ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE LOCALIZACIÓN Y MAPEO SIMULTÁNEOS (SLAM) PARA LA PLATAFORMA ROBÓTICA ROBOTINO®

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y CONTROL

NARVÁEZ TIPANTAXI VICTOR JULIO
victor_narvaez_t@hotmail.com

YANDÚN NARVÁEZ FRANCISCO JAVIER
fynsk8@yahoo.es

DIRECTOR:

DR. ANDRÉS ROSALES ACOSTA

androsaco@gmail.com

Quito, JUNIO 2013

DECLARACIÓN

Nosotros, Narváez Tipantaxi Víctor Julio y Yandún Narváez Francisco Javier, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Narváez Tipantaxi Víctor Julio	**Yandún Narváez Francisco Javier**

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Narváez Tipantaxi Víctor Julio y Yandún Narváez Francisco Javier, bajo mi supervisión.

Dr. Andrés Rosales Acosta

DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Agradezco a Dios, por darme la vida y siempre colmarme de bendiciones.

A la Virgen de El Quinche por ampararme y cuidarme a lo largo de los estudios.

A mi hermosa familia porque a pesar de tantos sinsabores nunca dejó de creer en mí.

A la Escuela Politécnica Nacional y a sus profesores, por todos los conocimientos adquiridos, vivencias y reflexiones que he logrado en mi estancia tanto como estudiante, al igual que como educador.

A la Universidad Técnica Federico Santa María, y al Departamento de Electrónica, acertadamente encabezado por el Dr. Mario Salgado, gracias por abrir las puertas y brindar todas las facilidades para así poder ampliar mis horizontes tanto investigativos como personales, pues gracias a su apoyo se pudo realizar una fructífera estancia de investigación en Chile.

Al Dr. Andrés Rosales porque además de guiarnos durante el transcurso de este proyecto, ha mostrado que un profesor además de su labor docente, puede brindar una sincera amistad.

Al Dr. Fernando Auat Cheeín, quien fue una vital guía para la consecución de esta meta, gracias por transmitir sus coocimientos, sabiduría y brindar su amistad durante la estancia en Chile, ya que así se logro sentir incluso un aire de familia.

Al Msc. David Pozo porque con su ayuda, experiencia y amistad, se logró culminar con éxito esta etapa; gracias por su amistad y apoyo.

A mis amigos, UNIVERSITARIOS F.C., JUNIORS DE BOCA, y LA ÚLTIMA Y NOS VAMOS, porque me han demostrado que la amistad sincera es un bien muy preciado, y que cada vez que los he necesitado han acudido a mi rescate, gracias por tener la oportunidad de compartir irrepetibles momentos tanto de tristeza como de felicidad.

A Dora, mi gran amiga, por estar presente cuando más lo he necesitado, por su amistad sincera y apoyo incondicional, te cu eme a secas.

A mi amigo, mi pana y mi hermano Pancho, porque sin su apoyo y dedicación, las cosas se hubieran tornado imposibles, por mostrarme el valor de la amistad, y que ésta se pone de manifiesto no solo en instantes de alegría sino en los momentos más difíciles, Gracias por el honor de ser tu compañero.

A Evelyn, por estar pendiente de mí, a pesar de cualquier circunstancia.

Víctor Narváez T.

DEDICATORIA

A mis tres mamás:

María del Carmen, por ser mi fortaleza, mi inspiración y el aliento para seguir adelante, por tus desvelos y sacrificio para que llegue a ser alguien en la vida, este triunfo es para ti, TE AMO MADRE.

Sofía, porque me has enseñado a nunca rendirme, por ser mi confidente, por apoyarme cuando las cosas se ponían color de hormiga.

Adriana, por siempre estar pendiente de mí, por cuidarme y protegerme, confiar siempre en mí y ser más que una hermana, una madre y amiga.

Ñañas las amo, y nunca tendré palabras suficientes para poder expresar todo lo que siento, a su esfuerzo y sacrificio le debo casi todo lo que soy.

A Julio, mi padre, por ser desde siempre mi referente, por enseñarme a expresar sin temor alguno y a usar el don de la palabra, siempre me sentiré orgulloso de ser tu hijo, TE AMO PAPÁ.

A mi pequeño Emilio, por ser mi nueva razón para seguir luchando, todo lo que haga siempre es pensando en tu futuro, eres una bendición en mi vida, y siempre luchare por tu bienestar, TE AMO HIJO MIO.

A mis sobrinos Martín y Macarena, por llegar a nuestras vidas para alegrarnos, y sepan que siempre tendrán en mi el respaldo y cariño que necesiten.

A todas las personas que de una u otra manera han aportado con su granito de arena para que este logro sea posible, quizás por espacio y tiempo no son nombrados pero están en mi corazón.

VICO.

AGRADECIMIENTO

A la vida, que me ha permitido transitar su camino y con ello cumplir esta meta, que me ha enseñado a apreciar los pequeños detalles del día a día y que me ha mostrado que a pesar de las circunstancias siempre se puede ser feliz.

A mis padres quienes con su ejemplo han forjado un ser humano que cada día desea ser mejor persona y los ama con todo su corazón.

A Edison, mi hermano, quien ha sido mi compañero y amigo desde siempre, con quien compartí muchas de las aventuras de mi niñez y adolescencia.

A Sonia y Ximena, mis hermanas quienes siempre me han apoyado desinteresadamente con entusiasmo y cariño.

A mi abuelita, a Rosi y a Nani quienes hicieron las veces de mis segundas madres, que cuidaron de mi cuando niño y a quienes debo gran parte de mi vida.

Al Dr. Andrés Rosales, director de este proyecto, quien nos brinda su apoyo incondicional, nos motiva para salir adelante, que con su ejemplo de lucha para sacar adelante a la Institución nos ha mostrado el valor de la perseverancia y quien además de un profesor, es un amigo.

Al Dr. Mario Salgado, y por su intermedio a la Universidad Técnica Federico Santa María, que nos permitieron realizar la estancia de investigación en Chile, prestando las facilidades necesarias para realizar la investigación en la Universidad.

Al Dr. Fernando Auat, quien nos recibió con los brazos abiertos en una tierra extraña y se convirtió en un amigo, quien nos supo guiar en varias partes del proyecto de una manera desinteresada y para quien guardo mis más sinceros sentimientos de gratitud, aprecio, admiración y respeto.

A Víctor, con quien compartimos muchas aventuras en nuestra vida universitaria, éste proyecto como una de ellas, lo hicimos amigo.

A Dora, una persona especial e importante, compañera de lucha durante todos estos años, a quien amo con todo mi corazón y con quien espero compartir el camino de la vida muchísimos años más.

A mis amigos los Juniors, un grupo de personas con quienes compartimos buenos y malos momentos y, sobretodo una pasión llamada fútbol, en mí siempre tendrán a un amigo incondicional.

A David Pozo, quien nos brindó su ayuda desinteresada en varios pasajes del proyecto, su colaboración fue de mucha importancia, gracias por todo.

A la Señora Mercedes, muchas gracias por su aprecio, apoyo y consideración.

A la Escuela Politécnica Nacional, y a todos sus docentes, quienes me han dado las herramientas necesarias para desenvolverme en el ámbito laboral.

A todas las personas quienes han compartido conmigo mi paso por la Institución y que han hecho de ello algo maravilloso e inolvidable.

Francisco

DEDICATORIA

A mi madre, una eterna luchadora, quien siempre ha confiado en mí y su fortaleza me ha inspirado para afrontar los obstáculos que se me presenten. Te amo mamá.

A mi padre, compañero de aventuras, quien me instruyó en muchos aspectos de la vida, tu ejemplo siempre estará presente en mi mente y tu recuerdo en mi corazón. Siempre te extrañaré viejo amigo. Te amo papá.

Francisco

CONTENIDO

RESUMEN

Dentro de la robótica móvil, el desarrollo de técnicas para la autonomía y navegación es esencial para la consecución de varias tareas, tales como desempeñarse en un entorno desconocido y brindar información de éste sin la necesidad de la intervención del ser humano.

La continua investigación en esta rama de la robótica, ha mostrado grandes soluciones para problemas cotidianos acerca de la navegación autónoma, tales como la exploración de varios lugares peligrosos, inhóspitos, o inaccesibles así como gran cantidad de tareas repetitivas y molestas, que pueden ser llevadas a cabo por robots móviles teniendo en cuenta las diversas técnicas y algoritmos desarrollados por la comunidad científica.

La navegación autónoma plantea los problemas de localización, y desconocimiento del entorno en el que se desplaza el robot móvil, por lo cual el presente trabajo busca solucionar esto mediante un mapeo en dos dimensiones que se logra con los valores que brinda el sensor láser utilizado, el cual devuelve lecturas de distancias, las cuales con un tratamiento de datos adecuado se convierten en un mapa del entorno.

Adicionalmente los sensores con los que cuenta la plataforma robótica Robotino® tales como encoders, brindan información acerca de la posición del móvil con respecto a un eje de coordenadas dispuesto con anterioridad, el cual cabe destacar, está sometido a gran cantidad de error, por lo que esto se resuelve a través de un Filtro Extendido de Kalman, que es una de las principales herramientas estudiadas y analizadas en este proyecto de titulación.

Para la implementación del Filtro Extendido de Kalman se ha considerado estimar la posición del robot móvil en el entorno con el modelo cinemático omnidireccional, el mismo que corresponde a un conjunto de ecuaciones matemáticas que describen el movimiento del robot.

Adicionalmente, en este trabajo se destaca la estancia de investigación que se llevó a cabo durante aproximadamente 30 días, en las instalaciones de la Universidad Técnica Federico Santa María, ubicada en Valparaíso-Chile. Esta

etapa de desarrollo estuvo bajo la acertada dirección del Dr. Fernando Auat Cheeín, gran referente y experto en estudios de Robótica Móvil, Localización y Mapeo Simultáneos (SLAM), quien dio luces acerca del proyecto aquí mostrado. Se ha considerado asimismo sus trabajos como se citan oportunamente en las referencias seguidas; además de la plataforma Robótica Robotino®, se realizaron varias pruebas y experimentaciones con el robot móvil de la marca MobileRobots cuyo modelo es el Pioneer 3-AT, en el aspecto de simulación.

Los entornos que se han manejado para evaluar la efectividad del algoritmo, son entornos estructurados, elaborados en un software especializado; éstos han sido probados en simulación, y en cuanto a ambientes reales se escogieron aquellos cuyas formas sean regulares como son pasillos largos y que contengan esquinas para ser detectadas.

El sistema realiza un mapa del entorno y muestra la trayectoria realizada por el robot móvil, y al final de su ejecución genera un gráfico con las líneas y esquinas encontradas, identificándose con colores distintos a los del mapa.

Se han cumplido a cabalidad los objetivos y el alcance planteado en el Plan del Proyecto de Titulación, y como se ha mencionado anteriormente, también se han realizado actividades adicionales que fortalecieron este proyecto.

PRESENTACIÓN

En este trabajo se realiza el estudio del problema del la localización y mapeo de un robot móvil, y de las diferentes soluciones desarrolladas para responder ante estos inconvenientes, como la Localización y Mapeo Simultáneos (SLAM), y algunos de los diferentes métodos de navegación que se han ido utilizando a lo largo de la historia.

En el Capítulo I, se realiza una introducción de la historia de la Robótica como tal, haciendo énfasis en la Robótica Móvil, sus inicios y continuo desarrollo hasta llegar a nuestros días, donde empiezan a plantearse nuevos retos y a identificarse nuevas interrogantes, llegando a la propuesta del SLAM. Se brinda un preliminar de los orígenes de esta temática; también se aborda lo concerniente a las plataformas robóticas utilizadas para las pruebas y experimentaciones, como son: Robotino® de Festo y Pioneer 3-AT de MobileRobots, con sus respectivas descripciones físicas, esto es sensores, actuadores y procesadores, y asimismo se pone en consideración los modelos cinemáticos omnidireccional y tipo uniciclo, respectivamente, los cuales describen sus movimientos.

En el Capítulo II, se realiza un análisis de los tipos de sensores existentes como son los propioceptivos y los exteroceptivos; asimismo los diferentes métodos de navegación como el uso de cámaras, sonar, y por último, el método utilizado en este trabajo como, es el barrido láser, y el estudio de las características que brinda el sensor láser usado como es el Hokuyo URG-04LX-UG01; otro tema estudiado en este capítulo es la utilización de la probabilidad para resolver situaciones existentes en la Robótica; se menciona la estimación probabilística de Cadenas de Markov como método de solución a las incertidumbres propias de la navegación de robots móviles, hasta llegar al tratamiento del algoritmo del Filtro Extendido de Kalman, sus ecuaciones tanto de predicción como de corrección y su posible implementación como solución al problema de la localización, para finalmente realizar una introducción al algoritmo de SLAM.

El Capítulo III, se realiza la descripción del algoritmo utilizado para el mapeo y la detección de características (esquinas) en el entorno, así como las técnicas usadas para el tratamiento de los datos devueltos por el sensor láser, el desarrollo del Filtro Extendido de Kalman, y todo el coste computacional que esto implica, como la validación del modelo cinemático de la plataforma robótica móvil utilizada; la transformada de Hough, su uso e implementación, y el proceso de detección de esquinas mediante el análisis de cruce entre todas y cada una de las líneas detectadas en el mapa elaborado.

El Capítulo IV, contiene una recopilación de los diferentes resultados obtenidos a partir de experimentaciones y pruebas realizadas tanto en forma virtual (simulación) como de manera real, colocando a la plataforma robótica móvil en un entorno estructurado, a fin de obtener un mapa y lograr la detección de líneas y esquinas; las gráficas existentes serán una comparación entre lo que se obtiene sin ningún proceso preliminar y lo que se logra aplicando el Filtro Extendido de Kalman.

En el Capitulo V se tienen las conclusiones y recomendaciones que durante la elaboración y desarrollo de este Proyecto de Titulación han surgido, las cuales además servirán como guía para futuros trabajos en el área que se ha tratado.

CAPÍTULO 1

INTRODUCCIÓN Y MARCO TEÓRICO

1.1 INTRODUCCIÓN

Desde la antigüedad el ser humano se interesó en la creación de “aparatos” a su semejanza, y que además lo releven y/o ayuden en la ejecución de un trabajo, fue así que se crearon distintos tipos de robots como son: manipuladores, móviles, androides, omnidireccionales, etc. Este desarrollo ha sido logrado a través de largos periodos de investigación, pruebas y análisis, además de vencer un sinnúmero de dificultades y limitantes debido a presupuesto, desarrollo tecnológico y el hecho de que la robótica requiere de un trabajo interdisciplinario entre varias ramas de la ingeniería, dicha problemática se fue solucionando con el paso de los años y el avance de la ciencia.

Cada tipo de robot requiere de un determinado estudio, puesto que, si bien es cierto existen similitudes en el desarrollo de ingeniería, también es cierto que surgen problemas específicos en cada robot. Al centrarse en los robots móviles, los problemas pasaron desde cómo lograr un movimiento controlado, hasta cómo hacer que se mueva sin chocarse o caerse, situaciones solucionadas mediante el uso de diversos tipos de sensores como los que se dispone en Robotino®, por ejemplo sensores: infrarrojos, ópticos, inductivos, detector de colisiones, etc.

En la actualidad los problemas se centran en dotar a los robots móviles de una inteligencia artificial capaz de reconstruir el entorno donde se mueven y además, darles la capacidad de localizarse dentro de un entorno desconocido, haciéndolo de manera simultánea, y a la vez, reduciendo el coste computacional que esto implica.

1.1.1 HISTORIA DE LA ROBÓTICA

En 1920 el escritor checo Karel Capek, usa por primera vez el término “Robot”, en su obra dramática “Rossum's Universal Robots / R.U.R.”, derivado de la palabra checa Robota, que significa servidumbre o trabajo forzado.

Posteriormente el escritor ruso de ciencia ficción, Isaac Asimov, acuñó el término Robótica como la ciencia que estudia a los robots. [1] Asimov, estableció, además las tres leyes de la robótica, que aparecieron por primera vez en el relato “Runaround”, y son las siguientes:

1. Un robot no puede hacer daño a un ser humano o, por inacción, permitir que un ser humano sufra daño.
2. Un robot debe obedecer las órdenes dadas por los seres humanos, excepto si estas órdenes entrasen en conflicto con la Primera Ley.
3. Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la Primera o la Segunda Ley. [2]

Con el advenimento de la tecnología, una de estas leyes antes enunciadas ha sido modificada para pasar a denominarse la “Ley Cero de la Robótica”, esta ley fue descrita por Asimov en la novela *Robots e Imperio*. Enunciándose de la siguiente manera:

“*Un robot no puede causar daño a la humanidad, o por inacción, permitir que la humanidad sufra daño."*, quedando las tres leyes de la robótica subordinadas jerárquicamente a esta nueva ley.[3]

En la TABLA 1.1 se expone de manera breve, el desarrollo de la robótica y los robots más sobresalientes de cada época.

TABLA 1.1 Evolución de la Robótica, Tomado de [1]

Fecha	Importancia	Nombre del Robot	Inventor
Siglo I a. C.	Descripciones de más de 100 máquinas y autómatas, incluyendo un artefacto con fuego, un órgano de viento, una máquina operada mediante una moneda, una máquina de vapor.	Autónoma	Ctesibio de Alexandria, Filón de Bizancio, Herón de Alexandria, y otros
1206	Primer robot humanoide programable	Barco con cuatro músicos robotizados	Al Jazarí
c. 1495	Diseño de un robot humanoide	Caballero mecánico	Leonardo da Vinci

1738	Pato mecánico capaz de comer, agitar sus alas y excretar.	Digesting Duck	Jacques de Vaucanson
1800s	Juguetes mecánicos japoneses que sirven té, disparan flechas y pintan.	Juguetes *Karakuri*	Hisashige Tanaka
1921	Aparece el primer autómata de ficción llamado "robot", aparece en *R.U.R.*	Rossum's Universal Robots	Karel Čapek
1930s	Se exhibe un robot humanoide en la Exposición Universal entre los años 1939 y 1940	Elektro	Westinghouse Electric Corporation
1948	Exhibición de un robot con comportamiento biológico simple	Elsie y Elmer	William Grey Walter
1956	Primer robot comercial, de la compañía Unimation fundada por George Devol y Joseph Engelberger, basada en una patente de Devol	Unimate	George Devol
1961	Se instala el primer robot industrial	Unimate	George Devol
1963	Primer robot "palletizing"	Palletizer	Fuji Yusoki Kogyo
1973	Primer robot con seis ejes electromecánicos	Famulus	KUKA Robot Group
1975	Brazo manipulador programable universal, un producto de Unimation	PUMA	Victor Scheinman
2000	Robot Humanoide capaz de desplazarse de forma bípeda e interactuar con las personas	ASIMO	Honda Motor Co. Ltd
2005	Robot que puede aprender a caminar en 20 minutos. Este modelo de robot es uno de los primeros robots en utilizar un programa de aprendizaje y es el primero en andar sin tener información previamente implantada en sus controles.	Toddler	MIT
2010	Robot Humanoide capaz de caminar de forma bípeda, bailar, cambiar la expresión de la cara y reconocer la voz	HRP-4C	Instituto Nacional de la Ciencia y Tecnología Industrial Avanzada
2012	Robot gigante controlado desde dentro, su tamaño es de 4m y pesa 4500kg, sus controles son paneles touch, pudiendo también ser controlado remotamente.	Kuratas	Suidobashi Heavy Industry

1.1.1.1 **Historia del SLAM**

A lo largo de la evolución de la Robótica, si bien se han desarrollado nuevos sensores y actuadores para los robots, también se han planteado nuevos retos que consisten en otorgar a los robots la capacidad de actuar o navegar de manera autónoma, por lo cual se necesita de sistemas inteligentes que les den la capacidad de aprender de su entorno y desenvolverse por sí mismos.

En los años 80, la comunidad científica, comienza a estudiar los temas de localización y mapeo de manera independiente, abordándolos como dos temas separados, hasta que surgió la pregunta ¿Se pueden fusionar estos dos temas para dotar de una mayor autonomía al robot?, con lo cual el objetivo cambió a fusionar la localización y el mapeo para que el robot mientras se desplace pueda construir un mapa de su entorno, y al mismo tiempo use las características encontradas en el mapa para localizarse. Esto presentó grandes dificultades, debido a errores presentes en las medidas de los sensores y debido al modelo del robot.

El origen del estudio probabilístico del SLAM ocurrió en 1986 en una conferencia de la IEEE[1] acerca de Robótica y Automatización, llevada a cabo en San Francisco, California. En esa época apenas se estaba abordando el tema de la estimación probabilística para la Robótica y la Inteligencia Artificial. En esa conferencia, varios investigadores que trabajaban en resolver problemas de localización y mapeo, como Peter Cheeseman, Jim Crowley y Hugh Durrant-Whyte [4], dieron las pautas que años más tarde permitirían obtener "*papers claves en la formulación de una base estadística para describir relaciones entre características del mapa y manipular la incertidumbre en la geometría del mapa*". [4]

Desde entonces se han desarrollado varios algoritmos basado en filtros probabilísticos para dar una solución práctica al problema de SLAM, unos más complicados y elaborados que otros, pero que arrojan buenos resultados si se aplican en los entornos para los que fueron diseñados, puesto que el algoritmo no

[1] IEEE: Institute of Electrical and Electronics Engineers

va a ser el mismo si se trabaja en un entorno cerrado y estructurado o si requiere una tarea bajo el mar por ejemplo.

En las próximas secciones se abordará el problema de SLAM, algunas de las técnicas usadas para resolverlo, su complejidad y el algoritmo para desarrollarlas.

1.1.2 CLASIFICACIÓN DE LOS ROBOTS

Dependiendo de la forma de agrupar a los diferentes tipos de robots que existen, se pueden tener dos grandes clasificaciones: Según su cronología y según su arquitectura.

1.1.2.1 **Según su cronología[5]**

1.1.2.1.1 Primera Generación.

Realizaban tareas repetitivas pre-programadas que se debían ejecutar secuencialmente, adicionalmente el control utilizado para este tipo de robots era en lazo abierto, razón por la cual pasaban por alto las perturbaciones o los errores producidos por el entorno.

1.1.2.1.2 Segunda Generación.

Ya se tiene un control en lazo cerrado, considerando las variaciones y perturbaciones provocadas por el entorno y detectadas por sensores, que brindan información necesaria para tomar medidas correctivas.

1.1.2.1.3 Tercera Generación.

Son robots que tienen la capacidad de adaptarse o reprogramarse de acuerdo a los cambios detectados por los sensores teniendo un comportamiento autónomo.

1.1.2.1.4 Cuarta Generación.

Son sistemas robotizados que apuntan hacia la toma autónoma de decisiones y a resolver problemas por sí mismos; a esta ingeniería se le conoce como Inteligencia Artificial.

1.1.2.2 **Según su arquitectura**

1.1.2.2.1 Poliarticulados

Este tipo de robots tiene pocos grados de libertad, su principal aplicación se denota en el campo industrial, en el cual se necesita abarcar un espacio extenso de trabajo, abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o reducir el espacio ocupado en el suelo. [6]

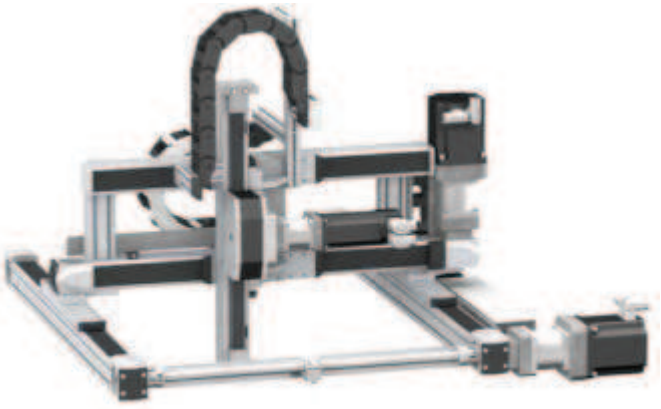


Figura 1.1 Robot Cartesiano Lexium, tomado de [7]

1.1.2.2.2 Móviles

Poseen un sistema de locomoción que les proporciona un gran rango de desplazamiento, sus diseños son basados en carros o plataformas de tipo rodante. Su trayectoria puede ser definida por telemando o ser generada de acuerdo a la información obtenida a través de los sensores que los conforman. Su utilización dentro de una cadena de producción radica en el trasporte de piezas de un lugar a otro, su recorrido puede ser seguido a través de un campo electromagnético con circuitos colocados previamente en el piso, o sino con

tecnología fotoeléctrica sorteándose obstáculos y memorizando caminos, además se observa un tipo de robots móviles que siguen una trayectoria definida previamente en un programa cargado en su cerebro. [6]

FIGURA 1.2: ROBOT MOVIL NISSAN BR23C, Tomado de [8]

1.1.2.2.3 Androides

Su aspecto es casi humano, intentando reproducir su comportamiento y respuesta ante situaciones cotidianas, su utilidad primordial radica en la investigación y experimentación, debido a las dificultades que presentan sobretodo en el ámbito de la locomoción bípeda, tanto así que es necesario un amplio control simultáneo entre la parte dinámica y el equilibrio en tiempo real del robot.

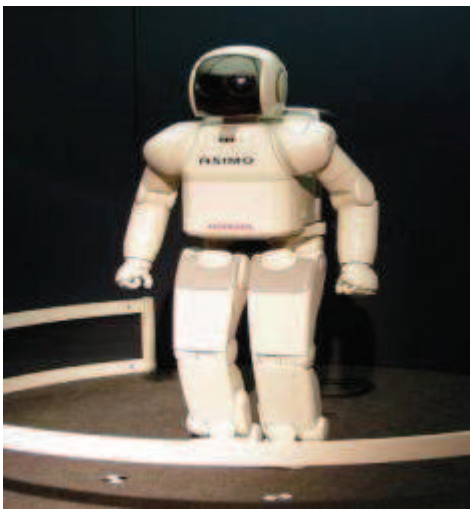


FIGURA 1.3: Robot Androide ASIMO-HONDA, Tomado de [9]

1.1.2.2.4 Zoomórficos

Los robots zoomórficos, que considerados en sentido no restrictivo podrían incluir también a los androides, constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos. A pesar de la disparidad morfológica de sus posibles sistemas de locomoción es conveniente agrupar a los Robots Zoomórficos en dos categorías principales: caminadores y no caminadores.

El grupo de los robots zoomórficos no caminadores está muy poco evolucionado. Hay experimentos efectuados en Japón basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación.

Los robots zoomórficos caminadores multípedos son muy numerosos y están siendo experimentados en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrestres, siendo estos piloteados o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos robots serán interesantes en el campo de la exploración espacial y en el estudio de los volcanes, asimismo en ambientes totalmente hostiles para la presencia de los seres humanos, como también en misiones de rescate.

FIGURA 1.4 Robot Hexápodo, tomado de [10]

1.1.3 ESTRUCTURA DE LOS ROBOTS

1.1.3.1 **Sensores**

Una parte importante de un robot son los sensores, los cuáles, análogamente a los sentidos humanos, permiten a las máquinas adquirir datos de su entorno para que posteriormente sean procesados. Existe una amplia variedad de sensores cuya utilidad depende de la aplicación o del robot en el que se va a trabajar, es así que existen sensores tanto con salida analógica como con salida digital. [11]

Para que un robot tenga conocimiento de su entorno, debe ser capaz de medir algunas magnitudes físicas básicas, tal como se muestra en la TABLA 1.2.

TABLA 1.2 Algunos tipos de sensores, Tomado de [11]

Tipos de Sensores	
Variable Medida	**Sensores**
Luz	Elementos sensibles LDRs o Fotorresistores Fotoceldas o celdas fotovoltaicas Fotodiodos Fototransistores Módulos integrados Reflectivos De ranura
Presión y Fuerza	Microinterruptores Sensores de Presión Sensores de Fuerza Piel robótica Sensores de contacto
Sonido	Micrófonos Captadores piezoeléctricos Rangers Ultrasónicos
Distancia	Medidores ultrasónicos Módulos Integrados Medidores Infrarrojos
Posición	Acelerómetros Giroscopios Sensores pendulares Contactos de Mercurio
Temperatura	Diodos Termistores RTD’s Termocuplas Pirosensores
Humedad	Sensores capacitivos Sensores resistivos Módulos Integrados

1.1.3.2 **Actuadores**

Los actuadores son los elementos llamados a recibir las órdenes provenientes de los controladores para posteriormente traducirlas en un movimiento. Los actuadores que generalmente se usan en la Robótica pueden ser neumáticos, hidráulicos o eléctricos. Al momento de considerar un actuador para un sistema robótico se debe tener en cuenta ciertos puntos:

- Potencia
- Controlabilidad
- Peso
- Precisión
- Velocidad
- Mantenimiento
- Costo

A la hora de dar prioridad a un ítem, es necesario tener en cuenta el robot con el que se está trabajando y la aplicación que se le va a dar al mismo, así por ejemplo, en un robot industrial se va a darle prioridad a potencia, mantenimiento, controlabilidad y costo, por otra parte en un robot móvil (como el que se usa en el presente trabajo) se puede darle prioridad a la controlabilidad, velocidad y precisión.

A continuación se describe a breves rasgos los tipos de actuadores con los que se puede trabajar en robótica:

- Actuadores Neumáticos

Con este tipo de actuadores se gana en fuerza, puesto que el aire comprimido permite generar una gran presión variando las áreas donde se aplica cierta presión (Principio de Pascal), pero se puede perder en parte el posicionamiento y un control fino no es posible, es así que son adecuados para un posicionamiento todo o nada. Por ejemplo, son utilizados en manipuladores sencillos, en apertura y cierre de pinzas o en determinadas articulaciones de algún robot (como el movimiento vertical del tercer grado de libertad de algunos robots tipo SCARA[2]). [12]

[2] SCARA: Selective Compliance Assembly Robot Arm

Las ventajas de este tipo de actuadores es que se puede trabajar en ambientes explosivos como una buena opción ante los elementos eléctricos.

- Actuadores Hidráulicos

Este tipo de actuadores son similares que los neumáticos, con la diferencia que el fluido que ocasiona el movimiento es un líquido (aceite generalmente), solo que las presiones a las que se trabaja son mucho mayores, se puede entender así que se usan para generar fuerzas mayores que en el caso de los actuadores neumáticos.

- Actuadores Eléctricos

Son los más usados, por sus características versátiles de control, tamaño y precisión. Dentro de esta categoría se tienen los actuadores más usados como son los motores DC, motores a pasos, servomotores, etc. Cada actuador tiene su utilidad específica, es así como un servomotor, por ejemplo, sirve para un posicionamiento preciso con un buen torque, mientras que los motores DC se usan generalmente para la locomoción de robots móviles.

1.1.3.3 **Autonomía**

Si bien existen robots que siguen órdenes o secuencias de forma casi perfecta, la evolución de la Robótica apunta hacia la creación de robots que se puedan desenvolver de una manera autónoma, teniendo la capacidad de tomar decisiones, seguir ordenes e incluso aprender de situaciones pasadas, para conseguir este comportamiento se han desarrollado técnicas de control avanzadas, tales como redes neuronales, control robusto, control difuso, etc., coadyuvadas con el desarrollo de sensores más precisos. Cabe anotar que con el aumento de la autonomía de un robot aumenta el costo de investigación y también las dificultades en la tarea del control.

Pues bien, buscando un robot autónomo los científicos se han topado, entre otros, con el problema de la localización y mapeo simultáneos, el mismo que es abordado en el presente trabajo, descrito y detallado en capítulos posteriores.

Cuando se habla de autonomía en Robótica es siempre necesario hablar de Inteligencia Artificial, que es la disciplina que se encarga de construir procesos que al ser ejecutados sobre una arquitectura física producen acciones o resultados que maximizan una medida de rendimiento determinada, basándose en la secuencia de entradas percibidas y en el conocimiento almacenado en tal arquitectura.[13]

Básicamente esta ciencia pretende simular o recrear el comportamiento del cerebro humano, con el fin de crear una máquina inteligente.

1.2 PLATAFORMAS UTILIZADAS

1.2.1 PLATAFORMA ROBÓTICA ROBOTINO®

1.2.1.1 **Características Principales**

La plataforma robótica Robotino® ha sido diseñada por Festo Didactic[3], siendo una herramienta capaz de enseñar de una forma pedagógica el entorno y manejo de un robot en el ámbito industrial, teniendo como objetivo mostrar el siguiente contenido didáctico:

• Mecánica

– Construcción mecánica de un sistema de robot móvil

• Electricidad

– Control de unidades de accionamiento

– Cableado correcto de componentes eléctricos

• Sensores

– Control de recorrido guiado por sensores

– Control de recorrido libre de colisiones con sensores de distancia

– Control de recorrido por procesamiento de imágenes de webcam

[3] www.festo.com

• Sistemas de control por realimentación

– Control de accionamientos omnidireccionales

• Utilización de interfaces de comunicación

– Redes LAN inalámbricas

• Puesta a punto

– Puesta a punto de un sistema robot móvil

Además cabe recalcar que esta plataforma robótica ha sido diseñada para un manejo en condiciones seguras y en busca de beneficios en la investigación, estudio y desarrollo de técnicas afines al robot, es por esto que se debe considerar lo siguiente:

Robotino® ha sido desarrollado según los más recientes desarrollos tecnológicos, cumpliendo con normas de seguridad reconocidas. Sin embargo, cuando se utiliza el sistema siempre puede existir el riesgo de lesiones físicas graves para el usuario o para terceras partes, o de daños causados a las máquinas u otros activos materiales. [14]

Robotino® debe utilizarse solamente para el fin que está previsto y en condiciones absolutamente seguras. [14]

1.2.1.2 **Accionamiento**

Robotino® es una plataforma móvil con accionamiento omnidireccional, puede desplazarse en cualquier dirección sin necesidad de girar cambiando de orientación, cuenta con tres unidades de accionamiento omnidireccional permitiéndole movimientos en todas las direcciones: adelante, atrás, y lateralmente, considerando que además puede girar sobre su propio eje (manteniendo o sin mantener su orientación); ello significa que dispone de tres grados de libertad, dos de traslación y uno de rotación.

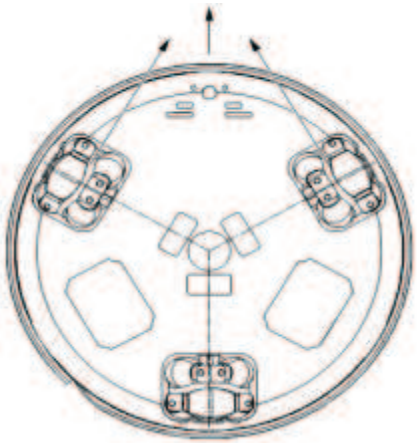


FIGURA 1.5 Sentido del Desplazamiento del Accionamiento Omnidireccional Robotino®, Tomado de [15]

1.2.1.2.1 Ventajas y Desventajas del Accionamiento Omnidireccional

La principal ventaja radica en que al variar la velocidad de los motores la plataforma puede desplazarse en cualquier sentido sin necesidad de tener que cambiar su orientación.

Un vehículo con esta característica de accionamiento puede girar sobre sí mismo sin desplazarse, es decir su radio de giro es cero, razón por la cual no implicaría efectuar maniobras para girar.

El peso de la plataforma descansa sobre las tres ruedas aumentando el equilibrio del sistema.

En cuanto al consumo de energía se estima una desventaja, debido a que existen muchas superficies de rodamiento, incrementándose las perdidas por fricción y aumentando la resistencia a la rodadura.

1.2.1.2.2 Accionamiento de un Actuador Omnidireccional

En el caso de un actuador omnidireccional se utilizan tres motores de accionamiento, se considera además la existencia de tres comandos para el accionamiento de cada motor (avanzar, retroceder, desconexión), así al tratarse de tres motores existirán 27 posibilidades en total (3^3 posibilidades), permitiendo que la plataforma móvil pueda desplazarse en varios sentidos.[15]

En la TABLA 1.3 se muestran las 27 posibilidades que se obtienen mediante los comandos:

TABLA 1.3 Posibilidades de Movimiento de Robotino®, tomado de [15]

MOTOR 1	MOTOR 2	MOTOR 3	SENTIDO DEL MOVIMIENTO
Avanzar	Avanzar	Avanzar	Giro en sentido horario sobre el propio eje central
Avanzar	Avanzar	Desconectar	Giro en sentido horario con radio pequeño
Avanzar	Avanzar	Retroceder	Giro en sentido horario con radio grande
Avanzar	Desconectar	Avanzar	Giro en sentido horario con radio pequeño
Avanzar	Desconectar	Desconectar	Giro en sentido horario con radio mediano
Avanzar	Desconectar	Retroceder	Círculo en 300°
Avanzar	Retroceder	Avanzar	Giro en sentido horario con radio grande
Avanzar	Retroceder	Desconectar	Movimiento en 0°
Avanzar	Retroceder	Retroceder	Giro en sentido antihorario con radio grande
Desconectar	Avanzar	Avanzar	Giro en sentido horario con radio pequeño
Desconectar	Avanzar	Desconectar	Giro en sentido horario con radio mediano
Desconectar	Avanzar	Retroceder	Círculo en 240°
Desconectar	Desconectar	Avanzar	Giro en sentido horario con radio mediano
Desconectar	Desconectar	Desconectar	Detenido
Desconectar	Desconectar	Retroceder	Giro en sentido antihorario con radio mediano
Desconectar	Retroceder	Avanzar	Círculo en 60°
Desconectar	Retroceder	Desconectar	Giro en sentido antihorario con radio mediano
Desconectar	Retroceder	Retroceder	Giro en sentido antihorario con radio pequeño
Retroceder	Avanzar	Avanzar	Giro en sentido horario con radio grande
Retroceder	Avanzar	Desconectar	Círculo en 180°
Retroceder	Avanzar	Retroceder	Giro en sentido antihorario con radio grande
Retroceder	Desconectar	Avanzar	Círculo en 120°
Retroceder	Desconectar	Desconectar	Giro en sentido antihorario con radio mediano
Retroceder	Desconectar	Retroceder	Giro en sentido antihorario con radio pequeño
Retroceder	Retroceder	Avanzar	Giro en sentido antihorario con radio grande
Retroceder	Retroceder	Desconectar	Giro en sentido antihorario con radio pequeño
Retroceder	Retroceder	Retroceder	Giro en sentido antihorario sobre el propio eje central

1.2.1.3 **Controlador**

El sistema puede ponerse en marcha inmediatamente sin necesidad de conectarlo a un PC.

El controlador de Robotino® consiste en un PC embebido con una tarjeta compact flash, en la cual se han instalado varias aplicaciones de demostración y el sistema operativo (Linux). Las aplicaciones de demostración pueden ejecutarse

directamente desde el panel de control del Robotino®. Para el caso específico de este proyecto de titulación se ha utilizado una tarjeta CF de 4 GB, debido a que la aplicación que se va a desarrollar requiere de un masivo espacio de almacenamiento y una versión de software actualizada, cuyo manejo se tratará en capítulos posteriores.

Robotino® puede programarse con el software Robotino View® en un PC a través del LAN inalámbrico. Robotino View® es capaz de transmitir señales al controlador del motor, así como visualizar, cambiar y evaluar valores de los sensores, además el robot puede programarse con Robotino View® incluso durante el funcionamiento real.

También están disponibles APIs, Linux y C++ para la programación del Robotino®. Cabe destacar que además existe la posibilidad de programar a Robotino® a través de LabVIEW mediante VIs y SubVIs, y que en el presente trabajo se utiliza el paquete computacional MATLAB, con el cual se usan tanto bloques de Simulink con cada uno de los sensores, actuadores y periféricos disponibles, como también mediante el manejo de M-files, con los que se logró mayor versatilidad en cuanto al desarrollo de programas de aplicación, siendo esta la forma preferida para el avance del presente proyecto.

Se dispone de una webcam que permite visualizar y evaluar una imagen de cámara en vivo con ayuda del Robotino®View. Con ello, pueden implementarse aplicaciones tales como el trazado de rutas y seguimiento de objetos. Numerosos sensores, una cámara y un controlador de altas prestaciones aportan al sistema la necesaria “inteligencia”.

Puede accederse al controlador directamente a través de la LAN inalámbrica (WLAN). Cuanto está correctamente programado, Robotino® realiza de forma autónoma las tareas asignadas.

Pueden conectarse actuadores y sensores adicionales a través de una interfaz de

E/S. [16]

1.2.1.4 **Sensores**

Robotino® ha incorporado varios tipos de sensores con el objetivo de medir distancias y detectar la velocidad a la que gira cada motor siendo éstos de tipo infrarrojo, ultrasónico y adicionalmente dispone de un sensor anticolisión montado alrededor del chasis en forma de circunferencia, el cual indica el contacto con objetos.

1.2.1.4.1 Detectores de Rayos Infrarrojos

Robotino® está equipado con nueve sensores de medición de distancia por infrarrojos, dichos sensores se encuentran distribuidos en el chasis generando un ángulo de 40° entre ellos, haciendo posible la detección de objetos en zonas próximas.

Cada uno los mencionados sensores, puede brindar una respuesta a través de la placa de circuito de E/S. Haciendo posible que se eviten obstáculos, que la plataforma móvil se mantenga a distancia prudente y tome medidas correctivas considerando un determinado objetivo.

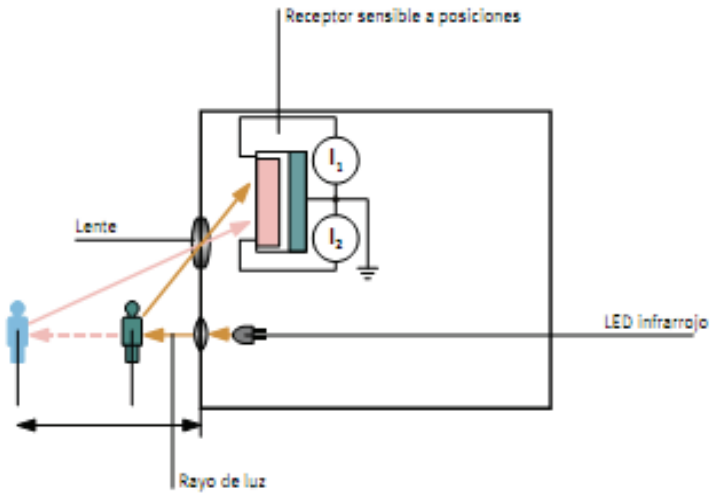


FIGURA 1.6 Detector de distancias por rayos infrarrojos, Tomado de [15]

Los sensores son capaces de medir distancias con precisión o relativas a objetos, con valores entre 4 y 30 cm. La conexión del sensor es especialmente sencilla e incluye tan sólo una señal de salida analógica y la alimentación. La electrónica de

evaluación del sensor determina la distancia, que puede leerse como una señal analógica. [15]

1.2.1.4.2 Detectores de Posición Ópticos

Robotino® además incluye en su estructura dos sensores de reflexión directa (de luz difusa). Los cables flexibles de fibra óptica de polímero son óptimos para manejar la longitud de onda infrarroja, debido a la baja amortiguación de luz.

Los sensores deben fijarse con los accesorios suministrados para ello y deben ser conectados a la interfaz de E/S. [16]

En este tipo de sensores el emisor y el receptor se encuentran localizados en un solo cuerpo, pues el objeto refleja una parte de la luz que hará que el receptor se active, brindando una respuesta que depende del tipo de receptor: NC (Normalmente Cerrado) o NO (Normalmente Abierto).

Además se debe considerar que la reflexión de la luz depende del tamaño del objeto, del tipo de superficie, de la forma, de la densidad, y del color del objeto, adicionalmente del ángulo de incidencia de la luz.

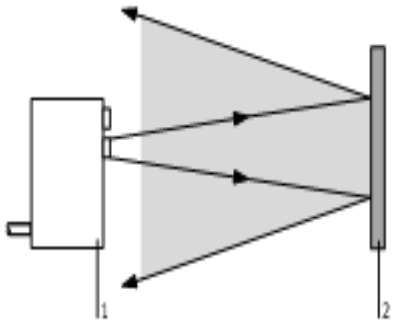


FIGURA 1.7 1. Emisor-Receptor 2. Superficie Especular Tomado de [16]

1.2.1.4.3 Encoder Incremental

La velocidad real de cada motor se mide en RPM por el encoder incremental. Si la velocidad real del motor difiere del punto de consigna, puede ajustarse para que coincida con el valor deseado por medio de un regulador PID, cuyos parámetros están configurados con la ayuda del software Robotino® View. [16]

Se ampliará su estudio en el Capítulo 2. (Véase Sensores Propioceptivos)

1.2.1.4.4 Sensor Anticolisión

El sensor anticolisión está formado por una banda de detección fijada alrededor de un aro que circunda el chasis. Una cámara de conmutación se halla situada dentro de un perfil de plástico. Dos superficies conductoras se hallan dispuestas dentro de la cámara, manteniendo una determinada distancia entre sí. Estas superficies entran en contacto cuando se aplica una mínima presión a la banda.

Con ello, una señal perfectamente reconocible es transmitida a la unidad de control.

Las colisiones con objetos en cualquier punto del cuerpo se detectan y, por ejemplo, se provoca la detención del Robotino® [16]

FIGURA 1.8 Encoder incremental (1) Banda de impacto, sensor anti-colisión (2) Sensores de medición de distancia (3), Tomado de [16]

1.2.1.5 **Modelo Cinemático**

El modelo que describe a la plataforma robótica Robotino® es un modelo cinemático omnidireccional, debido al tipo de accionamiento que lo gobierna.

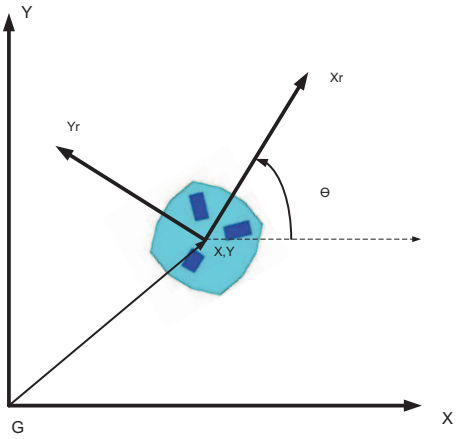


FIGURA 1.9 Cinemática Robotino®

El siguiente modelo cinemático describe el movimiento de la plataforma robótica Robotino®. Para tener más información acerca de la demostración y desarrollo matemático, dirigirse a [17] y [18].

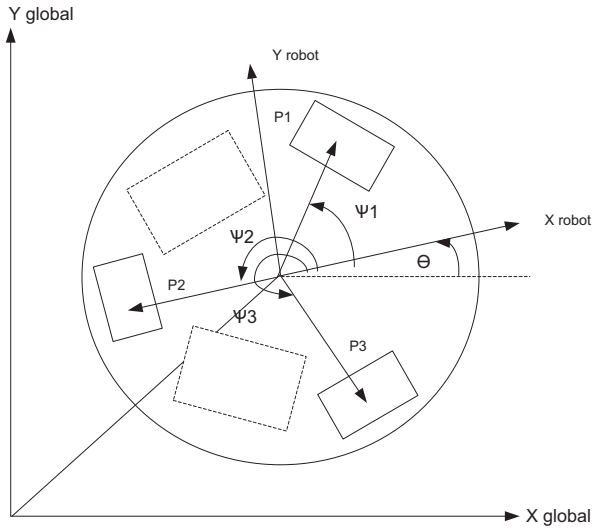


FIGURA 1.10 Modelo Cinemático Omnidireccional

De acuerdo a la FIGURA 1.10 se notan los siguientes términos:

$$\begin{bmatrix}\omega 1\\ \omega 2\\ \omega 3\end{bmatrix} = \frac{1}{r}\begin{bmatrix}-\sin(\theta+\psi 1) & \cos(\theta+\psi 1) & R\\ -\sin(\theta+\psi 2) & \cos(\theta+\psi 2) & R\\ -\sin(\theta+\psi 3) & \cos(\theta+\psi 3) & R\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} \qquad (1)$$

Donde:

$\omega 1, \omega 2, \omega 3$ son las velocidades angulares de cada uno de los motores del robot.

θ es el ángulo existente entre los ejes de referencia, global y local (robot).

$\psi 1, \psi 2, \psi 3$ corresponden a los ángulos en los cuales se disponen las ruedas del robot, 0°, 120° y 240°, respectivamente.

R es la distancia desde el centro de gravedad del robot hasta el centro de cada una de las ruedas. El valor de *R* para Robotino® es de 0.4m

r es el radio de las ruedas. El valor de *r* para las ruedas de Robotino® es igual a 0.125m. [17]

1.2.2 PLATAFORMA PIONEER 3-AT [19]

1.2.2.1 **Características**

La plataforma robótica Pioneer 3-AT posee 4 motores DC, 4 llantas todo terreno que hacen posible su circulación y desplazamiento por superficies de tierra, piedra o pavimento, adicionalmente puede subir pendientes de hasta 45 grados.

Tiene la capacidad de alcanzar velocidades de hasta 0.7 m/s tanto en avance como en retroceso, y su máxima capacidad de carga es de 15kg.

FIGURA 1.11 Robot Pioneer 3-AT

En la FIGURA 1.11 se observa la forma física de la plataforma Pioneer 3-AT. Se describen a continuación los componentes tanto internos como externos, que se encuentran en la plataforma:

- Microprocesador Renesas SH2.
- Tarjeta de potencia para la alimentación de los dispositivos.
- La fuente de alimentación está compuesta por un sistema de 3 baterías de 12 voltios a 7.5 Ah, estas baterías se encuentran dispuestas en paralelo brindando así un total de 22 Ah para todo el sistema.
- La conexión con una PC principal que gobernará sus acciones se realiza a través de un Cable Serial del tipo DB9, que a su vez tendrá la adaptación a USB.

1.2.2.2 **Controlador Renesas SH2**

El controlador Renesas SH2 posee una velocidad de procesamiento de 44.2368 MHz, además 32 bit RISC, cuenta con una memoria RAM de 32k y 128k de memoria flash; en esta se encuentra embebido el sistema operativo ARCOS, encargado de administrar todos los recursos que están implementados en la plataforma.

1.2.2.3 **Modelo Cinemático [18]**

A continuación se muestra el modelo que rige los movimientos de la plataforma robótica Pioneer 3-AT, en la FIGURA 1.12 se notan los sistemas de coordenadas:

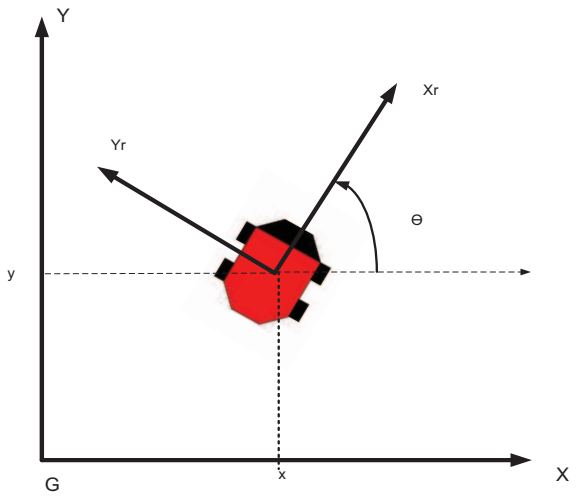


FIGURA 1.12 Cinemática Robot Pioneer 3-AT

El modelo cinemático que describe el movimiento del robot Pioneer 3-AT, corresponde a un robot de tipo uniciclo:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & -a*\sin\theta\\ \sin\theta & a*\cos\theta\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \qquad (2)$$

CAPÍTULO 2

SLAM Y SISTEMAS DE VISUALIZACIÓN

En este capítulo se describen los diferentes sistemas de visualización que se han ido desarrollando a lo largo de los tiempos. Así es como se puede llevar a efecto la navegación autónoma de robots móviles, que hoy en día se puede observar incluso en las plataformas enviadas a sitios inhóspitos sea en nuestro planeta o en planetas inexplorados. El término *visualización* es considerado debido a que los métodos a explicarse son aquellos que se asemejan a los sentidos de un ser humano, tanto es así que servirían como el sentido de la vista, razón principal por la que su estudio es necesario, pues las cámaras, sonar, o láser representan las formas en las que el robot móvil visualizará el mundo que le rodea, a fin de realizar una navegación autónoma.

Adicionalmente, se estudia acerca de la problemática de la robótica móvil, tal como es SLAM por sus siglas en inglés (Simultaneous Localization and Mapping), siendo el mapeo y la localización puntos importantes a tratar; asimismo el uso de los diversos métodos probabilísticos a fin de resolver estos problemas, como es el EKF (Extended Kalman Filter), para reducir los errores que se generan debido: a la incertidumbre del movimiento del robot, propio del modelo que lo gobierna; la incertidumbre de las medidas de los diversos tipos de sensores que lo conforman (Capítulo 1); y por último, la incertidumbre propia del entorno desconocido en el que se va a desenvolver; mediante las soluciones teóricas descritas en este capítulo se procederá a explicar en el capítulo siguiente la implementación del sistema propuesto para el presente Proyecto de Titulación.

2.1 SISTEMAS DE NAVEGACIÓN

La navegación de un robot móvil se ha convertido en una difícil labor para los investigadores, pues debe ser tal el desarrollo de ésta, que debe llegar a convertirse en una navegación autónoma, de esta manera se llega a la conclusión que la navegación es la tarea por la cual el robot móvil debe ser capaz de desplazarse de un punto a otro, dentro de un mapa y de forma segura; en

general, la navegación considera frecuentemente las siguientes preguntas: ¿Cuál es la posición actual del robot móvil? ¿Hacia dónde se dirige?, preguntas cuyas respuestas se resumen en: conocer la posición actual con el fin de tomar decisiones acertadas, y tener una percepción del mundo que le rodea, esto puede ser mediante sensores externos que brinden información, para de esta manera crear un mapa o modelo del entorno. [20]

Finalmente, esto explica que la navegación guarda una interesante relación con el conocimiento del mapa del entorno a ser explorado, dicho conocimiento puede ser total, parcial o nulo, y esto a su vez debe ser considerado detenidamente para la toma de decisiones basadas en estrategias de control, así como de los modelos y características de los sensores que serán usadas para navegar.

Para que la navegación sea posible, los robots móviles necesitan sensores que permitan conocer acerca del medio que los rodea, siendo capaces de medir varios estímulos tales como: color, distancia, fuerza, presión, inclinación, etc. Los sensores pueden estar agrupados de diversas formas, sin embargo una de las clasificaciones más comunes es separarlos en sensores propioceptivos y exteroceptivos.

2.1.1 SENSORES PROPIOCEPTIVOS

Los sensores propioceptivos son aquellos que miden los valores internos del robot, así como son las velocidades de los motores, la tensión de las baterías, etc. Los principales sensores de este tipo son: encoders, acelerómetros y giroscopios.

2.1.1.1 **Encoders Incrementales**

Los encoders son generalmente utilizados en robótica para realizar tareas de odometría, es decir llevar a cabo la tarea de estimar la posición de un robot móvil con respecto al punto inicial conocido; esto lo realizan en base al número de revoluciones de cada rueda del motor, y considerando la dirección de las ruedas.[21]

Los encoders pueden ser magnéticos, mecánicos y los más usados que son los ópticos, y dentro de éstos, los encoders del tipo óptico rotatorio relativo (o incremental), que está formado por disco con agujeros (ranuras) cerca de su borde, y está colocado de manera que el eje del disco llegue a coincidir con el eje de la rueda y del motor.

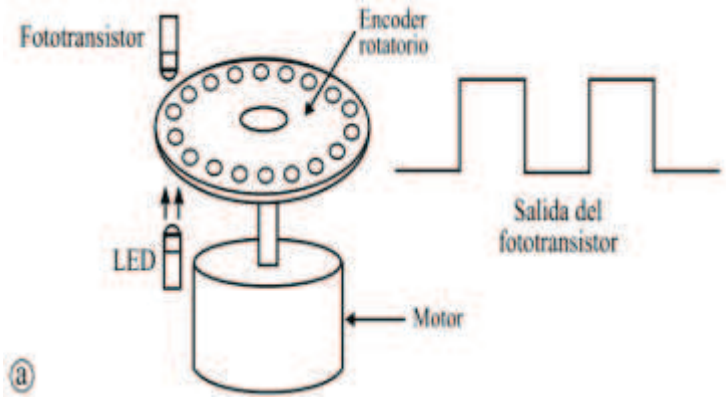


FIGURA 2.1 Encoders Óptico Rotatorio Unido a un Motor, Tomado de [21]

La precisión del encoder va en función del número de ranuras del disco, así mientras mayor sea el número de ranuras mayor será la precisión. Como se observa en la FIGURA 2.1, se coloca un LED infrarrojo en un lado del disco y un fototransistor receptor de infrarrojos en el otro lado. De esta manera cuando el disco gira las ranuras logran que el fototransistor reciba la luz de manera intermitente, por esto se genera una secuencia de pulsos como se muestra en la FIGURA 2.1; con estos pulsos se realiza una cuenta que considerará el diámetro de las ruedas y brindará un dato de distancia. [21]

2.1.2 SENSORES EXTEROCEPTIVOS

Los sensores exteroceptivos son aquellos que brindan información externa acerca del entorno del robot móvil, dicha información normalmente es utilizada para extraer características del exterior y lograr un modelo de éste. Los sensores que realizan esta función y que se usan con mayor frecuencia son: cámara, sonar y láser, etc.

2.1.2.1 **Cámara [22]**

Los sistemas basados en estereovisión son muy útiles para obtener información del entorno del robot, adicionalmente poseen la cualidad de estimar la profundidad, con esto se lograría una mejor navegación. Además con un sistema de visión estereoscópica se puede lograr la reconstrucción del entorno tridimensional que rodea al robot móvil, mediante información tridimensional que se obtendrá a partir de las imágenes correctamente tratadas.

Para lograr la obtención de información tridimensional se utilizan los sistemas estereoscópicos debido a su bajo costo y al menor tiempo de procesamiento comparado con otro tipo de sistemas, sin embargo, hay que considerar que su exactitud no es un punto a favor; por otro lado para la realización de aplicaciones de navegación no se requiere un modelo totalmente exacto de los objetos cercanos, sino es suficiente con adquirir imágenes con una aproximación buena para de esa manera tener una referencia para el robot móvil.

En un sistema de estereovisión se calcula la profundidad de cualquier punto con respecto a un plano predefinido, para esto se toma en consideración la diferencia de la proyección de dicho punto en varias fotografías.

En este proceso de visión estéreo se deben tomar en cuenta dos aspectos fundamentales como son: la correspondencia, el establecer relación de puntos de una y otra imagen; y el calcular la profundidad de los puntos, esto se puede lograr gracias a la geometría epipolar o geometría estéreo.

2.1.2.1.1 Correspondencia y Profundidad de Puntos

La correspondencia de imágenes se utiliza para establecer una relación entre los puntos de imágenes distintas pero de la misma escena, para esto se necesitan métodos de correlación, los métodos antes mencionados usan ventanas de tamaños fijos y además un criterio de similitud, el cual es una medida de la semejanza entre las ventanas; la relación entre las imágenes viene dada por la ventana que maximiza el criterio de similitud dentro de una zona de búsqueda.

Así, para cada pixel de una imagen se calcula la correlación entre los valores de intensidad de una ventana centrada en el mencionado pixel y una ventana del mismo tamaño centrada en el pixel que se desea analizar en la otra imagen, sin embargo, el principal problema radica en encontrar el punto que se ajustó de la mejor manera al primer punto.

Para resolver este limitante es necesario, tomar por cada pixel en una imagen, la vecindad cuadrada de un tamaño definido, y comparar con un número de vecindades de la otra imagen hasta llegar a seleccionar la mejor correspondencia.

Los parámetros de las cámaras a utilizarse, así como la configuración del sistema de visión estéreo, demarcan los posibles valores de profundidad a ser obtenidos, en la FIGURA 2.2 se puede observar que la distancia que existe entre las cámaras o también llamada línea base *b,* y el ángulo de visión de las cámaras limitan el valor mínimo de profundidad.

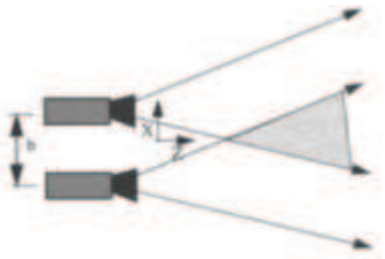


FIGURA 2.2 Zona en la que se puede obtener profundidad (Gris), Tomado de [22]

Una imagen de profundidad puede ser analizada como una representación en dos dimensiones de las distancias que existen entre los puntos en el espacio y un plano de referencia, para este caso ubicado en la cámara, dichas imágenes expresan estas distancias mediante un valor entero, por este motivo pueden visualizarse como un valor de color.

En la FIGURA 2.3, (derecha) se muestra un ejemplo de imagen de profundidad en la cual, la distancia se halla representada por niveles de gris y se obtuvo luego de que se le aplicaran algunos filtros de validación tales como: unicidad y semejanza de textura, mientras que la correspondiente imagen a color (izquierda) muestra el entorno que se trata de reconstruir.

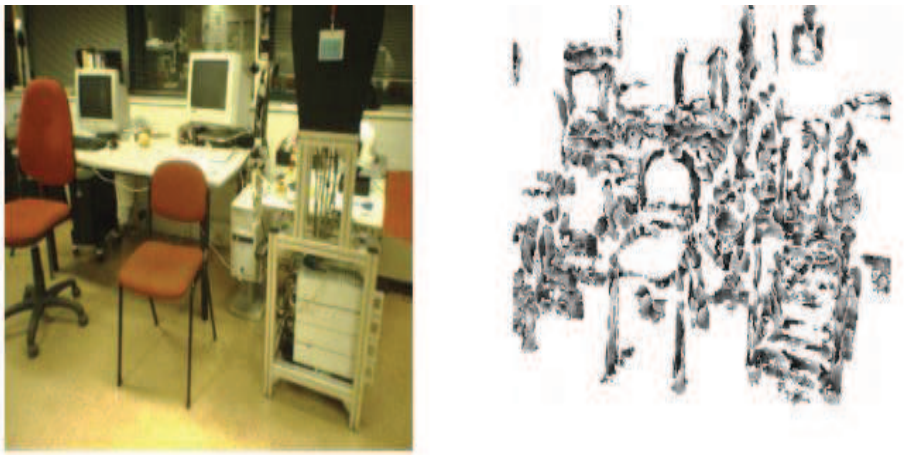


FIGURA 2.3 (Izq.) Imagen del entorno, (Der.) Imagen de Profundidad, Tomado de [22]

2.1.2.2 **Sonar**

SONAR es el acrónimo de “SOund NAvigation and Ranging”, en español “Navegación y Localización por Sonido”. Esta técnica usa la propagación del sonido, y ha sido usada frecuentemente en la navegación de robots móviles por diversas razones, como son: su facilidad de obtención, su sencillo manejo, y su bajo costo, pasando por el hecho de la gran cantidad de artículos de investigación existentes de este tema, siendo la principal motivación a su uso pues algoritmos de manejo están disponibles con facilidad.[23] y [24]

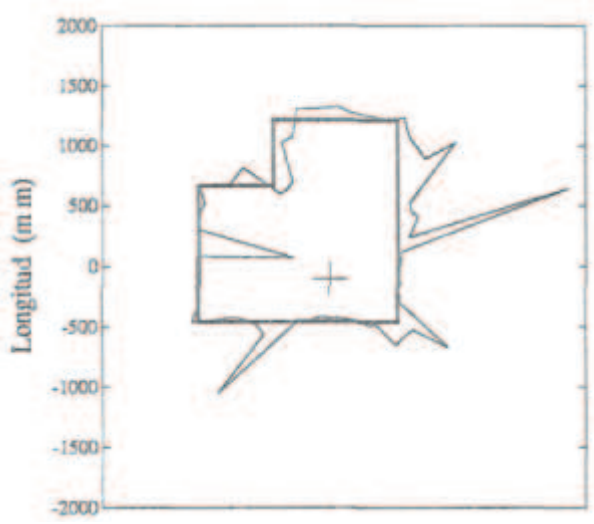


FIGURA 2.4 Barrido de un sonar en un entorno estructurado, Tomado de [24]

En la FIGURA 2.4 se observa un barrido en una habitación vacía, habiendo marcado la posición del sonar, con esto se puede decir lo siguiente:

- Las superficies no son detectadas en su totalidad
- Cierto número de medidas realizadas con este método no corresponden a objetos existentes en distancia o dirección.

A pesar de los puntos a favor que poseen los sensores de ultrasonido, se debe tener en cuenta que son poco fiables, y problemáticos, lo mencionado anteriormente responde a las características que se citan a continuación:

- Considerando la impedancia acústica del aire que es muy baja, y comparándola con la impedancia típica de los objetos (que es mucho mayor), se nota que las superficies de los objetos de un entorno van a actuar como reflectores de sonido (absorción baja).
- Si se observa la longitud de onda de la radiación, su valor es grande, en comparación con las frecuencias usadas típicamente en robótica que se encuentran en el orden de 40 kHz a 80 kHz (λ=4 mm y λ=8 mm), de aquí se concluye que la mayoría de superficies tienden a comportarse como espejos acústicos.
- De esta manera las superficies no ortogonales a la propagación del sensor podrían reflejar esta energía en otra dirección, por tanto haciendo que ciertos objetos sean acústicamente invisibles.
- Los efectos de difracción pueden no considerarse importantes cuando el objeto es muy grande, pero hay que tener cuidado con el efecto de múltiples reflexiones pues este fenómeno puede ocasionar medidas falsas, obteniéndose medidas mayores a las reales. [24]

2.1.2.2.1 Principio de Funcionamiento

Los dispositivos de tipo sonar utilizan energía acústica para de esta manera medir distancias, así, un emisor envía un paquete de ondas de presión ultrasónicas, luego se toma en cuenta el tiempo que se demora en reflejarse, rebotar y posteriormente regresar a un receptor, este tiempo se conoce como tiempo de vuelo (*time of flight*), con este dato la distancia a la que el paquete de ondas se reflejó puede ser rápidamente encontrada considerando la velocidad de

propagación del sonido en el aire (o cualquier fluido) y el tiempo de vuelo, obedeciendo a la ecuación siguiente:

$$d = c.t \qquad \text{(Ec. 2.1)}$$

En la Ec. 2.1 se debe considerar que el valor de c, corresponde a la velocidad del sonido en el aire, dicha velocidad generalmente responde a la ecuación:

$$c = \sqrt{\gamma.R.T} \qquad \text{(Ec. 2.2)}$$

En la Ec. 2.2:

γ: Coeficiente adiabático, (calculado experimentalmente en Lab. Física EPN) $\gamma = 1.47$ [25]

R: constante universal de los gases

T: Temperatura absoluta en grados Kelvin

Como la experimentación usualmente es en aire a condiciones normales de presión y temperatura, se conoce que el valor de *c = 343 m/s, (*Dato calculado en Lab. Física EPN) [25]

En la FIGURA 2.5 se muestra el proceso de envío y recepción de las señales por parte de un sensor de ultrasonido; esto inicia con la emisión de un paquete de ondas, al terminar la transmisión un integrador inicia un incremento, a fin de medir el tiempo a partir de la transmisión de las ondas hasta que se encuentre un eco.

Adicionalmente, se establece en el receptor un valor umbral, y éste al ser superado y recibir una onda de sonido, se considerara un eco de entrada de sonido válida, en la mayoría de los sensores de ultrasonido el emisor y el receptor son un mismo dispositivo, por tal razón no se puede hacer una transformación de uno en otro al instante, pues la membrana electrostática utilizada para la emisión de las ondas no puede reusarse hasta que deje de vibrar completamente, y dicho tiempo de demora se denomina usualmente *blanking time (*tiempo de borrado*)*.

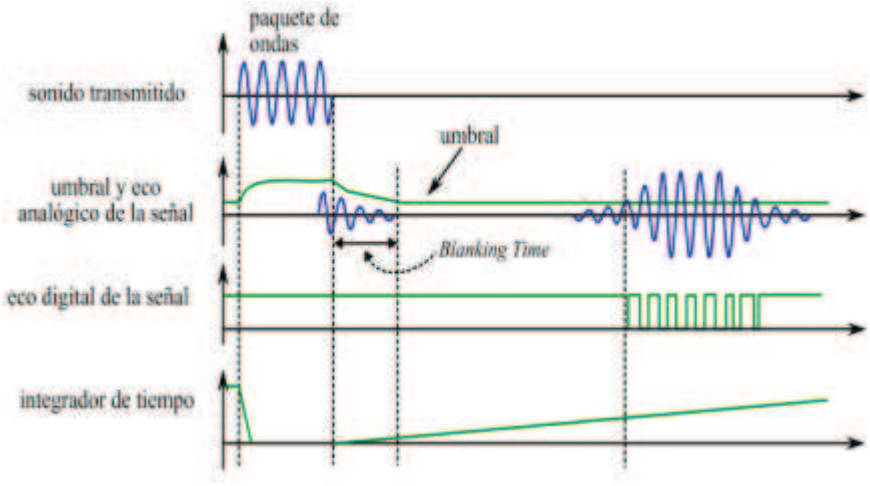


FIGURA 2.5. Emisión y recepción de señales en un sensor de ultrasonido, Tomado de [21]

Un problema que surge constantemente es el hecho de cubrir 360°, pues en estos casos existe un efecto llamado (*crosstalk*). Estas interferencias pueden ser directas o indirectas, así: si el sonar *x* emite una onda, ésta es reflejada en un objeto y al regresar es receptada por el receptor *x, y,* más los adyacentes, por el contrario en las interferencias indirectas el sonar *x* emite una onda y lo receptan los sonares *y* más adyacentes mas no el sonar *x*. Con la finalidad de evitar este tipo de interferencias se opta por no usar todos los sonares de manera simultánea. [21]

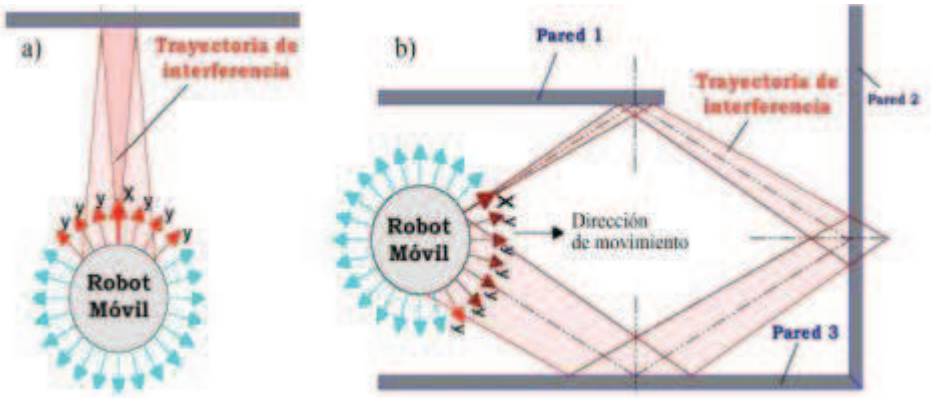


FIGURA 2.6 Interferencias (crosstalk) a) Directa, b) Indirecta, Tomado de [21]

2.1.2.3 **Láser**

El término Láser por sus siglas *(Light Amplification by Stimulated Emission of Radiation)*, usa al igual que el sonar estudiado anteriormente el tiempo de vuelo

para calcular la distancia, sin embargo, la principal diferencia radica en la velocidad de propagación, pues la velocidad de propagación del sonido es de 0.3 m/ms y la velocidad de una onda electromagnética es de 0.3 m/ns, de esta manera se observa que es un millón de veces mayor, no obstante la tecnología a ser usada a fin de medir ese tiempo de vuelo resulta ser muy costosa, entendiéndose porqué el uso de sonares es más frecuente a pesar de sus conocidos errores. [21]

Como ventajas conocidas se pueden citar muchas:

- Los datos que son entregados por un sensor láser pueden ser interpretados directamente; por otro lado, los datos de un sistema de visión estéreo (analizado anteriormente) requieren un análisis previo y aun más elaborado. Por ejemplo, con un sistema de visión se requiere de millones de pixeles por imagen, mientras que los datos provenientes del láser pueden ser 500, y en cada dato corresponde a una medida determinada.
- La precisión que tiene un láser es muy grande comparándola con la de un sonar, por ejemplo, muchos modelos de láser, como el SICK LMS221 poseen una desviación estándar de 1mm, un alcance máximo de 80 m y una resolución mínima de 0.25°.

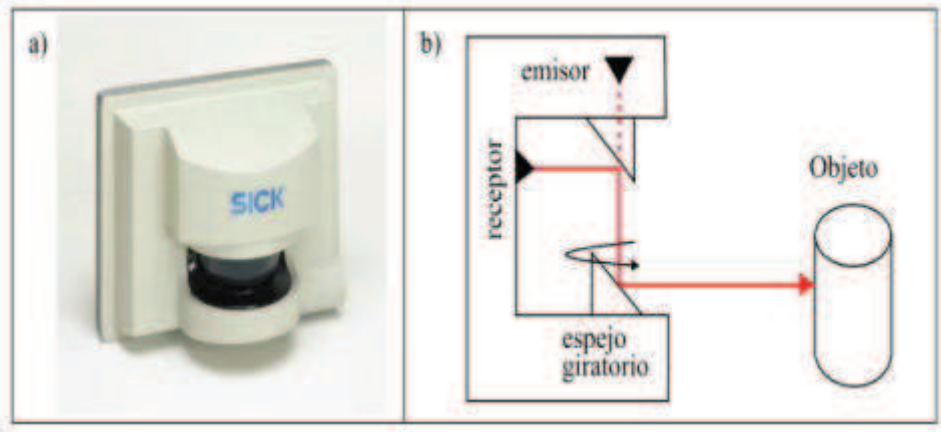


FIGURA 2.7 a) Sensor SICK LMS221, b) Funcionamiento, Tomado de [21]

2.1.2.3.1 Escáner Láser Hokuyo URG-04LX-UG01

Para el presente proyecto se utiliza el *Láser Range Finder Hokuyo URG-04LX-UG01*. Este modelo de medidor láser realiza un escaneo del contorno en 2D, y ha

sido diseñado por el fabricante para su uso únicamente en interiores. Este sensor ha sido escogido por ser económico, pues los medidores láser cuyo manejo es apto bajo condiciones ambientales difíciles (lluvia, niebla, nieve, frio) son de elevado costo.

En la FIGURA 2.8 que se encuentra a continuación, se puede observar al escáner en su montaje definitivo sobre Robotino® (izquierda), y además se muestra al escáner en perspectiva (derecha).

FIGURA 2.8 (Izq.) Escáner Montado en Robotino®, (Der.) Láser Hokuyo URG-04LX-UG01

Sin embargo, un limitante de este sensor es la necesidad de contar con una fuente de alimentación extra, y no solamente con el puerto USB que dispone Robotino®.

A fin de cubrir esto se considera colocar además un HUB del tipo EX-1163HM; este dispositivo brindará la energía necesaria para el correcto desempeño del escáner láser. La corriente necesaria para el arranque del sensor, es de 1.2A, aproximadamente. Además cabe destacar que el HUB para su alimentación toma una de las salidas digitales (descrita en Capítulo 1) que tiene la plataforma robótica Robotino® mediante un conector verde que se puede observar en la FIGURA 2.9.

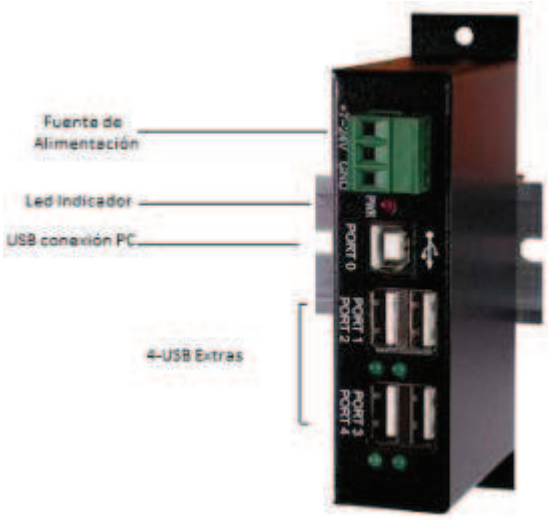


FIGURA 2.9 Partes del HUB EX-1163HM, Tomado de [26]

Adicionalmente, se necesitó de placas metálicas para lograr el acoplamiento del Láser Range Finder sobre Robotino®; estas placas brindan la posibilidad de colocar al láser en dos posiciones; en la FIGURA 2.10 se muestra al láser colocado de acuerdo a los soportes y sus respectivas posiciones.

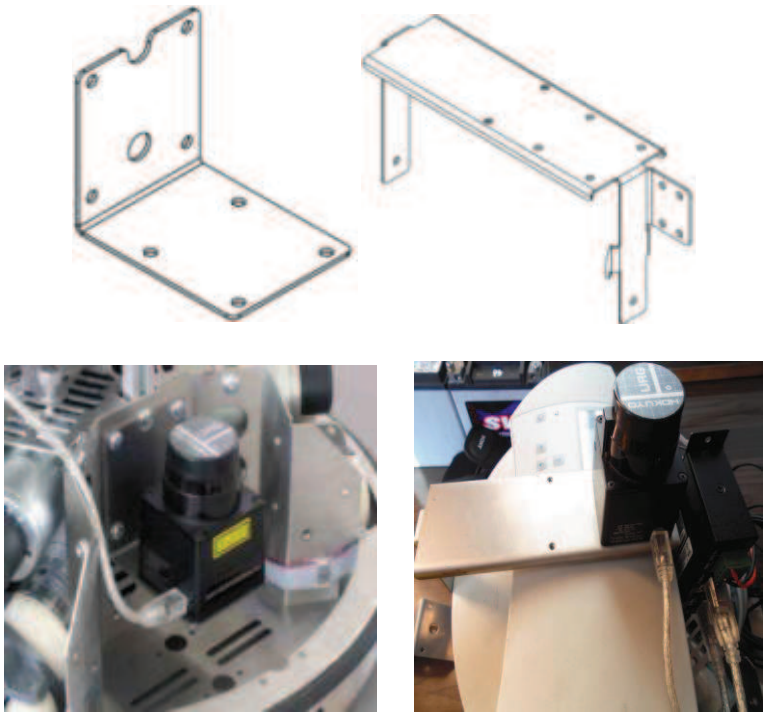


FIGURA 2.10 Posiciones de Montaje del Láser Hokuyo URG-04LX-UG01 sobre Robotino®

En cuanto al funcionamiento del láser URG-04LX-UG01, este sensor se basa en la medición del tiempo de vuelo, el láser emite un pulso de luz infrarroja (λ=785nm) que al encontrar un objeto cercano, rebota, al regresar al láser, este tiempo de vuelo (de ida y vuelta), hace que se determine fácilmente la distancia, sin embargo, cuando el haz de luz no encuentra un objeto en el cual reflejarse, enviará la distancia máxima posible (5.6 m en este caso), esto lograría medir las distancias en una sola dirección. Para solucionar esto y alcanzar una zona de 180° (240° es el máximo por default) el sensor utiliza un espejo rotatorio con el que se logra el barrido 2D. [27]

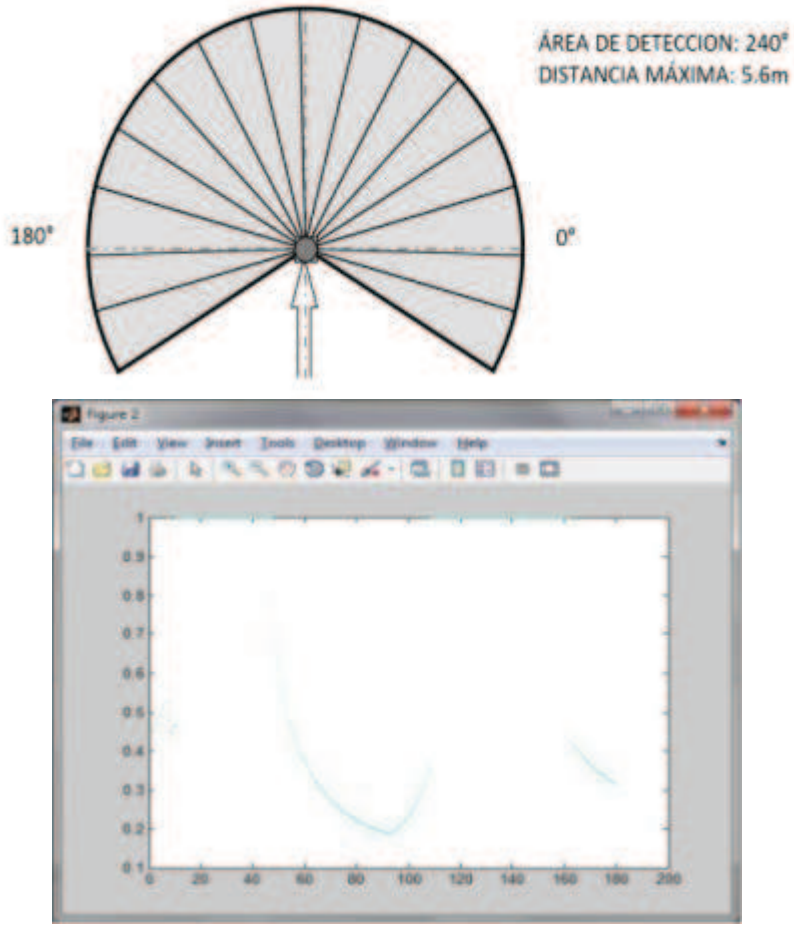


FIGURA 2.11 (Arriba) Características del Barrido de Hokuyo URG-04LX-UG01, (Abajo) Ejemplo de un barrido del Láser (distancia vs ángulo), Tomado de [27]

El láser Hokuyo URG-04LX-UG01, posee un área de detección de 180°, y una máxima distancia de 5.6m; estos datos son enviados hacia el procesador interno de Robotino® mediante una matriz de 513 elementos válidos, debido a que la

resolución angular es de 0.36°, valor dado por el fabricante (0.351° experimentalmente).

En la TABLA 2.1 se describen algunas de las características del láser Hokuyo URG-04LX-UG01

TABLA 2.1 Características Sensor Láser Hokuyo URG-04LX-UG01, Tomado de [27]

CARACTERÍSTICA	VALOR
Amplitud del Barrido	**180°**
Resolución Angular	0.36°
Distancia Máxima	5.6 m
Longitud de onda	λ=785nm
Fuente de Alimentación	5 Vdc (USB)
Consumo de Corriente	500mA (800 mA en el arranque)
Resolución	1 mm
Tiempo de Barrido	100 ms
Peso	160 g
Vida Útil	5 años
Dimensiones	50x50x70 mm

2.2 PROBABILIDAD Y ROBÓTICA

Actualmente los campos de aplicación de la robótica son muy variados, tales como líneas de ensamblaje, exploración espacial, medicina, etc. Muchas de esas aplicaciones se llevan a cabo dentro de entornos sumamente cambiantes, a veces, caóticos, que poseen alto grado de incertidumbre, es por ello que un control determinístico, es decir, con modelos que no contemplan el azar o la incertidumbre del ambiente, no es válido, y generaría resultados poco satisfactorios al momento de la implementación. Es así que se han buscado mecanismos para manejar estas incertidumbres con el fin de obtener resultados que sin ser perfectos sean los más óptimos desde el punto de vista práctico. En esa búsqueda se ha encontrado una solución bastante satisfactoria, la cual es, abordar el tema del control, desde un punto de vista probabilístico, tomando las relaciones entre variables, no como algo determinístico, sino como algo

estocástico, esto es un proceso cuya salida o salidas están determinadas por las variables que pueden ser predichas, así como por variables aleatorias.

2.2.1 MODELOS, SENSORES Y SU COMPORTAMIENTO EN LA REALIDAD [9]

Las relaciones entre el control y el estado del robot son descritas ampliamente en la literatura, teniendo modelos cinemáticos y dinámicos, que sirven para obtener una relación de la forma $x' = f(u, x)$, donde se suele tener la idea incorrecta que la función *f* es determinística, lo cual sería cierto en algunos casos ideales, donde se desprecian muchos factores, pero no en entornos reales. Por ejemplo, al realizar el modelado del robot utilizado en este proyecto de titulación, (Robotino®) se asume que no existe deslizamiento de las ruedas con la superficie, el centro de gravedad es el centro geométrico del robot, y otras consideraciones que se mencionaron en el Capítulo 1. Dicho esto, sería incorrecto suponer que las acciones de control se van a efectuar exactamente como son dadas; para citar un ejemplo, si se da al robot la orden de avanzar un metro adelante, éste lo hará aproximadamente, es decir, el robot avanzará y se ubicará en una posición desconocida, cerca de donde se le dio la orden (cuan cerca se coloca depende de la calidad del modelo matemático), pero no exactamente, sino éste estará en algún lugar dentro de una elipse de incertidumbre. Si se sigue dando órdenes de control, asumiendo que la posición final es la verdadera, pero lo que en verdad va a suceder es que el tamaño de la elipse va a ir incrementando conforme el robot avanza. Esta situación se muestra en la FIGURA 2.12, donde la elipse allí dibujada muestra la incertidumbre en la posición final del robot.

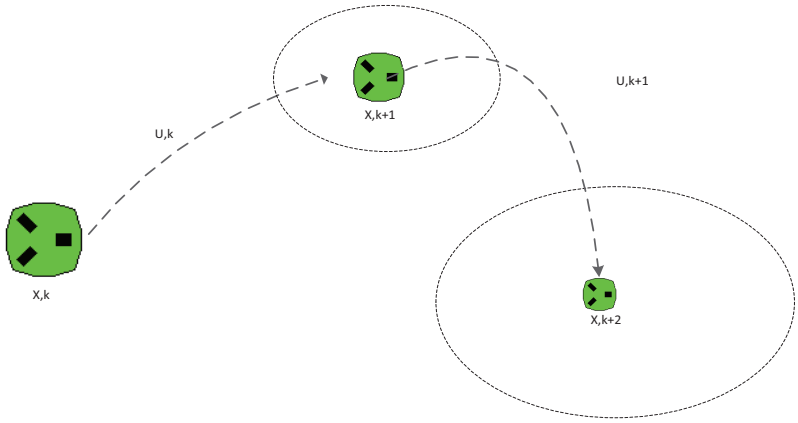


FIGURA 2.12 La incertidumbre en el movimiento aumenta

Con estos antecedentes, los modelos adoptados pasan a tener la forma $p(x'|u,x)$, con lo cual se obtiene la probabilidad actual (*p)* del estado del robot, dada la señal de control *(u)* y el estado anterior del sistema *(x)*. Con esto se consigue que en lugar de tener una predicción determinística, se tenga una distribución de probabilidad sobre todas las posibles salidas, lo que permite manejar la incertidumbre que se presenta y así poder adaptar el control a los cambios aleatorios que se dan.

Hasta ahora se ha supuesto que el estado *x* contiene únicamente la localización del robot, sin embargo la mayor parte de las veces no es así. Vale la pena mencionar que como estado se conoce a toda información necesaria para el control y posicionamiento del robot, esto puede ser pose, velocidad, detección de objetos circundantes, etc. Para poder determinar muchas de estas variables, es necesario un grupo, o al menos un *sensor* que va a proporcionar tal información. Siguiendo el mismo análisis que el realizado para el modelo del robot, sería incorrecto suponer que los sensores van a entregar información sin error. La información provista por los mismos puede ser representada a través de una función determinística *g* que va a relacionar el estado con las mediciones realizadas, hablando en el caso ideal, sin tener en cuenta que los sensores reales tienen ruido propio de sus sistemas de medición, y más aún limitaciones de rango. El enfoque probabilístico de este tema, es modelado nuevamente con distribuciones de probabilidad condicional. Tales modelos pueden ser tomados en forma directa como $p(z|x)$ que es la probabilidad de tener cierta medición teniendo un estado actual dado, o en forma inversa como $p(x|z)$, que es la probabilidad de obtener cierto estado, dada cierta medición; el uso de cualquiera de las dos depende de la aplicación en cuestión. [28]

En la FIGURA 2.13, se puede apreciar que el robot detecta una característica del ambiente y, a priori, la incertidumbre es alta, conforme se le aplican técnicas probabilísticas, tal incertidumbre va disminuyendo.

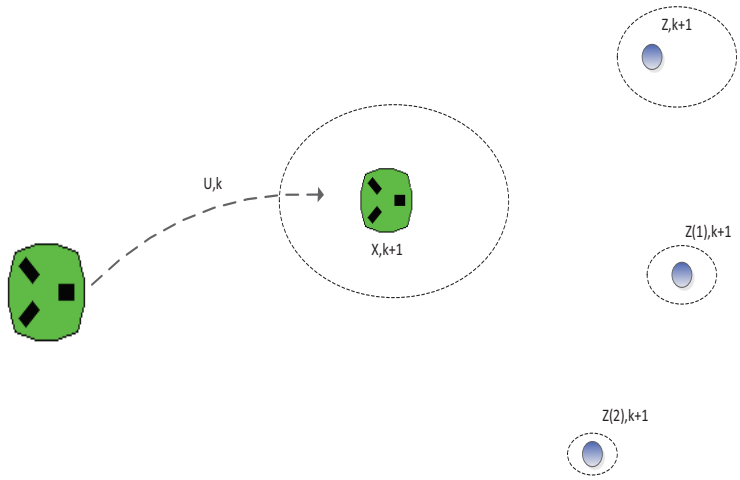


FIGURA 2.13 La incertidumbre disminuye aplicando técnicas probabilísticas

2.2.1.1 **Estimación Probabilística**

Se han expuesto las razones para abordar el tema del control desde un punto de vista probabilístico; ahora se van a revisar los diferentes métodos con los cuales se puede aplicar estas técnicas, siendo unos más sofisticados y complicados que otros, pero cada uno tiene o tuvo su aplicación, al final qué método se implementa depende de la aplicación en cuestión y del programador.

Los métodos más conocidos y desarrollados son los denominados “filtros bayesianos”, que permiten estimar las variables de estado (en este caso el vector de estados **X**), dado las variables conocidas. Estos filtros permiten de una manera recursiva estimar el estado actual del robot dado su estado anterior y/o las medidas de sus sensores.

Entre los filtros bayesianos más conocidos están las Cadenas de Markov, Filtro de Kalman, Filtro de Partículas, etc. En la siguiente sección se explicará brevemente las Cadenas de Markov y posteriormente se analizará el Filtro de Kalman, el cual se usa en este proyecto.

2.2.1.1.1 Cadenas de Markov

Las Cadenas de Markov describen un proceso estocástico discreto como aquel en el cual la probabilidad de que se tenga un resultado o que un evento ocurra, depende únicamente del resultado o evento inmediatamente anterior. Este tipo de

procesos recibe su nombre en honor al matemático ruso Andrei Andreevitch Markov, quien las introdujo por primera vez en 1907. [29]

Existen varios procesos estocásticos en los cuales los resultados del experimento no dependen de los resultados anteriores, sino que son independientes, en este caso el mejor ejemplo es el lanzamiento de una moneda o un dado; pero en la mayoría de los casos los resultados del experimento actual dependen de los resultados anteriores, como por ejemplo el precio de cierre de una acción en la bolsa, que se ve influido en cierta forma por el comportamiento de la bolsa en días anteriores. [30]

El uso de Cadenas de Markov para la localización se hace en dos etapas como lo hace un filtro recursivo: la predicción y la corrección basada en la observación. [20] La corrección se la hace tomando en cuenta dos datos: las medidas de los sensores y los datos que entrega la odometría del robot.

El uso de este método de localización se ilustra con el siguiente sencillo ejemplo [31]:

Se va a suponer que un robot únicamente se puede mover en la dirección *x*, y que el entorno en el cual se desplaza es un pasillo con tres puertas. Se ubicará al robot en una posición desconocida; al inicio del algoritmo, éste va a colocar una distribución de probabilidad uniforme a todos las posibles posiciones del robot, tal como lo muestra la FIGURA 2.14a.

Posteriormente, se obtienen las medidas de los sensores y se llega a la conclusión que se encuentra ubicado al lado de una puerta, con esto se logra incrementar la probabilidad de lugares cercanos a las puertas, disminuyendo así la probabilidad en otros lugares, resaltando que la misma no es cero puesto que los sensores pueden tener problemas de ruido y considerando que al tener únicamente la visualización de una puerta no se puede excluir el hecho de que el robot pueda estar en otro lugar, esto se muestra en la FIGURA 2.14b.

Luego el robot se va a desplazar, y el algoritmo va a incorporar esta información actualizando el estado predicho, además para tomar en cuenta el hecho de que existe ruido también en la odometría del robot, las distribuciones de probabilidad son menos ciertas que las anteriores, FIGURA 2.14c.

Finalmente, el robot lee los datos de sus sensores y resulta que detecta que está al lado de una puerta, esta observación es multiplicada por la estimación actual, lo que lleva a la distribución final centrada en la localización actual del robot, con lo cual el robot conoce su posición, como se muestra en la FIGURA 2.14d.

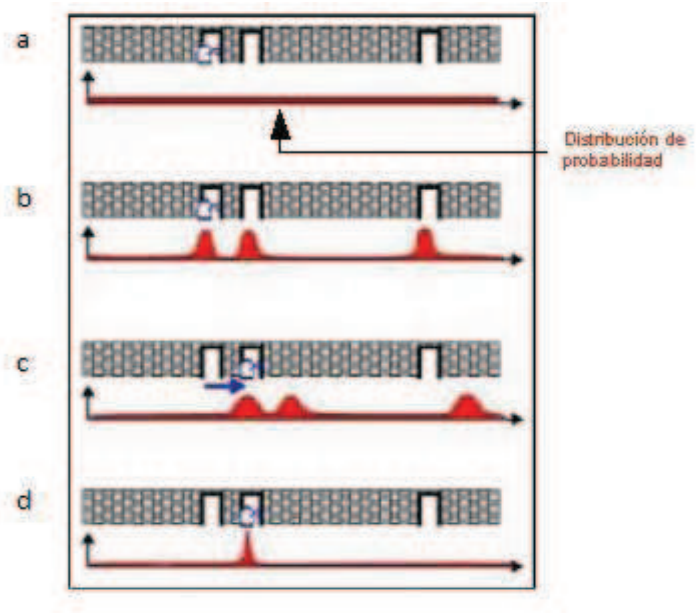


FIGURA 2.14 Ejemplo localización de Markov, Tomado de [33]

Las ventajas de este método son su simplicidad y la robustez que presenta en entornos estáticos, por otro lado las desventajas que presenta son el alto costo computacional que requiere y su falta de confiabilidad cuando se aplica en entornos cambiantes.

El tratamiento matemático de este método se escapa del objetivo de este proyecto, por lo que si se desea conocer acerca del mismo se puede encontrar más información en [31, 32, 33].

2.3 FILTRO DE KALMAN

El Filtro de Kalman es un algoritmo óptimo recursivo, que fue introducido por Rudolf E. Kalman en 1960, se le denomina óptimo porque incorpora y procesa todas las mediciones que se le suministra, además es recursivo puesto que no necesita almacenar los datos previos, únicamente toma los valores anteriores y

las mediciones recientes para calcular los valores actuales, lo que lo hace adecuado para ser usado en procesos on-line o en tiempo real. [34]

Este algoritmo toma como premisa que las perturbaciones en el proceso y en la medida son de carácter gaussiano, es decir, estas perturbaciones son procesos estocásticos con distribuciones gaussianas, lo cual permite asumir que las funciones de densidad de probabilidad de las perturbaciones van a estar definidas por su media y su covarianza, dado que se asume una distribución gaussiana, la media va a ser cero y la covarianza va a tener un valor finito, si la variable aleatoria es unidimensional, la covarianza va a ser un número, en cambio, si la variable aleatoria es multi-dimensional se va a tener un vector de medias y una matriz de covarianzas, que es diagonal y simétrica en la mayoría de los casos. [20]

Inicialmente el algoritmo estaba pensado para sistemas lineales, pero se le han hecho modificaciones para que el mismo sea aplicable a sistemas no lineales, en este caso se le da el nombre de Filtro Extendido de Kalman, que será descrito en la sección siguiente.

A continuación se muestran las ecuaciones del Filtro de Kalman y se explica cada uno de los factores que intervienen en ellas.

El sistema dinámico, al cual se le va a aplicar el filtro, debe ser representado por la siguiente ecuación de estado, donde se tiene las matrices que relacionan el estado actual con el estado anterior y con las entradas de control, además del ruido presente en la dinámica del proceso.

$$x_k = Ax_{k-1} + Bu_k + w_k \qquad \text{(Ec. 2.3)}$$

Donde:

A es la matriz que relaciona el estado anterior con el estado actual

B es la matriz que relaciona las entradas de control con el estado actual

w es el ruido en el proceso

También se tiene la ecuación que representa la medición tomada por parte de los sensores del robot con la matriz que relaciona el estado actual con la medición, y el ruido propio de los sensores.

$$z_k = Hx_k + v_k \qquad \text{(Ec. 2.4)}$$

Donde:

H es la matriz que relaciona la medición con el estado actual

v es el ruido en la medición

En este trabajo se va a utilizar el subíndice *k* indicando el tiempo.

Como se había mencionado anteriormente los ruidos del proceso (*w* y *v*) tienen una función de densidad de probabilidad con media cero y covarianza finita, de la forma:

$$p(w) \sim N(0, Q) \qquad \text{(Ec. 2.5)}$$

$$p(v) \sim N(0, R) \qquad \text{(Ec. 2.6)}$$

Descripción del Algoritmo

El algoritmo del Filtro de Kalman está divido en dos partes: la primera denominada de predicción, donde se tiene el modelo del proceso, y la segunda llamada de corrección, en la cual se incorpora la información de los sensores u observadores.

Ecuaciones de predicción

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \qquad \text{(Ec. 2.7)}$$

$$P_k^- = AP_{k-1}A' + Q \qquad \text{(Ec. 2.8)}$$

Ecuaciones de corrección

$$K_k = P_k^- H' (H P_k^- H' + R)^{-1} \qquad \text{(Ec. 2.9)}$$

$$\hat{x}_k = \hat{x}_k{}^- + K_k (z_k - H \hat{x}_k{}^-) \qquad \text{(Ec. 2.10)}$$

$$P_k = (I - K_k H)\, P_k^- \qquad \text{(Ec. 2.11)}$$

TABLA 2.1 Descripción de las variables del algoritmo del Filtro de Kalman, Tomado de [20]

VARIABLE	DIMENSIÓN	DESCRIPCIÓN
$\hat{x}_k{}^-$	nx1	Estado predicho
A	nxn	Matriz que relaciona el estado anterior con el estado actual
B	nx1	Matriz que relaciona el control con el estado actual
u_k	lx1	Vector de señales de control
P_k^-	nxn	Matriz de covarianzas predicha
Q	nxn	Matriz de covarianzas del ruido del proceso
K_k	nxm	Matriz ganancia de Kalman
H	mxn	Matriz que relaciona las medidas con el estado
R	mxm	Matriz de covarianzas del ruido en la medida
$\hat{x}_k$	nx1	Vector de estados estimado
z_k	mx1	Vector de medidas en el instante k
P_k	nxn	Matriz de covarianzas del vector de estado en el instante k
I	nxn	Matriz identidad

2.3.1 FILTRO EXTENDIDO DE KALMAN

Como se mencionó anteriormente, el Filtro de Kalman fue originalmente ideado para sistemas lineales, pero puesto que muchos sistemas son no lineales, se debía ampliar su funcionamiento, es así que se obtuvo el Filtro Extendido de Kalman (EKF), el cual brinda una solución linealizando el proceso en torno al estado actual, y el modelo de la medición en torno a la observación predicha. Este

par de linealizaciones se las formula como series de Taylor en las cuales los términos con mayor orden son despreciados. [35]

El EKF cambia las matrices ***A***, ***B***, ***H*** por funciones que entregan el estado estimado y las observaciones, incluyendo, por supuesto el ruido en ambos casos, así se tiene que:

$$\hat{x}_k = f_x(x_{k-1}, u_k, w_k) \qquad \text{(Ec. 2.11)}$$

$$z_k = h(x_k, v_k) \qquad \text{(Ec 2.12)}$$

La descripción de las etapas de estimación y corrección van a ser descritas a continuación [33]:

Ecuaciones de Predicción

$$\hat{x}_k^{\,-} = f(\hat{x}_{k-1}, u_k,) \qquad \text{(Ec. 2.13)}$$

$$P_k^- = A_k P_{k-1} A'_k + W_k Q_{k-1} W'_k \qquad \text{(Ec. 2.14)}$$

Ecuaciones de Corrección

$$K_k = P_k^- H'_k (H_k P_k^- H'_k + V_k R_k V'_k)^{-1} \qquad \text{(Ec. 2.15)}$$

$$\hat{x}_k = \hat{x}_k^{\,-} + K_k (z_k - h(\hat{x}_k^{\,-}, 0)) \qquad \text{(Ec. 2.16)}$$

$$P_k = (I - K_k H_k)\, P_k^- \qquad \text{(Ec. 2.17)}$$

Como se nota, aparecen términos nuevos en comparación al Filtro de Kalman lineal; en la TABLA 2.2 se explicarán cada uno de los términos que aparecen en las ecuaciones del algoritmo del Filtro Extendido de Kalman (EKF), los cuales se utilizan en el presente proyecto.

TABLA 2.2 Descripción de las variables del algoritmo del EKF, Tomado de [20]

Variable	Dimensión	Descripción
$\hat{x}_k^-$	nx1	Estado predicho
f()		Modelo del movimiento
A	nxn	Matriz jacobiana de derivadas de f con respecto a x
u_k	lx1	Vector de señales de control
P_k^-	nxn	Matriz de covarianzas del estado predicho
Q	nxn	Matriz de covarianzas del ruido del proceso
W	nxn	Matriz jacobiana de derivadas de f con respecto al ruido
K_k	nxm	Matriz ganancia de Kalman
h()		Modelo de la medición
R	mxm	Matriz de covarianzas del ruido en la medida
V	mxm	Matriz jacobiana de derivadas de h con respecto al ruido
$\hat{x}_k$	nx1	Vector de estados estimado
z_k	mx1	Vector de medidas en el instante k
H	mxn	Matriz jacobiana de derivadas de h con respecto a x
P_k	nxn	Matriz de covarianzas del vector de estado en el instante k
I	nxn	Matriz identidad

Existe también otra variación del filtro de Kalman que es el Filtro Descentralizado de Kalman, que se aplica cuando los sistemas tienen un comportamiento altamente no lineal y el EKF no tiene un buen desempeño [34]. Si se desea saber más acerca de esta variante de Filtro de Kalman se puede consultar en [36, 37].

2.4 SLAM (LOCALIZACIÓN Y MAPEO SIMULTÁNEOS)

Uno de los principales objetivos de la robótica móvil es brindar mecanismos para que los robots puedan desenvolverse de manera autónoma en entornos desconocidos, tal capacidad debe aportar con información suficiente tanto para la

navegación del mismo robot, como para quien necesite extraer información útil acerca del recorrido realizado. Entonces, el problema en cuestión se centra en colocar un robot en una posición inicialmente desconocida y que a partir de ahí, a través de las mediciones de sus sensores pueda construir un mapa del sitio por donde está desplazándose y usar este mapa para determinar su localización.

Este problema inicialmente estaba formulado como dos instancias separadas que se pueden enunciar en dos preguntas: ¿Dónde se encuentra el robot? (localización), y ¿Qué forma tiene el entorno por el cual el robot se moviliza? (mapeo). El responder estas dos preguntas es clave para tener robots autónomos, que sean capaces de reconocer un objetivo y decidir la forma en la cual se cumplirá con el mismo.

Con el objetivo de darle mayor autonomía a un robot móvil, se han fusionado estos dos problemas, naciendo así la técnica de SLAM (Simultaneous Localization and Mapping). El uso de este algoritmo le proporciona al robot habilidades para construir un mapa de su entorno y mientras lo hace, debe reconocer y asociar las características de su entorno (landmarks), lo que le permitiría ubicarse en el espacio que se moviliza.

Los algoritmos de SLAM no son los mismos en todas las condiciones, esto depende del lugar en el cual se va a desenvolver el robot, teniendo así algoritmos tanto como para entornos cerrados y estructurados, entornos cerrados y cambiantes, entornos al aire libre que en general son cambiantes, etc. Es así que la incertidumbre en tales ambientes va a ser mayor o menor dependiendo de ellos ya que obviamente va a ser más difícil detectar y asociar una característica en un entorno cambiante que en uno que no lo sea. Aquí radica una de las muchas complejidades que presenta el desarrollar un algoritmo robusto; otra dificultad con la que se encuentra es el coste computacional para implementar un algoritmo de SLAM, más aun en tiempo real, ya que conforme aumenta el tamaño del mapa, también aumenta la cantidad de datos a ser procesados (características del mapa), y así aumenta cuadráticamente el número de operaciones del proceso. El computador realiza un gran número de operaciones cuando calcula la matriz de covarianzas P y aún más cuando calcula la ganancia de Kalman [30]. Aunque la solución por filtro de Kalman brinda buenos resultados, si se aplica en entornos de

grandes dimensiones, el coste computacional es demasiado alto; esto se da porque la matriz de covarianzas almacena las varianzas de cada uno de los elementos que constituye el vector de estados, entonces agregar un objeto va a repercutir en el cálculo de tal matriz. No obstante, el añadir un objeto distante geométricamente del resto, va a influir de menor manera en el cálculo de *P*; es por esto que se suele crear submapas ordenados jerárquicamente para aliviar el trabajo computacional, generalmente las técnicas de submapeo conservan su estimado en las coordenadas globales. [38, 39]

2.4.1 DESCRIPCIÓN PROBABILÍSTICA DEL SLAM [4]

Es necesario considerar en primer lugar, el movimiento de un vehículo como se muestra en la FIGURA 2.15, en la cual se muestra el recorrido real y el estimado para un intervalo de tiempo.

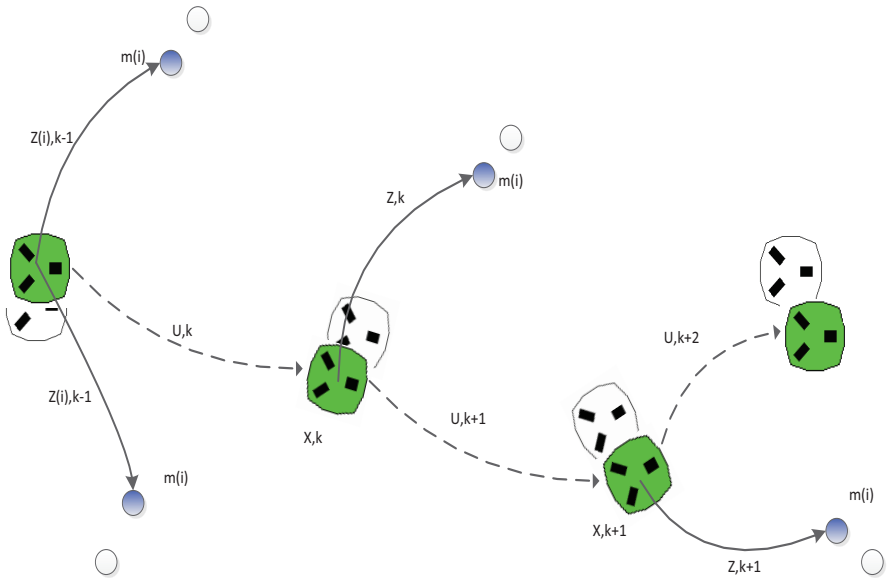


FIGURA 2.15 El robot debe estimar al mismo tiempo tanto su posición como la de las características del ambiente. A color la verdadera posición del robot y las características, en blanco y negro las estimaciones realizadas. Tomado de [4]

En el instante *k*, el robot ha realizado las mediciones con sus sensores y ha detectado la ubicación de algunas características, por lo que se tienen los siguientes datos:

x_k: Vector que describe la posición del robot.

u_k: Vector que contiene las señales de control aplicadas en el instante anterior (*k-1*) para mover el robot hasta la posición actual (*k*).

z_k: Observación de una o más características, realizada por los sensores del robot.

m_i: Ubicación de las características cuya ubicación se considera invariante en el tiempo (estacionarias).

Si se guarda estos datos durante el trayecto del robot se tendrán vectores que contienen todas las posiciones $X_{0:k} = \{x_0, x_1, \dots, x_k\}$, señales de control $U_{0:k} = \{u_0, u_1, \dots, u_k\}$, observaciones $Z_{0:k} = \{z_0, z_1, \dots, z_k\}$ y la ubicación de las características $m = \{m_0, m_1, \dots, m_k\}$.

Hablando en términos probabilísticos, el objetivo del SLAM se lograría calculando la probabilidad conjunta de la localización del robot y de las características del ambiente para todo instante *k*, dadas las señales de control, las observaciones de los sensores y el estado inicial del robot.

$$P(x_k, m | Z_{0:k}, U_{0:k}, x_0) \qquad \text{(Ec. 2.18)}$$

En general se aplica una solución recursiva, calculando primero la distribución estimada $P(x_{k-1}, m | Z_{0:k-1}, U_{0:k-1})$, para mediante la señal de control y la observación calcular la estimación posterior utilizando el teorema de Bayes. Para lograr esto es necesario tanto el modelo del movimiento del robot, así como el modelo de la observación.

El modelo de observación describe la probabilidad de encontrar una característica, dada la localización del vehículo y de la característica.

$$P(z_k, | x_k, m) \qquad \text{(Ec. 2.19)}$$

En cambio el modelo del movimiento del robot, puede ser descrito como un proceso de Markov, donde el estado actual depende únicamente del estado anterior y de la entrada de control que condujo al robot a esa posición.

$$P(x_k, | x_{k-1}, u_k) \qquad \text{(Ec. 2.20)}$$

Ahora el problema de SLAM puede implementarse como un algoritmo recursivo de dos pasos: predicción y corrección, representado en dos ecuaciones

probabilísticas, las cuales permitirán calcular la probabilidad conjunta $P(x_k, m|Z_{0:k}, U_{0:k}, x_0)$. Si el lector desea profundizar en el tema, encontrar las ecuaciones y su derivación matemática puede dirigirse a [20, 33]

Se ha descrito como se trata probabilísticamente al problema del SLAM, y como se mencionó antes existen varios métodos para aplicar estas soluciones de una manera práctica. En el presente trabajo se utiliza un Filtro Extendido de Kalman, que fue descrito en las secciones anteriores y su uso fue inicialmente presentado por Smith, Self y Cheeseman [40, 41, 42].

CAPÍTULO 3

DISEÑO Y CONSTRUCCIÓN DEL SISTEMA

En este capítulo se realiza la descripción del algoritmo utilizado en el proyecto, que consta básicamente de: filtrado de la pose, construcción del mapa, la detección de características (esquinas), así como de la teoría en la que se basa dicho algoritmo; también se describe el manejo del software de Robotino® en MATLAB, la obtención de datos de sus sensores y el tratamiento de los mismos.

3.1 IMPLEMENTACION DEL ALGORITMO DEL FILTRO DE KALMAN

Previamente se habló acerca de las bases teóricas del Filtro de Kalman, ahora se describe su implementación en el proyecto para poder eliminar los errores procedentes de la odometría del robot. El filtro toma como modelo para la predicción el modelo de un robot omnidireccional, el cual se describió previamente en el Capítulo 1, y como observador los datos de la odometría (x, y, θ) brindados por el software del robot, cuya obtención y tratamiento se describe de manera amplia en el Anexo C.

3.1.1 VALIDACIÓN DEL MODELO

Antes de pasar directamente al Filtro de Kalman fue necesario validar el modelo, para que los datos que proporcione el mismo sean correctos, puesto que Robotino® tiene sus propios valores de $\alpha 1$, $\alpha 2$, $\alpha 3$, R y r, (véase Capitulo 1). Además, se puede observar que las entradas de control son las velocidades angulares de las llantas del robot, pero en el caso de Robotino®, desde las librerías en MATLAB se obtienen las velocidades angulares de los motores, es así que se debe tener en cuenta la relación de la caja reductora que une los motores con las llantas para obtener valores correctos, aquí se resalta claramente la importancia de la etapa de validación del modelo. Cabe también mencionar que las unidades del mismo están en metros y radianes.

3.1.2 IMPLEMENTACION

3.1.2.1 **Predicción**

En esta etapa se usa el modelo cinemático del robot previamente validado para obtener una estima previa de la pose (*x*, *y*, *θ*) del robot, de acuerdo a Ec. (3.1)

$$\begin{bmatrix}\hat{x}^- \\ \hat{y}^- \\ \hat{\theta}^-\end{bmatrix} = \begin{bmatrix}\hat{x}(k-1) \\ \hat{y}(k-1) \\ \hat{\theta}(k-1)\end{bmatrix} 0.04 * \Delta T * \begin{bmatrix}-\sin\left(\hat{\theta}(k-1)+\frac{\pi}{3}\right) & \cos\left(\hat{\theta}(k-1)+\frac{\pi}{3}\right) & 0.125 \\ -\sin\left(\hat{\theta}(k-1)+\pi\right) & \cos\left(\hat{\theta}(k-1)+\pi\right) & 0.125 \\ -\sin\left(\hat{\theta}(k-1)-\frac{\pi}{3}\right) & \cos\left(\hat{\theta}(k-1)-\frac{\pi}{3}\right) & 0.125\end{bmatrix}^{-1} * \begin{bmatrix}\omega_1(k) \\ \omega_2(k) \\ \omega_3(k)\end{bmatrix}$$

(Ec. 3.1)

En la descripción del filtro también se mencionó una predicción de la observación, lo cual debería estar presente en el caso de un algoritmo de SLAM, en este proyecto se va a tomar las medidas del láser directamente sin aplicarle ningún filtrado.

El cálculo de la matriz de covarianza, considera el valor de la matriz de covarianzas predicho P_{k-1}, asimismo W_k que corresponde al Jacobiano del modelo con respecto al ruido, y se hace de acuerdo a la ecuación:

$$P_k^- = A_{x,k} P_{k-1} A'_{x,k} + W_k Q_{k-1} W'_k \qquad \text{(Ec. 3.2)}$$

Donde $A_{x,k}$ es el Jacobiano del modelo con respecto al vector *x.* Esta ecuación es completamente válida, pero si se desea tomar en cuenta las perturbaciones en las entradas de control, esta expresión puede extenderse y añadir un término que asocie también la dinámica de tales incertidumbres, quedando la predicción de la matriz de covarianza como:

$$P_k^- = A_{x,k} P_{k-1} A'_{x,k} + A_{u,k} Q_k A'_{u,k} \qquad \text{(Ec. 3.3)}$$

Donde $A_{u,k}$ es el Jacobiano del modelo con respecto a las entradas de control y Q_k es la matriz de covarianzas asociada al proceso, cuyo valor inicial es determinante para que el filtro funcione adecuadamente; este valor inicial fue encontrado experimentalmente.

$$U_k = \begin{bmatrix}{\sigma_x}^2 & 0 & 0 \\ 0 & {\sigma_y}^2 & 0 \\ 0 & 0 & {\sigma_\theta}^2\end{bmatrix} \qquad \text{(Ec. 3.4)}$$

Donde ${\sigma_x}^2$, ${\sigma_y}^2$, ${\sigma_\theta}^2$ representan las varianzas correspondientes al proceso en los ejes *x*, *y,* Θ, respectivamente.

3.1.2.2 **Actualización**

En primer lugar se calcula la ganancia de Kalman, de acuerdo a:

$$K_k = P_k^- H'_k (H_k P_k^- H'_k + R_k)^{-1} \qquad \text{(Ec. 3.5)}$$

Donde H_k es la matriz identidad de 3x3, ya que se obtiene directamente la pose desde la odometría. Además, R_k es la matriz de covarianza asociada a la observación (odometría del robot), cuyo valor inicial es determinante para que el filtro funcione adecuadamente, este valor inicial fue encontrado experimentalmente.

$$R_k = \begin{bmatrix} {\sigma_{xr}}^2 & 0 & 0 \\ 0 & {\sigma_{yr}}^2 & 0 \\ 0 & 0 & {\sigma_{\theta r}}^2 \end{bmatrix} \qquad \text{(Ec. 3.6)}$$

Donde ${\sigma_{xr}}^2$, ${\sigma_{yr}}^2$, ${\sigma_{\theta r}}^2$ representan las desviaciones estándar correspondientes al observador (odometría del robot) en los ejes *x*, *y,* Θ, respectivamente.

Posteriormente se calcula la pose corregida, con la que luego se va a realizar el mapeo, con la fórmula:

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - \hat{x}_k^-) \qquad \text{(Ec. 3.7)}$$

Se puede notar que el termino $h(\hat{x}_k^-, 0)$ de la Ec. 2.16, ahora es reemplazado únicamente por $\hat{x}_k^-$, ya que como se había dicho los datos de la odometría se los obtiene directamente. Cabe mencionar que el término $z_k - \hat{x}_k^-$ es conocido como vector innovación.

Finalmente se calcula la nueva matriz de covarianza con la fórmula:

$$P_k = (I - K_k H_k)\, P_k^- \qquad \text{(Ec. 3.8)}$$

Al final de todo el proceso de filtrado, los valores de la matriz de covarianzas deberían ser bajos, puesto que el objetivo del filtro es disminuir los errores y con esto la covarianza de los elementos del vector de estados (diagonal principal de *P*).

3.2 MAPEO Y TRANSFORMADA DE HOUGH

3.2.1 TRATAMIENTO DE DATOS PRELIMINAR AL MAPEO

Una vez que se han obtenido los datos por parte del sensor láser Hokuyo, y como se mencionará más adelante, para el uso de la transformada de Hough se necesita una imagen binarizada, teniendo que usarse la imagen final del mapeo realizado por el sistema.

Considerando que el láser brinda los valores de distancia en una matriz de 1x513 (como se describió anteriormente), se requiere calcular el ángulo al que se detecta dicha distancia, pues conociendo el índice de la matriz de distancias se puede determinar el ángulo, únicamente al multiplicar este índice por el valor de ángulo existente entre dos medidas consecutivas (0.351°) y realizarlo en un lazo repetitivo, el lazo que se utiliza para esto se detalla a continuación:

```
for i=1:514
ángulo(i)=(i-1)*0,351;
end
```

Luego de este lazo aparece una matriz de 513x1 que contiene los diferentes ángulos a los que se detectaron objetos, considerando siempre que la primera medida se llevara a efecto a los 0°, y como un lazo *for* no acepta que la variable empiece en cero se le resta uno a cada iteración. Ahora se requiere una transformación de la forma polar que se dispone *(r,Θ)* hasta llegar a la forma en coordenadas rectangulares *(x,y)*, que se logra tan solo utilizando funciones trigonométricas así:

$$x = r * \cos\theta \qquad \text{(Ec. 3.11)}$$

$$y = r * \sin\theta \qquad \text{(Ec. 3.12)}$$

Sin embargo, para este propósito es necesario hacer una concordancia entre los ejes del barrido del láser y el movimiento del robot, esto se debe a que la transformación a coordenadas rectangulares del marco de referencia global no corresponde con el marco de referencia del robot, tal como se nota en la FIGURA 3.1.

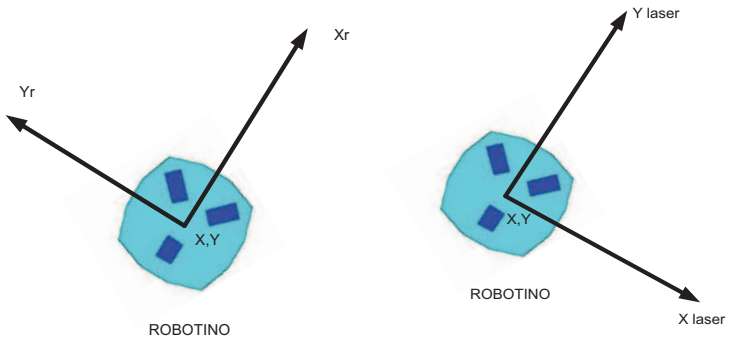


FIGURA 3.1. Izq. Ejes de coordenadas Robot, Der. Ejes de coordenadas Láser

Para lograr esto, se deberá realizar una corrección de ángulo para que las medidas coincidan con los ejes de referencia ya dispuestos, como se explica a continuación:

Medida 1:

Índice = 0, Angulo = 0°, Distancia = *r*

Transformación Ejes Láser:

$$x = r * \cos 0 \quad \text{Entonces } x=r$$

$$y = r * \sin 0 \quad \text{Entonces } y=0$$

Mientras que para los ejes del robot:

$$x = 0$$

$$y = -r$$

Por esta razón, se coloca el valor de (-90°), cabe recalcar el hecho de que todas las funciones de MATLAB operan de manera adecuada únicamente cuando los ángulos están expresados en radianes, así que es necesario realizar una transformación de grados sexagesimales a radianes (*pi/180°), después de realizar este ajuste las ecuaciones (Ec. 3.11) y (Ec. 3.12) quedan de la siguiente forma:

$$x = r * \cos\left(\theta - \frac{\pi}{2}\right) \qquad \text{(Ec. 3.13)}$$

$$y = r * \sin\left(\theta - \frac{\pi}{2}\right) \qquad \text{(Ec. 3.14)}$$

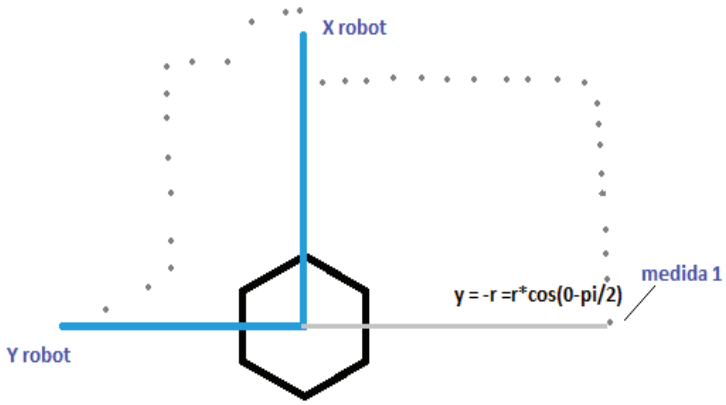


FIGURA 3.2 Concordancia de ejes Láser-Robot, Primera Medida

De esta manera todas las medidas expresadas en coordenadas cartesianas tienen como centro de coordenadas la posición del robot, es decir el marco de referencia local.

Para la realización del mapa del entorno en el cual se desenvuelve Robotino®, se ha limitado el valor máximo de detección, pasando del valor máximo anterior *r*=5.6m al nuevo valor de *r*=2.5m, de esta manera si los datos de distancia no se encuentran entre 0.2m < *r* < 2.5m el sistema los ignorará y no realizará ningún cálculo. Con esto se evita un solapamiento de datos, es decir, el sistema dejará de detectar puntos innecesarios y lejanos y se centrará en la observación y tratamiento de distancias cercanas evitando el riesgo de falsas lecturas que afecten el mapeo.

Para que durante la elaboración del mapa, éste se vaya construyendo continuamente a pesar del movimiento ejecutado por Robotino®, esto implica que

se traslade y/o rote y que no afecte el mapa, se debe considerar el uso de una matriz de rotación y traslación para que así, todos los valores de distancia expresados en coordenadas cartesianas sean referenciados al marco global. Una vez que se han realizado los procedimientos descritos hasta aquí, se obtienen los valores a ser dibujados como sigue:

$$xmapa = xx * \cos(tita) - yy * \sin(tita) + x \qquad \text{(Ec. 3.15)}$$

$$ymapa = xx * \sin(tita) + yy * \cos(tita) + y \qquad \text{(Ec. 3.16)}$$

Donde:

xx e *yy* corresponden a las medidas del láser expresadas en coordenadas cartesianas.

tita es el ángulo resultante de la corrección de los ejes robot-láser ($\theta - \frac{\pi}{2}$)

x e y corresponden los valores de la pose del robot una vez que ha pasado por el Filtro Extendido de Kalman (EKF) explicado anteriormente, en tal virtud los valores que se obtienen de éste (*x*, *y* ,*Θ*), se utilizan para realizar un mejor mapeo.

Es necesario conocer que además de la elaboración del mapa, las distancias detectadas, rotadas y trasladadas de acuerdo al movimiento y pose del robot, son almacenadas con el método denominado memoria dinámica, el cual se presenta como sigue:

```
x1=[x1;xx*cos(tita)-yy*sin(tita)+x];
y1=[y1;xx*sin(tita)+yy*cos(tita)+y];
```

En el extracto tomado del programa que gobierna este sistema, aquí se considera a *x1 e y1* como dos matrices que almacenan el valor anterior y actualizan estos datos con cada iteración, de esta manera al final del programa se obtendrá lo siguiente:

$$x1 = \begin{bmatrix} x1_1 \\ x1_2 \\ x1_n \end{bmatrix} \qquad y1 = \begin{bmatrix} y1_1 \\ y1_2 \\ y1_n \end{bmatrix}$$

Esta forma de almacenar se lleva a efecto debido a que en la matriz se ha colocado: el valor anterior (*x1* e *y1*), y luego la actualización del cálculo de estos separados por punto y coma (;), pero si se quiere otra forma de almacenamiento se deberá escribir el valor anterior, espacio ( ) y al final la actualización del cálculo como se muestra a continuación:

```
x1=[x1 xx*cos(tita)-yy*sin(tita)+x];
y1=[y1 xx*sin(tita)+yy*cos(tita)+y];
```

Habiendo escrito de esta manera las expresiones, la memoria dinámica proporciona los siguientes resultados:

$$x1 = [x1_1 \quad x1_2 \quad x1_n]$$

$$y1 = [y1_1 \quad y1_2 \quad y1_n]$$

Los procedimientos que se han descrito se desarrollan en la función llamada *datosláser*; las funciones serán descritas mediante un diagrama de flujo en el Anexo D.

Una vez que se ha logrado el mapeo por completo, la siguiente fase es analizar este mapa a través de la herramienta de la Transformada de Hough, con el fin de lograr la extracción de características que se puedan encontrar en el mapa.

3.2.2 TRANSFORMADA DE HOUGH

La Transformada de Hough es una técnica que brinda la posibilidad de detectar figuras dentro de una imagen, resulta ser muy robusta con respecto al ruido, sin embargo, previamente se debe obtener una imagen binaria de los pixeles que forman el objeto.

Esta técnica fue patentada y propuesta por Paul Hough en el año de 1962, en primera instancia solo se limitaba a la detección de rectas en una imagen, pero con el transcurrir del tiempo, se extendió hasta llegar a la identificación de cualquier figura que se pueda describir con parámetros, tales como circunferencias y elipses. En el presente proyecto se hace énfasis en el estudio del reconocimiento de líneas. [43, 44]

3.2.2.1 **Detección de Líneas Rectas**

La Transformada de Hough en su caso más simple contempla la detección de líneas rectas, así en el espacio de la imagen adquirida se define una recta con la ecuación más simple que se conoce:

$$y = mx + b \qquad \text{Ec. (3.9)}$$

Donde:

m: es la pendiente de una recta

b: es el corte de dicha recta con el eje de las coordenadas

De esta manera se considera la posibilidad de representar una recta de la forma (*m*, *b*), sin embargo, al encontrarse una recta vertical, esta representación es inconsistente, siendo necesario describir la recta en coordenadas polares (*r*, *Θ*), la variable r corresponde a la distancia entre el origen y la recta; asimismo el término *Θ* es el ángulo del vector dirección de la recta perpendicular a la recta original y que pasa por el origen.

De la siguiente parametrización, la ecuación de una recta puede escribirse así:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right) * x + \left(\frac{r}{\sin\theta}\right) \qquad \text{Ec. (3.10)}$$

Y luego ser reinscrita de la siguiente forma:

$$r = x * \cos\theta + y * \sin\theta$$

Siendo posible asociar a cada par de (*r*, *Θ*) una recta.

3.2.2.2 **Implementación**

Para la implementación de la Transformada de Hough, es necesaria una matriz que se denomina acumulador, la dimensión de esta matriz corresponde al número de parámetros desconocidos en el problema. Con esta consideración, para determinar una recta de la forma de la Ec. 3.9, se necesitará una dimensión de dos, pues las dos dimensiones corresponden a los valores cuantificados de (*r*, *Θ*).

Es necesario discretizar los parámetros que describen la figura a fin de construir el acumulador. Cada celda que tenga el acumulador representa una figura de la cual sus parámetros serán obtenidos a partir de la posición de la celda.

Para cada punto que tenga la imagen se debe buscar todas las posibles figuras a las que este punto pueda pertenecer, situación que se logra al buscar todas las combinaciones para los parámetros de la figura; de ser así se procede a calcular los parámetros de la misma para después encontrar la posición en el acumulador correspondiente, incrementando el valor en esa posición.

La búsqueda de figura se logra con la posición del acumulador con mayor valor. A continuación de muestra la descripción de un algoritmo de detección de rectas en una imagen:

1. Cargar una imagen

2. Binarizar la imagen

3. Detectar los bordes en la imagen

4. Para cada punto de la imagen

    Si (x,y) ϵ a un borde

        Para todo posible ángulo Θ

            Calcular r con un ángulo Θ

            Incrementar la posición (r, Θ)

5. Buscar las posiciones con mayores valores en el acumulador

6. Devolver las rectas cuyos valores son los mayores en el acumulador. [43]

3.2.2.3 **Descripción del Algoritmo**

Para el uso de la Transformada de Hough se han usado los siguientes scripts de MATLAB: *hough.p, houghpeaks.p, houghlines.p*; su forma de uso se explica a continuación:

3.2.2.3.1 Hough.p

Mediante este script de MATLAB se obtiene la transformada de Hough de una imagen binarizada, se tienen los siguientes parámetros de entrada y salida:

function [h,theta,rho]=hough(f,dtheta,drho)

h: Transformada de Hough

tetha: Vector que contiene los valores de *tetha*.

rho: Vector que contiene los valores de *rho*.

f: Imagen binaria de bordes

dtetha: Espacio en grados (°) de la Transformada de Hough a lo largo del eje *tetha*.

drho: Espacio de la Transformada de Hough a lo largo del eje *rho*.

3.2.2.3.2 Houghlines.p

Logra la extracción de segmentos de línea basándose en la Transformada de Hough.

functionlines=houghlines(f,theta,rho,rr,cc,fillgap,minlength)

f: Imagen binarizada a la cual se le aplicará la Transformada de Hough.

theta, *rho*: Vectores que son devueltos por hough.p

rr, *cc*: Filas y columnas donde buscar los segmentos de la transformada de Hough.

Fillgap: Separación de pixeles desde los cuales se considera distinto segmento (Por defecto 20).

Minlength: Mínimo número de pixeles que un segmento debe tener para que se considere segmento (Por defecto 40).

Lines: Estructura que se devuelve con el número de segmentos encontrados, contiene los siguientes ítems:

- *Point1*: Punto Inicial del segmento detectado, vector de dos elementos (*x*, *y*)
- *Point2*: Punto Final del segmento detectado, vector de dos elementos (*x*, *y*)
- *Length*: Distancia existente entre *Point1* y *Point2*.
- *tetha*, *rho*: Parámetros *tetha* (en grados °) y rho de la ecuación de la recta en la que el segmento detectado se encuentra, expresada en forma polar. [45]

Estos scripts son utilizados una vez que el mapa ha sido elaborado, o las matrices *x1* e *y1* han sido llenadas con los valores detectados expresados en coordenadas cartesianas, de esta manera se procede al tratamiento de estos datos a fin de encontrar líneas en el mapa y las posibles esquinas existentes. Para esto se requiere binarizar la imagen que se generara con los valores que contienen dichas matrices. Este procedimiento se describe a continuación con la función *esquina* que forma parte del algoritmo de detección de esquinas.

```
*************************************************************************
                        Extracto de la función Esquina
*************************************************************************
      xh=x1;yh=y1;
      %Distancia de grillado
      dgrid=0.06;
      %Grillado en x,y
      gx=-12:dgrid:12;gx=gx';
      gy=-12:dgrid:12;gy=gy';
      %Indices para llenar M con 1s para hacer la imagen
      indicex=zeros(length(gx),1);
      indicey=zeros(length(gy),1);
      %Creación de la imagen binarizada
      xr=roundn(xh,-2);
      yr=roundn(yh,-2);
      M=ones(length(gy),length(gx));
      for k=1:length(xh)
```

```
            [~,indicex(k)]=min(abs(xr(k)-gx));
            [~,indicey(k)]=min(abs(yr(k)-gy));
            M(indicey(k),indicex(k))=0;
        end
        M=logical(M);
**************************************************************************
```

En el extracto anterior se muestra la forma de generar una imagen binarizada a partir de los datos almacenados en *x1* e *y1,* en primera instancia se genera un plano lleno de cuadrículas, para luego crear una matriz con los índices a ser llenados con unos (1), luego se hace un redondeo de los valores de *xh* e *yh* (antes *x1* e *y1*), para finalmente realizar una resta entre cada unos de los valores con las cuadrículas, y encontrar el mínimo valor, es decir, el índice del valor más pequeño o más cercano a cada cuadrícula pasará de ser uno a ser cero, y eso se guardará.

3.2.2.3.3 Determinación de Esquinas

Una vez realizada la imagen binarizada, se procede a la detección de bordes previa la aplicación de los scripts descritos anteriormente, esto se logra con dos métodos establecidos en MATLAB como son: Método Roberts y Método Prewitt.

Estos métodos responden a la siguiente nomenclatura:

```
bw1=edge(M,'roberts','vertical',0.1);
```
Método Roberts

```
bw2=edge(M,'prewitt','vertical',0.1);
```
Método Prewitt

Donde:

M: es la imagen binarizada

Es importante destacar que estos métodos se aplicaron por separado pero a la misma imagen binarizada, pues los resultados experimentados se complementaban, pues ciertas líneas existentes en la imagen se detectaban con un método y otras se detectaban con otro método, razón fundamental por la que se optó por someter a las imágenes a los dos métodos, para recoger todos los datos de líneas al final del tratamiento de la imagen.

Luego se procede a aplicar los scripts de MATLAB que aplican la Transformada de Hough como sigue:

```
[H2,T2,R2]=hough(bw2);
peaks2=houghpeaks(H2,15);
lines2=houghlines(bw2,T2,R2,peaks2);
```

Después de esta secuencia de comandos, se consigue detectar las líneas que se encuentran en la imagen, y solo resta dibujar las estructuras que se han obtenido, pues como se citó anteriormente los scripts devuelven estructuras que contienen los puntos inicial y final de la línea; el algoritmo está elaborado para que una vez que se ha detectado por lo menos una línea, separe las estructuras y las convierta en puntos listos para ser dibujados.

Después de tener gran cantidad de líneas detectadas, lo que resta es encontrar probables esquinas, esto es encontrar las intersecciones de todas y cada una de las líneas, es decir, examinar todas y cada una de las combinaciones posibles entre líneas. Para esto, el algoritmo contempla las combinaciones a partir del uso de la función propia de MATLAB llamada *nchoosek,* la cual genera todas las combinaciones posibles a partir de los vectores en cuestión, adicionalmente una vez que se realizan las combinaciones, se procede a la revisión de las condiciones necesarias para determinar intersecciones entre líneas, pues se debe considerar si las líneas tienen pendiente (sea esta positiva o negativa), si son paralelas a cualquier eje de coordenadas (pendiente igual a cero o pendiente igual a infinito).

Lo siguiente es descartar las posibilidades de combinación entre líneas que sean paralelas entre sí, y adicionalmente, evaluar el ángulo existente entre dos rectas pues a pesar de existir intersección, si el ángulo resulta muy grande o muy pequeño no se considerará como una esquina. El ángulo se lo determina con la siguiente forma una vez calculadas las pendientes:

$$tan\,\emptyset = \left|\frac{m2 - m1}{1 + m2 * m1}\right|$$

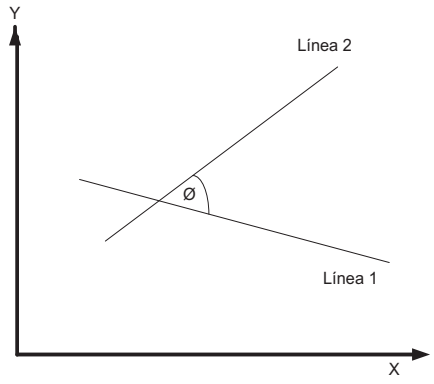


FIGURA 3.3 Típica intersección entre dos Líneas

En la FIGURA 3.3 se observa el ángulo existente entre dos líneas a ser evaluado para descartar falsas esquinas; en el algoritmo se hace un filtro para que no se consideren ángulos menores a 85°.

Ahora, siendo la forma de expresar una línea recta basada en la pendiente *m* y el corte en el eje de las abscisas *b*:

$$y = m * x + b$$

Se debe considerar que existen varias posibilidades de rectas como son:

- $y = b = constante$

En esta línea la pendiente *m* es 0 y es paralela al eje x.

- $x = constante$

El valor de *m* es infinito y no existe corte en el eje de y.

La respuesta que entrega el algoritmo (las esquinas), viene expresada en un vector que contiene los valores de *x* e *y* del punto de intersección.

A continuación se muestran las diversas intersecciones que se pueden encontrar considerando todos los tipos de líneas:

Caso 1:

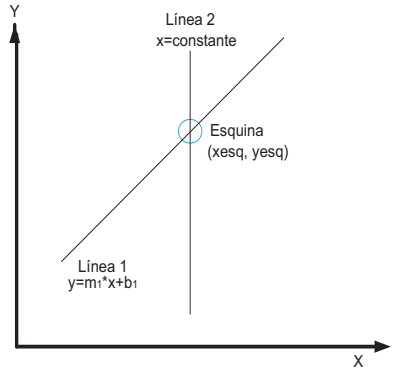


FIGURA 3.4 Intersección entre una línea común y una línea paralela al eje y

La respuesta que se ha considerado para el Caso 1 es:

Línea 1:

$$y = m1 * x + b1$$

Línea 2:

$$x = constante$$

Intersección:

$$xesq = constante$$

$$yesq = m1 * xesq + b1$$

Caso 2:

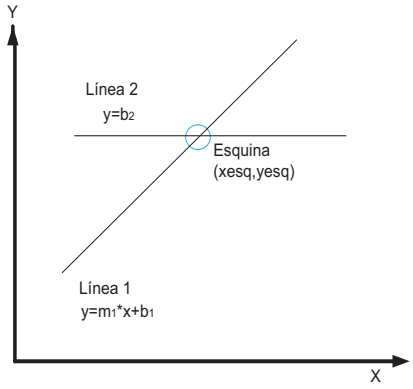


FIGURA 3.5 Intersección entre una línea común y una línea paralela al eje x.

La respuesta que se ha considerado para este caso es:

Línea 1:

$$y = m1 * x + b1$$

Línea 2:

$$y = b2$$

Intersección:

$$yesq = b2$$

$$xesq = \frac{b2 - b1}{m1}$$

Caso 3:

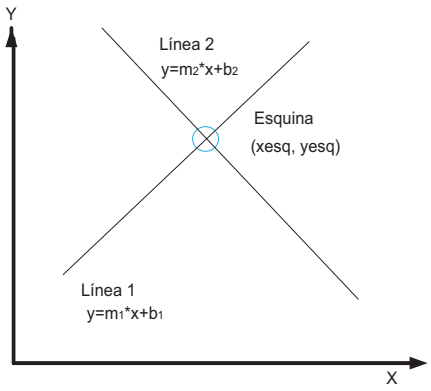


FIGURA 3.6 Intersección entre dos líneas comunes.

La respuesta que se ha considerado para este caso es:

Línea 1:

$$y = m1 * x + b1$$

Línea 2:

$$y2 = m2 * x + b2$$

Intersección:

$$xesq = \frac{b2 - b1}{m1 - m2}$$

$$yesq = m1 * xesq + b1$$

Caso 4:

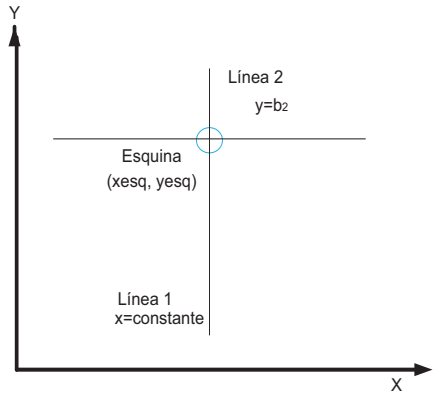


FIGURA 3.7 Intersección entre una línea paralela al eje x y una línea paralela al eje y

La respuesta que se ha considerado para este caso es:

Línea 1:

$$x = constante$$

Línea 2:

$$y2 = b2$$

Intersección:

$$xesq = constante$$

$$yesq = b2$$

Una vez contempladas las posibilidades y encontradas las esquinas resultantes de todas las combinaciones, se procede a revisar el vector que contiene los valores de *x* e *y* de cada punto (esquina), a fin de observar si existen valores repetidos y eliminarlos.

En el Anexo D, se encuentran los diagramas de flujo que describen cada uno de los procesos que posee este algoritmo, como son: *Modelo, Mapeo, Filtro de Kalman, Esquina* y el programa principal que es aquel que llama a las funciones descritas anteriormente.

3.2.3 ANÁLISIS DE LA POSIBILIDAD DE UN MAPEO EN TRES DIMENSIONES

En este proyecto de titulación se ha realizado mapeo y localización en dos dimensiones, para este efecto se utiliza un sensor láser, tal como se describió en este capítulo, sin embargo, para que sea posible la realización de un algoritmo capaz de reproducir ambientes en tres dimensiones, se necesitaría del manejo de dos sensores láser, de esta manera, se plantea ocupar un láser para realizar un barrido horizontal (la parte frontal a Robotino®), y otro láser para lograr un barrido vertical (la parte superior).

Considerando esto, se debe tener en cuenta, que el tratamiento de los datos obtenidos por un sensor, debe ir sincronizado con el grupo de datos que se adquieran por el otro láser, en otras palabras, que el vector de distancias del barrido horizontal y el vector que contiene las distancias captadas durante el barrido vertical, sean obtenidos simultáneamente. Se debe notar, que esta particularidad incrementará aún más el tiempo de procesamiento, así como el coste computacional.

Otro punto a tomar en consideración radica en que, si bien es cierto, la plataforma robótica Robotino® según sus manuales y teniendo las versiones de los drivers adecuadas instaladas en su CF card, es capaz de soportar hasta dos *laser rangefinder*, al momento de poner a prueba esta afirmación, preliminarmente se constata que no se obtienen los datos de los dos sensores láser, sino que, cada uno de los mismos entrega los respectivos vectores que incluyen distancias, pero no lo pueden hacer de una manera coordinada, sino que lo hacen alternadamente.

Una vez detectados algunos de los inconvenientes que se presentan al intentar el mapeo en tres dimensiones con Robotino®, la posible solución consiste en la utilización de dos sensores láser siempre y cuando, se incorpore algún dispositivo extra, que logre la sincronización de los datos de los barridos, tanto vertical como horizontal. O a su vez, utilizar sensores de escáner láser 3D, los cuales reúnen información acerca de la geometría del entorno en el que desenvuelven, teniendo como inconveniente el costo de los equipos y el mayor tiempo de procesamiento de datos, a pesar de su gran precisión.

Asimismo, el uso de cámaras de tipo estéreo sería de gran utilidad para el tratamiento de imágenes, y mediante esto analizar cada imagen adquirida y finalmente encontrar los algoritmos de reconstrucción en tres dimensiones basados en la técnica de estereovisión.

La diferencia de un sistema de estereovisión con respecto a un sensor de escáner láser 3D, radica en que la cámara se fija en los aspectos del color de las superficies del entorno, mientras que como ya se menciono el escáner 3D analiza la geometría del entorno.

Dependiendo de la aplicación en cuestión, se puede optar por alguno de estos dos métodos, o algún otro existente en el mercado, teniendo en cuenta además las características del ambiente, el costo del equipo y la cantidad de datos a ser procesados, lo cual se reflejaría en el tiempo de procesamiento.

Si se desea conocer más acerca de Mapeo 3D, se puede encontrar información en [22].

CAPÍTULO 4

PRUEBAS Y RESULTADOS

En el presente capítulo se exponen algunas de las pruebas realizadas, tanto en el robot Pioneer 3-AT, como en Robotino®. En el primer caso, se tiene únicamente entornos creados en el programa Mapper 3D (Véase Anexo A) y, tanto el robot como el láser, simulados con el programa MobileSim (Véase Anexo A), usando las librerías ARIA y programados en C++. Estas pruebas sirvieron para dar los lineamientos básicos de los algoritmos para la posterior implementación de los mismos en Robotino® y en entornos reales.

En el caso de Robotino®, se muestra la implementación en entornos reales, filtrando la odometría y variando diferentes parámetros del filtro o parámetros de las funciones que permiten detectar las características del entorno (esquinas). Para presentar las pruebas y resultados se han tomado datos en dos ambientes diferentes, ambos estructurados y estáticos.

Es pertinente mencionar, que el uso de MobileSim compatible con la plataforma Pioneer 3-AT en lugar de un simulador propio de Robotino®, responde al hecho de las limitaciones que presenta Robotino®View, simulador propio de Robotino®, dada su versión de prueba, que es aquella con la que se cuenta en la Institución; en contraste con las facilidades encontradas en MobileSim, pues en este simulador se logro trabajar con el sensor laser, haciendo que el robot se desenvuelva en entornos desconocidos y los reconstruya como se muestra más adelante.

4.1 PRUEBAS PIONEER (SIMULACION)

Las pruebas que aquí se muestran fueron realizadas con el robot Pioneer 3-AT, trabajando en entornos simulados, los mismos que se diseñaron de forma que sean sencillos para poder procesar cantidades relativamente pequeñas de datos, de tal forma que el algoritmo base pueda ser analizado y depurado de una manera eficaz y eficiente, para que estas experimentaciones sean replicadas

posteriormente en entornos reales con Robotino®, objetivo principal de este proyecto.

En la FIGURA 4.1, se muestran algunos de los entornos construidos y la simulación del robot navegando en ellos, recogiendo datos para procesarlos posteriormente. Se puede ver el haz de luz que emite el láser, indicando el campo de visión del mismo durante el recorrido, además, cabe mencionar que el manejo del robot fue manual, controlado por el teclado del computador, tal como se describe en el Anexo A.

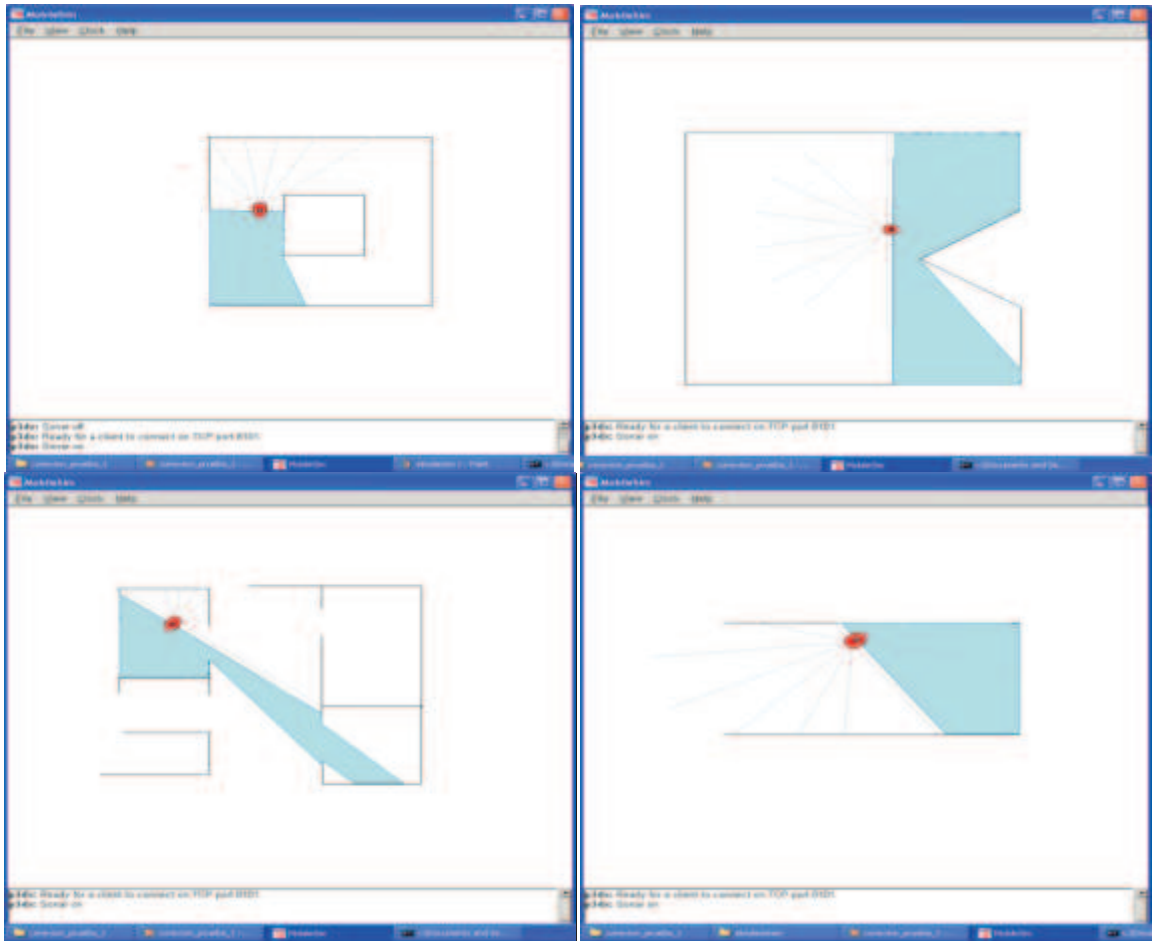


FIGURA 4.1 Entornos de simulación para el robot móvil PIONEER

A continuación, se presentan los resultados del procesamiento en MATLAB tanto para la construcción del mapa como para la detección de las características (esquinas), teniendo en cuenta que, como la simulación brinda datos “ideales”, el algoritmo en estas pruebas no usa el Filtro de Kalman, tomando los datos de odometría y láser, y trabajando con ellos directamente.

También se muestra la diferencia en el uso de los diferentes métodos para detección de bordes previo al uso de la Transformada de Hough (Ver Capítulo 3), en la parte de la detección de líneas. Se muestra claramente que, usando solo un método, las líneas encontradas no son todas las que corresponden al mapa (disminuyendo con ello el número de esquinas encontradas), mientras que con el uso de dos métodos juntos, las líneas encontradas son todas las que pertenecen al mapa.

Es necesario apuntar que los resultados mostrados dibujan los puntos tomados por el láser en azul, las líneas encontradas en verde, las esquinas detectadas como círculos magenta y la trayectoria seguida por el robot como puntos negros.

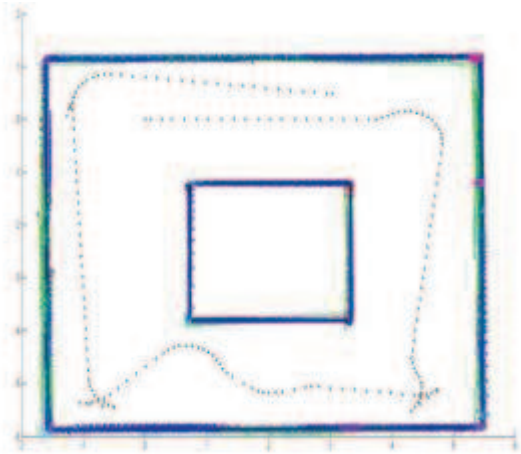


FIGURA 4.2 Resultado detección de bordes con método Prewitt para entorno de simulación uno

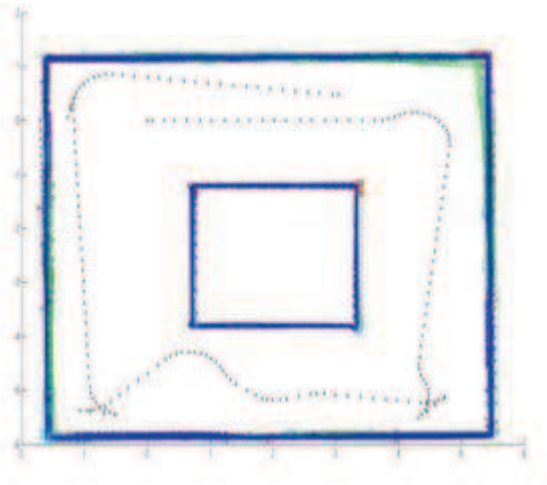


FIGURA 4.3 Resultado detección de bordes con método Roberts para entorno de simulación uno

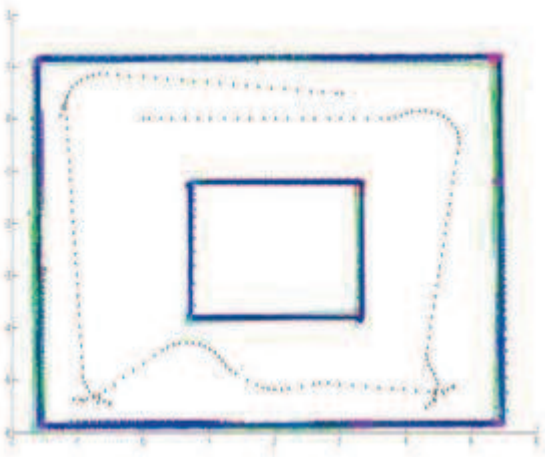


FIGURA 4.4 Resultado final para entorno de simulación uno

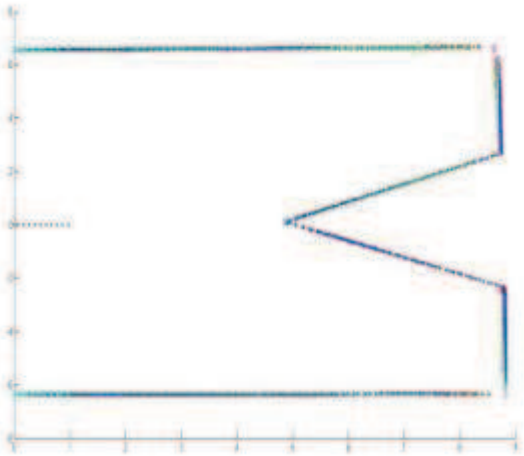


FIGURA 4.5 Resultado detección de bordes con método Roberts para entorno de simulación dos

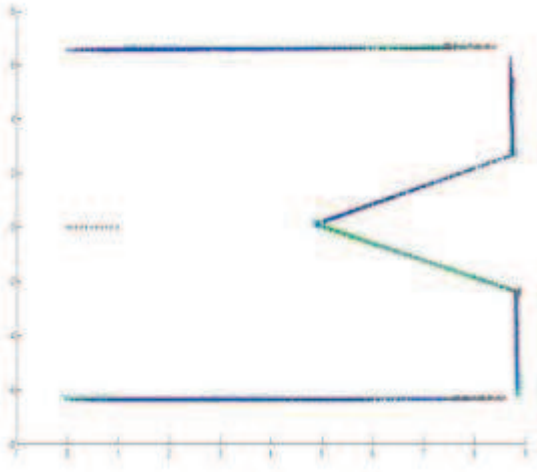


FIGURA 4.6 Resultado detección de bordes con método Prewitt para entorno de simulación dos

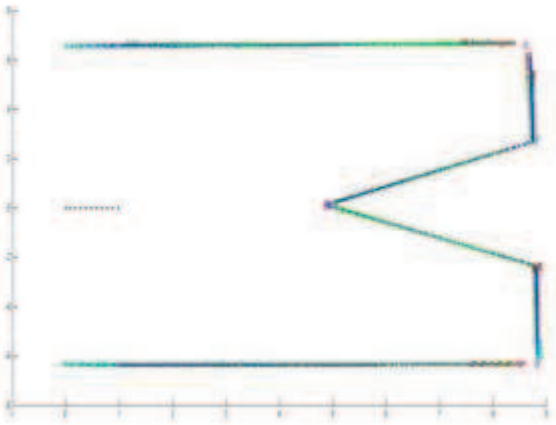


FIGURA 4.7 Resultado final para entorno de simulación dos

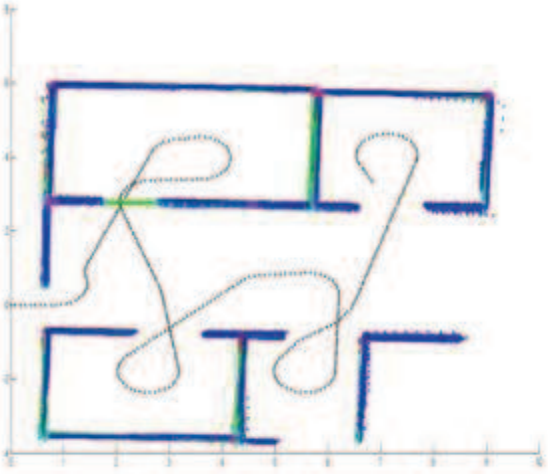


FIGURA 4.8 Resultado detección de bordes con método Roberts para entorno de simulación tres

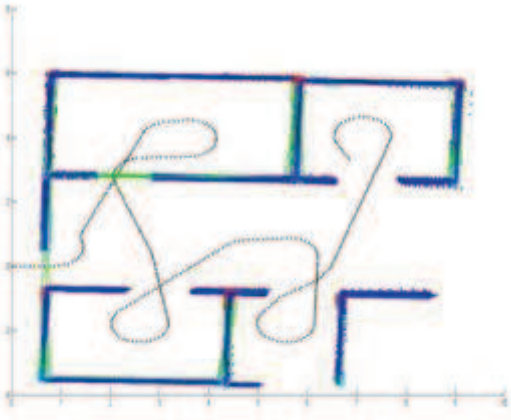


FIGURA 4.9 Resultado detección de bordes con método Prewitt para entorno de simulación tres

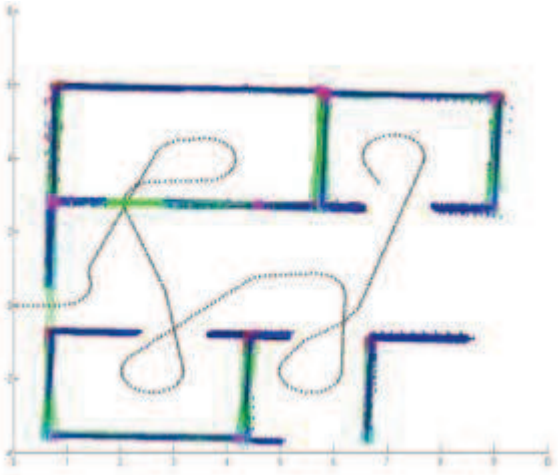


FIGURA 4.10 Resultado final para entorno de simulación tres

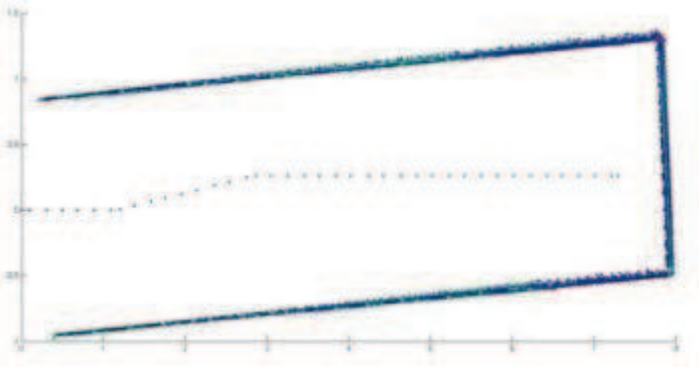


FIGURA 4.11 Resultado detección de bordes con método Roberts para entorno de simulación cuatro

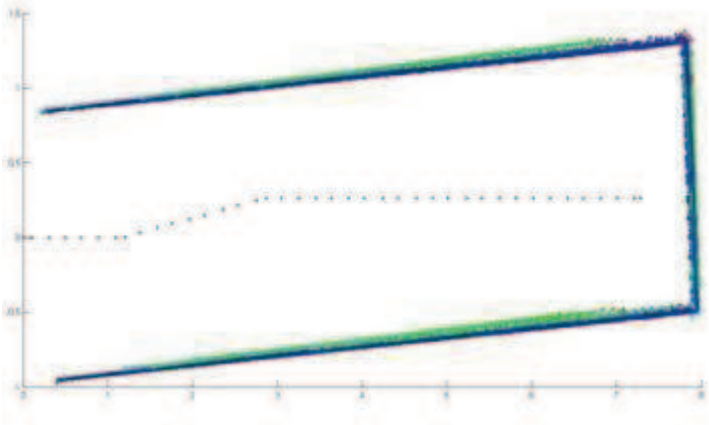


FIGURA 4.12 Resultado detección de bordes con método Prewitt para entorno de simulación cuatro

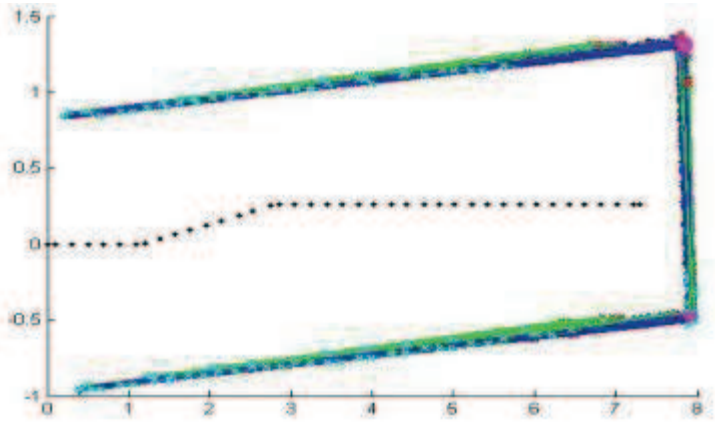


FIGURA 4.13 Resultado final para entorno de simulación cuatro

Se habrá notado que en algunos casos en los resultados finales, no todas las esquinas son detectadas, ni todas las detecciones realizadas corresponden a esquinas, esto no es muy crítico, mientras el número de falsas esquinas detectadas sea bajo y el número de esquinas verdaderas sea alto, ya que en un algoritmo de SLAM (Ver Capítulo 3), es el mismo algoritmo quien debe descartar posibles características falsas o muy “débiles” como para tomarlas en cuenta para la localización [35].

4.2 PRUEBAS ROBOTINO®

Estas pruebas fueron realizadas en dos ambientes dentro de la Escuela Politécnica Nacional, en la Facultad de Ingeniería Eléctrica y Electrónica; el primero, un pasillo en las afueras del Laboratorio de Sistemas de Control Automático, y el segundo, en las afueras del Aula Magna de la Facultad.

FIGURA 4.14 Entornos de Prueba
Izq. Pasillo afueras del Laboratorio de Control Automático Der. Pasillo afueras Aula Magna

4.2.1 FILTRO DE KALMAN

La idea del presente proyecto es filtrar los datos provenientes de la odometría del robot mediante un Filtro de Kalman Extendido, por lo cual se comparan los resultados entre el procesamiento usando únicamente los datos de la odometría, y el procesamiento usando el Filtro de Kalman; se puede apreciar claramente que, la odometría entrega datos relativamente aceptables hasta que en cierto punto los errores acumulativos entorpecen el procesamiento y el mapa comienza a desviarse de su correcta construcción. Por otro lado, cuando se le aplica el filtro, se tiene que el error se contrarresta y el mapa adquiere una mejor forma.

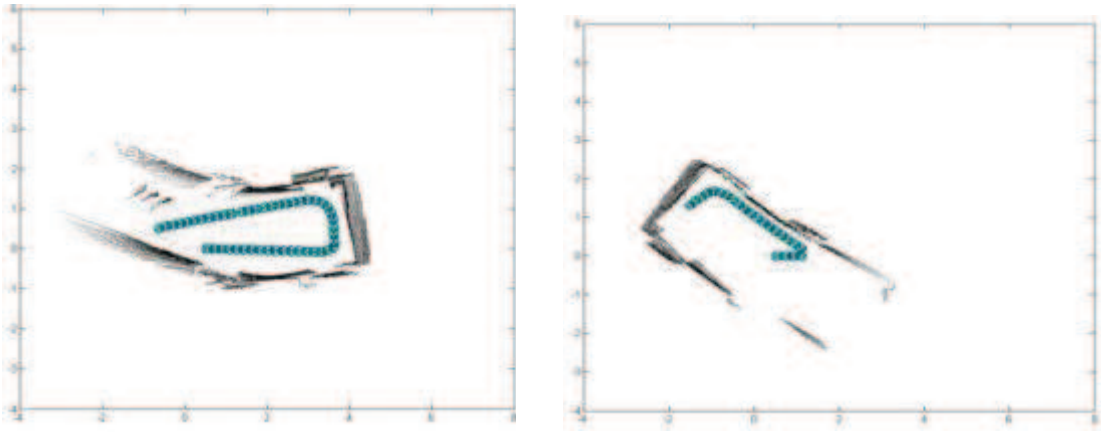


FIGURA 4.15 Mapas construidos: Izq. Primer entorno reconstruido con odometría; Der. Primer entorno reconstruido con datos filtrados

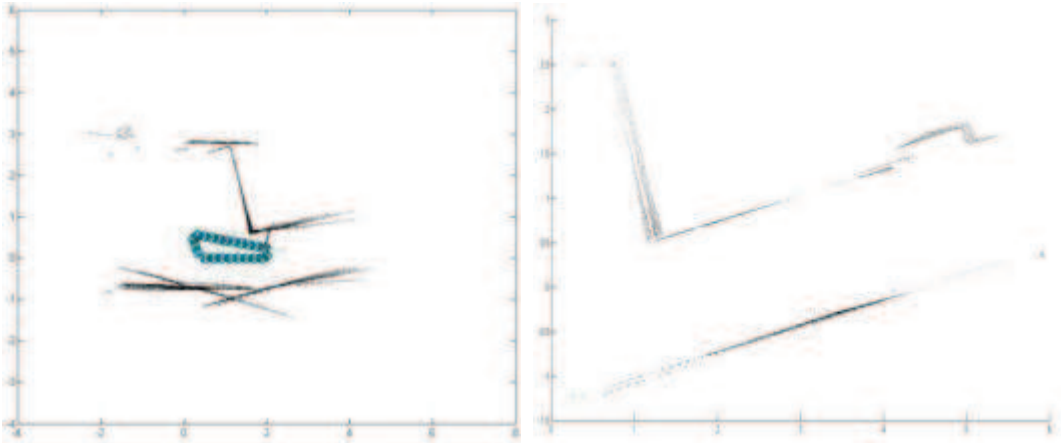


FIGURA 4.15 Mapas construidos: Izq. Segundo entorno reconstruido con odometría; Der. Segundo entorno reconstruido con datos filtrados

4.2.1.1 **Constantes del Filtro de Kalman**

En esta prueba se muestran los resultados al cambiar las constantes del Filtro de Kalman para la inicialización de las matrices de covarianza tanto de la observación (***R***) como en el proceso (***Q***). Estas variables contienen la ponderación del error presente en cada caso, los valores que arrojaron resultados más satisfactorios fueron encontrados vía prueba y error, considerando el criterio que si se les asigna valores pequeños se tiene perturbaciones pequeñas y si se entrega valores grandes la perturbación será considerable.

Cabe mencionar que esta no es la única forma de encontrar estos valores, ya que se puede realizar un análisis estadístico, tomando mediciones que indiquen la cantidad de perturbación presente tanto en el sistema como en la medición y con ello encontrar los mejores valores ***R*** y ***Q***.

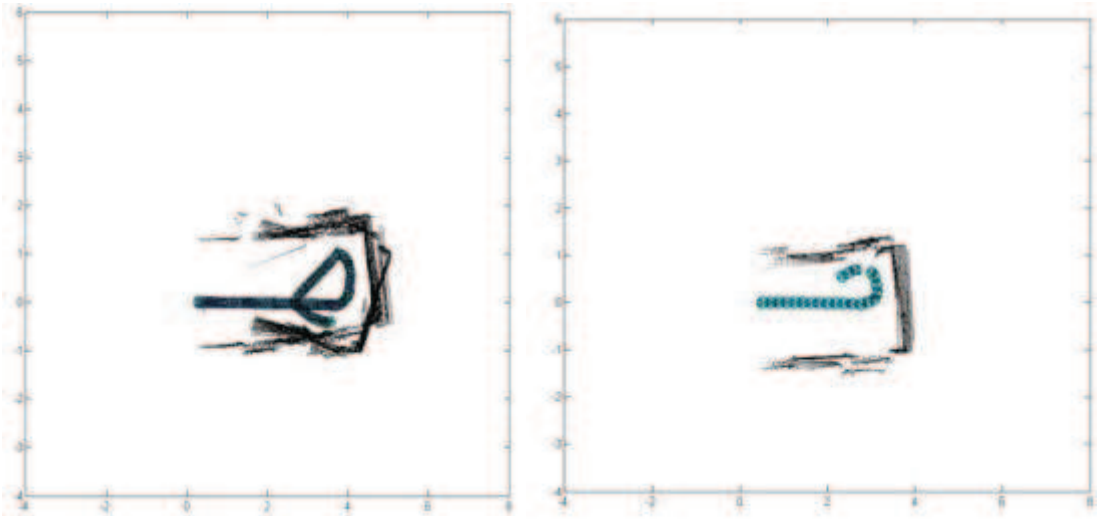


FIGURA 4.16 Mapas construidos usando el Filtro de Kalman; Izq. Primer entorno reconstruido con constantes incorrectas; Der.
Primer entorno reconstruido con constantes correctas

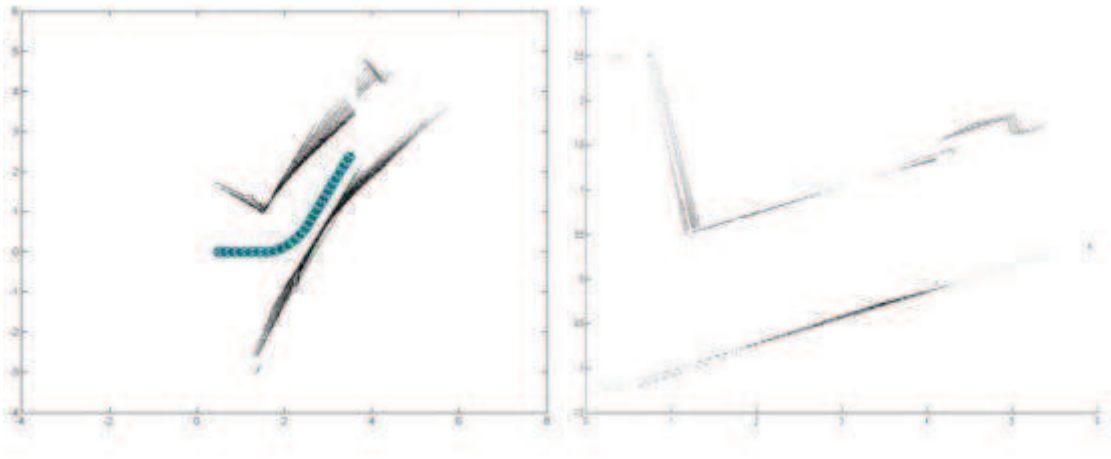


FIGURA 4.17 Mapas construidos usando el Filtro de Kalman; Izq. Segundo entorno reconstruido con constantes incorrectas; Der.
Segundo entorno reconstruido con constantes correctas

4.2.2 PRUEBAS EN ENTORNOS NO ESTÁTICOS

Además de las experimentaciones realizadas en los entornos estructurados y estáticos, se hicieron pruebas cuando personas se encontraban circulando en el ambiente de prueba, esto se puede notar en las marcas existentes dentro del mapa realizado. Esto es algo crucial para el algoritmo, puesto que las personas pueden ser confundidas con características, lo que llevaría a un error en la detección de las mismas, por esta razón el presente proyecto está orientado a entornos estáticos y estructurados.

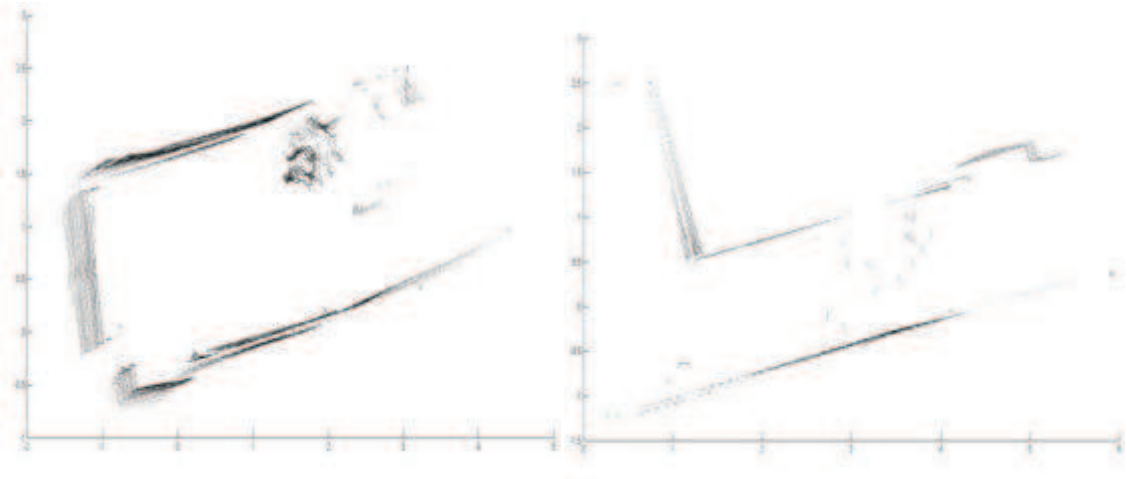


FIGURA 4.18 Mapas construidos en entornos no estáticos; Izq. Mapa del primer entorno; Der.
Mapa del segundo entorno

4.2.3 RESULTADOS FINALES

A continuación, se muestran los mapas finales construidos para los dos entornos reales, en los cuales se han afinado las constantes del Filtro de Kalman, y se ha logrado localizar al robot, cuya posición final se almacena en el vector llamado ***Xcorregida***, existente en el programa principal. Así como, un vector que contiene todas las características expresadas en valores cartesianos, tanto la posición como las esquinas están relacionadas al marco de referencia global.

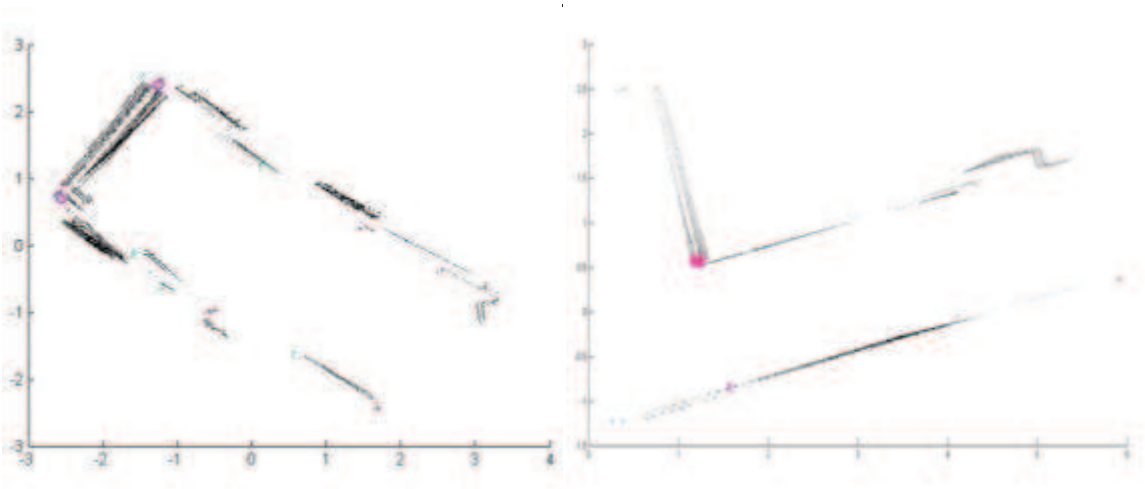


FIGURA 4.18 Mapas Finales construidos; Izq. Mapa del primer entorno; Der. Mapa del segundo entorno

4.3 ANÁLISIS DE RESULTADOS LOCALIZACIÓN

El presente análisis tiene como objetivo contrastar los datos obtenidos con odometría, el modelo del robot, y los datos filtrados con los datos medidos directamente en el entorno de trabajo, y así verificar la validez del algoritmo implementado.

Se analiza la posición final del robot y la precisión en la detección de esquinas luego de un recorrido, también se compara la posición del robot en cada iteración del algoritmo durante un mismo recorrido, teniendo los resultados que se muestran a continuación:

TABLA 4.1 Valores obtenidos en 4 pruebas en entornos reales.

		Odometría	Modelo	Kalman	Real
1	**x (m)**	2,887	2,45	2,648	2,66
	y (m)	0,6214	0,449	0,611	0,61
	tetha (°)	42,68	42,69	42,7	45
2	**x (m)**	2,978	2,572	2,748	2,78
	y (m)	0,32	0,137	0,3	0,38
	tetha (°)	45,97	46,32	45,980	45
3	**x (m)**	3,155	2,615	2,893	2,92
	y (m)	-0,121	-0,089	-0,107	-0,2
	tetha (°)	-3,85	-3,71	-3,83	-10
4	**x (m)**	3,162	2,616	2,9	2,88
	y (m)	-0,095	-0,066	-0,081	-0,15
	tetha (°)	-3,08	-3,29	-3,06	-5

En la TABLA 4.1 se muestran los valores obtenidos tanto de odometría, con el modelo y luego del Filtro de Kalman comparados con los valores reales medidos directamente, se puede apreciar que luego del filtrado, los datos tienden a aproximarse al valor real, a pesar que los datos de odometría difieren considerablemente de los valores reales.

TABLA 4.2 Cálculo de Errores en Pose del robot

Prueba		% Error A	% Error B	% Error C
1	**x (m)**	8,53	7,89	0,45
	y (m)	1,87	26,39	0,16
	tetha (°)	5,16	5,13	5,11
2	**x (m)**	7,12	7,48	1,15
	y (m)	15,79	63,95	21,05
	tetha (°)	2,16	2,93	2,18
3	**x (m)**	8,05	10,45	0,92
	y (m)	39,50	55,50	46,50
	tetha (°)	61,50	62,90	61,70
4	**x (m)**	9,79	9,17	0,69
	y (m)	36,67	56,00	46,00
	tetha (°)	38,40	34,20	38,80

Adicionalmente se realiza una comparación entre la ubicación de las esquinas en el entorno real y en el mapa realizado, teniendo que los resultados obtenidos con la odometría entregan mayor cantidad de error, debido a la acumulación de error en los sensores; en cambio con los valores corregidos con el Filtro de Kalman se reducen la cantidad de error.

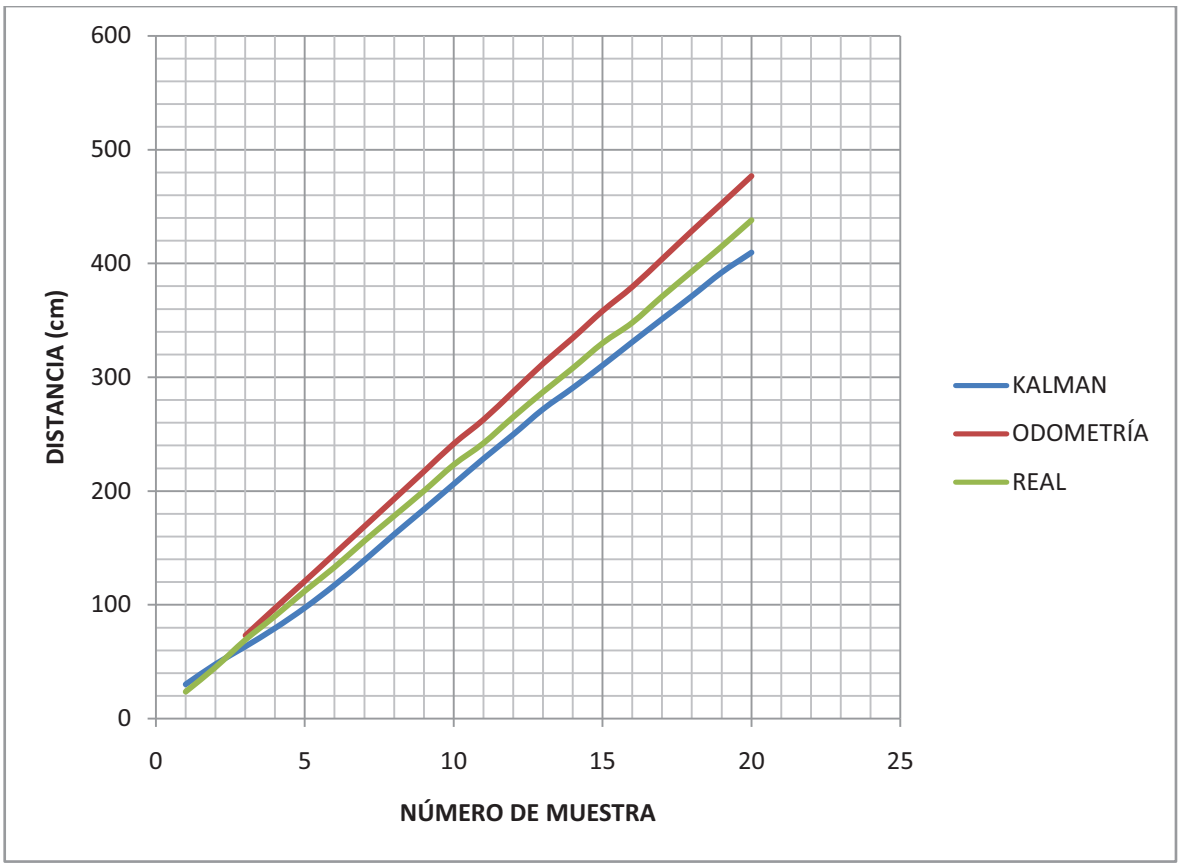
TABLA 4.3 Ubicación de Esquinas en el Primer Entorno

		Real	Estimada Odometría	% Error	Estimada Kalman	% Error
Prueba 1	**Xesq1**	6,43	6,86	6,69	6,49	0,93
	Yesq1	0,88	0,69	21,59	0,69	21,59
	Xesq2	6,47	6,58	1,70	6,52	0,77
	Yesq2	-1,3	-1,52	16,92	-1,57	20,77
Prueba 2	**Xesq1**	6,43	6,93	7,78	6,5	1,09
	Yesq1	0,88	0,79	10,23	1,02	15,91
	Xesq2	6,47	6,57	1,55	6,33	2,16
	Yesq2	-1,3	-1,45	11,54	-1,4	7,69

La FIGURA 4.19 muestra la posición del robot en cada iteración del algoritmo cada 5 segundos, mostrando los valores de desplazamiento obtenidos directamente por la odometría, como con el Filtro de Kalman, así como los valores reales medidos in situ.

Aquí se puede observar que si el número de iteraciones continúa creciendo, la línea de tendencia de la odometría difiere demasiado de la línea de tendencia de los valores reales, mientras que con el Filtro de Kalman se tiene un mejor resultado. Basados en la gráfica se puede afirmar que luego de que los valores son procesados, el algoritmo implementado corrige la odometría, mejorando la localización, pero debido a las condiciones propias del robot estos resultados no son los ideales. Se puede justificar eso debido a que existen muchas situaciones que no se han considerado y que influyen como por ejemplo el deslizamiento de las ruedas, el tipo de piso, nivel de baterías del robot, deterioro de sensores del robot, etc

FIGURA 4.19 Odometría vs. Medidas Reales



CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

Una vez terminado este trabajo, es decir habiendo analizado varios métodos de navegación y mapeo, y al implementar un método basado en el Filtro Extendido de Kalman para la localización y, la Transformada de Hough para el mapeo, se pueden describir las siguientes conclusiones y recomendaciones, a fin de resumir todo lo realizado en este Proyecto de Titulación.

5.1 CONCLUSIONES

- El control en lazo abierto de Robotino® es muy impreciso y se debe necesariamente cerrar el lazo para obtener, por ejemplo, un posicionamiento exacto.
- Los motores de Robotino® tienen una caja reductora, esta relación debe tomarse en cuenta al momento de calcular la pose (x, y, Θ) utilizando las ecuaciones del modelo del robot, puesto que las velocidades que se necesitan para este cálculo son la de las ruedas, más no la de los motores que es lo que se obtiene a partir del software de Robotino®.
- La plataforma robótica Robotino® es una muy buena herramienta de enseñanza pues se encuentra dotada de diferentes tipos de sensores como son infrarrojos, inductivos, encoders, cámara, salidas y entradas tanto analógicas como digitales, y además, la posibilidad de acoplar otros sensores que mejoran su capacidad de desenvolvimiento en entornos desconocidos, como por ejemplo, el láser Hokuyo URG-04LX-UG01 y el sensor de navegación NorthStar.
- La versatilidad en cuanto al lenguaje de programación de Robotino® es una gran ventaja, pues dependiendo de las necesidades de la aplicación a realizar se puede utilizar tal o cual lenguaje, como por ejemplo una interfaz gráfica de programación a través de LabVIEW o el mismo Robotino®View,

software propio de la marca Festo; o si se tiene una mayor familiarización con MATLAB, como es el caso del presente Proyecto de Titulación.

- La posibilidad de establecer una comunicación inalámbrica a través de la red Wireless propia de Robotino® es una gran ventaja pues elimina el hecho de tener un medio físico tal como un cable, sin embargo, esto incurre en una desventaja por el hecho de que esta comunicación se ve afectada por un sinnúmero de problemas como es la contaminación WiFi, esto quiere decir que en presencia de otras redes inalámbricas el desempeño del robot no es el óptimo, pues al trabajar en este tipo de ambientes la interferencia afecta a la emisión y recepción de datos.

- El Filtro Extendido de Kalman es un algoritmo muy útil para dar solución al problema de localización, pues reduce la incertidumbre en cuanto a la pose del robot (x, y, Θ) al desplazarse, adicionalmente disminuye notablemente los errores acumulativos que se generan en la odometría.

- Una de las ventajas del Filtro de Kalman es su relativamente sencilla implementación, en comparación por ejemplo, con el Filtro de Partículas, el cual en contraste es más robusto para resolver el problema de localización.

- La calibración de las matrices de covarianza en el algoritmo del Filtro de Kalman es un factor que genera una sensibilidad muy alta con respecto al cambio de dichas constantes, razón por la cual, al realizar un mínimo cambio en alguna de ellas los resultados cambian drásticamente.

- Si bien el sensor utilizado brindó buenos resultados, el mismo presenta la limitación de verse afectado por la cantidad de luz del ambiente y los colores de los objetos circundantes; para eliminar esto se pueden usar otros sensores como sonares, cámaras, etc.

- El sensor láser Hokuyo URG-04LX-UG01, con el que se trabajó durante este proyecto, posee buenas características como su distancia de alcance igual a 5.6m; no obstante hay que reconocer que tiene una zona muerta de 0.20m, es decir, que a pesar de que un objeto se encuentre muy cercano, el sensor láser puede o no detectarlo, asimismo otra desventaja es el

hecho de que solo puede ser usado en interiores y que a pesar de todos estos puntos en contra la fuente requerida para su funcionamiento debe tener una gran capacidad de corriente, pues al arrancar necesita de 1.2A.

5.2 RECOMENDACIONES

- Si se desea obtener datos de odometría más precisos se podría añadir sensores exteroceptivos, tal como el sistema de navegación NorthStar, con el que cuenta el Laboratorio de Sistemas de Control de la Institución, sin embargo, se debe considerar que al hacer uso de este sensor si bien es cierto mejorarían los datos de la pose del robot, los ambientes se verían limitados porque entre los componentes de NorthStar se encuentra un emisor de luz infrarroja que funciona únicamente en un rango de 2.5m de altura y hasta 4m de distancia.

- Luego de que durante la estancia de investigación en Chile se realizaran experimentaciones con la Plataforma Robótica Pioneer 3-AT, y al constatar de que éste puede desenvolverse en exteriores, se recomienda que para enriquecer la experiencia y ayudar al desarrollo de mas técnicas de navegación, además de la plataforma robótica Robotino® existente en el Laboratorio de Sistemas de Control, se adquiera otros robots móviles para exteriores.

- Se puede tener resultados desfavorables si las pruebas se las realiza en un ambiente con demasiada contaminación de redes WiFi, por lo que es recomendable tratar de evitar lugares con muchas redes de este tipo al momento de realizar las pruebas.

- La fuente de alimentación de Robotino® al parecer no representa una dificultad a la hora de realizar pruebas, sin embargo se debe tener en consideración el valor de tensión que brindan las baterías con las que el robot viene equipado, pues si dicho valor se encuentra por debajo de los 24V, puede que los resultados obtenidos no sean los más idóneos; esto es porque los sensores responden de distinta manera cuando la fuente de alimentación brinda toda la energía necesaria, como se enunció

anteriormente, como es el caso del valor de corriente de arranque del sensor láser.

REFERENCIAS BIBLIOGRÁFICAS

Capítulo 1

[1] Wikipedia, “Introducción de a la Historia de la Robótica”, 2012

http://es.wikipedia.org/wiki/Rob%C3%B3tica#Historia_de_la_rob.C3.B3tica

[2] Wikipedia, “Leyes de la Robótica”, 2012

http://es.wikipedia.org/wiki/Tres_Leyes_de_la_Rob%C3%B3tica

[3] Wikipedia, “Ley Cero de la Robótica”, 2012

http://es.wikipedia.org/wiki/Ley_Cero

[4] **DURRANT-WHYTE,** H., **BAILEY,** T., “Simultanoeus Localization and Mapping (SLAM) Part I”, IEEE Robotics&Automation Magazine, September 2006

[5] Santillana, “Generaciones de Robots”, 2012

http://ec.kalipedia.com/tecnologia/tema/generaciones-robots.html?x1=20070821klpinginf_88.Kes&x=20070821klpinginf_92.Kes

[6] Wikipedia, “Clasificación de los Robots según la Arquitectura”, 2012

http://es.wikipedia.org/wiki/Rob%C3%B3tica#Seg.C3.BAn_su_arquitectura

[7] Schneider Electric, “Robot Cartesiano”, 2012

http://www.directindustry.es/prod/schneider-electric-motion-deutschland/robots-cartesianos-4819-30854.html

[8] Nissan, “ Robot Móvil con evasión de obstáculos”, 2012

http://www.notodococches.com/nissan-inventa-un-robot-que-esquiva-obstaculos.html

[9] Wikimedia Commons, “Robot Humanoide”, 2012

http://commons.wikimedia.org/wiki/File:HONDA_ASIMO.jpg

[10] “Robot Hexapodo”, 2012

http://www.machinesthatgobing.com/images/lynxmotion-hexapod-1.jpg

[11] Robots, “Sensores, Conceptos Generales”, 2012

http://robots-argentina.com.ar/Sensores_general.htm

[12] Robótica-Proton, “Actuadores”, 2012

http://proton.ucting.udg.mx/materias/robotica/r166/r68/r68.htm

[13] Wikipedia, “Inteligencia Artificial”, 2012

http://es.wikipedia.org/wiki/Inteligencia_artificial

[14] **FESTO**, © Festo Didactic GmbH & Co. KG • 548606/555708-Manual (B) de Teoria, Denkendorf, 2007

[15] **FESTO**, © Festo Didactic GmbH & Co. KG • 544305 Manual, Denkendorf, 2007

[16] Robotics Equipment Corporation, “Valores de Robotino”, Copyright (C) 2004-2008

http://svn.openrobotino.org/trunk/openrobotino1/lib/rec/robotino/com/OmniDrive.h

[17] Aallborg University, “Modelo Cinemático Omnidireccional”, Denmark, 2012

http://www.control.aau.dk/~tb/wiki/index.php/Kinematics

[18] MobileRobots, “Pioneer 3, Operations Manual”, 2006

[19] **ANDALUZ**, G., **ANDALUZ**, V,. **ROSALES**, A., “Modelación, Identificación y Control de Robots Móviles”, Escuela Politécnica Nacional, Quito

[20] AUAT CHEEÍN**, Fernando Alfredo,** Universidad Nacional de San Juan, Instituto de Automática, “Localización y Reconstrucción Simultánea de Entornos por un Robot Móvil basada en la Navegacion Orientada a las Zonas de Máxima Incertidumbre ”, San Juan Argentina, Diciembre de 2008.

[21] **FERNANDEZ CARAMÉS**, Carlos, Universidad de Salamanca, Departamento de Informática y Automática, “Técnicas de Navegación de

Robots basadas en Sistemas de Medición por Láser”, Salamanca España, Septiembre 2007.

[22] **REALPE ROBALINO**, Miguel, “Hacia la Navegación Autónoma de Robots a partir de la implementación de un metodo de Localización y Mapeo Simultáneos (SLAM) mediante el uso de un sistema de vision 3D”, Escuela Superior Politécnica del Litoral, Facultad de Ingeniería en Electricidad y Computación, Guayaquil Ecuador, 2006.

[23] Wikipedia, “Sonar”, 2013

http://es.wikipedia.org/wiki/Sonar

[24] **GOMEZ ORTEGA**, Juan, “Navegacion en Robots Móviles basados en Técnicas de Control Predictivo Neuronal”, Universidad de Sevilla, Escuela Técnica Superior de Ingenieros Industriales, Sevilla, 1994.

[25] **VALLEJO**, Patricio, **GUTIÉRREZ**, Luis, “Laboratorio de Fisica II”, Quito Ecuador, Junio 2012.

[26] Exsys , “Características del Hub”, 2012

http://docs.roboparts.de/EX-1163-Manual.pdf

[27] Hokuyo, “Características Hokuyo URG-04LX-UG01”, 2012

http://docs.roboparts.de/URG-04LX_UG01/Hokuyo-URG-04LX_UG01_spec.pdf

[28] **S. THRUN**, “Probabilistic Robotics”, Carnegie Mellon University, Vol. 45, No. 3, Marzo 2002.

[29] Wikipedia, “Cadenas de Markov”, 2012

http://es.wikipedia.org/wiki/Cadena_de_Markov

[30] **ANÓNIMO**, Métodos Estadísticos en Ciencias de la Vida, “Cadenas de Markov”, 2012

http://www.bioingenieria.edu.ar/academica/catedras/metestad/Cadenas%20de%20Markov-1.pdf

[31] **RONCERO DEL REY**, Marta, “Estudio e Implementación de Métodos de Autolocalización para Robots AIBO ERS-7”, Universidad de Castilla-La Mancha, Ingeniería en Informática, Diciembre 2007

http://neithan.weebly.com/uploads/5/2/8/0/52807/pfc.pdf

[32] **SIEGWART**, R., **NOURBAKHSH** I. R., “Introduction to Autonomous Mobile Robots”. The MIT Press, 2004.

[33] **THRUN**, S., **BURGARD** W., **FOX** D., “Probabilistic Robotics”, MIT Press, Cambridge University, 2005.

[34] Wikipedia, “Filtro de Kalman”, 2013

http://en.wikipedia.org/wiki/Kalman_filter

[35] **ANDRADE-CETTO,** J., **SANFELIU** A., “Environment Learning for Indoor Mobile Robots”, Springer

[36] **WAN**, Eric A., **VAN DER MERWE**, Rudolph, "The Unscented Kalman Filter for Nonlinear Estimation"

[37] **DAUM**, F., “Nonlinear filters: beyond the Kalman filter”, Aerospace and Electronic Systems Magazine, IEEE, 2005

[38] **PEDRAZA,** G. , “SLAM geometrico con modelado basado en curvas Spline”, Universidad Politécnica de Madrid, Madrid, 2009

[39] **DURRANT-WHYTE**, H., **BAILEY**, T., “Simultanoeus Localization and Mapping (SLAM) Part II”, IEEE Robotics&Automation Magazine, September 2006

[40] **SMITH**, R., **SELF** M., **CHEESEMAN**, P. “Estimating uncertain spatial relationships in robotics”. Autonomous Robot Vehicles, 1990.

[41] **SMITH**, R., **SELF** M., **CHEESEMAN**, P. “On the representation and estimation of spatial uncertainly”. International Journal of Robotics Research. Vol 5, 1990.

[42] **SMITH**, R., **SELF** M., **CHEESEMAN**, P. “A Stochastic Map for Uncertain Spatial Relationships”. Autonomous Mobile Robots: Perception, Mapping and Navigation, IEEE Computer Society Press. 1991.

[43] Wikipedia, “Transformada de Hough, Teoría e Introducción”, 2013

http://es.wikipedia.org/wiki/Transformada_de_Hough ; teoria e introduccion de la transformada de hough

[44] **ÁREA DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA,** "Segmentación, Transformada de Hough", Universidad de Jaén, Departamento de Ingeniería electrónica, Telecomunicación y Automática, 2005

http://www4.ujaen.es/~satorres/practicas/practica4_vc.pdf

[45] **MARTÍNEZ JORDAN**, Alejandro, "Transformada de Hough", 2012

http://es.scribd.com/doc/60929263/Transform-Ada-de-Hough-MATLAB

ANEXOS

ANEXO A

SOFTWARE ADICIONAL

SOFTWARE ADICIONAL

Durante la etapa de experimentación e investigación realizada en Chile, en la Universidad Técnica Federico Santa María, fue necesaria la utilización de diferente software complementario, tanto para la simulación de entornos estructurados como para el manejo de la plataforma Pioneer 3-AT. Estos son descritos en este anexo:

A - 1 MOBILESIM

I. INTERFAZ INICIAL

Mediante este software se puede simular el comportamiento y funcionamiento de los algoritmos de control generados para el manejo del robot Pioneer 3-AT, siendo necesario realizar una comunicación serial entre el computador y el procesador del robot directamente a través de un cable serial. Una vez que se ha detectado el puerto virtual en el programa escrito en el compilador Visual C++ y ayudado de las librerías generadas por Mobile Robots (marca del Robot), como por ejemplo ARIA, se procede a operar con los ambientes estructurados creados a partir del manejo de Mapper 3D.

Lo primero que se hace es escoger el ambiente indicado para la simulación:

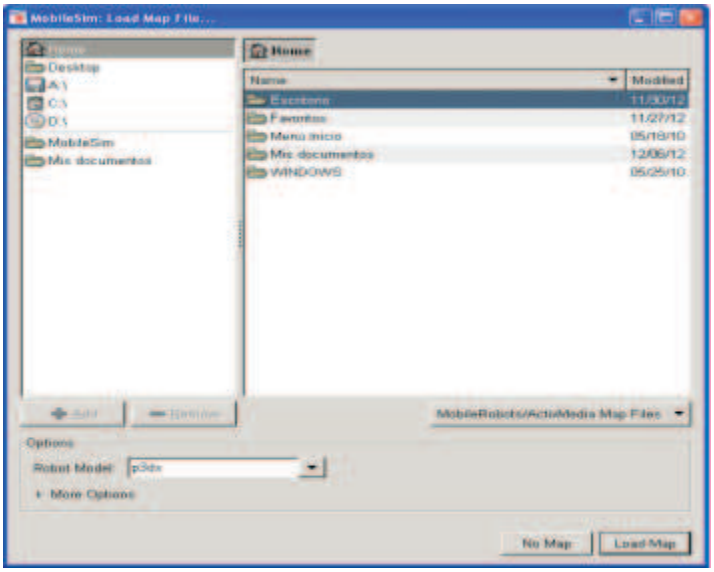


FIGURA A.1.1 Interfaz Inicial MobileSim

II. INTERFAZ DE SIMULACIÓN

Una vez elegido el entorno se procede a observar el desempeño del robot de acuerdo a las especificaciones que brinde el algoritmo. En este simulador se puede observar el uso del láser (efecto celeste en la gráfica) y adicionalmente, de los sensores sonares, los cuales están dispuestos en el chasis del Robot Pioneer.

El robot elegido se nota de color rojo en la parte superior del entorno, adicionalmente se puede observar información de los sensores utilizados durante la simulación, la cual se despliega en la parte inferior de la interfaz de simulación y además donde se presentan advertencias del uso del simulador.

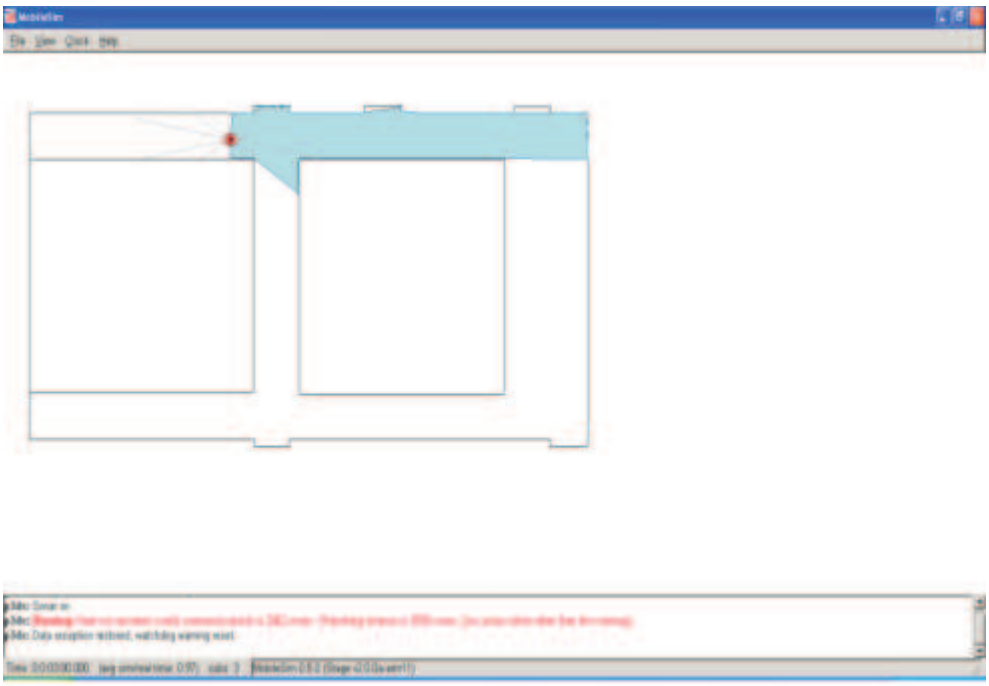


FIGURA A.1.2 Interfaz de Simulación MobileSim

A - 2 MAPPER 3D

El software Mapper 3D, brinda la posibilidad de crear o elaborar entornos estructurados, los cuales servirán para cargar estos mapas mediante el MobileSim, cabe destacar que la interfaz de desarrollo es muy amigable y parecida al entorno de Paint, por esta razón en el presente trabajo se realizaron muchos entornos, para de esta manera poner a prueba la consistencia del algoritmo de movimiento y mapeo con diferentes mapas.

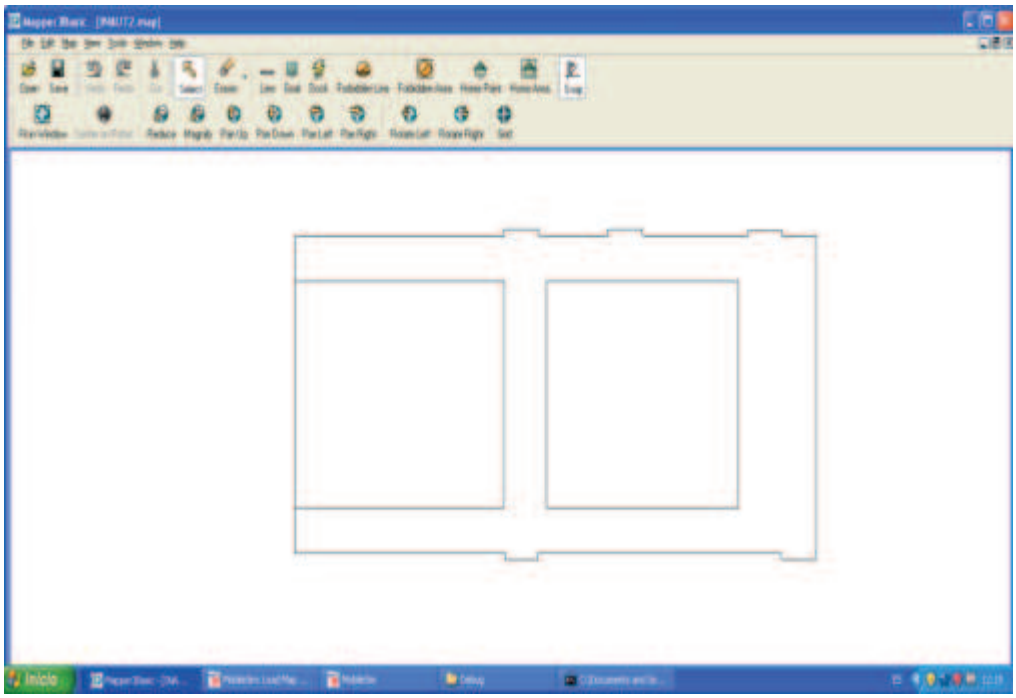


FIGURA A.2.1 Entorno creado con Mapper 3D.

A - 3 VISUAL STUDIO 2003

INTERFAZ DE PROGRAMACION

Para el control del robot Pioneer 3-AT se desarrollan los algoritmo en el lenguaje de programación C++, para esto es necesario contar con el paquete computacional Visual Studio 2003, con esta versión se garantiza la ausencia de errores e inconvenientes con la plataforma y demás, pues la mayoría de investigadores afirman que esta versión no genera problemas. Este compilador cuenta con librerías dedicadas para el manejo de estos robots, como es ARIA.

El software ARIA cuyas siglas representan (Interfaz Robótica Avanzada para Aplicaciones), que incluye ARNnetworking, corresponde a un ambiente desarrollado en lenguaje de C++ basado en código abierto, el cual proporciona una sólida interfaz de cliente a una variedad de sistemas robóticos inteligentes, incluyendo el procesador de la plataforma y los diversos sensores. El manejo de Aria incluye otras librerías que contienen los programas necesarios para controlar

el sensor láser, el sonar, la odometría y todos los demás periféricos de los cuales se dispone.

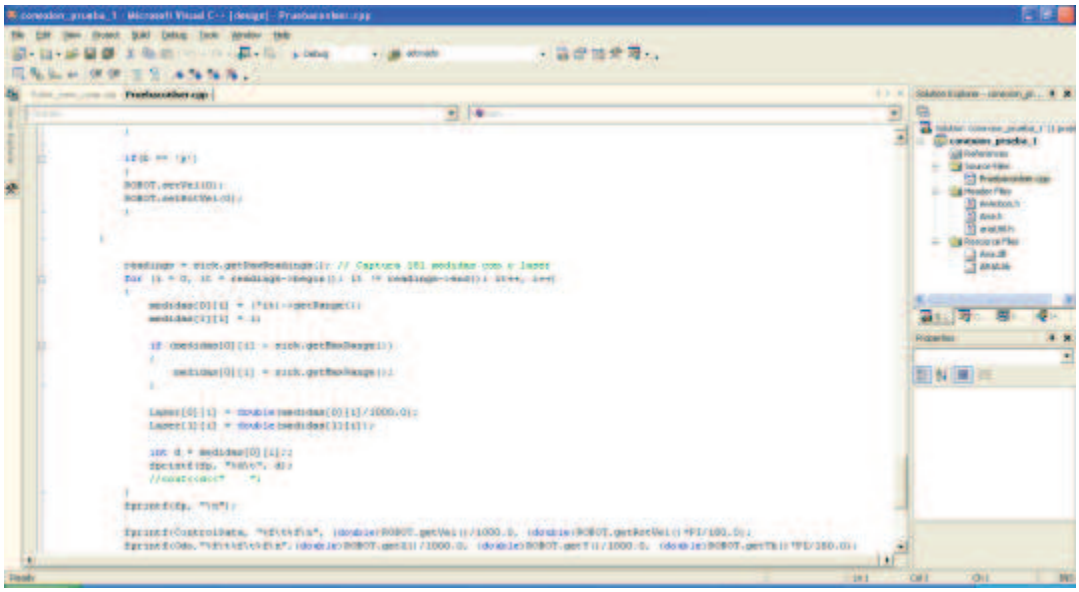


FIGURA A.3.1 Interfaz de Programación-Extracto de una prueba con Pioneer 3-AT

INTERFAZ DE MANEJO DEL ROBOT

Una vez ejecutado el programa escrito en C++, se despliega este mensaje en el cual se pide se ingrese el valor de la velocidad lineal con la que se va a mover el robot, e inmediatamente el valor de la velocidad angular, en el presente ejemplo se realizaba el movimiento del robot a partir de la selección de teclas, como sigue:

f --------> adelante (forward) b--------> atrás (back)

l---------> izquierda (left) r---------> derecha (right)

p--------> pausar (pause) x--------> salir (exit)

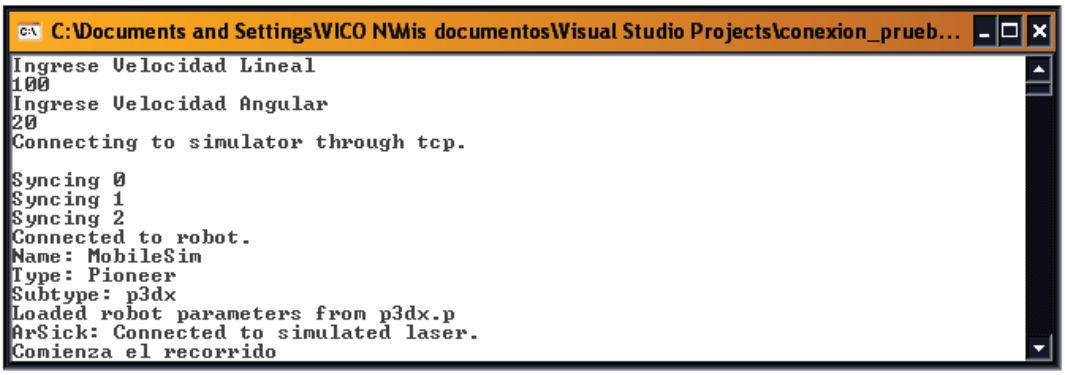


FIGURA A.3.2 Interfaz para control del movimiento del Robot Pioneer 3-AT

ANEXO B

MANUAL DE USUARIO

MANUAL DE USUARIO

El sistema desarrollado en este proyecto de titulación requiere de los siguientes requerimientos para un adecuado funcionamiento:

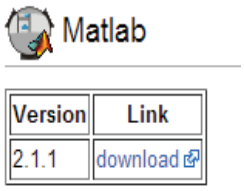
- ✓ MATLAB 2011b
- ✓ Robotino® MATLAB
- ✓ API2
- ✓ Visual Studio 2010 (o un compilador compatible con la versión de MATLAB)

Adicionalmente para la realización del presente trabajo se requirió la instalación de los drivers de la plataforma robótica Robotino®, que se obtienen previa la instalación de Robotino® MATLAB, disponible en la página:

http://wiki.openrobotino.org/index.php?title=Downloads

Aquí, se encuentran todos los programas compatibles con cada una de las versiones de sistemas operativos, y herramientas adicionales para el correcto uso de la plataforma robótica Robotino®.

En este caso se opto por escoger e instalar:



Habiendo instalado Robotino® MATLAB, se procede a habilitar los toolbox que contienen los m-files y bloques de Simulink que logran controlar los actuadores y demás sensores que posee Robotino®, esto es:

1. Seleccionar el compilador, esto se debe a que los programas se encuentran elaborados en lenguajes de programación C++, para escoger el compilador de acuerdo a las versiones de MATLAB y el número de bits del computador a usarse

(32 o 64), se logra con el siguiente comando: mex –setup, una vez que se escribe en el workspace se despliega lo siguiente:

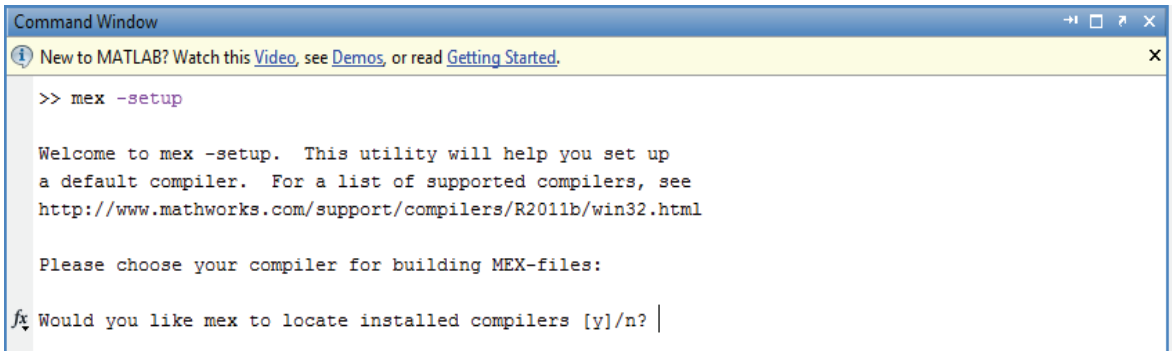


FIGURA B.1 Mensaje desplegado con el comando mex –setup

Al escoger “n” se muestra el listado que contiene todos los compiladores existentes y las versiones compatibles para el adecuado funcionamiento, pues en existen muchas versiones de compiladores cuyo desempeño en esta aplicación no es el correcto.

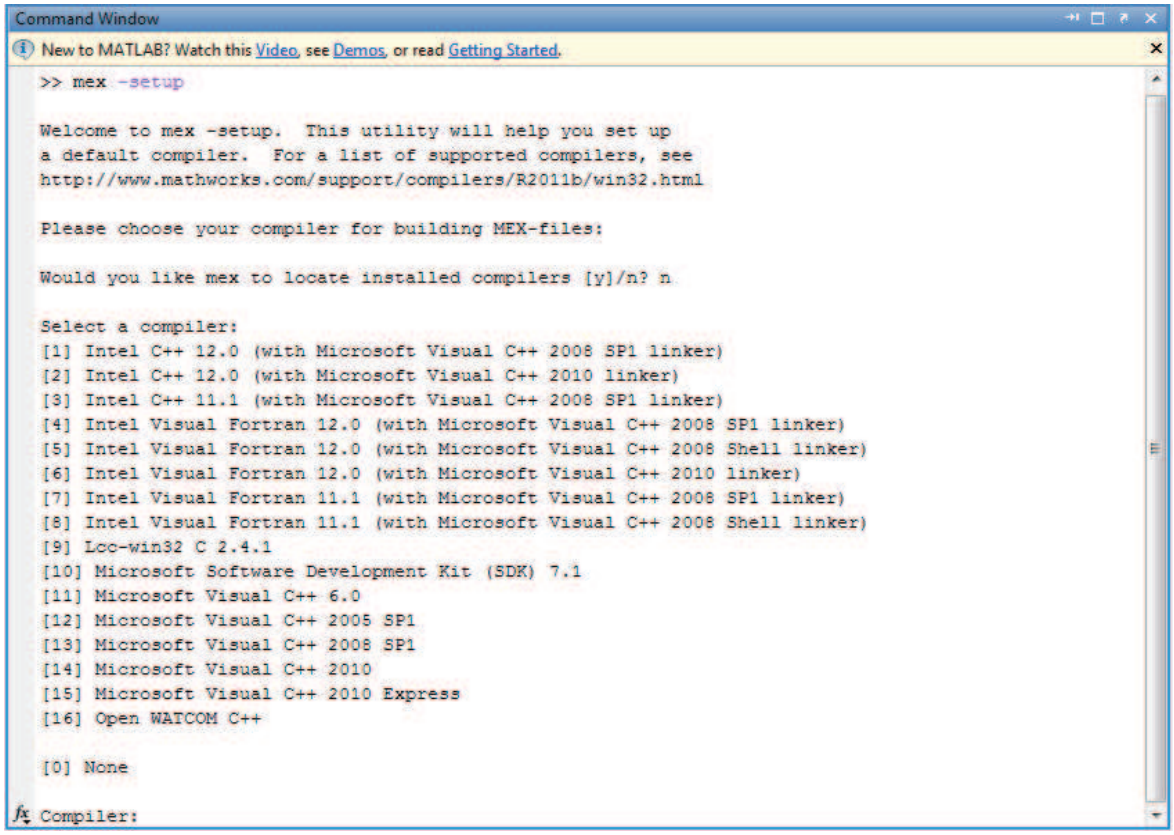


FIGURA B.2 Listado de Compiladores compatibles

Se escoge “y” para que se muestre el mensaje que hace posible el escoger el compilador:

```
Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Lcc-win32 C 2.4.1 in C:\PROGRA~1\MATLAB\R2011b\sys\lcc
[2] Microsoft Visual C++ 2010 in C:\Program Files\Microsoft Visual Studio 10.0

[0] None

Compiler:
```

FIGURA B.3 Mensaje para la Selección del compilador ya instalado

Una vez que se ha mostrado este mensaje se escoge el compilador Microsoft Visual C++ 2010, que previamente ha sido instalado, colocando el número 2.

```
Compiler: 2

Please verify your choices:

Compiler: Microsoft Visual C++ 2010
Location: C:\Program Files\Microsoft Visual Studio 10.0

Are these correct [y]/n? y
```

FIGURA B.4 Selección del compilador

Una vez confirmado el compilador seleccionado, se configura automáticamente y observan lo siguiente:

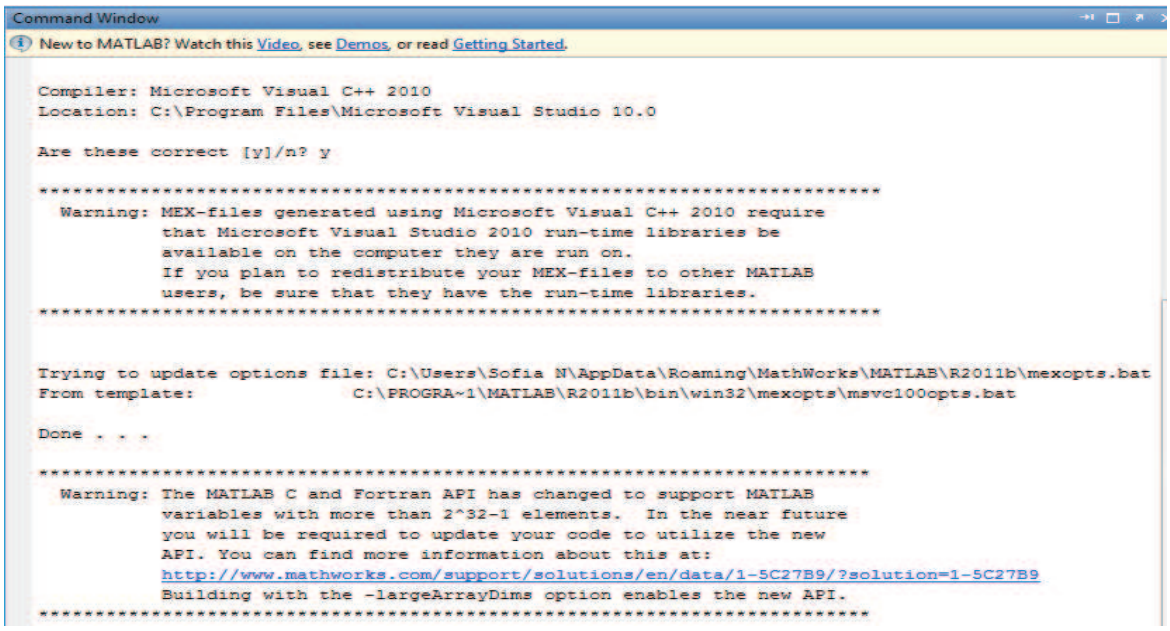


FIGURA B.5 Mensaje Final cuando MATLAB está listo para ser usado.

2. Cambiar el directorio raíz para que el MATLAB tenga acceso a los m-files y blockset para controlar Robotino, esto se logra con el siguiente comando:

```
cd ( getenv('ROBOTINOMATLAB_DIR'))
```

El directorio tenía la siguiente dirección:

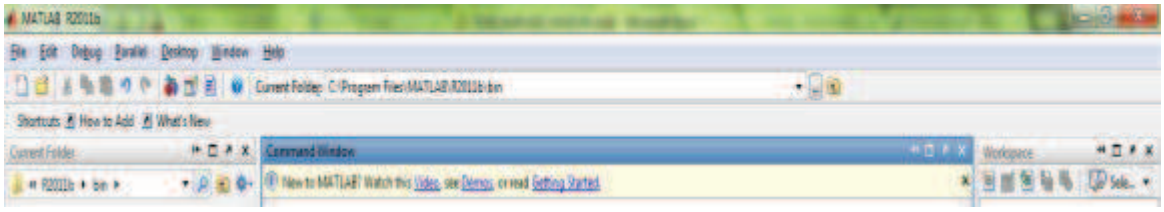


FIGURA B.6 Dirección del Directorio de MATLAB original

Una vez usado el comando anterior la dirección cambia a:

FIGURA B.7 Dirección nueva del Directorio de MATLAB

3. Añadir las carpetas de m-files y blockset al path de MATLAB.

Colocando *startup* en el command window.

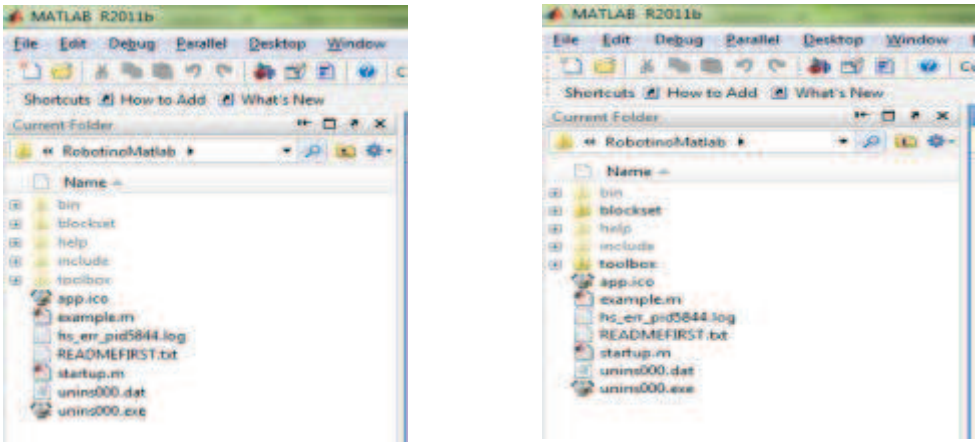


FIGURA B.8 Izq. Antes del comando, Der. Después del comando

4. MATLAB está listo para que se ejecute cualquier programa con Robotino® MATLAB, sea con M-files o con bloques de Simulink. Considerando siempre las restricciones propias de cada una de las funciones o bloques.

Adicionalmente dependiendo del modelo de computador utilizado, se puede ayudar al aprovechamiento de recursos de ésta, mediante un comando que hace posible que los procesos se separen y distribuyan en pequeños sub laboratorios de MATLAB, esto se realiza con el uso del comando:

```
matlabpool(n)
```

Se coloca dentro del paréntesis el número de sub laboratorios que se desea crear, esto si depende del número de núcleos que posea la laptop en cuestión, para el presente caso se coloca 2.

Además una prueba de que se ha realizado correctamente este paso, en la esquina inferior izquierda de la ventana de MATLAB aparecerá la imagen del número escogido rodeado de los símbolos del programa.

FIGURA B.9 Imagen que aparece cuando se han creado dos sub laboratorios

5. Resta realizar la conexión del computador con Robotino®, esto es a través de la selección de la Red Wlan propia del robot, esta aparecerá en el listado de las redes inalámbricas detectadas, y responde al nombre de Robotino APX1. Una vez encontrada esta red, se da clic en Conectar y se espera hasta que la red se haya conectado completamente, pues si se trata de correr el programa antes de que esto suceda, se producen inconvenientes con MATLAB, en el momento que la red se encuentre lista es hora de probar cualquier programa.

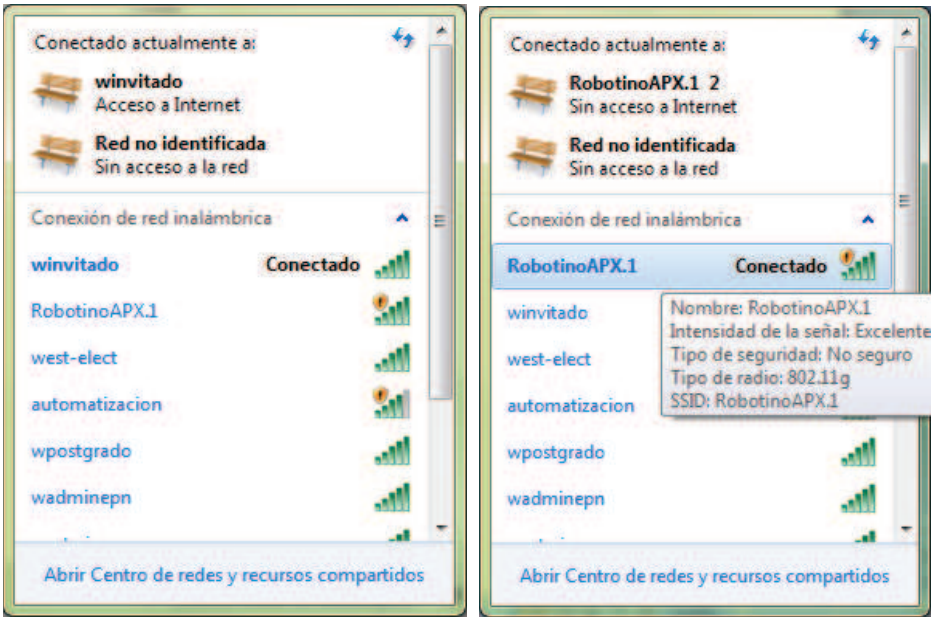


FIGURA B.10 Izq. Listado de Redes Inalámbricas, Der. Red Conectada

6. Luego de haber realizado estos pasos previos, es momento de ejecutar cualquier programa que controle la plataforma robótica Robotino®, sin embargo, pueden presentarse ciertos problemas como los descritos a continuación.

- Es necesario revisar si el sensor láser Hokuyo URG-04LX-UG01 está funcionando adecuadamente, esto es observar el led indicador (led verde), el cual estará encendido continuamente si el láser está listo para ser usado, caso contrario el led se encontrará titilando, si esto sucede el sistema el láser no se ha inicializado de manera adecuada, es decir, Robotino® se moverá normalmente pero el mapeo no se llevara a cabo, se notará en la pantalla del computador solamente una figura vacía de color gris, la solución es desconectar el cable USB del Hub, posteriormente apagar el robot, esperar un momento para conectar nuevamente, volver a conectarlo, y encenderlo de nuevo hasta que se note la total funcionalidad del láser.

FIGURA B.11 Forma de Desconectar el USB del Hub.

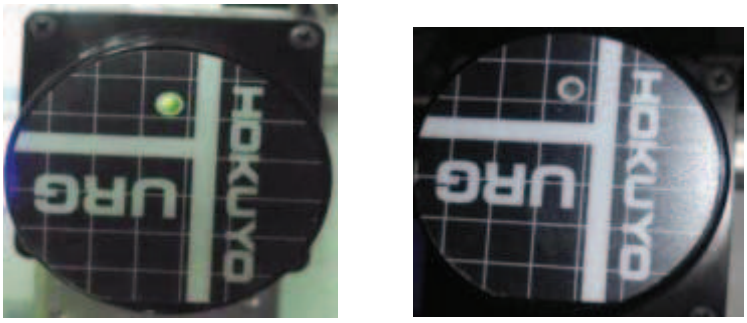


FIGURA B.12 Izq. Funcionamiento Correcto, Der. Funcionamiento Incorrecto

- Otro frecuente problema tiene que ver con la falla de conexión, sucede que mientras se encuentre funcionando en el display principal de Robotino® aparece el siguiente mensaje:

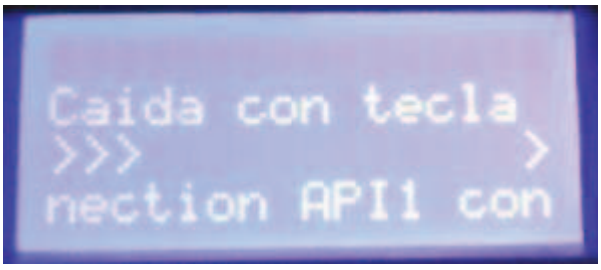


FIGURA B.13 Mensaje en el Display durante una ejecución Normal

Y cuando la comunicación falla se nota que desaparece este mensaje y se pone en la pantalla principal nuevamente.

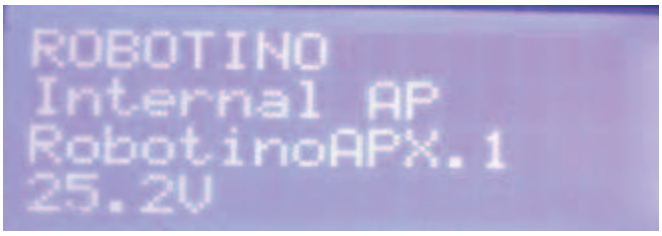


FIGURA B.14 Mensaje Principal que aparece en el Display

En este caso lo indicado es esperar el tiempo de duración del programa y apagar de la forma indicada (como se indica en FIGURA B-15 y B-16), para luego reiniciar la comunicación.

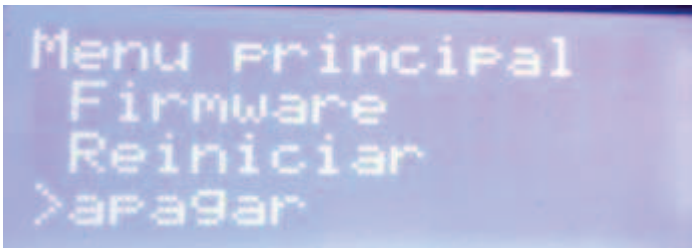


FIGURA B.15 Forma de Apagar Robotino®

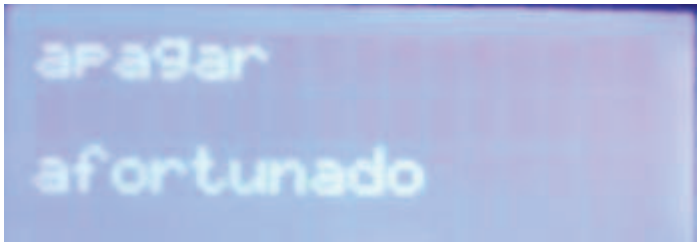


Figura B.16 Mensaje de Apagado

- En ocasiones suelen existir contratiempos con MATLAB, esto se nota cuando a pesar de presionar las teclas ubicadas en la parte superior del chasis de Robotino®, éste no responde, asumiendo que se encuentra en un lazo infinito, en este caso lo primero es desconectarse de la red inalámbrica, luego cerrar MATLAB y por ultimo apagar a Robotino®, como el robot no responde, se presiona prolongadamente el botón de apagado situado en el lado izquierdo de la botonera, sin embargo, cabe resaltar que esta no es la mejor forma de apagado, por lo que se recomienda solo hacer uso de esta opción en la última instancia, porque al apagar adecuadamente se vacían todos los registros existentes; mientras que al hacerlo de una

forma no recomendable, los registros permanecen ocupados, haciendo que el desempeño disminuya para la siguiente experimentación. Se asume que este problema se puede dar debido a la contaminación de redes WiFi en el ambiente de pruebas.

- WARNING: Como se explicó en el CAPITULO I, existe una tarjeta CF la cual contiene los drivers y sistema operativo de Robotino, esta nuca debe ser extraída mientras el robot se encuentre encendido y/o conectado a la red inalámbrica, pues de suceder esto la tarjeta puede echarse a perder irremediablemente.

ANEXO C

M-FILES DE ROBOTINO®MATLAB

M-FILES DE ROBOTINO®MATLAB

En el presente trabajo se utilizaron varias funciones proporcionadas por Robotino® MATLAB, a continuación se muestra un resumen de la utilización de estas en el programa que rige el sistema:

- **Com**

```
[ ComId ] = Com_construct
```

Esta función sirve para definir la identificación del objeto en este caso Robotino®, para luego establecer una conexión con el robot.

La función devolverá la identificación del nuevo elemento construido.

```
Com_setAddress(ComId, '172.26.1.4');
```

Mediante el uso de esta función se configura la dirección IP que luego establecerá la conexión del computador con Robotino®, es decir, se coloca el valor que aparece en el display principal ubicado en la parte superior del chasis.

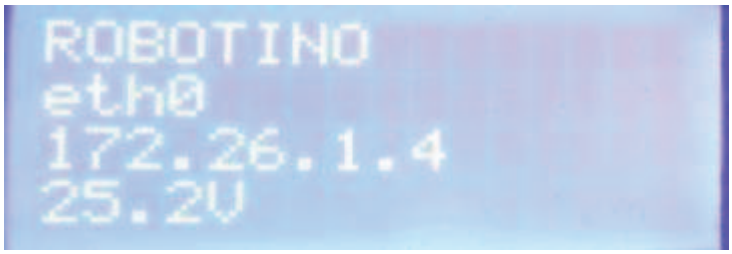


FIGURA C.1 Dirección IP mostrada en el Display

```
Com_connect(ComId);
```

Se logra establecer la conexión con Robotino.

```
Com_destroy(ComId);
```

Al finalizar el programa es recomendable destruir el objeto construido.

- **Omnidrive** 

  Mediante esta función se crea la capacidad de generar movimiento omnidireccional, es decir, accionar los motores en base a las diversas combinaciones que se explicaron en la TABLA 1.2 del Capítulo I.

  ```
  OmniDriveId = OmniDrive_construct;
  ```

  Se construye el objeto necesario para que se pueda lograr luego la manipulación de velocidades de los motores.

  ```
  OmniDrive_setComId(OmniDriveId, ComId);
  ```

  Se logra configurar el objeto y sincronizarlo con el objeto creado inicialmente.

  ```
  OmniDrive_setVelocity(OmniDriveId, Vx, Vy ,W);
  ```

  A través de esta función se puede configurar el valor adecuado de las velocidades *Vx* y *Vy*, asi como asignar la velocidad angular *W* que se requiera; es necesario recalcar que según pruebas realizadas la mejor velocidad lineal para el presente proyecto es de 70 mm/s.

  La forma de utilización de esta función es colocar directamente los valores de velocidad lineal en (mm/s) y de la velocidad angular en (°/s).

  ```
  OmniDrive_destroy(OmniDriveId)
  ```

  De la misma forma que los objetos anteriormente explicados, es necesario destruir el objeto construido, a fin de que no existan conflictos para un nuevo programa en MATLAB.

- Bumper 

  El sensor anticolisión, es decir la banda de color negro que rodea a Robotino® responde a la función *Bumper,* el uso se describe como sigue:

  ```
  BumperId = Bumper_construct;
  ```

  Se construye el objeto *Bumper* útil para activar el sensor anticolisión.

  ```
  Bumper_setComId(BumperId, ComId);
  ```

  Se logra relacionar el objeto *Bumper* creado con el objeto general Robotino®.

```
Bumper_value(BumperId) ~= 1
```

Esta variable sirve para saber el estado del sensor anticolisión, esto quiere decir si ha existido choque o no; la variable adquiere el valor de 1 en el instante en el que se detecte una colisión.

```
Bumper_destroy(BumperId);
```

Se requiere destruir el objeto creado para lo cual se usa esta función.

- **Odometry** 

Con las funciones correspondientes a *Odometry* se logra conocer la posición del robot con respecto a un marco de referencia global.

```
OdometryId = Odometry_construct;
```

Función necesaria para construir el objeto *Odometry* capaz de brindar información de odometría.

```
Odometry_setComId(OdometryId, ComId);
```

Útil para conectar y sincronizar el objeto *Odometry* con el Robotino® identificado anteriormente.

```
Odometry_set( OdometryId, x, y, phi )
```

Mediante esta función se puede configurar el eje de coordenadas del marco de referencia global, es decir, asignar los valores iniciales de cada eje, así como el ángulo de orientación inicial, colocando la posición de *x* e *y* en (mm), y el ángulo *phi* en (°), una vez que se ejecuta esta función se resetea el eje de coordenadas.

```
[ x, y, phi ] = Odometry_get( OdometryId )|
```

Esta función brinda información de la posición actual de Robotino®, tal como son los valores cartesianos de *x* e *y* (en mm), y el ángulo de orientación (en °).

```
OmniDrive_destroy( OmniDriveId );
```

A través de esta función se destruye el objeto creado inicialmente de forma que no se tengan conflictos una vez cesado el funcionamiento del programa.

- **LáserRangeFinder** 

  El sensor láser Hokuyo URG-04Lx-01 proporciona información de distancias en un rango de 0 a 5.6m, y posee ciertas funciones que se usan de acuerdo a las necesidades existentes.

  ```
  LáserRangeFinderId = LáserRangeFinder_construct;
  ```

  Es necesario el uso de esta función para crear el objeto láser y poder luego utilizarlo en el programa.

  ```
  LáserRangeFinder_setComId(LáserRangeFinderId, ComId);
  ```

  Se configura el objeto láser creado a fin de sincronizarlo con Robotino®.

  ```
  LáserRangeFinder_grab(LáserRangeFinderId) == 1
  ```

  Esta variable indica si los datos de un barrido provenientes del láser se encuentran listos para ser utilizados, en este caso, la variable tendrá valor de 1, cuando existen valores de distancias, y será 0 cuando aun no se ha logrado la obtención de datos.

  ```
  [ success, seq, stamp, angle_min, angle_max, angle_increment,
  time_increment, scan_time, range_min, range_max, ranges, numRanges,
  intensities, numIntensities ] = LáserRangeFinder_getReadings(
  LáserRangeFinderId );
  ```

  Todos los datos que proporciona el sensor láser, se encuentran definidos en esta función, sin embargo se nota que existen muchas variables que no se consideran de mayor importancia.

  ```
   [ ~, ~, ~, ~, ~, ~, ~, ~, ~, ~, ranges, ~, ~, ~ ] =
  LáserRangeFinder_getReadings( LáserRangeFinderId );
  ```

  Si se desea obviar una o algunas variables de acuerdo a las necesidades existentes, solamente se coloca el símbolo “**~**” en lugar del nombre de dicha variable.

  ```
  LáserRangeFinder_destroy(LáserRangeFinderId);
  ```

  Es utilizada para destruir el objeto creado una vez que ya se desee terminar la ejecución del programa.

- **DistanceSensor**

La plataforma robótica Robotino® está dotada de 8 sensores infrarrojos, los cuales se hallan distribuidos como se muestra en la figura siguiente:

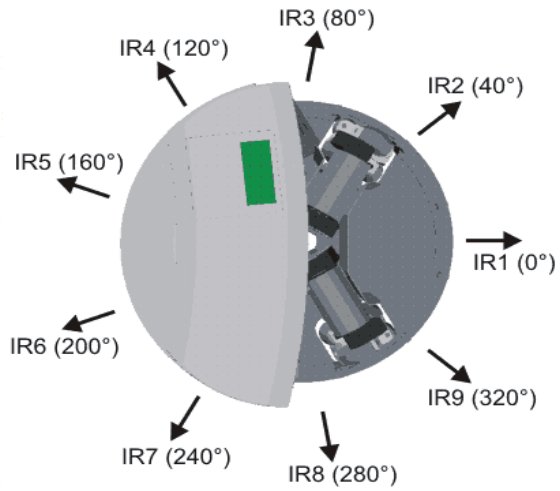


FIGURA C.2 Distribución de los Sensores Infrarrojos de Robotino

```
DistanceSensor0Id = DistanceSensor_construct(0);
```

Usando esta función se construye cada uno de los sensores infrarrojos que se ubican alrededor de Robotino®, los cuales responden al número que se muestra en la FIGURA C.2. Por cada sensor hay que utilizar esta función.

```
DistanceSensor_setComId(DistanceSensor0Id, ComId);
```

Mediante esta función se configura y sincroniza cada uno de los sensores infrarrojos creados con Robotino®, asimismo es necesario usar esta función para cada uno de los sensores creados.

```
value0 = DistanceSensor_voltage(DistanceSensor0Id);
```

Esta función brinda los valores de voltaje generados por cada uno de los sensores infrarrojos, el valor entregado está en voltios (V).

```
DistanceSensor_destroy(DistanceSensor0Id);
```

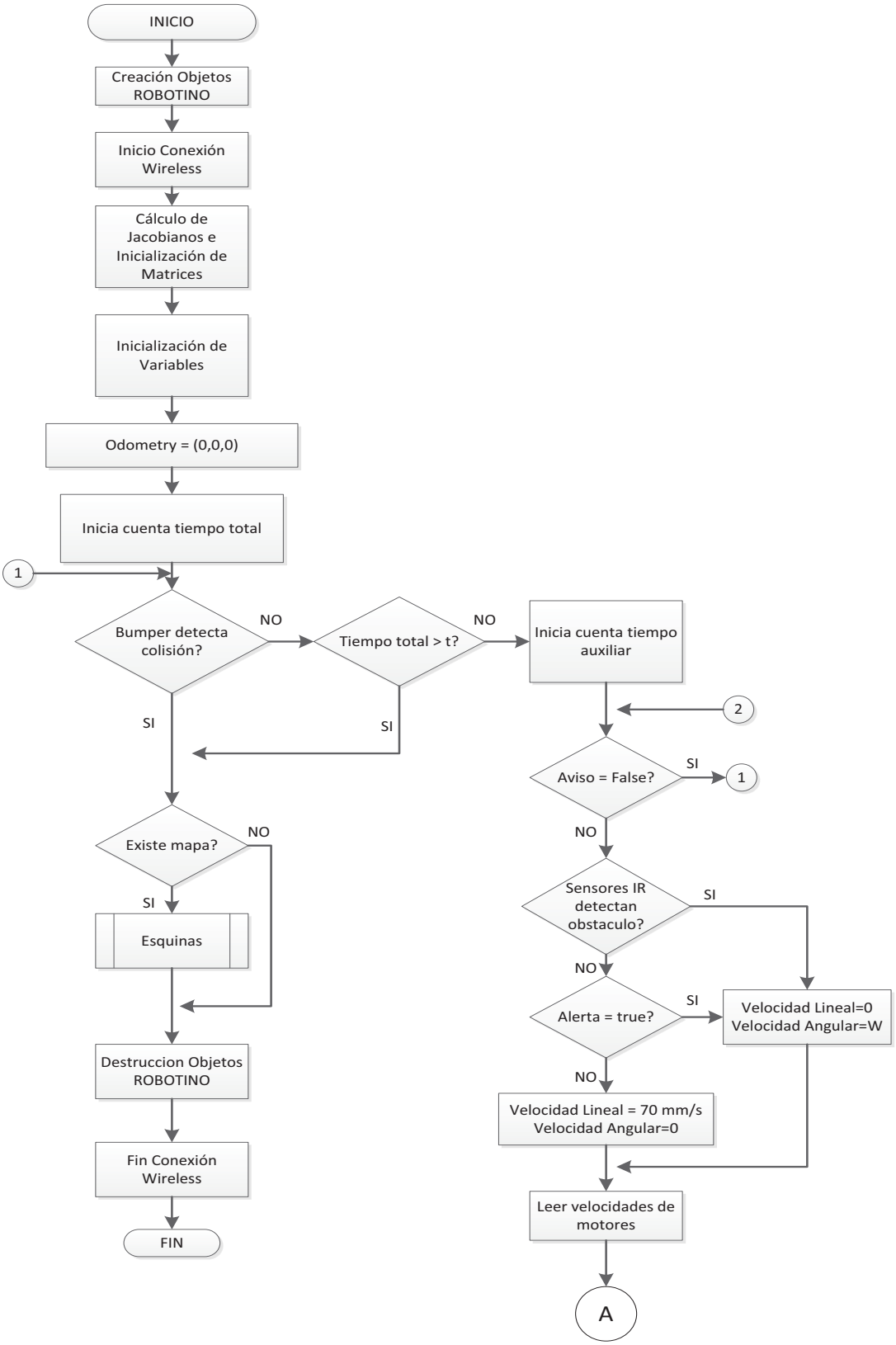
Se requiere el uso de esta función para destruir cada uno de los sensores que se hayan creado al inicio del programa.

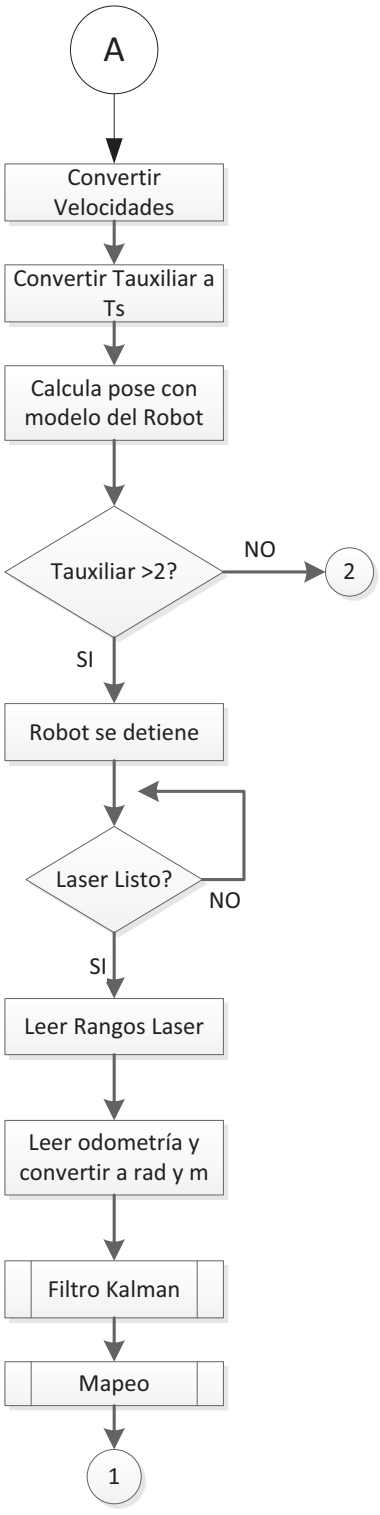
ANEXO D

DIAGRAMAS DE FLUJO

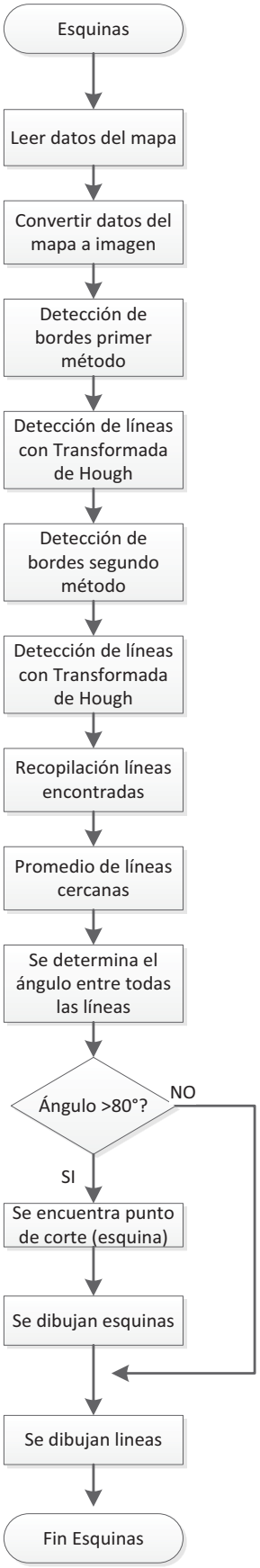
DIAGRAMAS DE FLUJO

Programa Principal

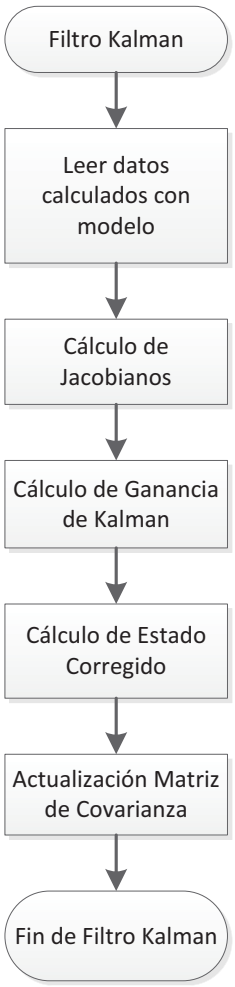



A
Convertir Velocidades
Convertir Tauxiliar a Ts
Calcula pose con modelo del Robot
Tauxiliar >2?
NO
2
SI
Robot se detiene
Laser Listo?
NO
SI
Leer Rangos Laser
Leer odometría y convertir a rad y m
Filtro Kalman
Mapeo
1

Función *Esquinas*



Función *Filtro de Kalman*



Función *Mapeo*