SISTEMA DE NAVEGACIÓN ROBÓTICA MÓVIL CON LÁSER 360°LIDAR PARA PLATAFORMA TURTLEBOT3

MOBILE ROBOTIC NAVIGATION SYSTEM WITH 360°LIDAR LASER FOR TURTLEBOT3 PLATFORM

Santiago Alberto Cortazar Palomeque.[1]*
sacortazarp@correo.udistrital.edu.co

Resumen*:* Los sistemas de navegación robótica cuentan con una gran variedad de algoritmos enfocados en la localización y mapeado simultaneo de objetos (SLAM), cada uno emplea técnicas distintas para generar un mapa y la calidad de este varía en función de la técnica que se emplee, existen tantos métodos documentados por la tendencia actual que está llevando la evolución tecnológica de innovar en los sistemas autónomos para que realicen este tipo de tareas y unas mucho más robustas como el manejo autónomo. Sin embargo, aún queda mucho campo de investigación para la implementación de nuevas técnicas que logren crear un mapa con el porcentaje de error mínimo posible o nulo en su defecto. El presente trabajo tiene como objetivo la implementación de un sistema de navegación robótica móvil con láser haciendo uso de una de las técnicas que existen en el SLAM, en donde la información capturada por el sensor láser LiDAR, tendrán un papel fundamental para la construcción del mapa, esa información se implementara en una aplicación que a partir de las distintas herramientas que ofrece el software libre, como ROS se planteará una GUI para llevar a cabo el control manual y autónomo del robot, además de la visualización en tiempo real de cada acción que esté realizando el sistema en el mapa, por último, se hará evidencia de los resultados de las funcionalidades propuestas para el proyecto realizando una prueba en un ambiente controlado con diferentes obstáculos.

Palabras clave: SLAM, ACML, Interfaz gráfica de usuario (GUI), Mapeo, Navegación.

Abstract: The robotic navigation systems have a wide variety of algorithms focused on the simultaneous location and mapping of objects (SLAM), each employing different techniques to generate a map and the quality of this varies depending on the technique used, there are so many methods documented by the current trend that is leading the technological evolution of autonomous systems to perform such tasks and some much more robust as autonomous management. However, there is still a lot of research to be done in order to implement new techniques that can create a map with the minimum possible error rate or none at all. This work aims to implement a mobile robotic navigation system with laser using one of the techniques that exist in the SLAM, where the information captured by the LiDAR laser sensor, will have a key role in the construction of the map, this information will be incorporated into an application that from the various tools offered by free software, As ROS, a GUI will be proposed to carry out the manual and autonomous control of the robot, in addition to the visualization in real time of each action that the system is performing on the map. Finally, the results of the functionalities proposed for the project will be made evident by carrying out a test in a controlled environment with different obstacles.

Key Words: SLAM, ACML, Graphical User Interface (GUI), Mapping, Navigation.

1. Introducción

Las investigaciones sobre sistemas de navegación para vehículos terrestres, marítimos y aéreos no tripulados ha sido dominio de las organizaciones militares. Hoy en día gracias al contexto tecnológico, la disponibilidad de componentes electrónicos, la propagación de programas informáticos de código abierto y el aumento de la potencia de computación, ha llevado a más organizaciones y empresas a interesarse por el concepto de automatización y robotización, como resultado los sistemas de navegación [1] [2] [3] se han convertido en un tema muy robusto dentro del ámbito de la investigación. La problemática para la cual se plantea el presente proyecto surge de la necesidad de la industria de reforzar sus procesos de producción y organización en una planta, la necesidad de optimizar en lo más posible las tareas siempre será la mayor prioridad, por lo tanto, se plantea una aplicación en la que se pueda crear un mapa de una zona determinada por el usuario, para luego manejar al robot en dicha

zona de manera eficaz monitoreando todo el recorrido y verificando que realice bien la tarea [4]. Al revisar el repositorio institucional de la universidad distrital se encontraron varios proyectos relacionados con los sistemas de navegación, algunos por ejemplo se utilizaban plataformas robóticas como medio para el transporte de material para empleados en la industria [5], otros en aplicaciones parcialmente acuáticas como la implementación de un sistema de navegación para un robot limpiador de piscinas [6] y otro en particular como el proyecto [7] que se enfoca enteramente en la adquisición de datos por un método novedoso como lo es la fusión sensorial. Teniendo un contexto claro en el ámbito de la necesidad industrial y en la gran capacidad investigativa que supone profundizar más en el desarrollo de nuevas tecnologías y algoritmos para robots, el presente proyecto consiste en implementar un sistema de navegación [8], con el láser 360°LiDAR y la plataforma robótica TurtleBot3 haciendo uso de una aplicación GUI en la cual se podrá visualizar todo el proceso de mapeado, control y navegación del robot [9] en tiempo real en un mapa [10]. Todo lo realizado en el proyecto cuenta con la implementación de herramientas libres (código abierto) como Ubuntu, este SO se incorporó por temas de compatibilidad con el resto de herramientas que se usaron, entre ellas está ROS, el cual es un sistema operativo diseñado para la programación en robots, facilita la comunicación entre el hardware y el software, para el diseño de algoritmos y nuevas herramientas con las que se puedan manipular estos sistemas, otra herramienta primordial en el desarrollo del proyecto fue Python que sirvió como base para generar los algoritmos de control y manejo autónomo presentes en el proyecto entre otros, por lo tanto, no se incurrieron en gastos para licencias. En la etapa inicial de desarrollo lo primero que se tuvo en cuenta fue la caracterización del sensor laser LiDAR sus características, su manipulación y como se realiza la modificación de parámetros para su uso ya que es un eje fundamental en el mapeado y la navegación [11]. Ya para las siguientes etapas de desarrollo se tuvo como eje fundamental los procesos relacionados al diseño e implementación del código para la GUI, debido a que a través de la interfaz se comunica todo el proyecto, fue necesario hacer uso de una buena cantidad de paquetes que ROS tiene como rosbridge (permite comunicarse a través de websockets) [12], gracias a ese método de comunicación se disminuyó la dificultad para realizar la implementación en el proyecto, ya en ese punto los pasos siguientes en el proyecto como el frontend (parte de una aplicación que

interactúa con los usuarios) y el backend (parte lógica en el desarrollo de una aplicación) se pudieron desarrollar sin muchas complicaciones.

2. Metodología

Para la realización del presente proyecto, se tuvo como punto de partida los objetivos propuestos y en base a dichos planteamientos, se procedió a realizar el diseño y desarrollo de todos los componentes que conforman el trabajo, se inició con la caracterización e implementación del sistema laser LiDAR [13] montado en el robot identificando de esa manera el impacto que generaría en el proyecto dada la necesidad de utilizar un sensor que pueda capturar todos los datos en un angulo de 360° [14]. Para el mapeo, el manejo y la navegación ubicar los obstáculos es la principal prioridad, si se desea conocer el entorno en que se encuentra algo [15], el sensor LiDAR suple esos requerimientos con creces gracias a sus características que más adelante se describirán. En segunda instancia se plantea el diseño de la interfaz web de usuario en base a ciertos lineamientos y parámetros que se deben realizar en un orden adecuado conforme al proceso que se quiera llevar a cabo [16], se desea que la aplicación sea lo suficientemente intuitiva, para que cualquier persona pueda ejecutar los procesos del robot sin conocer sus funcionamientos, por el enfoque que lleva el proyecto la GUI lleva toda la carga operativa puesto que todo gira en torno a las funcionalidades que provee, el conjunto de nodos y servicios [17] que se encargará de ejecutar y parar requieren de una arquitectura sólida y convencional para que dichas tareas se realicen de la manera más eficaz posible, más adelante se hará énfasis en este proceso y la importancia que tiene.

2.1 Sensor Laser LiDAR

Uno de los ejes fundamentales en el proyecto es la implementación de un sensor de distancia láser (LDS), que para este caso fue específicamente el de Alcance de Luz (LiDAR). Este LDS calcula la diferencia de la longitud de onda cuando la fuente láser es reflejada por el obstáculo si se encuentra como se muestra en la Figura 1 [18]. Para la integración del sensor en el proyecto se utilizaron las

librerías acondicionadas por el fabricante para manipular el LiDAR desde ROS, los datos que el sensor este transmitiendo se pueden visualizar y utilizar por medio del topic (método de conexión de ROS) “/scan”. El LiDAR está ajustado por defecto a trabajar a una frecuencia de 5Hz, Adicionalmente no hubo necesidad de cambiar ningún valor en las propiedades del láser, el angulo de medición está asignado a “6.28318548203 radiánes” equivalentes a 360°, su incremento se establece en 1° (0,0174532923847 rad = 1deg) y las distancias mínimas y máximas de medición son de 0,11 metros y 3,5 metros [19]. Para el tratado de información se tuvo en cuenta el contenido del mensaje que se recibe desde topic “/scan” el cual, devuelve la medición de distancia para cada angulo como un arreglo de “rangos”, esta información es esencial para generar el mapeado de una zona en conjunto con el robot, dado que se utiliza para determinar la posición de este en el mapa y también para ejecutar tareas más complicadas como la navegación en el mapa.

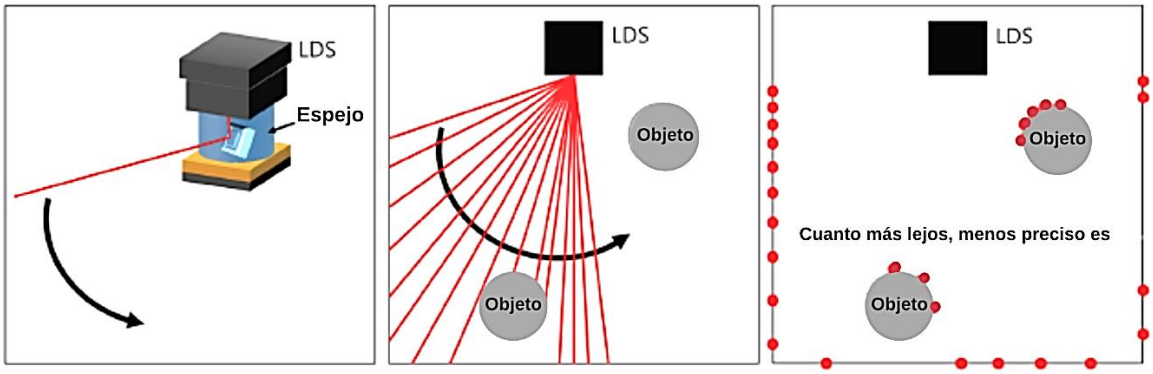

Figura 1. Medición de distancia usando LDS [18]

2.2 Diseño de la interfaz web

El principal enfoque ha sido diseñar e implementar una GUI, en la cual se permita a un usuario inexperto generar un mapa y navegarlo utilizando los algoritmos de generación de mapas y navegación con los que cuenta ROS. Para alcanzar esa solución se debe establecer una distribución para un programa de ejecución que permita gestionar el inicio y la parada de nodos (proceso de computación primario en ROS) necesarios para cada propósito en la aplicación, la comunicación de estas partes se realiza a través de websockets. La estructura que lleva la aplicación se ha desarrollado teniendo en cuenta el

diagrama de flujo expuesto Figura 2, donde queda en evidencia la operación que debe realizar el usuario para ejecutarla eficazmente.

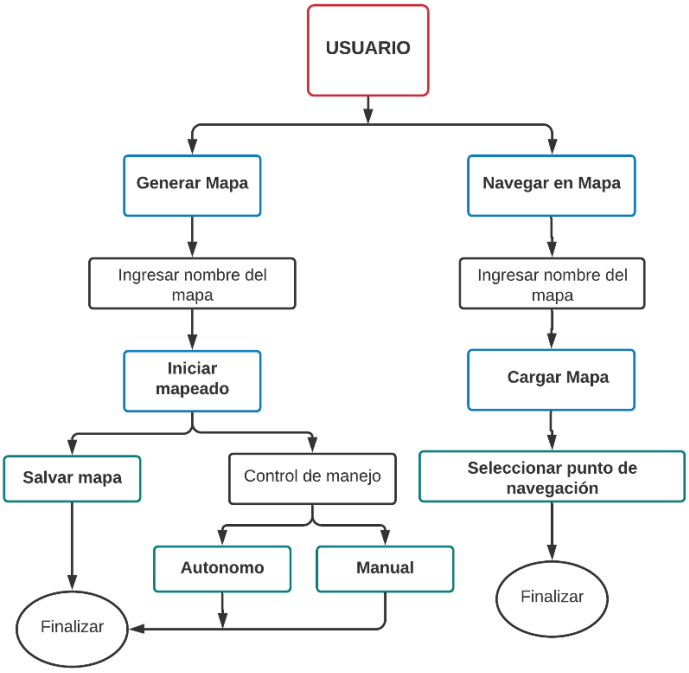

Figura 2. Estructura de uso de la interfaz grafica

El usuario como principal actor para hacer funcionar la aplicación puede elegir entre dos opciones primordiales generar un mapa o navegar en uno previamente creado, lo ideal es que primero se genere el mapa y luego se navegue en él, para aprovechar eficazmente la aplicación. Una característica destacable en el apartado de “generar mapa” se encuentra la opción de manejar el robot ya sea de manera autónoma y manual con el fin de disminuir el uso de la terminal de ROS para ejecutar un nodo que realice esas opciones. Existe una completa colección de paquetes que ofrecen algoritmos de procesado de visión, interpretado de nubes de puntos y localización simultanea y mapeado (SLAM), entre otros. En este caso solo se usaron 4 en específico, Rosbridge, Un paquete de ROS que aporta una capa de abstracción tal como se muestra en la Figura 3 [20], convierte toda la arquitectura ROS en un backend [21].

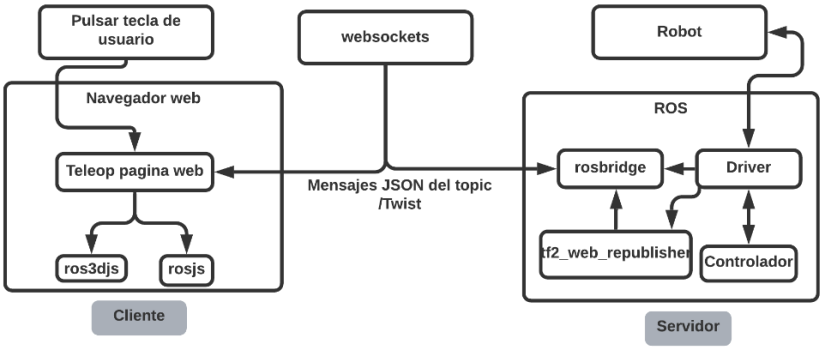

Figura 3. Relaciones de Rosbridge en el sistema [20]

Robot Web Tools, Contiene librerías de visualización web para robots de manera que se puedan desarrollar interfaces robot-humano más usables e intuitivas [22]. Paquete gmapping, proporciona localización simultánea y mapeado basándose en la información de un láser 2D montado sobre el robot. Paquete de navegación, recoge información de la odometría y del sensor láser y envía órdenes de velocidad para mover un robot móvil [23].

3. Implementación y desarrollo

El proyecto se implementó haciendo uso de la plataforma TurtleBot3 en su versión Burger debido a que fue necesaria para el desarrollo del trabajo por las características con las que cuenta, este sistema tiene un paquete de dos actuadores Dynamixel de alto rendimiento, una tarjeta Raspberry Pi3 modelo B que funciona como centro de cómputo, una tarjeta OpenCR como centro de control y una batería de polímero de litio de 11.1 V a 1800mAh como fuente de alimentación como se muestra en la Figura 4 [24], cada uno de esos componentes cumple un rol importante para el correcto funcionamiento de la aplicación y en la ejecución del proyecto en general. La TurtleBot3 es un producto de la empresa Sur Coreana ROBOTIS y son fabricados en colaboración de Open Robotics y ROBOTIS, su venta se realiza a través de distribuidores autorizados a nivel internacional en una gran variedad de precios.

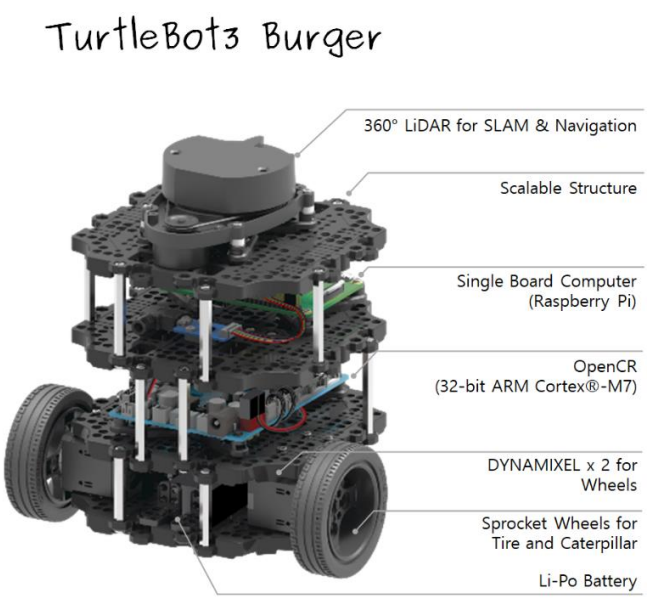

Figura 4. Especificaciones TurtleBot3 Burger [24]

El desarrollo de este proyecto está estructurado en dos fases, una representada como el backend, y la segunda como el frontend. La primera se caracteriza por estar conformada por el conjunto de nodos y servicios de ROS (Características del robot y del láser LiDAR, algoritmos de generación de mapa, navegación y la comunicación entre el server y el nodo de control), y la ejecución en un servidor web. En segundo lugar, para el frontend se optó por utilizar un camino más convencional para la creación de aplicaciones web como lo es HTML, para lograr un acabado más robusto y presentable se utilizó Bootstrap y Javascript para los estilos visuales, también como componente importante se incluyeron las librerías de Robot Web Tools que permiten añadir la visualización de objetos en 3D y las rejillas de navegación en 2D en la interfaz, cada librería tiene como código fuente Javascript, lo cual facilito en gran medida su implementación en el proyecto. En la Figura 5 se puede visualizar con más claridad la estructura planteada previamente.

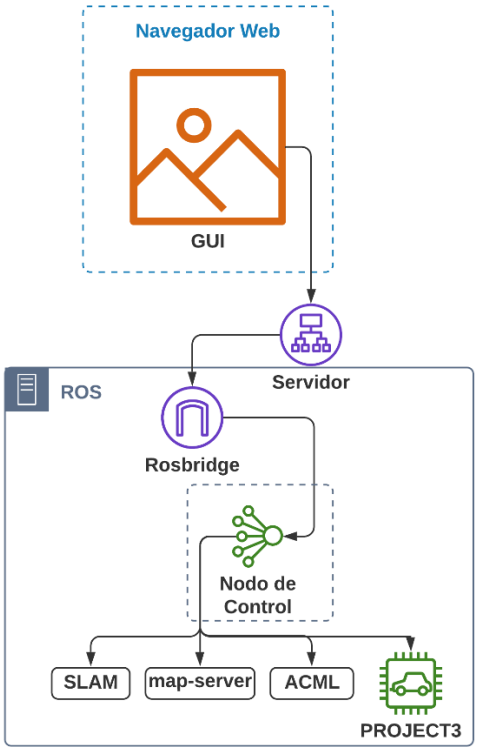

Figura 5. Estructura de desarrollo proyecto

3.1 Backend

Para esta fase de desarrollo se utilizó Ubuntu en su versión 16.04 y ROS en su distribución Kinetic, aunque no están en su versión más actualizada, siguen siendo estables y no presentaron ningún problema en el desarrollo del proyecto. El paquete usado para la creación del proyecto y para el manejo automático se muestra en la Figura 6 y 7 respectivamente.

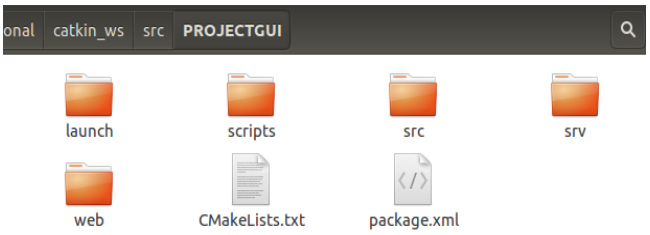

Figura 6. Paquete de desarrollo de la interfaz web

Es necesario destacar la arquitectura de directorios que componen un proyecto en ROS ya que gracias a su independencia entre cada apartado, se pueden incorporar nuevas herramientas que no fueron específicamente diseñadas para trabajar con ROS, de todos los directorios presentes se destacan la

Carpeta Launch**,** ya que contiene los archivos ".launch" que ejecutan, el servidor web, el servidor de rosbridge y los nodos necesarios para el mapeado, la navegación y la manejo automático, la Carpeta scripts**,** porque contiene los archivos ".sh" necesarios para la ejecución del servidor web mini-httpd el cual se encarga de llamar a los archivos y ficheros necesarios para visualizar la interfaz de usuario web desde el navegador.

Figura 7. Paquete de desarrollo nodo manejo automático.

A diferencia del paquete principal del proyecto, este solo aporta la ejecución del nodo del manejo automático sin otra utilidad relevante. De acuerdo a la perspectiva de control sobre el robot que se quería alcanzar, se optó por configurar como ROS_MASTER al PC con Ubuntu 16.04 y al ROS_HOST al TurtleBot3, realizando dicha configuración por medio de sus IP para el puerto WLAN. Teniendo esto en cuenta el nodo de control mencionado previamente ofrece los servicios que se muestran en la Figura 8, estos son necesarios para poder comunicarse con la GUI y ejecutar los nodos que hacen funcionar la aplicación.

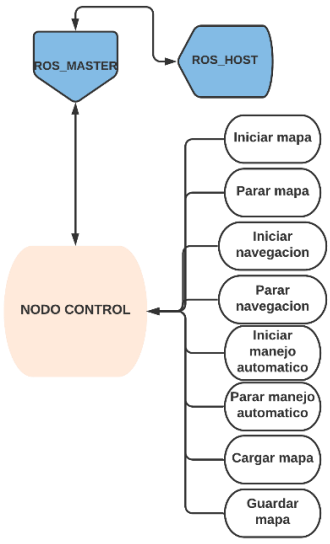

Figura 8. Esquema de comunicaciones con el nodo de control.

Nodo control, es el delegado para agenciar la ejecución y parada de otros nodos de ROS de acuerdo a la operación desde la interfaz web, el lenguaje de programación usado en este nodo es Python, ya que ROS ofrece soporte para el lenguaje gracias a la librería “rospy”. Para la ejecución de los otros nodos se utilizaron las librerías de Python “shlex” y “subprocess”. La librería “shlex” permite hacer análisis léxico de órdenes tipo UNIX y apartarlas en lexemas más simples para después utilizarlas con “subprocess” [23]. También era necesario utilizar librería de funciones de “rosparam” para borrar parámetros del servidor de parámetros de ROS cuando se paran los procesos. La lógica del nodo, es bastante sencilla. Cuando la interfaz web hace una llamada a un determinado servicio, una función de callback se encarga de procesar la petición y lanzar el proceso solicitado. Los servicios que se utilizan en el nodo son de tipo std_srvs/Trigger y PROJECTGUI/SetFilename.

3.2 Frontend

Como se mencionó anteriormente para la implementación de la interfaz de usuario se utilizó Bootstrap y Javascript para dar estilo visual a una estructura de documento HTML. La GUI cuenta con 3 ventanas de navegación index.html, mapping.html y navigation.html:

index.html: En esta ventana se encuentra la portada de la GUI, la cual está compuesta por dos opciones que permiten la navegación a las otras dos ventanas como se muestra en la Figura 9.

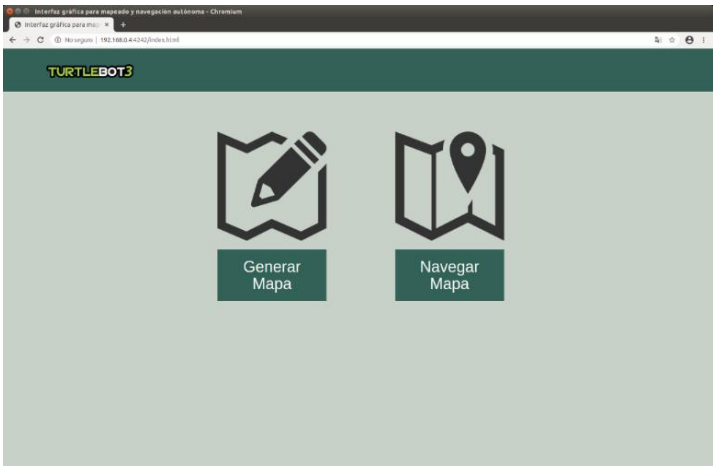

Figura 9. Ventana de portada de la GUI

mapping.html: En esta ventana se recoge la visualización del mapa que se está generando, también se encuentra la principal interacción del usuario con el manejo manual y autónomo del robot, por lo tanto, es la ventana que cuenta con más opciones en la GUI. Para comunicar la interfaz con ROS se utilizó la librería roslib.js de Robot Web Tools, que permite crear una comunicación mediante websockets con rosbridge, en adición para introducir el panel de visualización 3D se utilizó ros3d.js también de Robot Web Tools, que permite visualizar objetos 3D en un plano x, y, z como se muestra en la Figura 10.

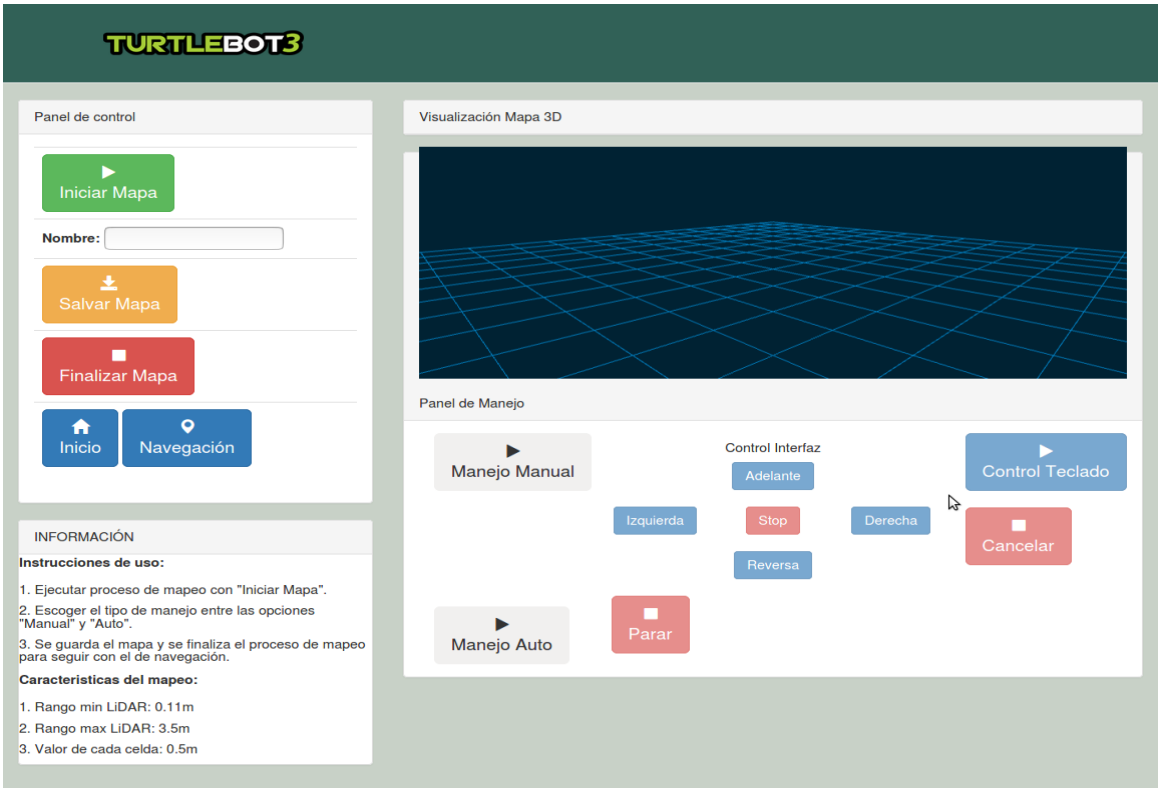

Figura 10. Ventana encargada del mapeo en la interfaz web.

navigation.html: En esta ventana se encuentra el panel de visualización 2D del mapa en el que se van a seleccionar la zona de navegación como tipo de interacción con el usuario y el sistema que llevara a cabo esta tarea. Para la visualización del mapa en 2D se utilizó un widget de la librería de Robot Web Tools, nav2d.js. Además, cuenta con un panel de información para guiar al usuario y darle información sobre la aplicación.

4. Resultados y análisis

Para realizar la prueba de funcionamiento se acondiciono un espacio con un obstáculo en el centro, además de diversos objetos que se encuentran generalmente en la sala de una casa para que el ejercicio de mapeado y navegación fuera lo más realista posible como se muestra en la Figura 11.

Figura 11. Zona de prueba

A continuación, se realizará la ejecución de todo el procedimiento necesario para generar un mapa mientras se visualiza todo el recorrido del robot en el panel de visualización 3D. El primer paso para generar un mapa es la ejecución de todos los nodos de ROS necesarios para iniciar el servidor web, el de rosbridge entre otros en el ROS_MASTER. Una vez iniciados todos estos procesos se debe iniciar un proceso de ROS en la TurtleBot3 la cual tiene el rol de ROS_HOST, esto es necesario para establecer la comunicación entre las dos máquinas y por último se ejecuta el nodo de control de la GUI. Una vez realizado todo el procedimiento anterior se introduce la IP asignada al ROS_MASTER y el puerto asignado en la configuración del servidor web mini-httpd en este caso el puerto es el 4141, luego se escribe la dirección de la siguiente manera en un navegador web que tenga compatibilidad con Bootstrap, “http:192.168.0.4:4141”. En la Figura 9 se puede observar la portada de la interfaz. Luego para iniciar el proceso de generación del mapa se presiona el botón “Generar Mapa” para navegar hacia la ventana de mapeo, se ingresa un nombre al mapa en el campo de “Nombre” y se ejecuta el

inicio del nodo encargado del SLAM con el botón “Iniciar Mapa”, la aplicación notifica con una ventana emergente al usuario cuando se inicia este proceso como se muestra en la Figura 12.

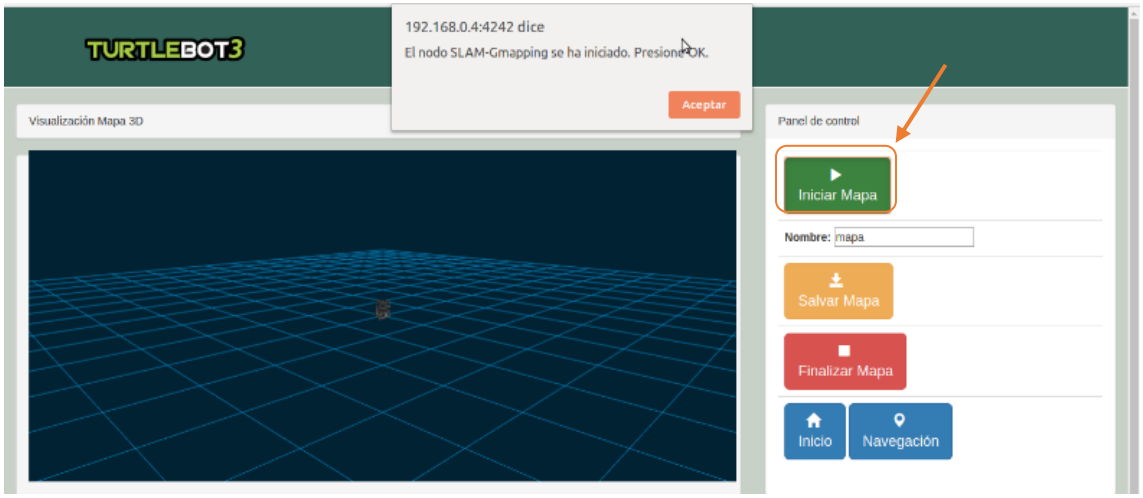

Figura 12. Inicio nodo de mapeo

En este punto se puede acceder al mando de control del robot en donde se encuentran las opciones para manipular los movimientos e indicarle que direcciones debe tomar o que se conduzca solo sin una dirección fija, y evite los obstáculos que se vaya encontrando mientras se llega al mapa deseado, la opción de “Manejo Auto”, ejecuta un nodo con la función de empezar el arranque del robot y si en su recorrido se encuentra con un obstáculo al frente lo esquive y gire hasta encontrar una ruta abierta, la opción “Manejo Manual” habilita la posibilidad de manejar al robot directamente desde la interfaz web o desde el teclado de acuerdo a la preferencia, los indicadores que se encuentran en el panel de manejo permiten manejar al robot desde la interfaz en las direcciones convencionales respectivamente, adelante, izquierda, derecha y reversa, además del stop que frena al robot, para manejarlo desde el teclado se debe presionar la opción “Control Teclado”, esta opción habilita la posibilidad de movilizar al robot en las mismas direcciones convencionales que desde los indicadores de la interfaz, solo que se debe presionar la tecla SHIFT y una de las flechas del teclado de acuerdo a la dirección a la que se quiera ir al mismo tiempo, en adición de la tecla SPACEBAR para frenar al robot, para el caso específico de esta prueba en su mayoría se realizó con la opción de “Manejo Auto” tal como se muestra en la Figura 13, pero para las zonas más estrechas surgió la necesidad de hacerlo manualmente.

Figura 13. Inicio manejo del robot

Sin importar la opción con la que se controle el robot; en el panel de visualización 3D se podrá ver como se mueve y cuanto del mapa se ha creado, la velocidad en que se genera un mapa varía mucho de acuerdo al tamaño y la cantidad de obstáculos que se encuentran en la zona, para este caso en particular tomo entre 5 – 10 minutos mapear todo el espacio visualizado en la Figura 14, Se obtuvo un mapa bien definido en sus bordes y en su semejanza con lo que se tiene en la realidad gracias al rango efectivo del láser y la altura del robot (19.2cm), a pesar de ese resultado hay que tener cuidado en chocarse mucho con los bordes del mapa por que se pueden generar errores en el proceso de mapeo, estos errores ocurren por el algoritmo que se implementó para mapear si se mejora el algoritmo ese tipo de errores no ocurrirán, una vez terminado se procedió a guardar el mapa y terminar el proceso de mapeado.

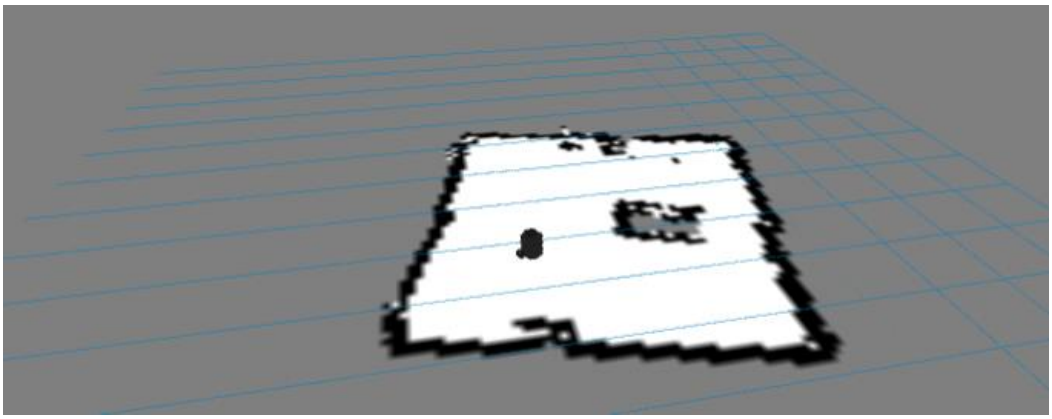

Figura 14. Mapa terminado y guardado

Justo después de la creación del mapa se procedió a empezar las pruebas de los procesos para ejecutar correctamente la opción de ACML en la GUI y de esa manera poder navegar en el mapa creado previamente, se presionó la opción “Navegación” para saltar a la ventana de navegación, seguido de eso se escribió el nombre del mapa y se cargó en el sistema de la interfaz web con el botón “Cargar Mapa”, luego para poder ejecutar todos los nodos de ROS encargados de la navegación se presionó el botón “Iniciar Navegación”.

Para la prueba de navegación se seleccionó un punto en el mapa para que el robot en tiempo real navegara desde su posición actual hasta el punto B en la zona de pruebas de manera rápida sin chocarse con ningún objeto que se encontrara tal como se muestra en la Figura 15 y 16 respectivamente.

Figura 15. Ruta de navegación seleccionada

Figura 16. Ruta de navegación

El resultado fue muy satisfactorio el robot llego hasta las ruta indicada sin ningún inconveniente ni retraso en el recorrido tal como se muestra en las Figuras 17, esta tarea tomo 1 minuto para este caso, el tiempo varia si el punto se encuentra entre muchos obstáculos o si es una esquina del mapa, también tenemos el caso en el que el robot no ejecute el proceso de navegación por que se seleccionó un punto fuera del mapa creado y por lo tanto seria algorítmicamente imposible para el robot desplazarse hasta dicho punto si no se conoce la zona.

Figura 17. Ejecución exitosa de la navegación a partir de la GUI

A continuación, se mostrarán resultados de pruebas similares a la propuesta anteriormente en otro tipo de escenarios las variables que se tuvieron en cuenta son la distancia desde un punto A hasta un punto B y el tiempo que se demoró el robot en realizar el recorrido, estos datos se muestran en la Tabla 1.

Tabla 1. Pruebas recorrido

Distancia (metros)	Tiempo (segundos)
0.2	5
0.45	20
1	30
1.2	50
2	130

Los datos presentados en la Tabla 1 demuestran que mientras más distancia exista entre el punto A y el punto B más tiempo le tomara al robot recorrer dicha distancia. Esto ocurre porque el algoritmo que se implementó para llevar a cabo esta tarea realiza muchas verificaciones del terreno si la distancia es muy extensa, se realizó de esta manera por los errores futuros que podría ocasionar un usuario al momento de ejecutar la opción de navegación y escoger mal un punto en el mapa para que se movilice el robot. Estas verificaciones eran necesarias si se quería que la aplicación fuera viable para implementarla en otro tipo de entorno.

5. Conclusiones

- ✓ El proyecto cumplió su principal objetivo y fin de implementación, navegar en una zona previamente creada en base a una opción de mapeo, integrando herramientas novedosas que todavía no se hubieran utilizado en la facultad Tecnológica de la universidad Distrital Francisco José de Caldas como lo son el sensor laser 360°LiDAR y la plataforma TurtleBot3.
- ✓ La interfaz gráfica de usuario ofrece la manipulación especifica de la plataforma TurtleBot3, la visualización de la zona que se está mapeando, está muy bien conseguida, también de la zona por la que se navega, si bien existe cierto retardo en la transmisión de información, este es lo suficientemente corto para asemejarse a la realidad, sin embargo, si se utiliza en una zona lejana al punto de acceso a la red, pueden ocurrir errores en la visualización.
- ✓ La implementación del mando a distancia en la aplicación cumplió con su rol de permitir al usuario controlar inalámbricamente el manejo del robot desde la interfaz o desde el teclado según se requiera, además de que se integró una opción más como el manejo auto, el cual es muy útil si se desea realizar otra actividad mientras se está mapeando una zona.
- ✓ El sensor laser LiDAR, resulto ser indispensable en cada apartado del proyecto tanto en la navegación como en el mapeo fue necesario implementarlo, sin embargo, si no se configura bien, se podrán evidenciar muchos errores en su funcionamiento como la presentación de datos erróneos en el mapa. Para el caso de su implementación en el trabajo no se evidenciaron estos problemas, ya que se configuración bien.
- ✓ La arquitectura de los paquetes que se crean en ROS para hacer aplicaciones deja en evidencia una independencia entre estructuras de desarrollo que representa una gran ventaja para crear aplicaciones como esta, todo gracias a los beneficios que ofrece ROS, sin embargo, se debe ejecutar una gran cantidad de procesos en segundo plano para iniciar solo con la captura de la información del láser y el robot.

Referencias

[1] Martínez F., Jacinto E., Montiel H., “Neuronal Environmental Pattern Recognizer: Optical-by-Distance LSTM Model for Recognition of Navigation Patterns in Unknown Environments. In: Tan Y., Shi Y. (eds) Data Mining and Big Data”. Communications in Computer and Information Science, vol 1071. Springer, Singapore, 2019. https://doi.org/10.1007/978-981-32-9563-6_23.

[2] Martínez F. H. & Montiel, H., “HYBRID FREE-OBSTACLE PATH PLANNING ALGORITHM USING IMAGE PROCESSING AND GEOMETRIC TECHNIQUES”. In ARPN Journal of Engineering and Applied Sciences, vol 14 (18), 3135 – 3139, 2019.

[3] Montiel H., Jacinto E. & Martínez F. H., “Generación de Ruta Óptima para Robots Móviles a Partir de Segmentación de Imágenes”. *Información tecnológica*, *26*(2), 145-152, 2015. https://dx.doi.org/10.4067/S0718-07642015000200017.

[4] K. Riaño Tolosa and J. Diaz Rios, "Sistema de visión artificial para la optimización por algoritmo de colonia de hormigas de trayectorias de taladrado de circuitos impresos (PCB) a partir de procesamiento digital de imágenes". Universidad Distrital Francisco José de Caldas. Facultad Tecnológica. 2019

[5] S. Velandia Arenas and A. Buitrago Anzola, "Robots cooperativos y coordinados para el transporte de materiales largos de forma autónoma en un entorno controlado". Universidad Distrital Francisco José de Caldas. Facultad Tecnológica. 2019

[6] L. Cardona Rendón, P. A. Ortiz Valencia, and J. S. Botero Valencia, “Sistema de navegación para un robot limpiador de piscinas,” *Rev. Tecnura*, vol. 18, no. 39, p. 22, Dec. 2013, doi: 10.14483/udistrital.jour.tecnura.2014.1.a02.

[7] “Sistema de navegación para robots móviles utilizando fusión sensorial,” *Tecnura*, vol. 13, no. 25, pp. 128–135, 2009, doi: 10.14483/22487638.6675.

[8] P. Lopez Torres, “Análisis De Algoritmos Para Localización Y Mapeado Simultáneo De Objetos,”. Universidad de Sevilla. Departamento de Ingeniería de Sistemas y Automática. Máster en Ingeniería Industrial. p. 51, 2016.

[9] A. Garcia Blanco, “IMPLEMENTACIÓN EN UNA PLATAFORMA MÓVIL DE LA TÉCNICA SLAM MEDIANTE CONTROL INALÁMBRICO,” *Zaguan.Unizar.Es*, vol. 27, pp. 1–100, 2019.

[10] F. Gallego Valcarcél, "Desarrollo y construcción de un robot móvil que simule el comportamiento individual básico de un arquero de fútbol". Universidad Distrital Francisco José de Caldas. Facultad Tecnológica. 2018

[11] F. Roda Herrero, "Módulos software para un sistema de asistencia a la movilidad basado en entorno ROS (Robot Operating System)". Universidad de Alcalá. Escuela Politécnica Superior. 2018

[12] A. Luzardo Alliey, "Diseño de la interfaz gráfica web en función de los dispositivos moviles. Caso de estudio: diarios digitales", *Fido.palermo.edu*, 2020. [Online]. Available: https://fido.palermo.edu/servicios_dyc/publicacionesdc/cuadernos/detalle_articulo.php?id_libro=321&id_articulo=6997. [Accessed: 18- Aug- 2020]

[13] D. D. Pham and Y. S. Suh, “Pedestrian navigation using foot-mounted inertial sensor and LIDAR,” *Sensors (Switzerland)*, vol. 16, no. 1, Jan. 2016, doi: 10.3390/s16010120.

[14] P. Hyde, R. Dubayah, W. Walker, J. B. Blair, M. Hofton, and C. Hunsaker, “Mapping forest structure for wildlife habitat analysis using multi-sensor (LiDAR, SAR/InSAR, ETM+, Quickbird) synergy,” *Remote Sens. Environ.*, vol. 102, no. 1–2, pp. 63–73, May 2006, doi: 10.1016/j.rse.2006.01.021.

[15] Morcillo Martínez, A. F. L., and Miguel, “Sistema de detección de obstáculos para drones basado en sensor láser,” Universitat Politècnica de València, Valencia, 2018.

[16] J. Almeida, H. Fernandes, V. Filipe, and J. Barroso, “Web platform architecture to support the Geographic Information System of the University of Trás-os-Montes and Alto Douro Campus,” in *Proceedings - 2009 International Conference on New Trends in Information and Service Science, NISS 2009*, 2009, pp. 1112–1117, doi: 10.1109/NISS.2009.177.

[17] A. Rodriguez Rodriguez and M. Fernàndez Muñoz, “WebGuide: Aplicació Web (HTML5) i Asterisk utilitzant VoIP (WebGuide: HTML5 and Asterisk application using VoIP),” Universitat Politècnica De Catalunya.

[18] Y. Pyo, H. Cho, L. Jung and D. Lim, *ROS Robot Programming (English)*, 1st ed. ROBOTIS, 2017, p. 487.

[19] L. Galtarossa. "*Obstacle Avoidance Algorithms for Autonomous Navigation System in Unstructured Indoor Areas".* Politecnico di Torino, Corso di laurea magistrale in Mechatronic Engineering (Ingegneria Meccatronica), 2018

[20] C. Crick, G. Jay, S. Osentoski and O. Jenkins, "ROS and Rosbridge", *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction - HRI '12*, 2012. Available: 10.1145/2157689.2157846 [Accessed 18 August 2020].

[21] “Teleoperating and visualizing a robot on a web browser - ROS Robotics Projects.” [Online]. Available: https://subscription.packtpub.com/book/hardware_and_creative/9781783554713/12/ch12lvl1sec106/teleoperating-and-visualizing-a-robot-on-a-web-browser. [Accessed: 29-Jul-2020].

[22] R. Toris *et al.*, “Robot Web Tools: Efficient messaging for cloud robotics,” in *IEEE International Conference on Intelligent Robots and Systems*, 2015, vol. 2015-December, pp. 4530–4537, doi: 10.1109/IROS.2015.7354021.

[23] J. Rapado García, "Diseño e implementación de una interfaz gráfica de usuario para mapeado de entornos y navegación en Ros". Universitat Politècnica de València. Departamento de

Informática de Sistemas y Computadores. 2016.

[24] ROBOTIS. "TurtleBot3 E-Manual". Especificaciones de hardware. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/#hardware-specifications. [Accessed: 20 August 2020].