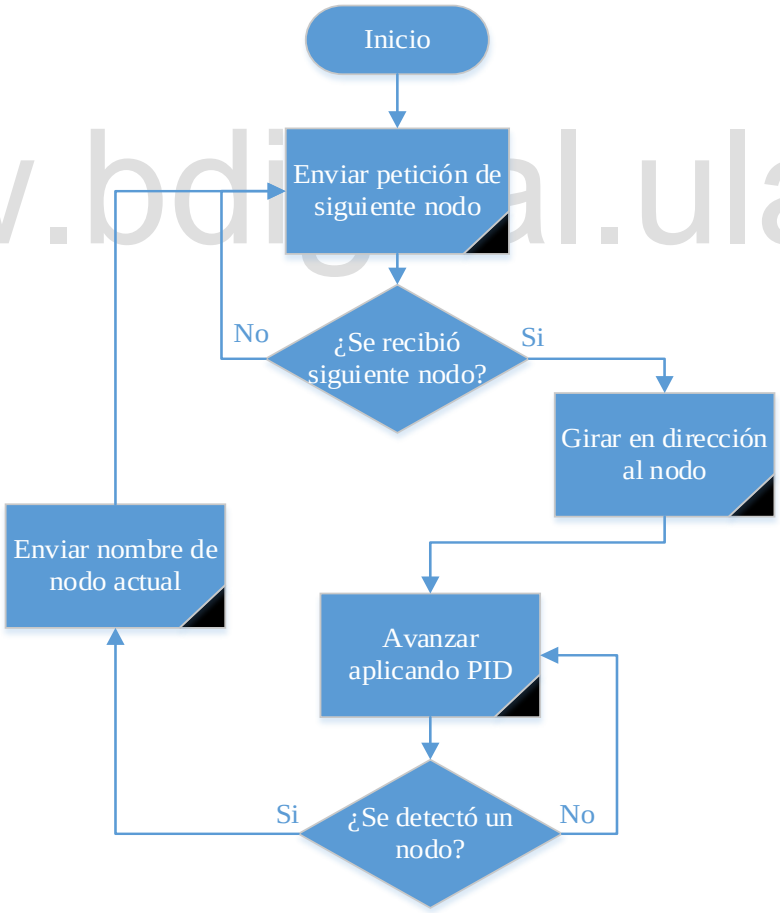

Figura 5.6 Diagrama de flujo de seguimiento del objetivo dinámico del robot

CAPÍTULO VI
PRUEBAS Y RESULTADOS

Todas las pruebas definitivas se realizaron en base al mismo laberinto, debido a que con simplemente cambiar el punto desde el cual inicia el robot, éste se enfrenta a problemas totalmente distintos y, de esta manera, no ha sido necesario cambiar la forma del laberinto para comprobar su correcto funcionamiento. El laberinto en cuestión se muestra en la figura 6.1.

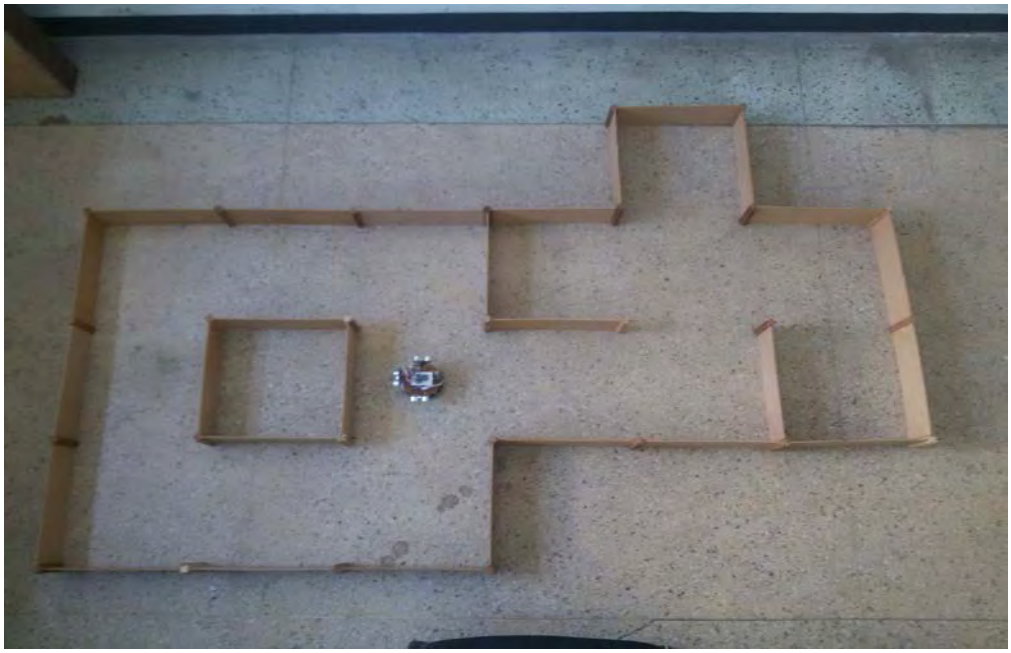

Figura 6.1 Laberinto propuesto

Sin importar el punto en que sea iniciado el robot, éste se verá enfrentado a todos los tipos de decisiones a los cuales se debe enfrentar, como lo son: retornos, intersecciones, cruces a la izquierda, derecha, decisiones de seguir hacia adelante, nodos repetidos y circuitos cerrados. Todo esto con la finalidad de corroborar la correcta implementación de los algoritmos de exploración, así como las técnicas de búsqueda ya expuestas.

6.1 SISTEMA DE EXPLORACIÓN Y MAPEO DEL LABERINTO

En primer lugar se comprobó el funcionamiento del algoritmo de exploración, así como el funcionamiento del sistema como un todo, al momento de explorar el laberinto propuesto en la figura 6.1.

6.1.1 Detección de nodos repetidos y pendientes

La detección de nodos repetidos, así como el marcado y búsqueda de orientaciones pendientes en un nodo dado, son los puntos de mayor importancia, debido a que ellos son la propuesta que se plantea para lograr alcanzar los objetivos planteados; por lo tanto, han sido el punto inicial de enfoque al momento de realizar las pruebas.

Se realizaron numerosas pruebas, para así corroborar el correcto funcionamiento del papel que interpreta cada parte del sistema bajo distintas situaciones y, en caso de ser necesario, ajustar o afinar aquellos detalles que siempre emergen al momento de efectuar las pruebas.

Finalmente se lograron observar resultados positivos, que indican el buen funcionamiento de los algoritmos propuestos. Como se esperaba, la combinación de estas técnicas ya explicadas en la sección 5.1, lograron servir su propósito: evitar recorrer por un tiempo indefinido una zona del laberinto, debido a la desventaja que presentan las reglas de exploración básicas; así como optimizar la exploración y obtención de información en un espacio desconocido.

Se llega a esta conclusión, no solamente debido al correcto desenvolvimiento del robot en el laberinto al momento de explorar, sino porque en las bases de datos manejadas por el

servidor, se observa claramente la información de cada nodo, así como la relación que existe entre cada uno de ellos; se aprecian las direcciones pendientes en cada nodo a medida que se va explorando, y la asignación de un nombre en esa dirección al momento de explorar dichos nodos pendientes. En definitiva, los algoritmos codificados para el manejo de memoria dinámica, creación de nodos y asignación de relaciones entre ellos, se comportan apropiadamente.

Así mismo se logró corroborar que el robot al pasar por un nodo por el cual ya ha visitado anteriormente, el servidor se da cuenta y le envía el correcto mensaje indicando que es un nodo repetido y, a su vez, el nombre de dicho nodo. Con esto, el robot procesa adecuadamente la información, se adapta y toma la decisión de ir al nodo pendiente detectado.

Se observó el seguimiento del robot hacia dicho nodo pendiente, así como el acertado direccionamiento hacia la orientación pendiente cuando se encuentra en el nodo en donde quedó otro camino al cual ir.

Como medida de indicación, se programó el robot para que indicara mediante sonidos la correcta o incorrecta creación de un nodo; dependiendo de si se pudo reservar un espacio de memoria. De la misma manera el robot indica cuando ha terminado de explorar el laberinto, y se encuentra a la espera de indicaciones.

Dicho esto, el robot y el servidor repiten todas las acciones ya mencionadas cuantas veces sean necesarias, hasta que finalmente todo el laberinto ha sido explorado. En este punto el robot ya no consigue nodos pendientes y termina su programa de exploración.

6.1.2 Mapeo del laberinto

En un segundo plano, pero no de menor importancia, se encuentra la página web encargada de mostrar el laberinto a medida que es descubierto por el robot, para ser observado por el usuario y así obtener una vista general y completa del espacio explorado por él.

Como se ha planteado, la página web muestra el laberinto a medida que se va explorando, hasta que el robot finaliza su exploración, y el mapa del laberinto queda debidamente

planteado en dicha página. En las figuras 6.2 y 6.3 muestran el laberinto en proceso de formación y completamente graficado, respectivamente.

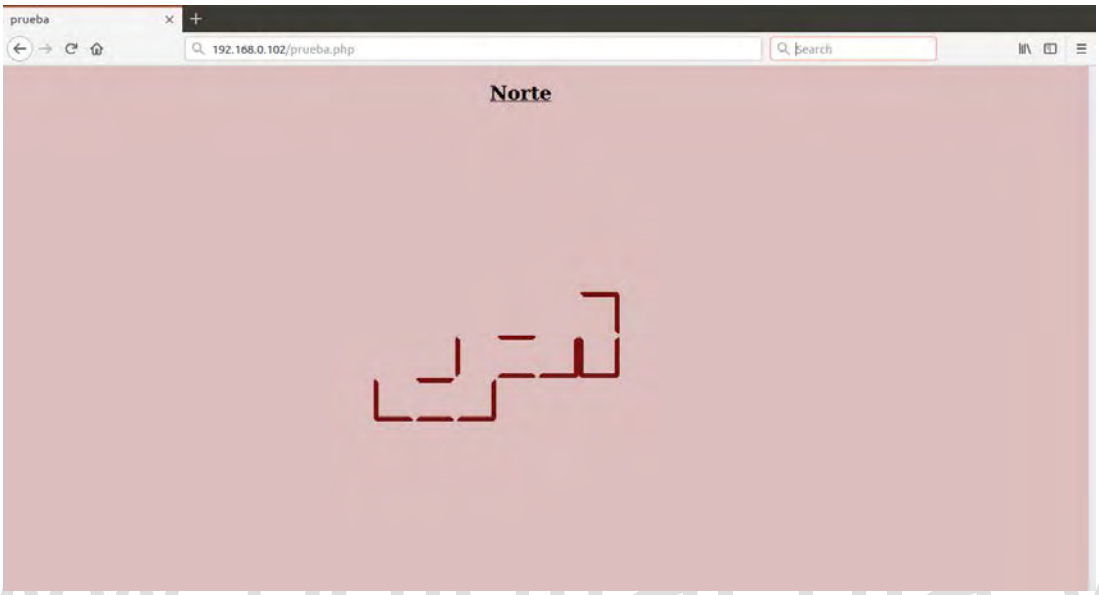

Figura 6.2 Mapeo parcial del laberinto (En proceso de exploración)

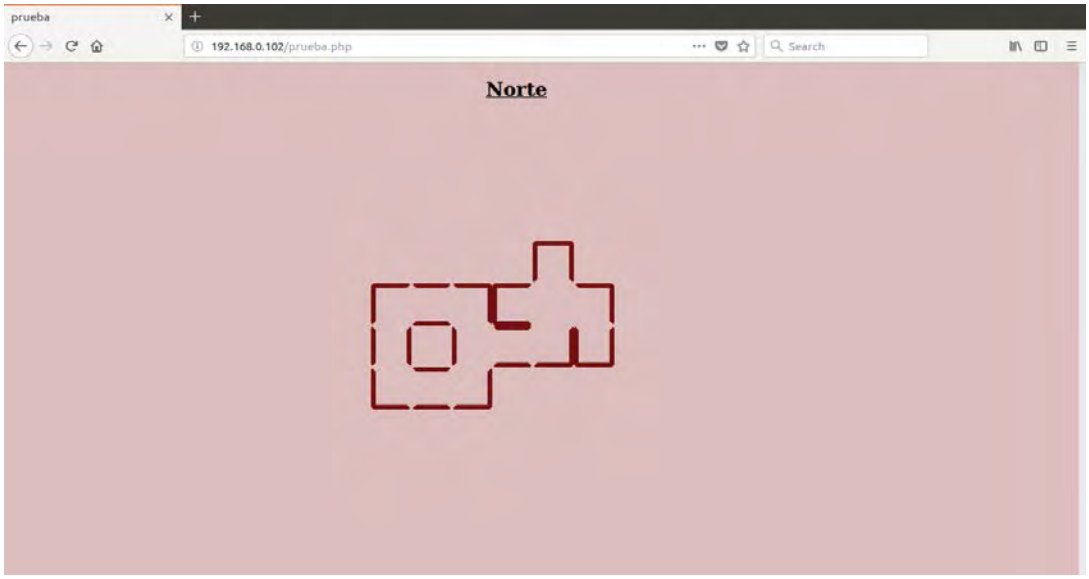

Figura 6.3 Mapeo finalizado del laberinto

Estos resultados demuestran, asimismo, el correcto funcionamiento del sistema de exploración, debido a que si ocurriese algún inconveniente por parte del robot, servidor o ESP8266, no sería posible mostrar correctamente el laberinto explorado.

De la misma manera, se realizaron numerosas pruebas, y se inició el robot desde distintos puntos, por ende, el orden de los nodos es distinto, así como el orden de las decisiones de giro, nodos pendientes y repetidos. Esto ocasiona que el laberinto mostrado por la página web cambie de orientación, a raíz de que el norte cambia. Esto demuestra el acertado funcionamiento del sistema propuesto. En la figura 6.4 se puede observar lo que sucede cuando el robot inicia desde otro punto, y el norte es otro.

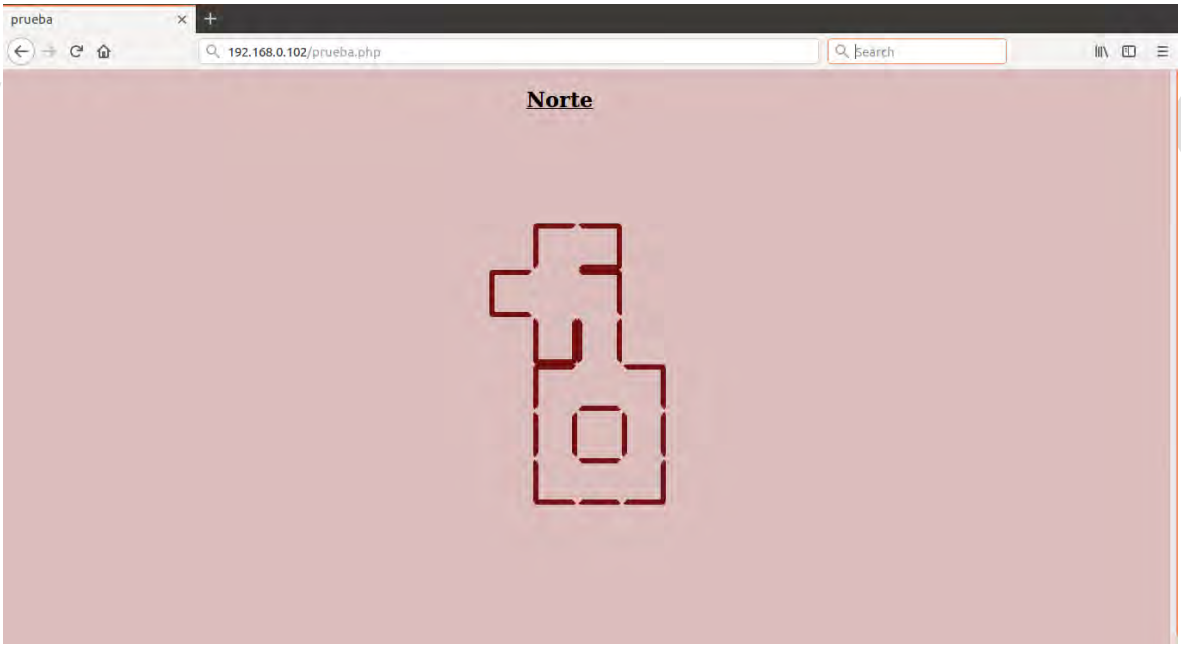

Figura 6.4 Mapeo del laberinto finalizado e iniciado desde otro punto y dirección

Como se observa en la figura 6.4, el laberinto resultante es claramente el mismo, pero las orientaciones son otras y, de la misma manera, el sentido en que se explora y encuentran los nodos.

6.1.3 Problemas de instrumentación

A pesar de los resultados positivos obtenidos, se encontraron ciertos errores de instrumentación al momento de detectar caminos libres y encontrar la orientación actual o, a la cual se desea ir dependiendo del sentido del giro.

El problema al momento de ubicar la correcta orientación, se debe a la alta sensibilidad al ruido que presenta el magnetómetro integrado en el MPU-9250. De las figuras de caracterización del magnetómetro, se aprecia que a pesar de todos los esfuerzo por obtener la medición más limpia posible (calibración más 2 filtros), de todas maneras el magnetómetro muestra presencia de ruido.

Además de esto, todas estas pruebas se realizaron con el MPU-9250 encima de un Arduino Uno, el cual no contaba con la presencia de motores que contiene el 3pi. En cambio, al ser usado en el robot, el MPU-9250 se ve sometido a la alteración del campo magnético producido por los motores DC.

Estos motores DC, como ya se ha mencionado, cuentan con imanes permanentes en el exterior y bobinas electromagnéticas montadas en su eje, las cuales junto al resto de sus partes, producen campos magnéticos que provocan la rotación del motor. Todo esto somete al magnetómetro a un campo magnético que no fue tomado en cuenta al momento de la caracterización y, por ende, existe un ruido de magnitud desconocida que, además, afecta al magnetómetro de una forma que tampoco se conoce.

Aparte del ruido de los dos motores, se tiene la interferencia electromagnética del circuito del propio 3pi, así como el de la placa de expansión y los cables que conectan a estas dos; que como ya se ha observado en figuras anteriores, son cables que sobresalen de la placa de expansión hacia el 3pi, las cuales debido a su longitud y cruce entre ellos, generan pequeños campos electromagnéticos que seguramente son detectados por el AK8963.

Claramente la interacción de todos estos elementos, además de otros factores externos variables, afecta la medición obtenida del MPU-9250, causando que exista un margen de error considerable.

Por esta razón, a pesar del mayor de los esfuerzos para disminuir la ventana de error presente en la corrección de orientación, el robot de todas maneras presenta problemas para encontrarla. Esto causa que el robot tarde en encontrar la orientación, provocando que se quede durante un tiempo considerablemente largo en busca del valor registrado al comienzo de su recorrido, para una dirección dada.

Luego de ser detectada dicha orientación, suele suceder que a pesar del pequeño rango en el cual se considera que la lectura es la correcta, la orientación encontrada presenta un error lo suficientemente grande como para que el robot colisione y no pueda continuar con la exploración.

Este error es eventual y aleatorio, permitiendo que el robot a veces logre finalizar la exploración con éxito. No obstante, sucede lo suficientemente seguido como para que sea un error siempre latente en todas las etapas de la exploración.

Con respecto a los sensores de ultra sonido, se logró filtrar el ruido debido a las uniones de las tablas del laberinto, así como el ajuste de los valores umbral para la detección de caminos libres o paredes en cada nodo. No obstante, existe un error al momento de calcular la distancia recorrida entre dos nodos.

Dicho error provoca que la medición de la distancia recorrida sea menor a la distancia correcta, resultando en un cálculo erróneo en las coordenadas de los nodos, lo cual puede provocar que no se detecten los nodos ya explorados, así como la mala representación del laberinto en la página web. Este error es menos común que el del magnetómetro, pero igual presente en cualquier momento.

A pesar de la presencia de estos errores en la instrumentación, se ha logrado registrar el funcionamiento completo y correcto del robot, lo cual quiere decir que de no funcionar el robot en un momento dado, no es por mala implementación de la solución planteada, ni

porque la solución no sea la correcta, sino por falta de buena instrumentación más resistente al ruido, o mejor adaptada para esta aplicación.

6.2 CÁLCULO Y SEGUIMIENTO DE RUTA

Una vez explorado y mapeado el laberinto, se procede a indicarle al robot cuál será su destino. Los resultados corroboran la buena implementación del algoritmo de búsqueda, así como el uso de la heurística y el cálculo de múltiples rutas.

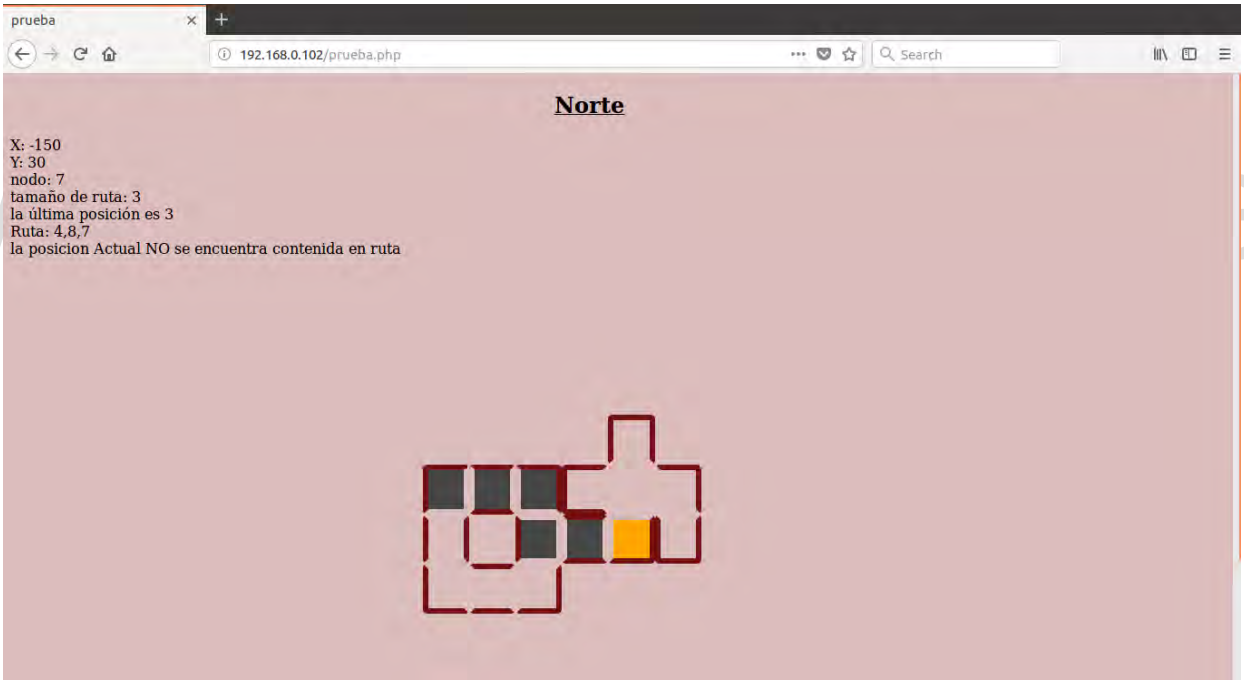

Figura 6.5 Cálculo de la ruta más corta por parte de la página web

Como se observa en la figura 6.5, la página web logra encontrar el camino más óptimo al objetivo, resaltando el punto de inicio, o posición actual en la que se encuentra el robot al momento de indicar el destino.

De la misma manera se va actualizando la posición del robot a medida que éste llega a los nodos contenidos en la ruta hasta llegar al destino, como se logra apreciar en las figuras 6.6, 6.7 y 6.8.

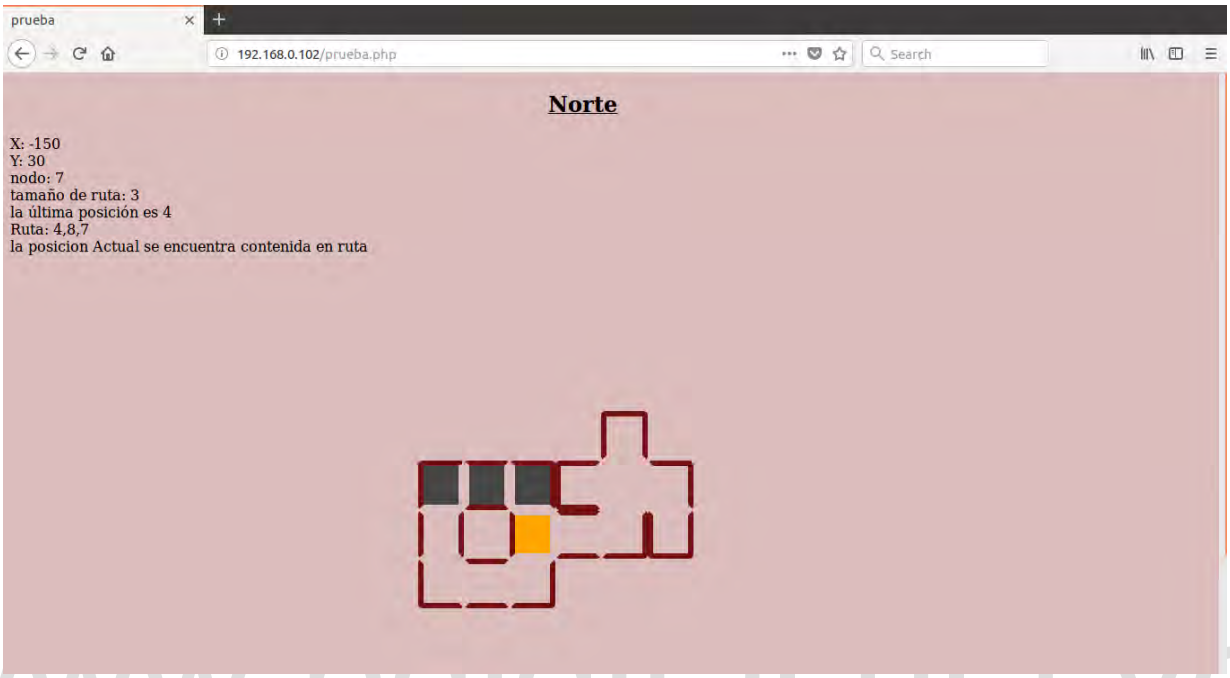

Figura 6.6 Actualización de posición mientras el robot sigue la ruta asignada

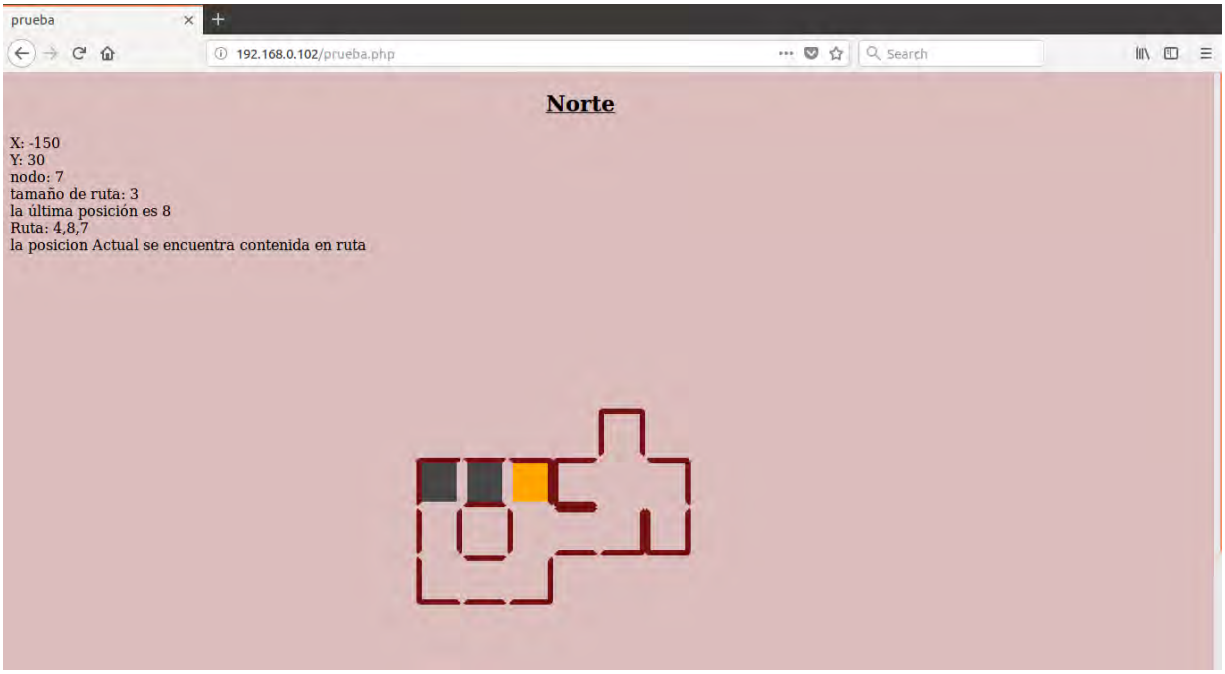

Figura 6.7 Actualización de posición del robot

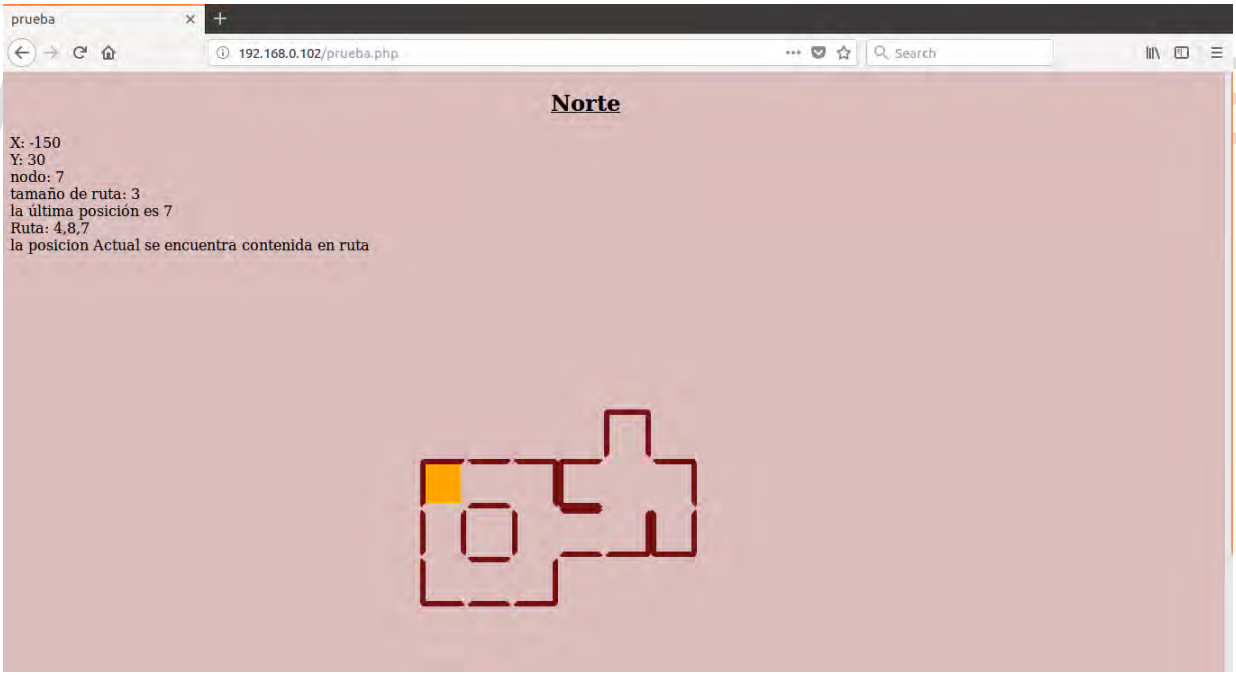

Figura 6.8 Finalización de seguimiento de ruta

Una vez finalizado el seguimiento de la ruta por parte del robot, se puede indicar un nuevo nodo destino para que la página web calcule nuevamente la ruta más corta, como se muestra en la figura 6.9.

Esto es posible debido a la actualización de la posición del robot, ya que el servidor conoce en todo momento en dónde se encuentra el robot, lo que le permite calcular nuevamente otra ruta a partir de la última posición registrada.

De esta manera, el usuario puede indicar otro nodo destino, cuantas veces éste lo desee; el servidor calculará la nueva ruta a partir de la última posición registrada, y el robot solicitará el próximo nodo al cual se debe dirigir.

Es importante mencionar que el problema en el magnetómetro también afecta al robot cuando se hace el seguimiento de la ruta al objetivo dinámico y, por lo tanto, dicho seguimiento también se puede ver afectado por el rendimiento del AK8963. Esto se debe a que el robot depende del magnetómetro al momento de girar y encontrar la orientación deseada, dependiendo de la posición del siguiente nodo en la ruta, con respecto al nodo actual.

Sin embargo, el problema con el cálculo de distancia recorrida ya no se obtiene en esta etapa, debido a que el robot no envía dirección y distancia al servidor, sino directamente el nombre del nodo en el cual se encuentra. Esto es porque el robot ya tiene toda la información de todos los nodos del laberinto y, por ende, ya él conoce en dónde está.

No obstante, a diferencia de los módulos de ultrasonido HC-SR04, el módulo WiFi ESP8266 sigue manteniendo una participación vital en todo momento, ya que sin éste el servidor no sabría si el robot llegó al siguiente nodo y este último tampoco podría recibir el nombre del nodo al cual se debe dirigir; es decir, es de suma importancia en todo momento que el ESP8266 filtre de manera correcta lo que se le envía al servidor.

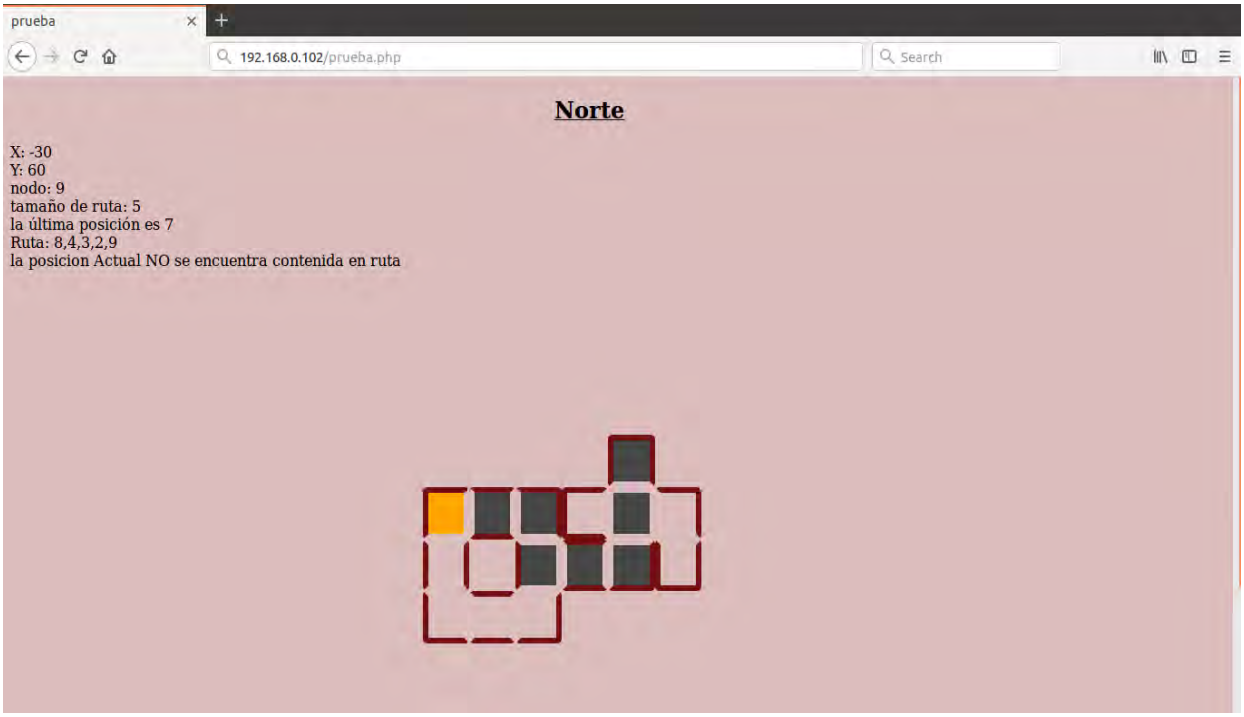

Figura 6.9 Cálculo de otra ruta a partir de la posición actual

CONCLUSIONES

- El uso del IoT permite obtener una gran gama de opciones y ampliar en gran medida el alcance que puede tener cualquier aplicación o proyecto. Se hace increíble poder acceder a los datos producidos por un robot desde cualquier parte del mundo y, además, brindarle al usuario una rica experiencia interactiva llena de posibilidades. El aumento del manejo de memoria, la facilidad para el manejo de distintas bases de datos, las ventajas inherentes de los lenguajes de programación orientados a objetos; estos son sólo algunas de las características más resaltantes al momento de trabajar con el internet de las cosas.
- Se logró el correcto uso y entendimiento de técnicas de búsqueda de la inteligencia artificial e, incluso, la modificación de estas con la finalidad de mejorar los resultados obtenidos.
- Se integraron nuevas tecnologías que son estudiadas hoy en día y que se cree que definirán el futuro de la raza humana.
- Es resaltante la necesidad que se tiene de una instrumentación más adaptada a la navegación de espacios desconocidos, que permita un desenvolvimiento más adecuado y preciso.

RECOMENDACIONES

A pesar de los resultados satisfactorios que han sido obtenidos a partir de las técnicas propuestas, claramente se deben estudiar otros puntos que permitan la mejora de este proyecto. Por estas razones se plantean ciertas recomendaciones a tomar en cuenta, para aquellos interesados en esta rama de la robótica y, de esta manera, obtener robots realmente autónomos que puedan adaptarse a cualquier espacio en el cual se desee que se desenvuelvan:

- Estudiar la exploración de laberintos que no cuenten solamente con paredes que sean perpendiculares o paralelas entre sí.
- Se propone el uso de otro tipo de compás que permita una mejor detección de orientación.
- Se plantea la posibilidad de usar una cámara como instrumentación visual para el robot y, de esta manera explorar y reconstruir de mejor manera el espacio en el cual se encuentra.
- Explorar espacios no basados en laberintos, que sean más parecidos a lugares reales. Esto supone espacios que contengan paredes que no necesariamente son rectas; pueden presentar oscilaciones o irregularidades. De la misma manera puede existir la presencia de otros objetos como sillas o personas.
- Ahondar en el uso de la inteligencia artificial, incluyendo áreas como el procesamiento de lenguaje y aprendizaje de máquinas. Se hace interesante pensar en las posibilidades que se pueden obtener al juntar éstas con el internet de las cosas.

BIBLIOGRAFIA

[1] Pololu corporation, «Robot Pololu 3pi Guia de usuario,» Las Vegas, Nevada, 2001 - 2009.

[2] G. A. R. Robredo, «Electrónica Básica para Ingenieros,» Universidad de cantabria, Santander, 2011.

[3] Atmel, «8-bit AVR Microcontrollers ATmega328/P datasheet sumarry,» San jose, 2016.

[4] HobbyTronics LTD, «ATmega328P Microcontroller - TQFP | Atmelx,» [En linea]. Available: http://www.hobbytronics.co.uk/atmega328-tqfp. [Accesado el 11 Enero de 2018].

[5] Prometec, «Tutoriales Arduino | Prometec,» [En linea]. Available: https://www.prometec.net/. [Accesado el 11 de enero de 2018].

[6] ITEAD Intelligent Systems Co.Ltd., «Ultrasonic Raging Module HC-SR04,» Shenzhen, 2010.

[7] Virtualbotix, «Virtualbotix,» [En linea]. Available: https://www.virtuabotix.com/img_5822/. [Accesado el 12 de Enero de 2018].

[8] Team, Espressif systems IOT, «ESP8266EX datasheet,» Espressif Systems, Shanghai, 2015.

[9] Fiore Basile, «ESP8266 introduction,» Fiore Basile, 2015. [En linea]. Available: http://fabacademy.org/archives/2015/doc/networking-esp8266.html. [Accesado el 13 de Enero de 2018].

[10] InvenSense inc, «MPU-9250 Product Specification,» San Jose, 2016.

[11] Play-zone GmbH, «Play-zone.CH MPU-9250 Acelerometer + gyro + Kompass,» [En linea]. Available: https://www.play-zone.ch/en/mpu-9250-accelerometer-gyro-kompass.html. [Accesado el 13 Enero de 2018].

[12] SparkFun Electronics, «I2C - learn.sparkfun.com,» [En linea]. Available: https://learn.sparkfun.com/tutorials/i2c. [Accesado el 15 de Enero de 2018].

[13] Refsnes Data, «HTTP Methods GET vs POST,» W3.CSS, [En linea]. Available: https://www.w3schools.com/tags/ref_httpmethods.asp. [Accesado el 2 Febrero de 2018].

[14] J. R. Fielding, «Hypertext Transfer Protocol (HTTP): Message Syntax and Routing,» Internet Engineering Task Force (IETF), 2014.

[15] H. Schildt, *Utilización de C en inteligencia artificial,* McGraw-HILL, 1989.