ASISTENTE ROBÓTICO PARA PERSONAS CON DISCAPACIDAD VISUAL

JUAN CAMILO CÁRDENAS CAICEDO, JUAN DAVID GÓMEZ ARCINIEGAS,
CRISTIAN MARTINEZ BEDOYA

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES
FACULTAD DE INGENIERÍAS
PROGRAMA DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.
2016

ASISTENTE ROBÓTICO PARA PERSONAS CON DISCAPACIDAD VISUAL

JUAN CAMILO CÁRDENAS CAICEDO, JUAN DAVID GÓMEZ ARCINIEGAS,
CRISTIAN MARTINEZ BEDOYA

Trabajo de grado para optar el título de ingeniero electrónico

Director,
Andrés Camilo Jiménez Álvarez
MsC en Ingeniería Electrónica

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES
FACULTAD DE INGENIERÍAS
PROGRAMA DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.
2016

Nota de aceptación

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Bogotá, Marzo 28 del 2016

AGRADECIMIENTOS

Los autores expresan sus más sinceros agradecimientos a Dios por las bendiciones y oportunidades que nos ha dado a lo largo de nuestra carrera profesional y a todas las personas, familiares, amigos e instituciones que han colaborado y hecho posible el desarrollo y culminación de este proyecto de grado; Especialmente al director de proyecto de grado MsC. Andrés Camilo Jiménez, por su apoyo, paciencia y dedicación.

A la universidad Los libertadores por el apoyo técnico, metodológico y hábil que brindó durante este tiempo.

Las directivas de la Universidad los libertadores, los jurados calificadores y el cuerpo docente no son responsables por los criterios e ideas expuestas en el presente documento, estos corresponden únicamente a los autores.

Se autoriza a los usuarios interesados para consultar y reproducir parcial o totalmente este trabajo de grado con fines académicos y de investigación, siempre y cuando se haga la correspondiente citación bibliográfica para darle crédito al trabajo.

CONTENIDO

RESUMEN.....

ABSTRACT.....

INTRODUCCIÓN.....

JUSTIFICACIÓN……...

1. OBJETIVOS.....
    1.1 OBJETIVO GENERAL.....
    1.2 OBJETIVOS ESPECÍFICOS.....

2. MARCO TEÓRICO

    2.1 ANTECEDENTES……..
        2.1.1 DISPOSITIVOS GENERALES PARA PERSONAS CON DISCAPACIDAD VISUAL
        2.1.1.1 ASISTENTE AUTÓMATA DE PASEO PARA PERSONAS CIEGAS
        2.1.1.2 BASTÓN DE NAVEGACIÓN PARA DISCAPACITADOS VISUALES
        2.1.1.3 DISPOSITIVO ETA MULTIMODAL
        2.1.1.4 ASISTENTE DE NAVEGACIÓN INDOOR CON RFID
        2.1.1.5 SISTEMA DE NAVEGACIÓN NOVEDOSO EN ENTORNOS CERRADOS PARA PERSONAS CON DISCAPACIDAD VISUAL
        2.1.2 PRODUCTOS COMERCIALES
        2.1.2.1 BASTÓN K-SONAR
        2.1.2.2 MINIGUIDE
        2.1.2.3 BRAINPORT V100
        2.1.2.4 KAPTEN MOBILITY GPS

    2.2 SLAM……..
        2.2.1 MAPAS TOPOLOGICOS
        2.2.2 MAPAS METRICOS
            2.2.2.1 GMAPPING
            2.2.2.2 HECTOR SLAM
            2.2.2.3 RTABMAP SLAM

    2.3 CINEMÁTICA
        2.3.1 ROBOTS MÓVILES
        2.3.2 RUEDAS
        2.3.3 MODELADO CINEMÁTICA DIRECTA

2.3.4 MODELADO CINEMÁTICA INVERSA

2.4 ROS

3. DISEÑO DE VItour

3.1 APLICACIÓN ANDROID (REMOTE CONTROL)
3.1.1 INTEGRACIÓN CON ROS
3.1.2 DESARROLLO
3.1.3 APARIENCIA UI

3.2 CINEMÁTICA ESTRUCTURA FÍSICA
3.2.1 CINEMÁTICA DIRECTA
3.2.2 CINEMÁTICA INVERSA

3.3 ARQUITECTURA
3.3.1 ROS
3.3.2 KINECT
3.3.3 PC
3.3.4 DISPOSITIVO ANDROID
3.3.5 ESTACIÓN DE MOVIMIENTO

4. RESULTADOS
4.1 CONTROL Y MOVIMIENTO
4.2 MAPAS OBTENIDOS

5. TRABAJOS FUTUROS
5.1 ETAPA DE NAVEGACIÓN AUTÓNOMA
5.2 ETAPA DE LOCALIZACIÓN DE USUARIO
5.3 IMPLEMENTACIÓN DE SISTEMA EMBEBIDO (TARJETA DE DESARROLLO)

6. CONCLUSIONES

7. ANEXOS

8. BIBLIOGRAFÍA

LISTA DE IMÁGENES

Figura 1. Diagrama de flujo del asistente propuesto

Figura 2. Concepto del sistema basado en navegación por líneas de color

Figura 3. Diagrama de bloques del sistema para sensar el color en las líneas

Figura 4. Arquitectura general del sistema ETA Multimodal

Figura 5. Campo potencial del RG

Figura 6. Modelo del funcionamiento de la aplicación móvil

Figura 7. Bastón K-SONAR de la compañía BAT Ltd

Figura 8. Miniguide

Figura 9. Representación del entorno por Brain Port V100

Figura 10. Brain Port V100

Figura 11. Dispositivo Kapten

Figura 12. Mapa construido usando SLAM

Figura 13. Mapa topológico

Figura 14. Mapa métrico

Figura 15. Mapa generado con Gmapping (Grisetti, Stachniss e Burgard 2006)

Figura 16. Esquema básico del sistema de navegación y mapeo – Hector SLAM

Figura 17. Mapa obtenido sobrepuesto con el plano real de la Rescue Arena RoboCup 2011

Figura 18. Diagrama de flujo del algoritmo del proceso de selección

Figura 19. Ejemplo de aplicación de RANSAC en un conjunto de datos atípicos

Figura 20. Esquema del funcionamiento del administrador de memoria

Figura 21. Proceso general del RTABMap

Figura 22. Tipos de ruedas

Figura 23. Centro de curvatura instantánea

Figura 24. Configuración de los robots móviles con ruedas

Figura 25. Diagrama del robot móvil

Figura 26. Conexión típica de un sistema de ROS

Figura 26. Etapas de diseño Vitour

Figura 28. Interfaz de usuario de la aplicación desarrollada

Figura 29. Esquema general de ROS para VItour

Figura 30. Diagrama eléctrico de VItour

Figura 31. VItour

LISTA DE ECUACIONES

Ecuación 1. Ecuación que normaliza el brillo de las dos imágenes

Ecuación 2. Cálculo del umbral

Ecuación 3. Descripción por medio de matrices de estados

Ecuación 4. Cálculo de las velocidades de las ruedas

Ecuación 5. Cálculo de similitud entre los espacios más recientes

Ecuación 6. Probabilidad para aceptar una hipótesis de un bucle cerrado

Ecuación 7. Velocidad lineal del robot

Ecuación 8. Velocidad angular del robot

Ecuación 9. Dinámica sobre el eje X

Ecuación 10. Dinámica sobre el eje Y

Ecuación 11. Dinámica sobre el eje de rotación

Ecuación 12. Matriz de rotación

Ecuación 13. Matriz de velocidad

Ecuación 14. Matriz de velocidades de la plataforma

Ecuación 15. Velocidad angular de las ruedas de tracción

Ecuación 16. Inversa de la matriz de referencia A

Ecuación 17. Pseudoinversa de la matriz A

Ecuación 18. Cinemática inversa

Ecuación 19. Cálculo de la resistencia de protección R1

Ecuación 20. Estimación de la corriente de salida I_o

LISTA DE TABLAS

Tabla 1. Propiedades usadas en las distancias entre el usuario y el obstáculo

Tabla 2. Cuadro comparativo entre los métodos Gmapping, Hector SLAM y RTABMap

Tabla 3. Comparación técnica entre el sensor Kinect y los láser 2D

Tabla 4. Dirección hacia adelante (La desviación fue hacia el lado derecho)

GLOSARIO

ALGORITMO: Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas.
BROADCAST: Es una forma de transmisión de información donde un nodo emisor puede enviar datos a una multitud de receptores de forma simultánea, sin necesidad de reproducir la misma transmisión nodo por nodo.
BÚSQUEDA DICOTÓMICA: Es un algoritmo de búsqueda que reduce el tiempo del proceso al disminuir exponencialmente el número de iteraciones necesarias para buscar un elemento dentro de un vector previamente ordenado.
ENTORNO DINÁMICO: A diferencia del entorno estático son los ambientes o lugares en donde los objetos cambian de posición en diferentes instantes de tiempo.
ENTORNO ESTÁTICO: Hace referencia a un lugar o espacio en donde no existen cambios de posición de los objetos que se encuentran dentro del mismo.
EOA: Son dispositivos de ayuda en orientación a personas con discapacidad visual en entornos desconocidos.
ESTEREOPSIS: Es el fenómeno por el cual, a partir de dos imágenes con disparidad horizontal se puede reconstruir una imagen tridimensional permitiendo la percepción de profundidad.
ENCODER: Dispositivos que permiten el monitoreo electrónico de la posición de un eje giratorio.
ETA: Son las tecnologías asistidas cuyo propósito es mejorar la movilidad para las personas con algún tipo de discapacidad visual.
FORK: Hace referencia cuando algún desarrollador a partir de una copia de un código fuente comienza un desarrollo independiente sobre el mismo, creando paquetes de software distintos.
FPGA: Es un dispositivo lógico programable que contiene bloques cuya interconexión y funcionalidad se puede configurar hacienda uso de un lenguaje de descripción especializado.
HOST: Es un ordenador que funciona como el punto de inicio y final de las transferencias de datos entre varias computadoras conectadas a una red.
IMÁGENES ESTEREOSCÓPICAS: Es cualquier técnica capaz de recoger información visual tridimensional y/o crear la ilusión de profundidad mediante una imagen estereográfica, un estereograma, o una imagen 3D.
LAYOUT: Cuadrícula imaginaria que divide en espacios o campos la página que se diseña para facilitar la distribución de elementos como textos o gráficos en la misma.
LIDAR: Es una tecnología que permite determinar la distancia de un emisor láser a un objeto o superficie utilizando un has láser pulsado. La distancia al objeto se determina midiendo el tiempo de retraso entre la emisión del pulso y su detección a través de la señal reflejada.

NODO: Espacio en el que confluyen parte de las conexiones de otros espacios reales y abstractos que comparten sus mismas características.
NUBES DE PUNTOS: Conjunto de vértices en un sistema de coordenadas tridimensional.
ODOMETRÍA: Es el estudio de la estimación de posición de vehículos con ruedas durante la navegación.
P2P: O peer-to-peer es una red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre si.
PALABRAS VISUALES: Son pequeñas partes de una imagen que contienen algún tipo de información relacionado a las características de las mismas, por ejemplo color, forma, cambios en los pixeles, descriptores de características de bajo nivel, etc.
PLD: Componente electrónico empleado para la fabricación de circuitos digitales reconfigurables. A diferencia de las puertas lógicas que tienen una función fija, este dispositivo tiene una función indefinida al momento de fabricarse.
RFID: Identificación por radiofrecuencia es Sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, tarjetas, transponedores o tags.
RMR: Robot móvil de ruedas
ROBOT: Es una entidad virtual o mecánica artificial capaz de realizar determinadas operaciones de manera autónoma y sustituir a los seres humanos en algunas tareas, en especial las pesadas, repetitivas o peligrosas.
ROS: Sistema operativo robótico por sus siglas en inglés *(Robot Operating System)* es un framework libre (open source) el cual contiene las herramientas y librerías suficientes para el desarrollo de software dedicado a robots.
SISTEMA EMBEBIDO: Es un sistema de computación que al contrario de lo que ocurre con equipos de propósito de general, está diseñado para cumplir tareas específicas en tiempo real.
SLAM: Localización y mapeado simultáneos es una técnica usada por los robots y vehículos autónomos para construir un mapa de un entorno desconocido en el que se encuentra, a la vez que estima su trayectoria al desplazarse dentro de este entorno.
UNIDAD DE PROCESAMIENTO: Es el hardware dentro de los sistemas programables que interpreta instrucciones de un programa informático mediante la realización de operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.
WEARABLE: Es aquel dispositivo que se lleva sobre, debajo o incluido en la ropa y que está siempre encendido, no necesita encenderse ni apagarse.

RESUMEN

A nivel mundial existe una gran parte de la población que presenta alguna enfermedad o deterioro de la vista, algunas de tipo progresivo con el paso del tiempo mientras que otras presentes desde el nacimiento. Para estas personas el desplazarse dentro de un entorno que no reconocen puede resultar ser algo complicado ya que no pueden percibir distancia alguna o profundidad con cada objeto ubicado en dicho espacio con respecto a su propia posición, motivo por el cual muchas veces pueden chocar durante su desplazamiento con cualquier elemento.

En este trabajo de grado se muestra el desarrollo de un asistente robótico que permita ayudar a las personas con discapacidad visual a desplazarse en un entorno cerrado desconocido. Basado en técnicas confiables para realizar el proceso de *Localización y Mapeado Simultáneo o SLAM (del inglés Simultaneus Localization And Mapping);* el robot crea un mapa del entorno y es capaz de navegar por el mismo a partir de los algoritmos desarrollados. Adicionalmente el robot puede conectarse e interactuar con cualquier terminal Android lo que permite al usuario comunicarse directamente con el robot de forma inalámbrica y dar las órdenes necesarias para que el asistente acuda al llamado del usuario.

El funcionamiento del robot está basado en un framewrok de tipo open source que cuenta con innumerables herramientas y librerías que facilitan de gran manera la labor de desarrollar software de interpretación robótica en lenguajes de programación básicos como C++, Python y Java. *Sistema Operativo Robotico o ROS (del inglés Robot Operating System*) cuenta con una comunidad de desarrolladores por todo el mundo que dan gran soporte a la plataforma y solventan de muy buena manera las inquietudes presentadas por otros desarrolladores, siendo esta una ventaja del software de tipo open source.

Palabras clave:

- Asistente robotico
- SLAM
- Algoritmos
- Navegación
- Software open source

ABSTRACT

Worldwide there is a lot of people who have a disease or impairment of vision, some of that diseases are progressive with the pass of time, while others are present since the birth. For that people displaced in an unrecognized environment could result be some complicated because they cannot perceive distance or deep measure between each object inside the environment and its own position, why can often collide with any element.

This work shows the development of a robotic assistant that allows visual impaired people to move on an unknown close environment. Based on reliable techniques for perform the SLAM *(Simultaneus Localization And Mapping)* process, the robot build a map of the ambience and it is able to move through it from the algorithms developed. Additionally the robot has the ability to connect and interact with an android terminal allowing the user to communicate with the robot in a wireless way and give it the instructions to go to the called.

Robot´s operation is based in an open source framework that has countless tools and libraries to make easier the work to develop robotics software interpretation with basic programming languages as C++, Python and Java. ROS (Robot Operating System) has a developer's community around the world who gives support to the platform and solve in a very good way the concerns raised by any other developer, being this an advantage of the open source software.

Key words:

- ➢ Robotic assistant
- ➢ SLAM
- ➢ Algorithms
- ➢ Navegation
- ➢ Open source software

INTRODUCCIÓN

En el mundo existen aproximadamente 285 millones de personas con discapacidad visual, de los cuales 39 millones son ciegas y 246 millones presentan baja visión, según la OMS (Organización Mundial de la Salud); por otra parte, señala que tres cuartas partes de los casos de ceguera son prevenibles o tratables, y que de no mediar ninguna intervención, el número de personas ciegas irá en aumento hasta alcanzar los 75 millones para 2020. Esta misma entidad afirma que un 90% de la concentración mundial de discapacidad visual se concentra en los países en desarrollo.[1]

Las personas que presentan algún tipo de limitación visual ya sea por pérdida parcial o total de la vista normalmente tienen ciertas dificultades para desplazarse, aún más si el entorno no es familiar para ellos ya que requieren algún tiempo de adaptación al nuevo entorno. Se conocen algunos desarrollos tecnológicos que atacan la problemática mencionada anteriormente, innovaciones que en algunos casos tienen un costo elevado como el proyecto de la compañía NSK *"guide dog"*[2] o *"guido"* una solución enfocada a los adultos mayores desarrollado por Haptica Ltd. Otros sistemas de tipo *"wereable",* que en su mayoría, presentan una solución poco óptima en cuanto a la comodidad para su uso, pues el diseño de los mismos implica un maletín con un computador portátil que el usuario debe cargar e incluso un casco donde es adaptada una cámara estéreo y por otro lado las soluciones cómodas no presentan un mayor rango de variables para considerar como lo es el caso del asistente basado en unos lentes con sensores de proximidad y un chip que emite una alerta sonora al detectar un obstáculo en el rango del sensor (Sheikh Sadi, Mahmud, Mostafa Kamal, & Ibne Bayazid, 2014), donde no se contempla la posibilidad de evitar elementos que estén a baja altura.

El asistente robótico que se presenta a continuación solventa este tipo de dificultad al buscar un prototipo de un menor costo, orientado a facilitar el proceso de desplazamiento de éstas personas en espacios cerrados, con condiciones estáticas, aún si la persona no está familiarizada con el lugar (casa, oficina, etc.). Lo que se busca es brindar cierto nivel de autonomía a las personas que presentan éste inconveniente, desarrollando un asistente que brinde la suficiente fiabilidad, en la totalidad del proceso, para guiar al usuario desde un punto A a un punto B dentro del entorno que se desee.

---

[1] WHO (World Health Organization); DE BODE, Chris . 10 Facts on visual impaired and blindness [en línea]. <http://www.who.int/features/factfiles/blindness/blindness_facts/en/>. [citado Junio 26, 2015].

[2] HANLON, Mike. The evolution of of NSK's guide robot for the visually-impaired - NSK (Nippon Seiko Kabushiki Kaisha) [en línea]. <http://www.gizmag.com/nsk-guide-robot-visually-impaired-irex/29814>

En ésta propuesta se diseña el robot llamado *Vitour,* que desempeña la tarea de construir mapas haciendo uso de la técnica de SLAM con las imágenes captadas por el sensor Kinect que se instala en el mismo. Por otra parte, el robot es incapaz de crear el mapa de manera autónoma, es decir que el asistente es controlado remotamente desde un terminal Android para la fabricación del mapa. Los resultados tienen gran acierto y brindan una información fiable sobre la cual es posible trabajar la etapa posterior de navegación.

De esta manera el prototipo diseñado busca brindar una mejor experiencia al usuario en comparación con los sistemas mencionados anteriormente y minimizando en lo posible los costos en su construcción. Con una unidad central de procesamiento, que opera bajo ROS[3], un framework que se basa en una red de procesos peer-to-peer *(P2P)*, donde se controla cada aspecto del sistema desde la comunicación con el terminal Android hasta la construcción del mapa, pasando por el desplazamiento y el control de los motores, todo gracias a las herramientas con las que cuenta ROS para la elaboración de un sistema robótico completo. La comunidad de ROS ha venido creciendo debido a que varios desarrolladores ven en este framework una herramienta poderosa y muy útil, el monitoreo y visualización de las comunicaciones y el flujo de datos entre los diversos nodos hacen de ROS una opción apropiada para el desarrollo robótico adicional a ello cuenta con una acertada integración con otros ambientes entre los cuales se incluye el sistema operativo para dispositivo móviles Android.

[3] <http://www.ros.org/>

JUSTIFICACIÓN

Como se ha mencionado anteriormente existe una probabilidad sobre el crecimiento de enfermedades visuales que inducen a la pérdida parcial o total de la vista, motivo por el cual se considera necesario a través de la tecnología y sus avances brindar herramientas y sistemas orientados a personas con éste tipo de limitaciones. En nuestro caso es planteado un sistema inteligente que brinde asistencia en el desplazamiento de dichos sujetos en espacios con los cuales no se encuentran familiarizados y mejorar en cierta medida la calidad de vida de las personas con algún tipo de enfermedad visual que afecten el desarrollo normal de dicha tarea.

La robótica es un campo de estudio que ha tenido un crecimiento exponencial, motivo por el cual alrededor del mundo existen diferentes comunidades que enfocan sus esfuerzos en desarrollar técnicas y teorías que hagan de los sistemas inteligentes y robots instrumentos óptimos y útiles al desarrollo de la humanidad. Es por esta razón que hemos decidido crear un robot con un enfoque no industrial y más de servicio humano haciendo uso de plataformas de tipo open-source que no han sido utilizadas en la institución Fundación Universitaria los Libertadores y de esta manera fomentar la investigación en éste campo al implementar herramientas de software que faciliten la construcción y desarrollo de aplicaciones robóticas.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Diseñar un asistente robótico para el desplazamiento de una persona con limitaciones visuales a través de un entorno estático, cerrado y controlado.

1.2 OBJETIVOS ESPECÍFICOS

- Evaluar los algoritmos para la medición de profundidad y segmentación en imágenes bajo el entorno de ROS.
- Diseñar e implementar una aplicación en plataforma Android para establecer el puente de comunicación entre el usuario y ROS.
- Modelar, diseñar e implementar la estructura física del asistente robótico.
- Implementar y evaluar un algoritmo de mapeo en entornos cerrados y controlados en el asistente robótico.
- Validar el funcionamiento del asistente robótico.

2. MARCO TEÓRICO

2.1 ANTECEDENTES

Los robots han cobrado gran importancia a partir del desarrollo tecnológico, las tareas que requerían de mucho esfuerzo para el hombre los robots las han ejecutado de una manera notable y esto ha permitido que estas máquinas tomen un papel considerable en el desarrollo de la humanidad. Y es que el desarrollo de sistemas expertos y mecanismos robóticos han impulsado la investigación profunda en el campo de la robótica y la inteligencia artificial, motivo por el cual se han abierto las puertas a la implementación de robots no sólo en el sector industrial, sino que en diversos campos expandiendo de esta manera su estudio. Una de las áreas de enfoque de la robótica son los robots asistenciales, área en la que aún falta por explotar pero que ha evolucionado, pues los diferentes servicios que requiere el hombre han obligado a que la investigación con éste enfoque tome cierta importancia. Entre este tipo de robots, se encuentran los especialmente diseñados para brindar soluciones y constante apoyo a personas con cierto tipo de discapacidad física, en éste caso la discapacidad corresponde a la vista, bien sea por pérdida parcial o total; no cabe duda que el sentido de la vista es uno sino el de mayor importancia de los cinco sentidos, ya que con él se recoge información espacial del entorno lo que permite una interacción muy completa con el mismo.

2.1.1 DISPOSITIVOS GENERALES PARA PERSONAS CON DISCAPACIDAD VISUAL

El desarrollo tecnológico ha permitido encontrar nuevas tecnologías que ayuden a las personas con algún tipo de discapacidad visual a mejorar su calidad de vida, surgiendo de esta manera tres categorías principales de este tipo de sistemas según (Dakopoulos & Bourbakis, 2010), ETAs; ayudas de navegación electrónicas del inglés *(Electronic Travel Aids),* EOAs; ayudas de orientación electrónicas del inglés *(Electronic Orientation Aids)* y finalmente los PLDs; dispositivos localizadores de posición del inglés *(Position Locator Devices)*, éste artículo se centra en el rendimiento de algunos sistemas wereables para la detección y evasión de obstáculos. Se analizan sistemas basados en la ecolocación, el mismo sistema usado por los murciélagos, donde se tienen unos sensores de ultrasonido adaptados en gafas donde haciendo uso de un microprocesador y un conversor A/D la entrada resulta en un sonido estéreo audible que indica al usuario la proximidad en la que se encuentra con respecto a un objeto.

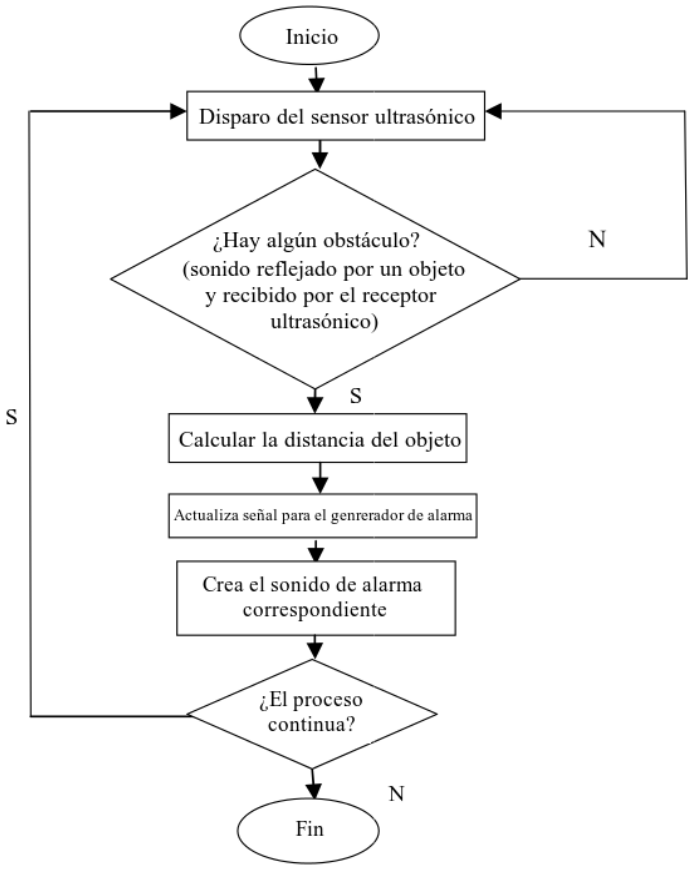
Por otra parte en la universidad internacional de Florida (Aguerrevere, Maroof, & Barreto, 2004), se creó un sistema que haciendo uso de sonidos especializados 3-D basado en lecturas de un sistema sonoro multidireccional, lograba detectar objetos de su entorno. Las investigaciones continuaron y se comenzaron a incluir sistemas con la capacidad de obtener información visual, el prototipo diseñado por (Sainarayanan, Nagarajan, & Yaacob, 2007) de la universidad de Malaysia Sabah daba mayor importancia en los objetos que se encontraban en el centro de la visión, por ende a través de una video cámara digital se captaban imágenes y se realizaba una segmentación de las mismas donde se permitiera distinguir entre el fondo y los obstáculos, finalmente la imagen se dividía en derecha e izquierda transformando esto en sonidos de tipo estéreo enviados por audífonos, a pesar de ser un sistema que trabaja en tiempo real al tener una respuesta auditiva no se logra tener información sobre distancias con respecto a un objeto dado. En otros dispositivos la propuesta consiste en transformar la información visual en señales sensibles al tacto con el fin de no obstaculizar la audición del usuario, modelos como los expuestos por los investigadores (Ito et al., 2005) con el dispositivo CyARM o el TVS *(del inglés Tactil Vision System)* planteado por (Johnson & Higgins, 2006) de la Universidad de Arizona, alertaban al usuario cuando se encontraba cerca de un obstáculo por medio de señales de vibración con diferentes frecuencias en función a la proximidad del obstáculo. En el caso del CyARM en las diferentes pruebas se demostró que más del 90% de las veces las personas eran capaces de detectar obstáculos ubicados frente a ellos o en lo posible pasar entre ellos, pero la desventaja se encontraba en que el usuario debía sostener el artefacto en su mano y realizar escaneos constantes de su entorno a través de sus sensores ultrasónicos. Por otra parte, el TVS hacia uso de dos cámaras para captar la información visual y crear una representación 2-D del espacio en tiempo real a velocidades normales al caminar, al ser un sistema de tipo wereable el usuario no tiene la necesidad de sostener algún objeto en sus manos pero el procesamiento se realiza en una computadora portátil la cual se carga en un maletín lo que agrega peso por cargar al usuario.

A pesar que se limiten otras características, sistemas algo más sencillos pueden llegar a tener una gran utilidad, ya que cuando se desarrollan prototipos wereables lo que se busca es la comodidad en el usuario y su fácil manejo sin necesidad de algún aprendizaje de codificación en la salida como otros sistemas lo proponen, conllevando a una experiencia de usuario que realmente mejore su calidad de vida. Algunos modelos se encargan de brindar ese tipo de experiencia incluyendo factores como los son la comodidad, diseño y el cumplimiento de alguna tarea simple como lo es la evasión de algún obstáculo.

2.1.1.1 ASISTENTE AUTÓMATA DE PASEO PARA PERSONAS CIEGAS

Éste asistente automático, que a pesar de sus limitaciones no deja de ser una interesante propuesta por parte de los investigadores (Sheikh Sadi et al., 2014) de la Universidad de Ingeniería y Tecnología de Khulna, Bangladesh, consiste en unas gafas con un sensor de ultrasonido adaptado en el marco el cual puede detectar obstáculos de 3cm de diámetro o superiores, a un bajo costo el sistema incorpora tres componentes principales, 1) el sensor de ultrasonido 2) Microcontrolador 3) Generador de alerta. El sensor de ultrasonido puede detectar objetos a una distancia máxima de 3 metros, dicha distancia es medida por el microcontrolador al calcular el tiempo que se demora el ultrasonido en viajar y regresar, dicho sensor se dispara continuamente en lapsos breves de tiempo para tomar medidas constantes. El generador de alerta genera la alarma acorde a la señal que recibe del microcontrolador. En la figura 1 se observa el diagrama de flujo del sistema descrito.

Figura 1. Diagrama de flujo del asistente propuesto.



Fuente: SADI, M.S.; MAHMUD, S.; KAMAL, M.M.; BAYAZID, A.I Electrical Engineering and Information & Communication Technology (ICEEICT). Automated walk-in assistant for the blinds, vol., no., pp.2, 10-12. April 2014

El sonido emitido por el generador incrementa su intensidad en función a la distancia entre el obstáculo y la persona, en éste sistema dicha intensidad se divide en 4 niveles, como se puede observar en la tabla 1, cada nivel corresponde a un intervalo de distancia en metros, en el caso del primer nivel éste intervalo es de 2-3 metros donde el volumen es el mínimo indicando que el usuario no se encuentra en riesgo de chocar con el obstáculo. A diferencia del primer nivel, el último nivel tiene unas propiedades que alertan al usuario que se encuentra muy cerca de un obstáculo, factores como el volumen, la duración del sonido y el intervalo de tiempo de la emisión de la alerta varían considerablemente en éstas condiciones.

Tabla 1. Propiedades usadas en las distancias entre el Usuario y el obstáculo

Distancia del obstáculo	Duración del sonido	Intervalo de Tiempo	Volumen del sonido
2m-3m	1s	1s	40
1.5m-2m	.8s	.8s	45
1m-1.5m	.5s	.5s	50
< 1m	.3s	.3s	55

Fuente: SADI, M.S.; MAHMUD, S.; KAMAL, M.M.; BAYAZID, A.I Electrical Engineering and Information & Communication Technology (ICEEICT). Automated walk-in assistant for the blinds, vol., no., pp.3, 10-12. April 2014

2.1.1.2 BASTÓN DE NAVEGACIÓN PARA DISCAPACITADOS VISUALES

No hay que olvidar los elementos comunes de ayuda para las personas de baja o nula visión, entre estos el más popular es el bastón, el cual no provee información alguna de la forma de un objeto ni mucho menos crea una representación en 2-3-D del entorno donde se encuentra ubicada la persona. Sin embargo, existen algunos proyectos que brindan un nuevo concepto para éstos bastones. Lo desarrollado por (Shiizu, Hirahara, Yanashima, & Magatani, 2007) de la Universidad de Tokai, es un sistema que agrega una funcionalidad de navegación al bastón que normalmente solo sirve para detectar obstáculos alrededor, la navegación se basa en cintas de colores que marcan la ruta por donde la persona puede transitar. Cada color usado es asignado para un destino diferente, permitiendo así fijar una ruta que garantice al usuario llegar a su destino.

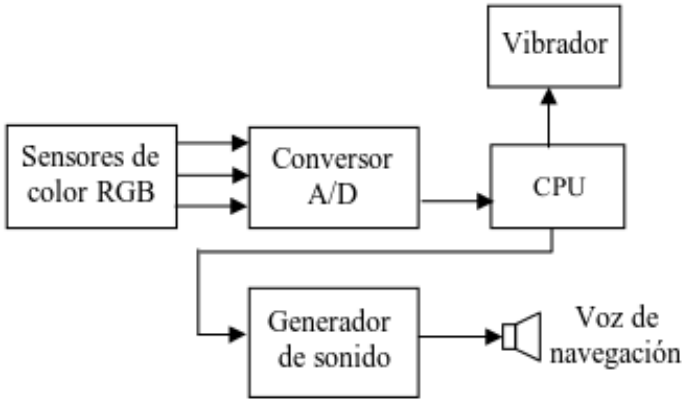
El concepto se ve claramente en la figura 2, un sistema compuesto por la navegación por líneas de colores, la incorporación de etiquetas de identificación por radio frecuencias (por sus siglas en inglés: RFID) y un bastón inteligente el cual incluye un sensor de color, una antena para captar las etiquetas RFID y un sistema de vibración.

Figura 2. Concepto del sistema basado en navegación por líneas de color.

Fuente: Shiizu, Y.; Hirahara, et, al., "The development of a white cane which navigates the visually impaired," in Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, vol., no., pp.5005-5008, 22-26 Aug. 2007

Las líneas de navegación se ponen sobre el piso, en caso de haber varios destinos se usa una línea para cada uno, en cada punto de interés se ubica una etiqueta de RFID la cual indica un código de área puesto sobre la línea de navegación. El bastón inteligente por medio del sensor de color RGB reconoce el color de la línea y de esta manera guía al usuario a través de la ruta, cuando identifica la ruta el usuario es informado a través de la vibración del bastón. Cuando el bastón reconoce una de las etiquetas RFID el procesador de voz notifica al usuario la información correspondiente a esa zona y brinda las indicaciones para llegar a su destino. La figura 3 muestra el diagrama de bloques del sistema implementado para sensar el color en las líneas de navegación. Lo que la señal captada por el sensor RGB entra a un conversor A/D, cuando se tiene una salida digital, la información es procesada por una CPU (microcontrolador) que en función de la señal de entrada coordina la activación del vibrador o el generador de sonido. Básicamente éste es el sistema que hace del bastón convencional un bastón inteligente para otorgar una prestación adicional en el momento de la navegación.

Figura 3. Diagrama de bloques del sistema para censar el color en las líneas



Fuente: Shiizu, Y.; Hirahara, et, al., "The development of a white cane which navigates the visually impaired," in Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, vol., no., pp.5005-5008, 22-26 Aug. 2007.

A pesar de ser un sistema realmente útil, en las pruebas realizadas por los autores se evidenció que el sistema no detecta el uso de los puntos críticos implementados en diferentes rutas, por lo que en las zonas donde existen cambios de dirección el sistema no logra activar correctamente el generador de sonido y el usuario no es notificado sobre las indicaciones necesarias para llegar a su destino.

2.1.1.3 DISPOSITIVO ETA MULTIMODAL

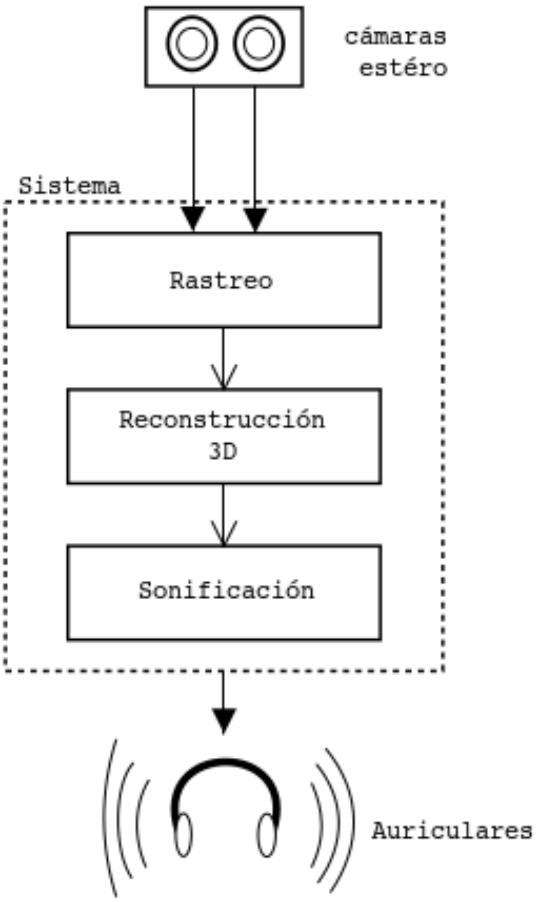
El sistema propuesto por (Fusiello, Panuccio, Fontana, Murino, & Rocchesso, 2002), tiene la capacidad de crear una representación en 3-D a partir de frecuencias sonoras haciendo que este prototipo brinde mayor información al usuario de su alrededor. El sistema consiste en crear reconstrucciones, con técnicas que permitan al usuario tener una percepción en 3-D del escenario, a través de sonidos sintéticos una vez se realice la segmentación de imágenes en 3-D. Haciendo uso de audífonos, dos micro cámaras y un computador el sistema cumple su objetivo, un sistema wereable se construye a partir de un computador portátil y ajustando las cámaras en unos lentes de sol. En la figura 4 se observa la arquitectura general del sistema, el usuario hace uso de un láser como bastón, el sistema se encarga de realizar el rastreo del haz de luz emitido por el láser y calcula la profundidad del mismo y suministra al usuario sonidos adecuados que entreguen la información necesaria para describir el entorno. Adicional a ello lo que se propone son tres métodos de uso del sistema acorde al gusto de cada usuario.

- Láser: la posición en 3-D del punto del láser produce un sonido.

- Sonificación global: todo lo captado por las cámaras produce un sonido. En este caso el láser no se usa. Las imágenes son segmentadas en una región homogénea y la posición y área de cada región distinta es caracterizada por sonidos.

- Profundidad máx/min: similar al anterior, pero el sonido solo caracteriza las regiones más cercanas y más lejanas.

Figura 4. Arquitectura general del sistema ETA multimodal.



Fuente: Fusiello, A.; Panuccio, A.; Murino, V.; Fontana, F.; Rocchesso, D., "A multimodal electronic travel aid device," in *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, vol., no., pp.39-44, 2002

El análisis visual implementado en el sistema y descrito por el autor es una representación del proceso de estereopsis usado por los seres humanos para percibir información de profundidad de su entorno a partir de la disparidad entre las imágenes entre la retina de cada ojo. Al usar un láser el sistema permite localizar el punto en cualquiera de las dos imágenes captadas por cada cámara, lo que soluciona el problema de encontrar cuales puntos de cada imagen son proyecciones del mismo punto del escenario, conocido como un par conjugado, un problema común en la estereopsis computacional.

El proceso que se aplica en cada imagen (derecha e izquierda) para ubicar el objetivo, en este caso el centro del láser, corresponde a la aplicación de un filtro óptico pasa banda de color rojo para cada cámara. Luego con la aplicación de un algoritmo que computa los parámetros α, β de la transformación global en escala de grises, se normaliza el brillo en las imágenes (Ver ecuación 1)[4] ajustando una línea recta a la trama del histograma acumulativo de izquierda versus el histograma acumulativo de derecha.

Ecuación 1. Ecuación que normaliza el brillo en las dos imágenes.

$$I_l(x,y) = \alpha I_r(x,y) + \beta \qquad \forall(x,y)$$

Fuente: Fusiello, A.; Panuccio, A.; Murino, V.; Fontana, F.; Rocchesso, D., "A multimodal electronic travel aid device," in *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on* , vol., no., pp.39-44, 2002

Después de suavizar la imagen evitando el efecto blur que se puede introducir al promediar los píxeles en la imagen, haciendo uso de un umbral (T) automático la imagen es binarizada:

Ecuación 2. Cálculo del umbral

$$T = max\{I(x,y)\} - k$$

Fuente: Fusiello, A.; Panuccio, A.; Murino, V.; Fontana, F.; Rocchesso, D., "A multimodal electronic travel aid device," in *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, vol., no., pp.39-44, 2002

Una vez se minimicen los posibles objetivos de la detección del punto emitido por el láser dentro de la imagen haciendo uso de un filtro de tamaño y la restricción

[4] El autor hace uso del algoritmo descrito en I. J. Cox, S. Hingorani, B. M. Maggs, and S. B. Rao. A maximum likelihood stereo algorithm. Computer Vision and Image Understanding, 63(3):542–567, May 1996.

epipolar, el número correspondiente a la cantidad de objetivos se usa como retroalimentación para ajustar el umbral de binarización en la ecuación 2. El valor de k varía al aplicar una búsqueda dicotómica hasta que la cantidad de objetivos se encuentren en un intervalo entre 0-5. La implementación de una predicción de la posición del punto láser en la imagen incrementa la precisión del rastreo construyendo una matriz de estados la cual describe la posición y velocidad del objetivo en el tiempo discreto y una matriz de transición que describe la evolución del sistema a través del estado actual a un estado siguiente lo que permite predecir el estado futuro. En la ecuación 3 $X(n)$ es la matriz de estados mientras que $\Phi(n,n+1)$ describe la matriz de transición, la posición predicha se usa para coincidir con el objetivo actual con uno de los objetivos candidatos hallados anteriormente. En el caso de obtener un sólo objetivo se toma el candidato más cercano.

Finalmente, por medio de la triangulación se recogen las coordenadas en un espacio 3-D del punto del láser. Cada conjugado par define dos rayos que se intersecan en algún punto en el espacio. Debido a los errores de medición de los puntos de posición y la calibración de las cámaras estéreo los rayos no siempre se intersecan, por lo que en el proceso de triangulación computa las coordenadas del punto 3-D más cercano a los dos rayos.

Ecuación 3. Descripción por medio de matrices de estados.

$$\hat{X}(n+1) = \Phi(n, n+1)X(n) = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_n & y_n \\ \acute{x}_n & \acute{y}_n \end{pmatrix}$$

Fuente: Fusiello, A.; Panuccio, A.; Murino, V.; Fontana, F.; Rocchesso, D., "A multimodal electronic travel aid device," in *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, vol., no., pp.39-44, 2002

Éste sistema depende fundamentalmente de la eficiencia en el seguimiento e identificación del láser a través del espacio, lo que implica que cuando la escena tiene más brillo que el haz de luz emitido por el láser o si éste se encuentra a una distancia no reconocible para las cámaras se presenten problemas en términos de fiabilidad y estabilidad del sistema. Lo que implica que es necesario garantizar unas condiciones de luminosidad máximas que permitan el correcto funcionamiento del sistema multimodal.

2.1.1.4 ASISTENTE DE NAVEGACIÓN INDOOR CON RFID

El prototipo RG *(Robot Guide)* es un proyecto desarrollado por (Kulyukin, Gharpure, Nicholson, & Pavithran, n.d.) del Departamento de Ciencias de la Computación de la Universidad estatal de Utah (USU) y el Centro para Personas con Discapacidad de la misma universidad. La construcción de un robot que permitiera asistir a personas con discapacidad visual en términos de localización y navegación autónoma haciendo uso de sensores de bajo costo.

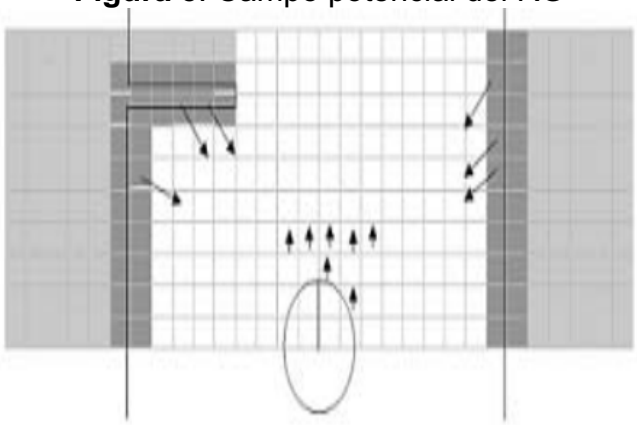
En términos de hardware el RG cuenta con una plataforma Pioneer 2DX, una plataforma robótica comercial que consta de tres ruedas y 16 sensores ultrasónicos, con una batería que ofrece una autonomía de 5 horas .Para más información ver anexo A. Sobre esta plataforma se monta una computadora portátil encargada de realizar el procesamiento, adicional a ello se implementa un lector RFID conectado a una antena y de esta manera hacer uso de las etiquetas RFID distribuidas por el entorno. En cuanto al sistema de navegación, el RG hace uso de campos de potencial y la búsqueda de espacios vacíos a su alrededor. El concepto básico de los campos de potencial es ubicar dentro de la grilla de detección un vector en el cual el robot es rechazado de los obstáculos y atraído hacía el objetivo, la dirección deseada es la dirección de espacio vacío máximo alrededor del RG, una vez encontrada se convierte en la dirección de destino que guía al robot a través del campo potencial. Este tipo de estrategia permite al robot seguir a través de pasillos, evadir obstáculos y girar sin hacer uso de sensores de orientación como los compases digitales.

El robot hace uso de lecturas láser tomadas cada milisegundo para hallar el máximo espacio vacío. Se establece un límite de distancia inicial en las lecturas, en caso de no hallar un espacio vacío el límite se hará menor hasta hallarlo. Después hallando el punto medio entre las dos lecturas se encuentra la dirección objetivo. El campo potencial del robot es una grilla 10x30 que cubre un área de 12 metros cuadrados según los autores[5], dentro de la grilla se pueden encontrar tres tipos diferentes de celdas, 1) Celdas ocupadas: Las cuales almacenan los vectores de repulsión generado por los obstáculos; 2) Celdas libres: Las cuales tienen los vectores con la dirección obtenida al encontrar el máximo espacio vacío; y 3) Celdas desconocidas: cuyo contenido no se conoce.

---

[5] Kulyukin, V.; Gharpure, C.; Nicholson, J.; Pavithran, S., "RFID in robot-assisted indoor navigation for the visually impaired," in Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on , vol.2, no., pp.1979-1984 vol.2, 28 Sept.-2 Oct. 2004

Figura 5. Campo potencial del RG



Fuente: Kulyukin, V.; Gharpure, C.; Nicholson, J.; Pavithran, S., "RFID in robot-assisted indoor navigation for the visually impaired," in Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on , vol.2, no., pp.1979-1984 vol.2, 28 Sept.-2 Oct. 2004

En la Figura 5 las celdas de color blanco son las que se encuentran libres, las grises son las ocupadas, mientras que las grises de tono más claro son las que no poseen ningún tipo de información. Los vectores de repulsión denotados son sumados para obtener un vector de repulsión resultante, así mismo los vectores de las celdas libres se suman para formar un vector objetivo resultante y finalmente se tiene el vector de resultado final sumando el vector resultante de repulsión con el objetivo. El robot ajusta las velocidades de las ruedas V_1 y V_2 en función de m_r y a_r, ver ecuación 4, donde v es la velocidad del robot y S una constante que determina la finura del giro. Bajo estas consideraciones es que el robot RG logra mantener en la mayoría de los casos una velocidad moderada de 0.7m/s, una velocidad adecuada que permite realizar los giros de manera suave.

Ecuación 4. Cálculo de las velocidades de las ruedas

$$V_1 = V_2 = v + (\alpha_r * S)/m_r$$

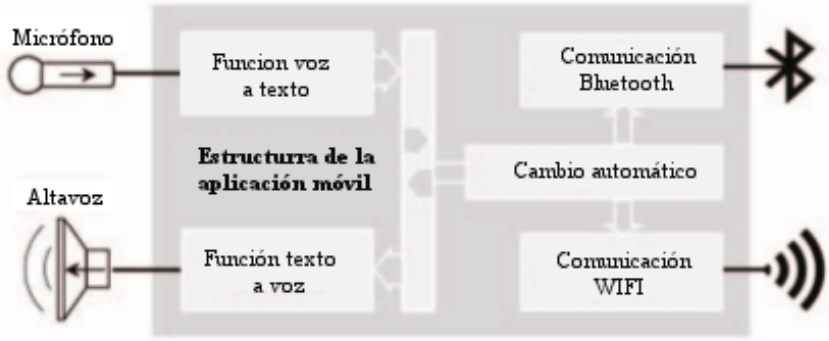
Fuente: Kulyukin, V.; Gharpure, C.; Nicholson, J.; Pavithran, S., "RFID in robot-assisted indoor navigation for the visually impaired," in Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on , vol.2, no., pp.1979-1984 vol.2, 28 Sept.-2 Oct. 2004

La interacción de los usuarios con el robot RG se puede realizar de manera oral, ya que la voz se reconoce y sintetiza con el Microsoft Speech API (SAPI), o haciendo uso de un teclado portable, así mismo se incorporan sonidos no verbales que permiten la interacción del usuario con el entorno.

2.1.1.5 SISTEMA DE NAVEGACIÓN NOVEDOSO EN ENTORNOS CERRADOS PARA PERSONAS CON DISCAPACIDAD VISUAL

El objetivo principal de los desarrolladores (Chaccour & Badr, n.d.) Del sistema que se describe a continuación, es brindar una navegación segura en entornos cerrados con la menor complejidad posible, la construcción de un sistema fiable y de fácil operación para el usuario con el mínimo costo. Éste sistema cuenta con arquitectura muy sencilla donde sus principales componentes son el usuario y el sistema de procesamiento remoto, el usuario será capaz de interactuar con el sistema haciendo uso de su teléfono inteligente, pues la aplicación móvil implementada por los autores, tiene como funciones principales la transcripción de voz a texto y de texto a voz, pues el usuario por medio de la aplicación tiene la capacidad de enviar y recibir comandos e instrucciones al sistema de procesamiento remoto, los mensajes de texto se transmiten a través de comunicación WIFI o Bluetooth. Para éste tipo de sistemas mantener la conexión entre la aplicación y la unidad de procesamiento en cualquier punto del entorno es fundamental, por ésta razón es que se implementan conexiones punto a punto dentro de una red inalámbrica doméstica y si en algún momento se pierde la conexión WLAN la aplicación automáticamente cambia la conexión a Bluetooth. La figura 6 ilustra el modelo de funcionamiento de la aplicación la cual es desarrollada para terminales Android, mientras que actualmente los desarrolladores trabajan en la versión para usuarios con terminales IOS.

Figura 6. Modelo de funcionamiento de la aplicación móvil



Fuente: Chaccour, K.; Badr, G., "Novel indoor navigation system for visually impaired and blind people," in Applied Research in Computer Science and Engineering (ICAR), 2015 International Conference on, vol., no., pp.1-5, 8-9 Oct. 2015

En el entorno de desplazamiento es necesaria la instalación de diferentes cámaras que capturan las imágenes para su procesamiento enviándolas al sistema de procesamiento remoto (unidad central del sistema), así mismo la segunda entrada de dicho sistema es la interfaz WLAN encargada de recolectar los mensajes provenientes del usuario, el sistema analiza la información contenida en cada una de las dos entradas para producir una respuesta textual con las indicaciones

pertinentes a la navegación del usuario. El procesamiento de imágenes se activa cuando se detecta movimiento en el entorno, inicialmente con el procesamiento el sistema busca el marcador que el usuario lleva en su cabeza, una vez encontrado el algoritmo estima las coordenadas e identifica la ubicación. A continuación, el usuario indica el punto de destino, entonces el sistema a través de los comandos de voz guiará al usuario durante el recorrido brindando asistencia no solo en la ruta sino en la detección y evasión de obstáculos.

Éste sistema fue probado en una representación a escala en una maqueta, así mismo el usuario fue modelado con un muñeco de madera, una vez el sistema recibe el comando de texto que indica el lugar de destino enviado por el sujeto, la información se analiza en la unidad central remota y comienza a realizar la planificación de la ruta, una vez el sistema indica al usuario la rotación adecuada para comenzar la navegación, el sujeto recibe las indicaciones adecuadas y el número de pasos requeridos para llegar al punto de destino.

2.1.2 PRODUCTOS COMERCIALES

Diferentes compañías involucradas en la industria tecnológica desarrollan robots de tipo asistencial orientado a un nicho específico en el mercado, aun así los robots comerciales orientados a brindar algún tipo de asistencia a personas con discapacidad visual no ofrece el auxilio requerido, ya que no abarca aspectos importantes que deben ser considerados y sin embargo su precio es considerado muy alto por algunas personas en relación con la baja utilidad durante su uso.

2.1.2.1 BASTÓN K-SONAR

La compañía Bay Advanced Technologies Ltd (por sus siglas en inglés: BAT) ofrece al mercado el bastón K-SONAR[6], un dispositivo basado en el concepto de ecolocación con un funcionamiento muy similar al asistente autómata planteado por los investigadores de la Universidad de Ingeniería y Tecnología de Khulna, Bangladesh, descrito anteriormente en la sección 2.1.1.1, al igual que aquel dispositivo el K-SONAR hace uso de un sensor de ultrasonido con tecnología KASPA reconocida por la Sociedad Acústica de América *(del inglés Acoustical Society of America)*, que proporciona un mejor ancho de banda que los sensores comunes otorgando 10 veces más poder.

[6] http://www.ksonar.com/spanish/how-ksonar-works.php

Para tener un adecuado uso del K-SONAR el usuario debe tener un periodo de aprendizaje para acostumbrar a su cerebro a reconocer las diferentes secuencias de sonido emitidas por el dispositivo al detectar un objeto para su correcto reconocimiento, por lo que el proceso de adaptación para el uso del K-SONAR varía en cada persona. Este dispositivo se puede encontrar en el mercado con un precio entre 1000 a 1100 USD.

Figura 7. Bastón K-SONAR de la compañía BAT Ltd.

Fuente: <http://www.ksonar.com/spanish/how-ksonar-works.php> [en línea]

2.1.2.2 MINIGUIDE

De manera similar al K-SONAR, el Miniguide[7] implementa la técnica de ecolocación para detectar objetos en el entorno del usuario, la diferencia con respecto al dispositivo de BAT Ltda., es que el Miniguide alerta al usuario con una salida vibradora que indica la distancia del objeto, aunque los fabricantes (GDP Research), incluyen una retroalimentación auditiva como segunda opción de respuesta. Su mecanismo es simple, la frecuencia de la vibración es directamente proporcional a la distancia del objeto, por lo que para objetos cercanos las vibraciones serán más rápidas.

El dispositivo de dimensiones (80 x 38 x 23 mm) cuenta con una batería que brinda hasta 100 horas de autonomía en vibración continua, además cuenta con diferentes modos de operación en función de la distancia que se desee detectar los objetos; 0.5, 1, 2, 4 y 8 metros son los modos principales que ofrece el dispositivo que según los fabricantes no es recomendable usarlo sin otro tipo de ayuda primaria ya sea el perro guía o un bastón. Su precio está alrededor de 500 USD.

[7] http://www.gdp-research.com.au/minig_1.htm

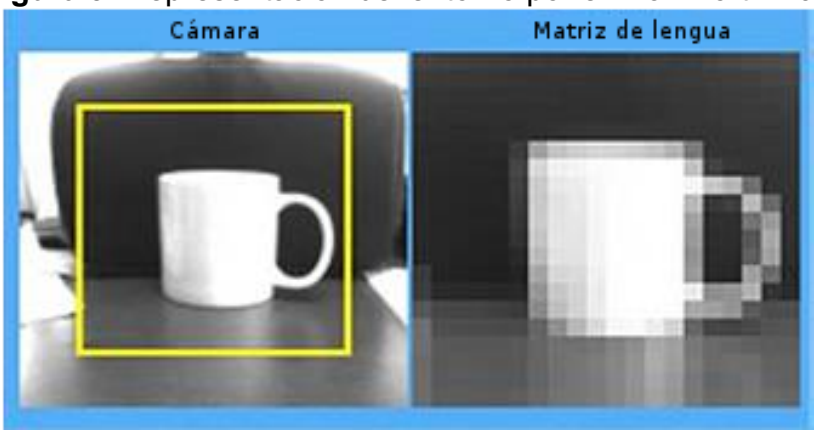
Figura 8. Miniguide

Fuente: <http://www.gdp-research.com.au/minig_1.htm> [en línea]

2.1.2.3 BRAINPORT V100

La compañía norteamericana Wicab Inc., lanzó al mercado el BainPort V100, un dispositivo que usa una cámara ubicada en unas gafas de sol, un control y un electro-simulador que el usuario debe ubicar en su lengua. Éste último elemento va conectado a las gafas y consta de 400 electrodos los cuales estimulan la lengua del usuario según las imágenes captadas por la cámara. El color blanco por defecto genera intensas simulaciones, mientras que cuando se capta el color negro hay ausencia de las mismas. De esta manera el usuario tiene una representación de tipo táctil en vivo de su entorno. La figura 9 muestra el concepto del funcionamiento de este dispositivo el cual para su correcto uso es necesario un entrenamiento adecuado que involucre un proceso de aprendizaje y adaptación por parte del usuario. El precio de éste dispositivo es de 10.000 USD.

Figura 9. Representación del entorno por el BrainPort V100



Fuente: <http://www.wicab.com/> [en línea]

Figura 10. BrainPort V100

Fuente: <http://www.wicab.com/> [en línea]

2.1.2.4 KAPTEN MOBILITY GPS

En el año 2012 la empresa Irie AT™ pone en el mercado el dispositivo Kapten Mobility GPS[8], un artefacto pequeño que brinda una navegación a manos libres que incluye reconocimiento de voz y diferentes modos de operación según las necesidades del usuario. Cuenta con dos métodos de entrada de datos, voz y texto a través del micrófono y el teclado incorporado. Adicional cuenta con mapas embebidos pre cargados de los países de Estados Unidos y Canadá.

Dentro de sus modos de navegación se encuentra la navegación libre el cual entrega al usuario información detallada de su entorno, incluyendo nombres de calles, intersecciones, lugares de interés, etc. Un dispositivo enfocado a otorgar libertad y autonomía al usuario para hacer recorridos fuera de su casa. Su costo es de 499 USD.

[8] http://irie-at.com/content/kapten-mobility-gps

Figura 11. Dispositivo Kapten

Fuente: <http://irie-at.com/content/kapten-mobility-gps> [en línea]

2.2 SLAM

Localización y mapeo simultáneo (por sus siglas en inglés: SLAM), es un término general para los algoritmos que construyen mapas de un entorno desconocido y al mismo tiempo realizan una estimación de su posición dentro del área. Este método se utiliza generalmente en robots y en vehículos que se desea sean autónomos. Lo primero que debe realizar un robot para ser autónomo, es crear un mapa del entorno donde desea navegar y además conocer su posición dentro del mismo, luego de esto, se procede a trabajar en la navegación y planificación de rutas.

Por último, se tienen en cuenta las decisiones inteligentes del vehículo que depende de la información que tenga disponible y las acciones que puede tomar con respecto a dicha información; hecho esto, podemos determinar si realmente el robot es autónomo.

Figura 12. Mapa construido usando SLAM

Fuente <http://robohub.org/wp-content/uploads/2014/10/RTAB_14.jpg>

Las primeras discusiones sobre el SLAM se presentaron en 1986 durante la conferencia de Robótica y Automatización del IEEE en San Francisco, California. Desde aquel evento se comenzó a indagar la forma en que los métodos probabilísticos pudieran integrarse con la robótica; algunos asistentes hablaron acerca de cómo implementar métodos teóricos de estimación en los problemas de localización y mapeo. Una de las grandes conclusiones al final de este evento fue el reconocimiento y la importancia de las bases estadísticas como parte fundamental en la robótica.

En años posteriores, se construyeron estudios clave sobre el tema; uno de ellos fue el trabajo de (Smith & Cheeseman, 1986) y (H. F. Durrant-Whyte, 1988), los cuales establecieron la base probabilística para demostrar la relación existente entre los puntos de referencia y la manipulación de la incertidumbre geométrica.

Durante la misma época, también se produjeron los primeros estudios sobre navegación visual y navegación por sónar, usando algoritmos basados en los filtros de Kalman. Todo esto llevó a que la localización y el mapeo fuera formulado como un único problema de estimación; además se reconoció que debía existir correlación entre los objetos de referencia, los cuales en investigaciones de otros científicos habían sido reducidos o eliminados, pero que representaban una parte importante

del problema ya que entre más ricas son las correlaciones, la solución es mucho mejor según la investigación de (H. Durrant-Whyte & Bailey, 2006).

El SLAM es considerado como el problema de construir un mapa de un entorno desconocido por un robot móvil mientas que al mismo tiempo navega por el mismo utilizando el mapa. Algunos componentes para el estudio del SLAM son: extracción de *landmarks* (bases estadísticas para describir los objetos de referencia), asociación de datos, estimación de estado, actualización de estado y actualización de landmark. A lo largo del tiempo, se ha utilizado con frecuencia *laser-scanners* para el proceso de captura de información que se usa para el mapeo del entorno, este tipo de dispositivos a pesar de ser bastante precisos resultan ser muy costosos. En este proyecto el *láser-scanner* será remplazado por un Kinect™, el cual es más económico y está acorde con las necesidades del proyecto.

Para lograr obtener un mapa consistente y que el robot logre ubicarse rápidamente, se debe tener en cuenta algunas características para los puntos de referencia, ya que deben ser fáciles de distinguir y puedan ser re-observados en el entorno. A continuación se describen algunas pautas que se deben considerar para tener un buen landmark de acuerdo con (Riisgaard & Blas, 2004) en su libro *"SLAM for Dummies"*:

- Deben ser distinguidos desde diferentes puntos de vista.
- Deben ser los suficientes como para que el robot no pase demasiado tiempo sin tener una referencia ya que el robot puede perderse.
- Los puntos de referencia deben ser estáticos ya que, por su misma razón de ser, en caso de re ubicarse el objeto el robot podría desubicarse.
- Los landmarks deben ser lo suficientemente distinguible de otros objetos que tengan características en común, ya que, si el objeto reconoce dos puntos de referencia que son parecidos y en el futuro los vuelve a encontrar, él debe ser capaz de reconocer cual es cual.

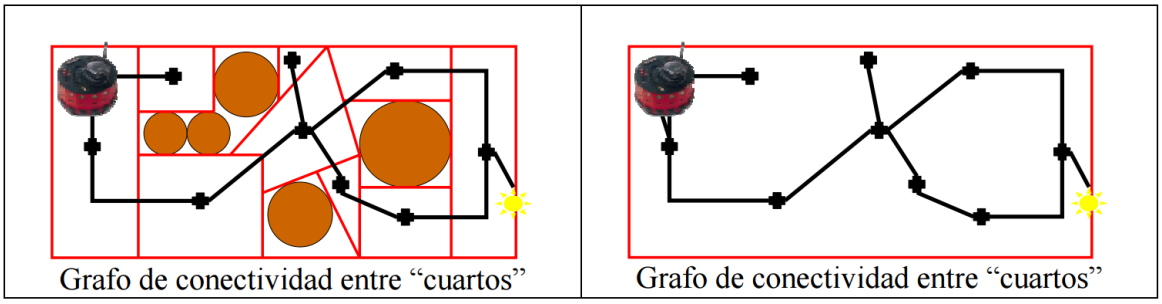
El mapa se construye a partir de la información obtenida del entorno, esta se utiliza de tal forma que sirva para poder representar gráficamente el espacio y planificar rutas de navegación. Esto a nivel general, se puede ver desde dos puntos de vista: *topológico* y *métrico*.

2.2.1 MAPAS TOPOLÓGICOS

Los mapas que son generados desde el punto de vista topológico, normalmente se representan en forma de grafos en los cuales se representan todas las posibles ubicaciones donde puede estar el robot, utilizando nodos y en forma de aristas se representa la posibilidad de acceder a una ubicación desde la otra. En pocas palabras, únicamente se representa los lugares donde puede ubicarse el robot y las conexiones que existen entre ellos. A medida que se explora el entorno, se van almacenando la descripción de cada lugar para ir trazando una trayectoria. Cada vez que encuentra un nuevo lugar, lo asocia con el lugar visitado anteriormente para poder posteriormente llegar al sitio. La construcción del mapa se hace de forma lineal hasta que encuentre un lugar que ya se halla almacenado con antelación y es donde se cierra el ciclo o bucle.

Esta técnica presenta algunas falencias con respecto a la asociación de datos, debido a que el mapa almacenado si esta correlacionado con el mapa que se está capturado, podría generar problemas de ubicación al confundirlo de entorno, además, al utilizar esta técnica en entornos abiertos (tales como habitaciones grandes o halls, etc.) no captura de forma correcta el mapa del entorno, debido a que los sensores tienen un alcance limitado para el reconocimiento de objetos.

Figura 13. Mapa topológico.



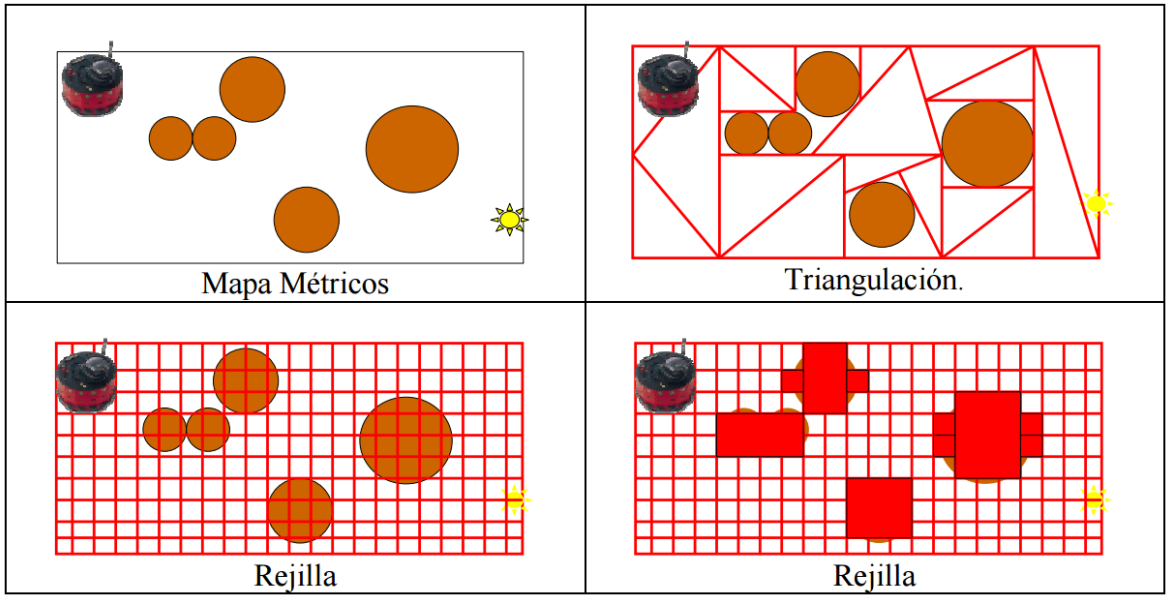
Fuente <http://es.slideshare.net/jfk791021/robots-mviles>

2.2.2 MAPAS MÉTRICOS

Las representaciones métricas consideran todos los objetos ubicados en el espacio bidimensional en el que se llevara a cabo el desplazamiento. Esta es la representación más usada para resolver el problema del SLAM ya que dentro de sus ventajas se encuentra la precisión de las coordenadas donde se ubican los objetos en el entorno.

Las representaciones métricas representan el espacio libre y/o obstáculos mediante una desratización del espacio, el cual se divide en celdas de tamaño predefinido en forma geométrica y estas a su vez se clasifican como ocupadas o vacías acompañado de un nivel de confianza utilizando probabilidad.

Figura 14. Mapa métrico.



Fuente <http://es.slideshare.net/jfk791021/robots-mviles>

Las técnicas que se evalúan a continuación son bajo las cuales son creados los paquetes más comunes que trabajan la tarea de SLAM dentro del framework que se implmenta en el sistema propuesto en el presente documento, ROS (Ver sección 2.4). Dichas técnicas obtienen mapas métricos de los entornos donde son aplicadas, brindando así resultados con gran precisión por las bondades que ofrecen los mapas métricos en el desarrollo de SLAM.

2.2.2.1 GMAPPING

El paquete Gmapping pertenece al proyecto OpenSlam ubicado en los repositorios de ROS. Es una técnica de SLAM desarrollada por (Grisetti, Stachniss, & Burgard, 2005) basados en los artículos publicados por Kevin P. y otros colegas (Doucet, Freitas, Murphy, & Russell, 2000; Murphy, 2000), quienes dieron a conocer el filtro de partículas llamado Rao-Blackwellized, el cual resultó ser muy eficiente para resolver problemas de localización y mapeo simultaneo.

El principal problema del Rao-Blackwellized es su complejidad que crece con el número de partículas que requiere para construir un mapa. Es por esto, que los investigadores se propusieron reducir la complejidad de este algoritmo para construir mapas con una cantidad menor de partículas manteniendo un porcentaje alto de exactitud(Van Der Merwe, Doucet, De Freitas, & Wan, 2001).

Uno de los grandes desafíos para los investigadores que usan este filtro es lograr reducir la cantidad de partículas necesarias para construir un mapa preciso, a esto se le debe añadir que durante la etapa de re-muestreo puede eliminarse la partícula correcta; a este efecto se le conoce como *el problema de agotamiento de partículas o empobrecimiento de partículas*.

En la presentación del algoritmo *GMapping*, se hace referencia a dos enfoques usados para mejorar sustancialmente el rendimiento de los filtros de partículas Rao-Blackwellized.

- Una distribución propuesta que considera la precisión de los sensores del robot y nos permite extraer las partículas de una manera altamente precisa.

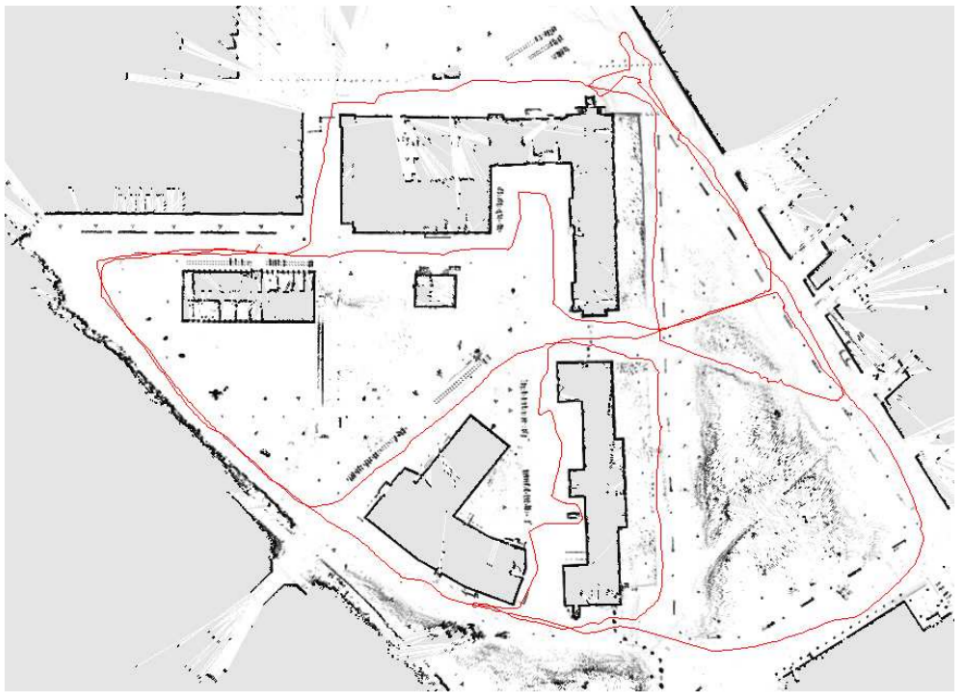
- Una técnica de re-muestreo adaptativo que mantiene una variedad razonable de las partículas y de esta manera permite al algoritmo aprender un mapa exacto al tiempo que reduce el riesgo de agotamiento de las partículas.

La distribución propuesta se calcula mediante la probabilidad alrededor de la partícula que más depende de la pose obtenida por el procedimiento de *scan-matching* junto con la información de odometría. De esta manera, se tiene en cuenta la observación más reciente del sensor para crear la próxima generación de partículas permitiéndonos obtener el estado del robot de acuerdo a más información que la obtenida en un modelo en el que sólo se tiene en cuenta la odometría.

El método propuesto calcula la distribución exacta teniendo en cuenta no sólo el movimiento del robot sino además la observación más reciente. De esta manera, la drástica disminución de la incertidumbre sobre la posición del robot se ejecuta en el

paso de predicción del filtro. Además, aplica un procedimiento selectivo para llevar a cabo la actividad de re-muestreo que se encarga de reducir notablemente el problema de reducción de partículas.

Figura 15. Mapa generado con Gmapping (Grisetti, Stachniss e Burgard 2006)



Fuente: < http://www2.informatik.uni-freiburg.de/~stachnis/pdf/grisetti07tro.pdf>

Esto ayuda a estimar el estado del robot de acuerdo con un modelo dotado de mayor información y por lo tanto más preciso que el obtenido únicamente con información de odometría. El uso de este modelo tiene dos efectos fundamentales:

- El mapa es más preciso, ya que la información actual se incorpora a los mapas individuales después de considerar este efecto sobre la posición del robot. Esto reduce significativamente el error de estimación, de modo que para la siguiente representación se requieren menos partículas.

- El segundo enfoque es la estrategia de re-muestreo adaptativo, que permite actualizar un nuevo muestreo sólo cuando sea necesario y así mantener una diversidad de partículas razonable. Esto resulta una reducción significativa del riesgo de agotamiento de partículas.

Podemos explicar la metodología utilizada en este algoritmo de forma resumida con algunos pasos:

- **Muestreo**: La siguiente generación de partículas que se obtienen de la generación anterior por muestreo de la distribución propuesta.

- **Importancia de los pesos:** A cada partícula se le asigna un peso de acuerdo a su importancia según el principio de muestreo.

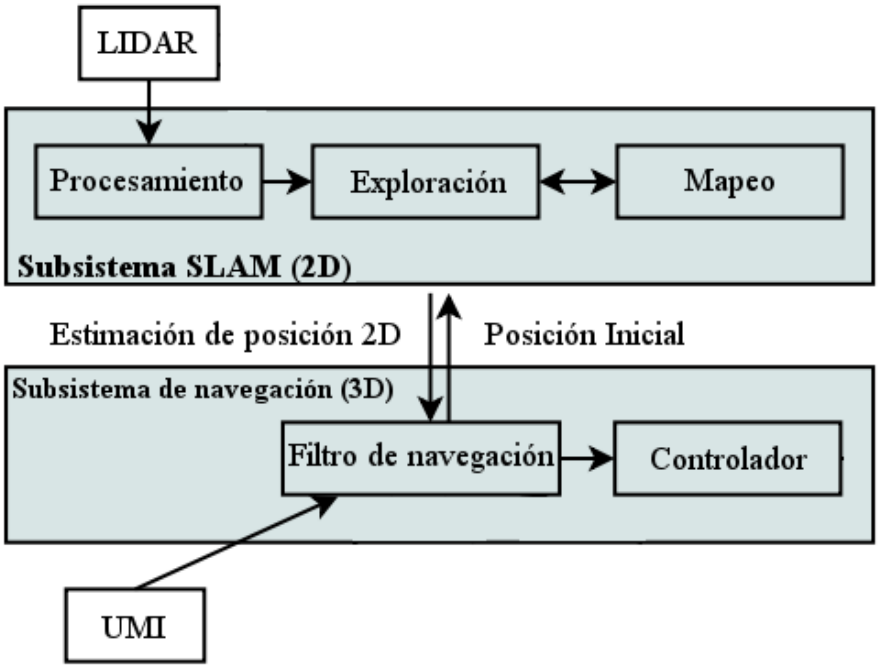
- **Re-muestreo:** Las partículas que tienen menor peso son remplazadas, esta etapa sólo es necesaria cuando un número finito de partículas se usa para aproximarse a una distribución continua. El re-muestreo permite usar un filtro de partículas cuando la distribución se aleja de la propuesta. Luego del re-muestreo todas las partículas tienen el mismo peso

- **Estimación del mapa:** En cada partícula, el mapa estimado correspondiente se obtiene basado en la trayectoria que ha tenido la partícula y en el historial de sus observaciones.

2.2.2.2 HECTOR SLAM

El sistema propuesto por los autores de éste artículo (Kohlbrecher, Von Stryk, Meyer, & Klingauf, 2011) , permite tener una percepción del entorno lo suficientemente precisa sin consumir muchos recursos de máquina lo que lo hace ideal para vehículos cuyos procesadores son de bajo costo, haciendo uso de hardware moderno en términos de LIDAR *(del inglés Laser Imaging Detection and Ranging)* para medir las distancias del sistema con respecto a un objeto en el entorno, se puede obtener la suficiente información para obtener un proceso de SLAM en 2D, esto combinado con un sistema de navegación basado en unidades de medida inercial (por sus siglas en ingés UMI), ver figura 16, el cual interpreta la información proveniente del proceso de SLAM en 2D como una fuente de ayuda para calcular los movimientos de desplazamiento.

Figura 16. Esquema básico del sistema de navegación y mapeo – Hector SLAM



Fuente: S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," *9th IEEE Int. Symp*

Una de las consideraciones más destacadas de éste sistema es que está orientado a plataformas o robots que requieran hasta seis grados de libertad del movimiento, es bien conocida la importancia en la libertad de movimiento en la robótica, por ende el sistema dispone todo el procesamiento de los movimientos de traslación y rotación del agente. Esto se logra a causa de los dos subsistemas que están ligeramente acoplados en términos de sincronización pero en términos de resultados en estimación de posición están separados individualmente, el filtro de navegación combina la información de la UMI y otras variables provenientes de sensores (Compas, Altímetro, GPS, etc.) obteniendo como resultado información en 3D, por otra parte el subsistema de SLAM en 2D proporciona la información correspondiente a la estimación de posición y el rumbo del robot dentro del plano.

Al ser una técnica adecuada para las plataformas basadas en hardware de tipo LIDAR, se aplica un mapa con grilla para la representación gráfica del SLAM 2D. La exploración o escaneo por parte del láser se transforma en un cuadro de coordenadas permitiendo estimar la orientación de la plataforma, así el escaneo se transforma en una nube de puntos o *point cloud* de puntos finales de escaneo. Dicha nube de puntos permite mejor procesamiento de la información al permitir implementar reducciones de resolución o eliminar valores atípicos. Los autores optimizan la alineación de los puntos finales de los rayos del láser con el mapa construido hasta el momento, empleando el modelo Gauss-Newtoniano de ésta

propuesta de visión por ordenador[9] citada por los autores, haciendo inexistente la necesidad de realizar búsquedas de asociación de datos entre los puntos finales de los rayos o una búsqueda rigurosa de la posición. Tras emplear las operaciones que entregan la mejor alineación del láser con el mapa, al minimizar algunas expresiones que caracterizan las coordenadas de los puntos finales de los rayos, se obtiene la expresión con la que se optimiza la alineación del escáner con respecto al mapa. Adicionalmente se almacenan diferentes mapas con distintas resoluciones que son actualizados simultáneamente, con la información correspondiente de la estimación de posición como resultado del proceso de alineación entre el láser y el mapa asegurando que los mapas sean consistentes y no pierdan detalles sin importar la escala de los mismos.

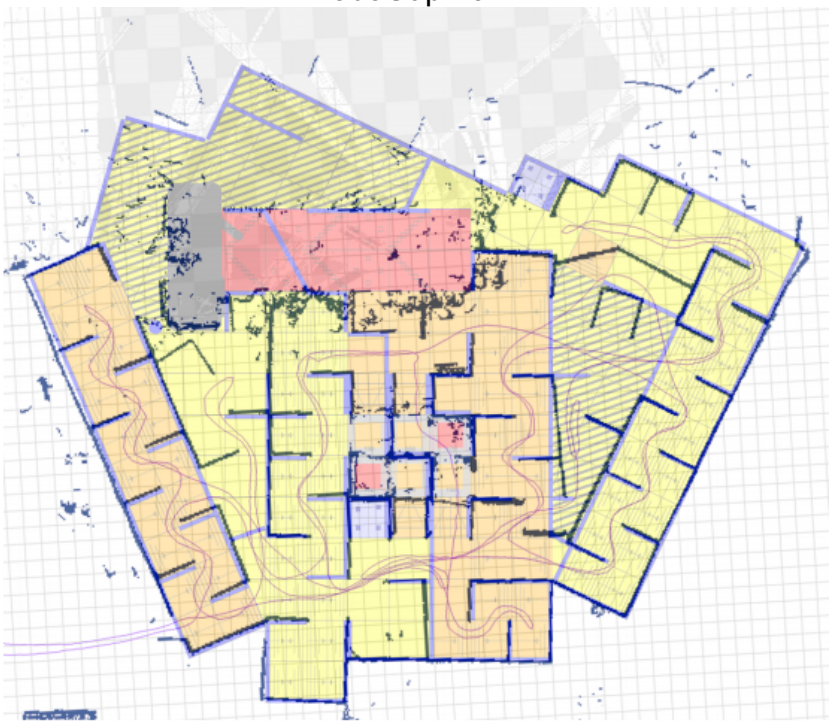
En cuanto a las técnicas de navegación de la propuesta de estos autores, se aplica un filtro de Kalman extendido para estimar la posición en los seis ejes (6D), la posición en 2D es actualizada por la coincidencia en el escaneo, mientras que para una estimación 3D es necesario un sensor de altura adicional en el sistema.

El artículo presenta resultados en donde fue puesto a prueba el sistema de SLAM descrito anteriormente en plataformas y entornos diferentes, especialmente llaman la atención dos plataformas, la primera es el robot móvil Hector UGV el cual está equipado con un láser que ofrece un amplio rango de lectura, y un sensor inercial de 6 grados de libertad. EL anexo B contiene mayor información de Hector UGV. Ésta plataforma móvil fue puesta a prueba en el escenario USAR, Búsqueda y Rescate Urbano, en la Liga Robótica de Rescate de RoboCup[10], obteniendo un muy buen rendimiento en el proceso de SLAM como se puede apreciar en la figura 17.

El otro sistema a destacar es un sistema embebido muy portable, consta simplemente del sistema LIDAR Hoyuko UTM-30LX (Ver anexo C) y una tarjeta de desarrollo basado en un procesador Intel así como el sensor de Unidad de Medida Inercial (por sus siglas en inglés: IMU). Como se muestra en éste video[11], es un sistema de fácil transporte llevado a mano con el que se construye con precisión el mapa del entorno, en éste sistema se observa que la propuesta de los autores trabaja de gran manera con recursos mínimos de cómputo.

[9] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision (darpa),” in DARPA Image Under-standing Workshop, Apr 1981, pp. 121–130.
[10] http://www.robocup.org/
[11] https://www.youtube.com/playlist?list=PL0E462904E5D35E29

Figura 17. Mapa obtenido sobre puesto con el plano real de la *Rescue Arena* RoboCup 2011



Fuente: S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," *9th IEEE Int. Symp. Safety, Secur. Rescue Robot. SSRR 2011*, pp. 155–160, 2011.

La solución propuesta por los investigadores de la Universidad de Darmstadt a pesar de ser optimizada para trabajar con hardware de procesamiento central de bajo costo y con recursos mínimos, no la hace la técnica más viable en términos de implementación para el proyecto presentado, debido a que los sistemas LIDAR son de difícil acceso en Colombia además que tienen un costo muy elevado para el presupuesto manejado para el proyecto, recordando que se quiere plantear una solución de bajo costo.

2.2.2.3 RTAB-MAP SLAM

Real-Time Appearance-Based Mapping *(por sus siglas en inglés RTAB-map)*, es una técnica de SLAM gráfica basada en la implementación de un detector de bucles de cierre incremental por apariencia. Ésta técnica se basa en la obtención de imágenes de tipo RGB-D ofreciendo seis grados de libertad en el proceso de la construcción del mapa como en el sistema Hector SLAM descrito en la anterior

sección. Además solventa un problema denominado por los autores como el secuestro del robot, haciendo referencia al momento en el que la plataforma se apaga por cualquier motivo y es trasladado a otro espacio dentro del mismo entorno, lo que implica que el robot incurra en comenzar nuevamente el proceso de reconocimiento del espacio y la construcción del mapa. Por otra parte, éste método tiene contemplados los requerimientos necesarios para lograr realizar procesos de SLAM a través de multi-sesiones online, de igual manera la propuesta de los autores permite crear mapas a través de diferentes agentes interconectados en línea.

El sistema detallado por los autores en ésta publicación (Labb & Michaud, 2014) se compone de tres partes esenciales para lograr un correcto y óptimo proceso de SLAM; Detección de bucle cerrado, la optimización gráfica (RANSAC) y el administrador de memoria.

- DETECCIÓN DE BUCLE DE CIERRE

  Se puede afirmar que es el núcleo de la propuesta presentada por Mathieu Labbé y François Michaud, autores del sistema. La detección de bucles cerrados a partir de imágenes RGB-D se utiliza para determinar si la observación actual proviene de un espacio ya visto o si corresponde a uno nuevo, de esta manera se van interconectando las imágenes necesarias que son seleccionadas por un eficiente algoritmo y de esta manera se construye un mapa visual en tres dimensiones del entorno. El método tiene como objetivo brindar la posibilidad de realizar SLAM independientemente del tiempo de operación y tamaño del lugar, requisitos necesarios para la operación en línea. Para ello es indispensable una clasificación de espacios o nodos y limitar los mismos que se empleen al realizar una detección de bucle cerrado, entonces cuando el número de espacios en el mapa hace que el tiempo de procesamiento sea mayor a un tiempo umbral establecido, se transfieren los nodos que tengan un menor peso o probabilidad para causar una detección de bucle de cierre desde la memoria de trabajo (MT) a una memoria a largo plazo (MLP) y en caso de que se requieran para completar alguna detección futura pueden retornar a la memoria de trabajo para completar el proceso.

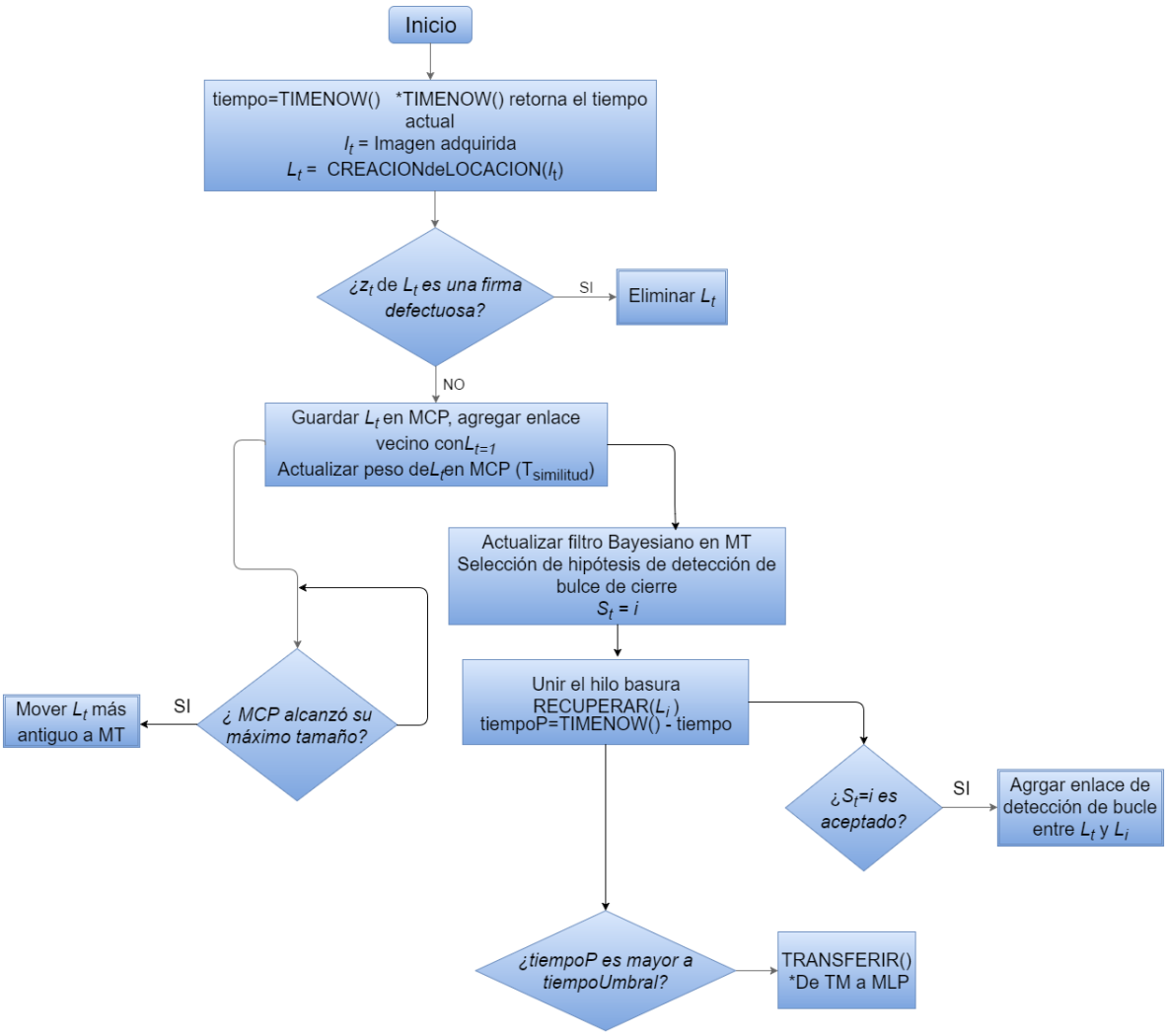
  El peso calculado para cada nodo depende del tiempo en el que dicho espacio ha sido visto consecutivamente, entonces cuando se requiera hacer el proceso de transferir nodos de la memoria de trabajo a la memoria de largo plazo, son transferidos aquellos nodos de menor peso. En caso que existan varios nodos con el mismo peso, el más antiguo es el transferido.

  Un módulo de percepción toma las imágenes del entorno y las lleva a la memoria sensorial (MS), allí se evalúan las ilustraciones para reducir dimensionalmente

los datos en las mismas y extraer las características útiles que permitan realizar el proceso de detección de bucle cerrado. Una vez completado dicho proceso, se crea un nuevo nodo con los detalles de la imagen y se almacenan en la memoria de corto plazo (MCP), allí se actualiza el peso del nodo y en caso tal que el nuevo lugar sea similar al último almacenado en la MCP se fusionan para crear un nuevo nodo con mayor peso. Hay un tamaño fijo de la MCP en función de la tasa con la que se adquieren las imágenes y la velocidad del robot, cuando los nodos superan éste tamaño el más antiguo se envía a la MT para que se tome en cuenta en el proceso de detección de bucle de cierre.

En síntesis la MCP se dedica a observar similitudes entre las imágenes secuenciales que se captan a través del sensor en pro de calcular el peso para cada nodo, por otra parte la MT tiene el papel de detectar los bucles de cierre entre los nodos en el entorno, mientras que la MLP “hiberna” los nodos que contienen un peso bajo para liberar el procesamiento en la MT, aunque pueden ser utilizados en caso que sean requeridos. La figura 18 muestra el diagrama de flujo con las generalidades del proceso descrito anteriormente.

Figura 18. Diagrama de flujo del algoritmo del proceso de selección



Fuente: M. Labbe and F. Michaud, “Appearance-based loop closure detection for online large-scale and long-term operation,” *IEEE Trans. Robot.*, vol. 29, no. 3, pp. 734–745, 2013

❖ Creación de nodo o espacio

De acuerdo a lo mencionado por los autores en (Mathieu Labbe & Michaud, 2013), la identidad de cada imagen z_t tomada en un tiempo t, se representa por palabras visuales de un vocabulario incremental construido en línea, basado en el modelo propuesto en (Sivic & Zisserman, 2003). Las características de aceleración robusta conocidas como SURF *(del inglés Speeded-Up Robust Features)*, se extraen de las imágenes para asociarlas a una de las palabras incluidas en el vocabulario gráfico. Los detalles más predominantes en cada ilustración son seleccionados de acuerdo a la respuesta de característica, de esta manera se desechan características que no superen una constante de respuesta $T_{response}$. Dentro del proceso se consideran como identidades erróneas aquellas que tengan pocas características SURF de acuerdo a la constante de características promedio por imagen T_{bad}. Así, por ejemplo, para entornos cerrados no se consideran identidades válidas las provenientes de una imagen de una pared blanca, puesto que no tiene ningún detalle diferenciador.

Para relacionar las identidades de las imágenes con su correspondiente palabra se comparan las características SURF usando la relación de distancia entre el vecino más cercano y el segundo más cercano, de esta manera si la distancia del vecino más cercano es inferior a *N* veces el segundo vecino más cercano las dos características se representan por la misma palabra visual. Dentro de la fase de búsqueda de aquellos vecinos con mayor cercanía, es necesaria la implementación de algoritmos eficientes de búsqueda, puesto que los vectores de descripción de las características SURF tienen una gran dimensión (vector de dimensión 64), por esta razón los autores emplean cuatro árboles k-dimensionados de acuerdo al algoritmo de búsqueda FLANN (del inglés Fast approximate nearest neighbors), presentado en (Muja & Lowe, 2009). Al tener un menor tiempo de construcción los autores prefieren implementar árboles aleatorios, sin embargo la técnica FLANN obliga a que el árbol sea reconstruido en línea en cada cambio dentro del vocabulario visual. Una nueva palabra se agrega al diccionario en el momento en que una característica extraída de una imagen I no satisface la relación de distancias entre los vecinos más cercanos mencionada anteriormente. Cuando se crea una nueva palabra ésta se agrega al vocabulario y a su correspondiente identidad.

Finalmente un espacio L_t se crea con la información de la identidad de una imagen z_t y el tiempo t. Todos los nuevos espacios tienen un peso inicial de cero. A continuación se muestra el algoritmo de creación de un nuevo espacio extraído del artículo (Labb & François, n.d.).

Algoritmo 1. Algoritmo para la creación de nuevos espacios

Creación de un nuevo espacio L con una imagen I
1: función LOCATIONCREATION($\boldsymbol{I}$) **2:** $\boldsymbol{f}$ = detección de un máximo de características SURF de una imagen I con respuesta de característica superior a $\mathbf{T_{response}}$ **3:** $\mathbf{d}$ = extracción de descriptores SURF de I con las características $\mathbf{f}$ **4:** Construcción de los árboles k-dimensionados **5:** $\mathbf{z}$ = cuantización de los descriptores $\mathbf{d}$ **6:** $\mathbf{L}$ = creación de un espacio con identidad $\mathbf{z}$ y peso 0 **7:** retornar $\mathbf{L}$ **8: fin de la función**

Fuente: M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Trans. Robot.*, vol. 29, no. 3, pp. 734–745, 2013

- Actualización de peso para los nodos

La ecuación 5 evalúa la similitud entre el presente nodo con el último nodo guardado en la MCP para calcular el peso correspondiente a cada nodo nuevo.

Ecuación 5. Cálculo de similitud entre los espacios más recientes

$$\mathfrak{s}(z_t, z_c) = \begin{cases} N_{par}/N_{zt}, & N_{zt} \geq N_{zc} \\ N_{par}/N_{zc}, & N_{zt} < N_{zc} \end{cases}$$

Fuente: M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Trans. Robot.*, vol. 29, no. 3, pp. 734–745, 2013.

Para la ecuación anterior N_{par} corresponde al número de pares de palabras que coinciden entre las identidades de los nodos de ubicación que están siendo comparados, mientras que N_{zt} y N_{zc} son los números de palabras totales y los números de palabras de identidades comparadas respectivamente. Cuando el resultado de la estimación de similitud entre los nodos es mayor a una constante cuyo valor oscila entre cero y uno, el nodo de comparación L_c es fusionado con el nodo L_t, guardando en el vocabulario visual, aquellas palabras que se encuentran en las dos identidades z_c y z_t. Finalmente el peso del nodo L_t se incrementa por el peso del nodo de comparación L_c incrementado en uno, los enlaces que relacionan

los vecinos y los bucles de cierre son redireccionados al nodo resultante L_t y L_c se elimina de la MCP.

❖ Filtro Bayesiano

Éste filtro discreto es indispensable, ya que almacena el seguimiento de las hipótesis de bucle cerrado al estimar la probabilidad con la que un nodo o lugar actual coincida con un lugar visitado anteriormente (almacenado en la MT). S_t Es la variable aleatoria con la que se representa el estado de todas las hipótesis de lazo cerrado. $S_t = i$ Es la probabilidad de que L_t tenga alta coincidencia con un espacio anterior L_i. $S_t = -1$, define la probabilidad para que L_t se considere como un nuevo nodo. La probabilidad completa que estima el filtro para la hipótesis está dada por la ecuación 6.

Ecuación 6. Probabilidad para aceptar una hipótesis de bucle cerrado.

$$\boldsymbol{p}(S_t|L^t) = \eta\, \boldsymbol{p}(L_t|S_t) \sum_{i=-1}^{t_n} \boldsymbol{p}(S_t|S_{t-1} = i)\, p(S_{t-1} = i|L^{t-1})$$

Fuente: M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Trans. Robot.*, vol. 29, no. 3, pp. 734–745, 2013.

❖ RANSAC

Abreviación del inglés *Random Sample Consensus,* es un algoritmo iterativo usado para adaptar parámetros de un modelo matemático a datos experimentales, siendo ideal para aplicaciones donde las mediciones tienen un bajo porcentaje de validación a causa de los altos errores de medición lo que lo convierte en un método predilecto para el análisis de imágenes.

Las condiciones para el algoritmo se dan al tener un modelo con n parámetros los cuales se estiman con un conjunto de medidas P. El número de elementos de P debe ser mayor a n. De igual manera se tienen dos subconjuntos de P denominados S y T. El algoritmo trabaja como se menciona a continuación.

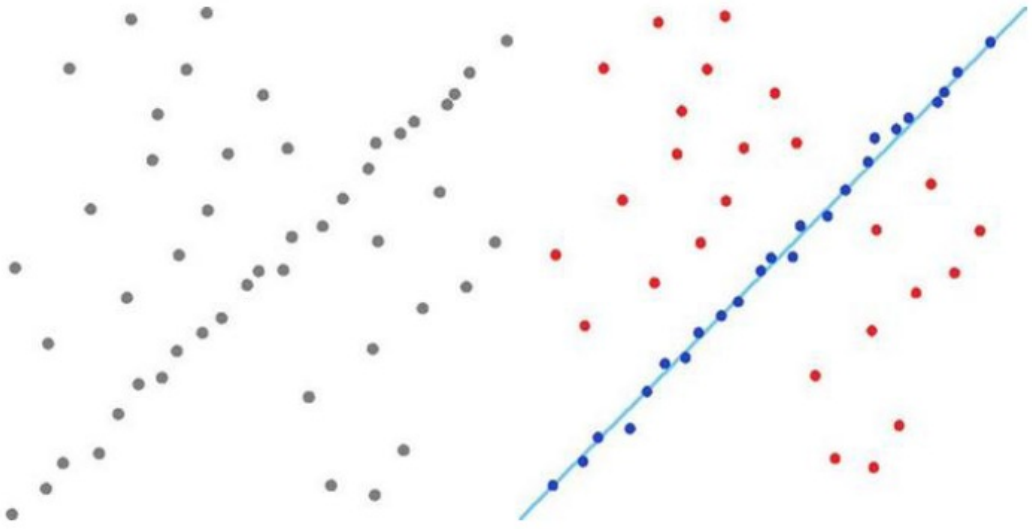
Inicialmente se selecciona un subconjunto de mediciones en P denominado S, con el cual se realiza la primera estimación de los n parámetros del modelo. La estimación actual del modelo selecciona un nuevo subconjunto de puntos, T. Provenientes de las mediciones con cierta tolerancia a errores del modelo. Si T contiene mayor cantidad de mediciones que un límite previamente establecido entonces los parámetros deben ser re-estimados de acuerdo a éste nuevo subconjunto. De esta manera se determina la coincidencia de T con el modelo,

almacenando dicho valor y seleccionando un nuevo subconjunto S. Si las mediciones de T no superan el límite establecido, aleatoriamente se selecciona u nuevo subconjunto S para iniciar una nueva estimación. En el caso en que ninguno de los subconjuntos contengan mayores mediciones que el límite, se da salida con fallo, o si el número máximo de iteraciones es alcanzado, simplemente se da salida.

El método se debe caracterizar por tres parámetros: la tolerancia del error usada para determinar cuáles datos pertenecen al modelo, el número máximo de iteraciones para el algoritmo y el valor límite de mediciones en un subconjunto en la tarea de estimación de parámetros.

Debido a estas características, es realmente útil la implementación de éste algoritmo para detectar líneas rectas dentro de un entorno de tres dimensiones presentes en los ambientes cerrados (paredes, puertas, suelos, etc.) minimizando el cuadrado del error producido a causa de la diferencia de la posición de la estimación con respecto a los puntos estimados. Realizada la estimación, el algoritmo comprueba cuantas mediciones se encuentran cerca de dicha aproximación. Según el autor (Pons, 2012), si el número de coincidencias es superior al límite, se puede determinar la detección de una línea recta, en el caso experimental, algún segmento correspondiente a una puerta, ventana, mueble, pared, etc.

Figura 19. Ejemplo de aplicación de RANSAC en un conjunto de datos atípicos



Fuente: P. F. De Carrera and E. Bernabeu, “Localización y generación de mapas del entorno (SLAM) de un robot por medio de una Kinect .,” 2012

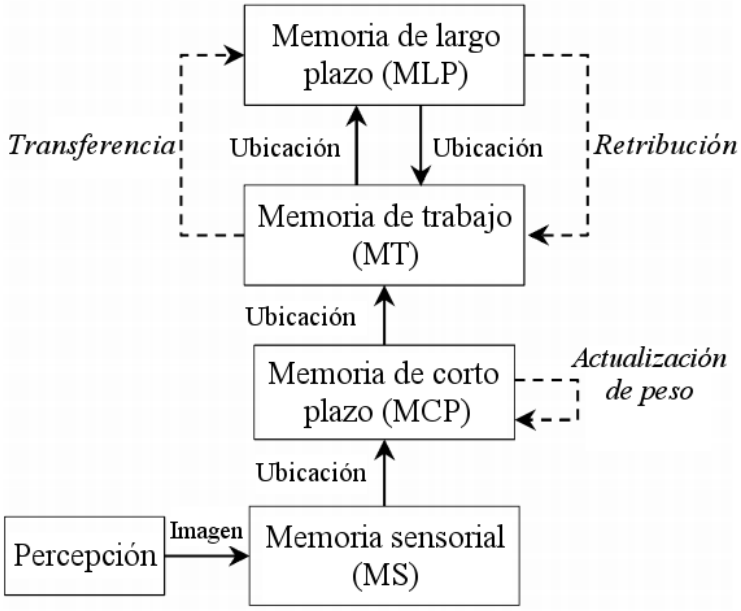
❖ ADMINISTRADOR DE MEMORIA

La figura 20, muestra el esquema del funcionamiento del administrador de memoria manejada en el método descrito, este administrador es el que otorga la facultad de trabajar correctamente en la construcción de mapas online sin importar las dimensiones del entorno ni el tiempo de operación. Cuando el número de ubicaciones en el mapa hacen que el tiempo de procesamiento supere cierto umbral, los nodos con menor probabilidad de causar una detección de cierre son transferidos de la MT a la MLP. Aun así, al detectarse un bucle de cierre las ubicaciones vecinas pueden ser restauradas en la MT para ser consideradas en la detección.

El proceso de transferencia de las ubicaciones a la MLP se debe aplicar cuidadosamente para no transferir ubicaciones que sean de gran importancia en la detección de algún bucle de cierre, como solución los autores le otorgan un peso a cada nodo de acuerdo a que tan propenso sea el mismo para causar una detección válida de igual manera el peso varía en las ubicaciones que tienden a ser más re-visitadas. Así los espacios con menor peso son los que se transfieren a la MLP.

Por otra parte la administración de la MCP se basa en el tamaño fijo de la misma, cuando el número de ubicaciones alcanza el tamaño máximo de la memoria, la ubicación más antigua se mueve a la MT.

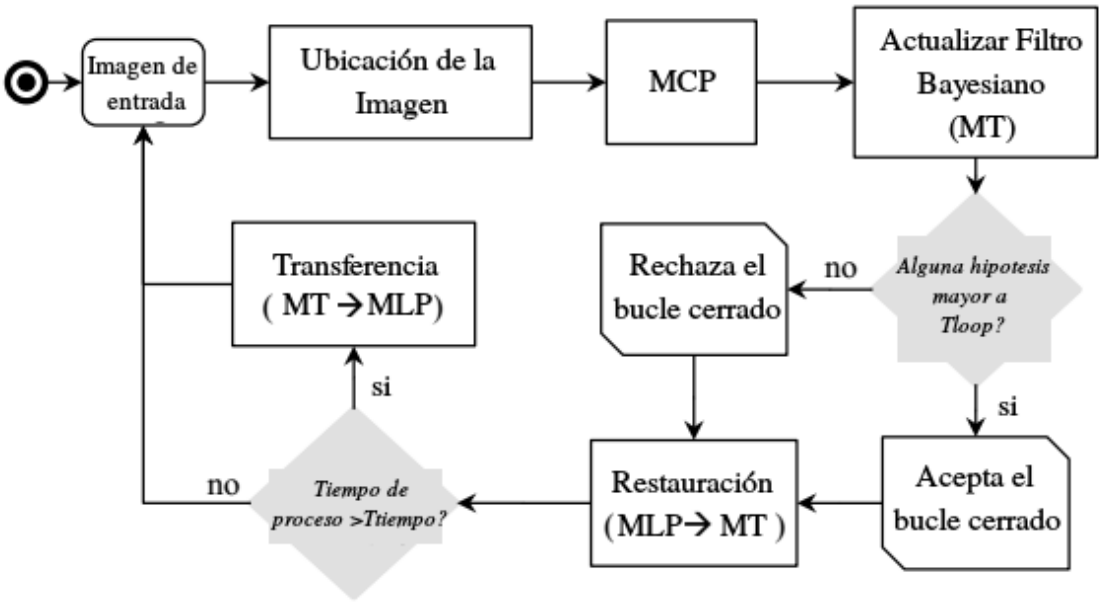
Figura 20. Esquema de funcionamiento del administrador de memoria



Fuente: M. Labb and M. François, “Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation.”

Éste es el modo de operación del *Real-Time appareance-based mapping* planteado como una alternativa para el proceso de SLAM a través de la captura de imágenes RGB-D, en la figura 21 se puede observar las generalidades del metodo descrito (M Labbe & Michaud, 2011). Las propiedades de éste método otorgan una gran ventaja para los sistemas que no hacen uso de dispositivos láseres de escaneo, permitiendo así utilizar sensores de tipo RGB-D como alternativa suficiente.

Figura 21. Proceso general del RTABMap



Fuente: M. Labbe and F. Michaud, “Memory management for real-time appearance based loop closure detection,” *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 1271–1276, 2011

2.3 CINEMATICA

Los robots móviles se pueden clasificar por el tipo de locomoción que utilizan para realizar sus desplazamientos, bajo éste parámetro se encuentran tres tipos de robots móviles: robots con ruedas, por patas y por orugas(SOLAR & SALAZAR, n.d.). Dada la importancia del desarrollo e innovación de los robots móviles, cada uno de los tres anteriores tipos ha ido evolucionando a través del tiempo, sin embargo, los más implementados en los diferentes campos son los robots móviles por ruedas, dado que estos brindan ventajas considerables en aspectos como el diseño mecánico, control y su eficiencia en el ahorro de energía cuando se movilizan en superficies lisas.

2.3.1 ROBOTS MÓVILES CON RUEDAS

A pesar que existen diferentes configuraciones cinemáticas de robots móviles (acuáticos, bípedos, hexápodos, etc); únicamente vamos a hacer énfasis en los robots móviles de ruedas diseñados para ambientes controlados. Los robots móviles terrestres tienen aplicaciones en varios campos de la industria, tales como el almacenamiento, control de producción, control y transporte. Una ventaja principal por la que es conveniente utilizar ruedas como medio locomotor es la facilidad de adaptarlas a cualquier prototipo, puede desplazar un peso mayor que el soportado por otro tipo de locomoción; el control de las ruedas es mucho menos complejo que la actuación de patas o el tipo oruga, ya que requieren menor cantidad de partes y los problemas de balance son menores dado que el robot siempre se encuentra en contacto con la superficie. Sin embargo, entre las desventajas que se encuentran es que difícilmente pueden subir o bajar escalones.

2.3.2 RUEDAS

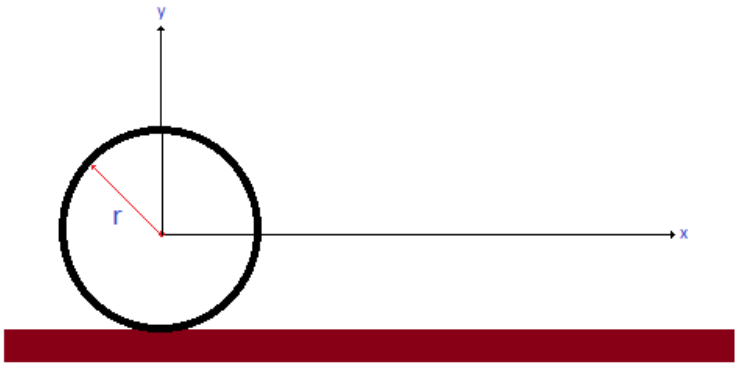
Las ruedas de un robot móvil se mueven por contacto con alguna superficie o fricción, de forma ideal una rueda se desplaza $2\pi r$ por cada vuelta. Los robots móviles utilizan para su locomoción cuatro tipos de ruedas las cuales se clasifican a continuación(Barrientos Sotelo et al., 2007):

Figura 22. Tipos de ruedas

Fuente: V. R. Barrientos Sotelo, R. Silva Ortigoza, V. M. Hernandez Guzman, G. Silva Ortigoza, J. R. Garcia Sanchez, and M. A. Molina Vilchis, “Una panorámica de los robots móviles,” *Télématique*, vol. 6(3), pp. 4–5, 2007

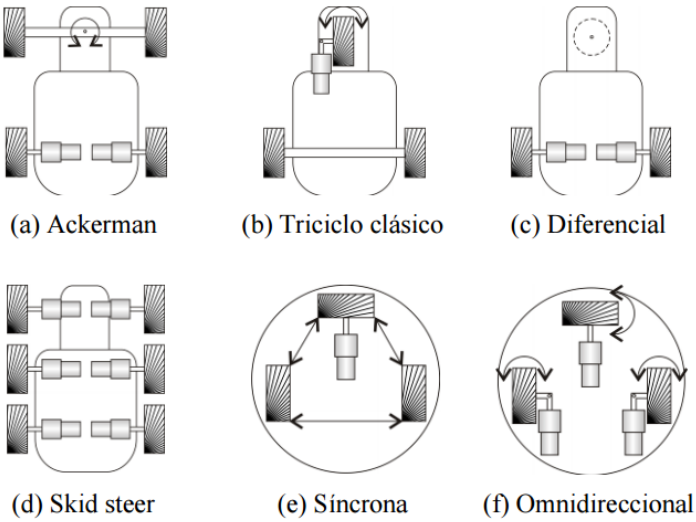
Un robot móvil puede contar con varias ruedas, por lo cual cada rueda se rige por un movimiento circular el cual tiene un punto llamado centro de curvatura instantáneo (ICC).

Figura 23. Centro de curvatura instantáneo.



La locomoción de los robots móviles puede desarrollarse con diferentes tipos de configuración las cuales dependen de manera directa de la aplicación que se le quiere dar al robot; sin embargo, a nivel general se tienen las configuraciones que se describen a continuación: Tracción diferencial, síncrono, Skid Steer, triciclo, ackerman y tracción omnidireccional.

Figura 24. Configuración de los robots móviles con ruedas



Fuente: V. R. Barrientos Sotelo, R. Silva Ortigoza, V. M. Hernandez Guzman, G. Silva Ortigoza, J. R. Garcia Sanchez, and M. A. Molina Vilchis, “Una panorámica de los robots móviles,” *Télématique*, vol. 6(3), pp. 4–5, 2007

Actuadores de los robots móviles

Comunmente los robots moviles terrestres son dotados de motores para su movimiento, los cuales podemos encontrar de gran variedad dependiendo de la necesidad que se tenga. En robotica móvil los motores DC son usados con mayor frecuencia ya que el control de estos motores es menos complejo debido a su modelo lineal.

Existen dos tipos de motores DC de imán permanente: con escobillas y sin escobillas. Los motores sin escobillas tienen algunas ventajas frente a los motores con escobillas a pesar de que tienen caracteristicas similares; los motores sin escobillas pueden alcanzar una velocidad 10 veces mayor a los motores con escobillas, no cuentan con escobillas por tanto no requieren mantenimiento por residuos almacenados o remplazo de las mismas, son mucho mas seguros ya que no generan chispas y esto evita accidentes en ambientes inflamables, se minimiza la interferecia que los motores con escobillas generan por la conmutación mécanica de las escobillas ya que los motores sin escobillas ejercen conmutación electrónica.

2.4 ROS (ROBOT OPERATING SYSTEM)

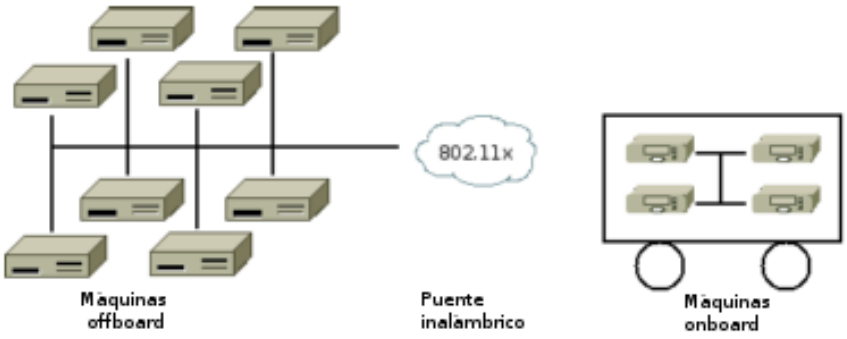
ROS es un framework que permite escribir software orientado a robots el cual incluye un conjunto de herramientas, librerías y paquetes de software que brindan una experiencia de desarrollo completa y modular de manera tal que se puedan crear comportamientos robóticos complejos de una manera más sencilla. ROS nace como un proyecto de robótica propio de *Willow Garage* en el año 2007 aunque después el desarrollo y mantenimiento del framework paso a *Open Source Robotics Foundation*, a partir de su creación el proyecto creció basado en el desarrollo de software en conjunto o colaborativo, es decir el proyecto de ROS al ser construido bajo licencia BSD (open-source) cuenta con la colaboración de muchos laboratorios y comunidades enfocadas a la robótica, por ende un laboratorio especializado en la construcción de mapas en entornos cerrados puede crear un sistema robusto en este campo, mientras que otro grupo de desarrolladores es experto en la navegación y desarrolla los paquetes de software esenciales para cumplir a cabalidad dicha tarea, creando una red que contribuye a un sistema global y general, ROS.

De acuerdo a (Quigley et al., 2009) éste framework cuenta con los siguientes criterios de construcción y diseño:

- ✓ Peer-to-peer
- ✓ Basado en herramientas
- ✓ Multilenguaje
- ✓ Ligero
- ✓ Libre distribución (Open-source).

Los criterios mencionados anteriormente hacen de ROS uno de los frameworks más poderosos para escribir código orientado a la robótica, una de las grandes ventajas es que cualquier sistema construido bajo ROS consiste en diversos *hosts* conectados en tiempo de ejecución bajo topología P2P (Ver figura 26) y de esta manera se evita la congestión de tráfico de datos innecesarios en los sistemas comunes basados en un servidor central, cuando se tienen diferentes centros de procesamiento conectados entre sí a través de una conexión LAN.

Figura 26. Conexión típica de un sistema de ROS

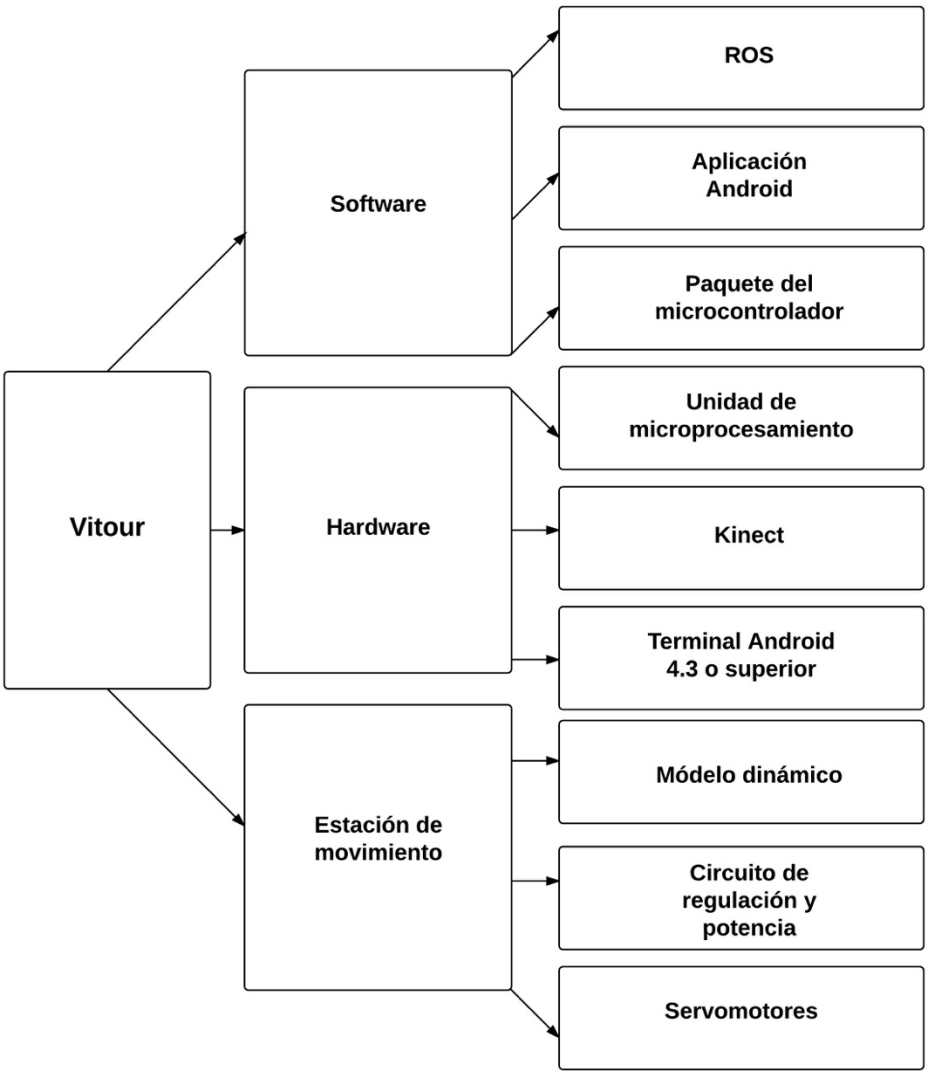


Fuente: M. Quigley, K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," *Icra*, vol. 3, no. Figure 1, p. 5, 2009.

3 DISEÑO DE VItour

En esta sección se describe el proceso de diseño de la plataforma construida para la aplicación de pruebas en entornos reales, cerrados y controlados. VItour (del inglés Visual tour) es el concepto transformado en producto de la creación de un asistente para personas con discapacidad visual a un bajo costo en relación con los demás productos que se ofrecen en el mercado y que han sido relacionados a lo largo del presente documento.

Figura 27. Etapas de diseño Vitour



Fuente: Propia

3.1 APLICACIÓN ANDROID (REMOTE CONTROL)

3.1.1 INTEGRACIÓN CON ROS

Dentro de las virtudes con las que cuenta el framework de ROS se encuentra una completa integración con diferentes librerías y plataformas lo que incurre en una gran ventaja para cualquier desarrollador, dentro de la lista de entornos compatibles se encuentra el lenguaje Java, uno de los lenguajes de programación orientado a objetos con mayor concurrencia y desarrollo en el mundo. Como bien es conocido Android, el sistema operativo orientado a dispositivos móviles de Google lanzado en el año 2007, fue construido como un fork de Linux y sus aplicaciones se desarrollan en su mayoría en lenguaje Java. Por ende, es posible la construcción de aplicaciones móviles bajo esta plataforma que tengan una completa sinergia con ROS. Los desarrolladores de Android tienen a su disposición el Kit de desarrollo para Android o Android SDK (por sus siglas en inglés Software Development Kit), adicionalmente Google en el año 2013 lanzó su entorno de desarrollo integrado (por sus siglas del inglés: IDE) oficial para la construcción de aplicaciones Android conocido como Android Studio, el cual reemplazo a Eclipse como IDE oficial, distribuído de forma gratuita bajo licencia Apache y cuenta con la virtud de ser multiplataforma.

El paquete que permite la integración entre el framework de ROS y Android es *rosjava* el cual se encuentra perfectamente documentado en el wiki[12] donde se describe el proceso de instalación del mismo. Desarrollado en colaboración de Google con Willow Garage rosjava fue anunciado durante el evento de Cloud Robotics en el Google I/O 2011, éste paquete define el núcleo con el cual se pueden desarrollar programas de Java que interactúen adecuadamente con ROS, el módulo rosjava_core contiene la librería cliente que habilita de manera simple la comunicación con los tópicos, servicios y parámetros de ROS, de esta manera se pueden crear clases correspondientes a algún tipo de nodo y éste a su vez puede interactuar con diferentes nodos de igual manera como se describe en el anexo D.

El paquete android_core es una colección de componentes y ejemplos para el desarrollo de aplicaciones ROS en Android, su núcleo se basa en el paquete rosjava y contiene proyectos sencillos que muestran la construcción de aplicaciones basadas en la nueva estructura *gradle* de desarrollo de aplicaciones móviles para Android.

[12] http://wiki.ros.org/rosjava

3.1.2 DESARROLLO

❖ APLICACIÓN REMOTE CONTROL

La aplicación para el proyecto está desarrollada en lenguaje Java en su versión de JRE 1.8.1 a través del IDE Android Studio v 1.5.1. La aplicación móvil permite enlazar el dispositivo Andorid como un nodo al máster central de ROS (Ver anexo D), permitiendo así tener la comunicación entre el dispositivo con el nodo que requiere intercambiar información con el terminal Android. La aplicación tiene una interfaz de usuario simple, y para lograr el enlace con el máster se solicita la dirección IP donde se esté ejecutando el mismo.

El rol de la aplicación dentro del sistema es brindar un control remoto hacia los movimientos del robot VItour, movimiento de avance, retroceso, giro sobre el eje hacia la derecha y giro sobre el eje hacia la izquierda. Para cumplir ésta tarea el terminal se comporta como un nodo publicador, Ver anexo D, el cual envía las señales de control de los tres motores implementados para el movimiento de la plataforma a un nodo que está suscrito al tema o tópico correspondiente al control de los servos motores. A pesar que las instrucciones de control para el giro de cada uno de los motores es independiente, se envían en un mismo dato como vector, optimizando así el uso de variables y el análisis de las mismas por la unidad de micro procesamiento.

En cuanto al comportamiento de la aplicación, cada uno de los botones incluidos en el *layout* del control de movimiento, envían una trama consecutiva del vector correspondiente a la pulsación mientras que el botón se mantenga presionado, una vez se deje de presionar, la aplicación depura los datos residuales que no completaron el proceso de transmisión y de esta manera al evitar un comportamiento inesperado cuando otro botón se presione.

Cabe resaltar que hasta que la conexión P2P entre el nodo y el máster no se genere con éxito la aplicación se encontrará inhabilitada para enviar los datos de control.

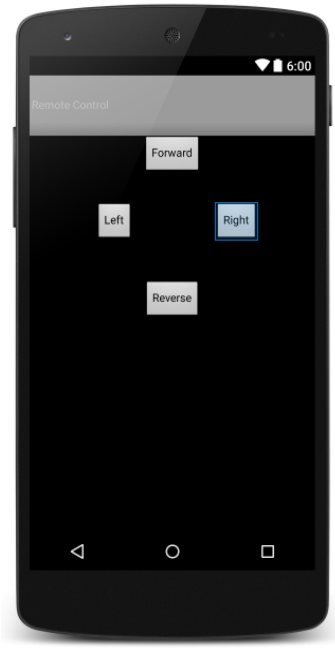
3.1.3 APARIENCIA UI

La interfaz de usuario de la aplicación es muy simple. Como muestra la figura 27(a), inicialmente se observa un menú con dos opciones de uso, la primera denominada “modo mapa” y la segunda “modo navegación”, como infieren sus respectivos nombres con la primera opción se accede a la configuración con la cual se tiene la capacidad de construir un mapa del entorno accediendo a un control manual sobre los movimientos del robot. Para tener control sobre el robot la aplicación pedirá al usuario ingresar el master URI o dirección IP donde está siendo ejecutado el master, ver figura 27(b). Finalmente, la figura 27(c) muestra la interfaz de control remoto a

través de una conexión P2P. El control es sencillo con cuatro botones correspondientes a los movimientos de avanzar, retroceder, derecha e izquierda.

Figura 28. Interfaz de usuario de la aplicación desarrollada

(a) (b)



(c)

Fuente: Imágenes generadas en el IDE Android Studio.

Con estas tres simples interfaces el usuario interactúa con la aplicación y de manera remota con el robot para generar el mapa del entorno que desea. Es importante destacar que es un requisito indispensable que el master URI sea el mismo del master ejecutado con *roscore* en el computador, ya que si no coincide simplemente la comunicación no se podrá establecer, pues hay que recordar que la comunicación de ROS se realiza por medio de P2P.

3.2 CINEMATICA

Modelo cinemático

Para el ejercicio del modelado del robot, se tomó como referencia la cinemática que es el estudio del movimiento de una plataforma móvil cuando no se tienen en cuenta las fuerzas que ocasionan el movimiento.

En el modelado del robot, se tuvieron en cuenta algunas consideraciones con el fin de hacer más tratable la obtención de modelos cinemáticos del robot móvil de ruedas (por sus siglas en inglés: RMR), los cuales se describen a continuación:

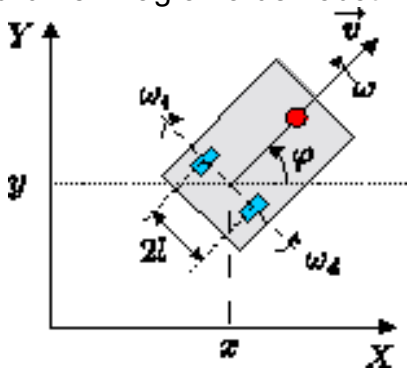
- El robot se mueve sobre una superficie plana, libre de irregularidades de textura y nivel.
- El deslizamiento de las ruedas es despreciable
- Los ejes de desplazamiento son perpendiculares a la superficie de trabajo
- La estructura es totalmente rígida, no cuenta con partes flexibles
- El robot tiene restricciones no holonómicas
- Se desprecia la rueda trasera que únicamente se encarga de apoyar la estabilidad del robot

2.3.3 MODELADO DE CINEMÁTICA DIRECTA

En la configuración cinemática que conforma el robot móvil implementado se consideró que el movimiento se realiza en el plano XY, con una configuración de tracción tipo triciclo. Dado que cada rueda motriz posee su propio motor y cada uno posee su propio circuito de control, es necesario emplear un sistema de tracción diferencial.

Para realizar el modelado del robot, es necesario conocer las condiciones físicas del sistema, en especial las dimensiones del robot. En la Figura 25, (x, y) muestran la posición del punto medio del eje que une las dos ruedas traseras, con respecto al punto de referencia fijo XY, φ describe el ángulo que forma el eje de simétrica del robot con respecto al eje X positivo, wd y wi son las velocidades angulares de las llantas derecha e izquierda, r es el radio de las ruedas y l es la distancia de separación entre el eje medio del robot y la rueda de tracción.

Figura 25. Diagrama del robot móvil.



Fuente: http://publicaciones.urbe.edu/index.php/telematique/article/viewArticle/849/2085

Definidas las variables que son de nuestro interés en el sistema, procedemos a determinar los movimientos que puede realizar el robot móvil. El robot puede desplazarse con una velocidad lineal v y rotar con una velocidad angular φ como se denota en la Figura 25.

Dado que en nuestro prototipo es posible variar la velocidad de cada rueda, podemos variar las trayectorias que el robot toma. Para que el robot se desplace en línea recta, se define que la velocidad de las ruedas de tracción debe tener la misma magnitud; teniendo en cuenta lo anterior, podemos definir que la velocidad lineal es el promedio de la velocidad de las ruedas, siendo proporcional al radio de estas (Enrique, Alejandro, & Leonardo, 2014).

Ecuación 7. Velocidad lineal del robot.

$$v = r\frac{v_r + v_i}{2}$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, “Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial,” *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

Para lograr que el robot pueda girar de forma rotacional en su centro de masa, se define que la velocidad de las ruedas de tracción debe ser la misma, pero con sentido de rotación diferente; dado esto podemos definir que la velocidad angular de rotación es igual a la diferencia de la velocidad de sus ruedas sobre la longitud que hay entre ellas.

Ecuación 8. Velocidad angular del robot.

$$w = r\frac{v_r - v_i}{l}$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, "Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial," *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

Definidas las velocidades que representan el desplazamiento del robot, se procede a integrarlas para definir las ecuaciones de la dinámica del movimiento del robot en cada uno de sus ejes.

Ecuación 9. Dinámica sobre el eje X.

$$\dot{x} = v * \cos\theta$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, "Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial," *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

Ecuación 10. Dinámica sobre el eje Y.

$$\dot{y} = v * \sin\theta$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, "Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial," *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

Ecuación 11. Dinámica sobre el eje de rotación

$$\dot{\emptyset} = w$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, "Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial," *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

Para obtener el modelo diferencial con el cual trabajará el robot, se remplazan las velocidades lineales y angulares obtenidas anteriormente; se plantea una matriz de rotación dado que no siempre el robot va estar alineado con el eje global.

Ecuación 12. Matriz de rotación

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\emptyset}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ w\end{bmatrix}$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, "Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial," *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

Ecuación 13. Matriz de velocidad

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\emptyset}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}r\dfrac{v_{r+}v_i}{2}\\ r\dfrac{v_{r-}v_i}{2}\end{bmatrix}$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, "Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial," *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

De esta manera se pueden obtener las ecuaciones que determinan el movimiento del robot con el modelo cinemático directo hallado. Inicialmente se multiplican las matrices, de rotación y de velocidad del robot, con el cual obtenemos una ecuación matricial que representa las velocidades en los ejes X e Y así como la velocidad angular del robot en el sistema global.

Ecuación 14. Matriz de velocidades de la plataforma

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\emptyset}\end{bmatrix} = \begin{bmatrix}r\dfrac{\cos\theta}{2} & r\dfrac{\cos\theta}{2}\\ r\dfrac{\sin\theta}{2} & r\dfrac{\sin\theta}{2}\\ \dfrac{r}{l} & -\dfrac{r}{l}\end{bmatrix}\begin{bmatrix}w_r\\ w_i\end{bmatrix}$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, "Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial," *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

2.3.4 MODELADO DE CINEMÁTICA INVERSA DEL ROBOT

La cinemática inversa se obtiene teniendo como referencia las velocidades lineal y angular deseadas para el correcto funcionamiento del robot. Dado que las variables de entrada que se tienen de la plataforma son la velocidad angular de las ruedas, es necesario un modelo matemático que permita encontrar los parámetros de velocidad que deben aplicarse al robot para obtener el comportamiento deseado.

Se toma como referencia la ecuación 15 para despejar las variables de interés.

Ecuación 15. Velocidad angular de las ruedas de tracción

$$\begin{bmatrix} w_r \\ w_i \end{bmatrix} = \begin{bmatrix} r\frac{\cos\theta}{2} & r\frac{\cos\theta}{2} \\ r\frac{\sin\theta}{2} & r\frac{\sin\theta}{2} \\ \frac{r}{l} & -\frac{r}{l} \end{bmatrix}^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\emptyset} \end{bmatrix}$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, “Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial,” *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

Se debe encontrar la inversa de la matriz que no es cuadrada, para ello se halla la pseudoinversa de la matriz en referencia definida como:

Ecuación 16. Inversa de la matriz de referencia A

$$A^{+} = (A^{T} * A)^{-1} * A^{T}$$

$$A = \begin{bmatrix} r\frac{\cos\theta}{2} & r\frac{\cos\theta}{2} \\ r\frac{\sin\theta}{2} & r\frac{\sin\theta}{2} \\ \frac{r}{l} & -\frac{r}{l} \end{bmatrix}^{-1}$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, “Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial,” *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

A partir de lo anterior, se determina la pseudoinversa de la matriz A.

Ecuación 17. Pseudoinversa de la matriz A

$$A^{+} = \begin{bmatrix} \frac{\cos\theta}{r} & \frac{\sin\theta}{r} & \frac{l}{r} \\ \frac{\cos\theta}{r} & \frac{\sin\theta}{r} & -\frac{l}{r} \end{bmatrix}$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, "Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial," *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

Dado esto, se determina la cinemática inversa como:

Ecuación 18. Cinemática inversa

$$\begin{bmatrix} w_r \\ w_i \end{bmatrix} = \begin{bmatrix} \frac{\cos\theta}{r} & \frac{\sin\theta}{r} & \frac{l}{r} \\ \frac{\cos\theta}{r} & \frac{\sin\theta}{r} & -\frac{l}{r} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\emptyset} \end{bmatrix}$$

Fuente: L. Enrique, M. Alejandro, and E. Leonardo, "Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial," *Rev. Ing. USBMed*, vol. 5, no. 1, pp. 26–34, 2014.

3.3 ARQUITECTURA DEL SISTEMA

A continuación, se describe la arquitectura general del sistema y como cada elemento que conforma la plataforma VItour interactúa con los demás componentes y así llegar a un producto final que cumple con los objetivos establecidos para el proyecto. VItour es una plataforma diseñada con componentes de bajo costo que permite realizar el proceso de SLAM gráfico dentro de entornos cerrados. Los mapas generados por la plataforma podrán ser usados para el proceso de navegación autónoma entre los diferentes espacios del mapa y así completar la labor de asistir como guía en ambientes cerrados o de tipo *indoor* a personas con alguna deficiencia visual.

3.3.1 ROS

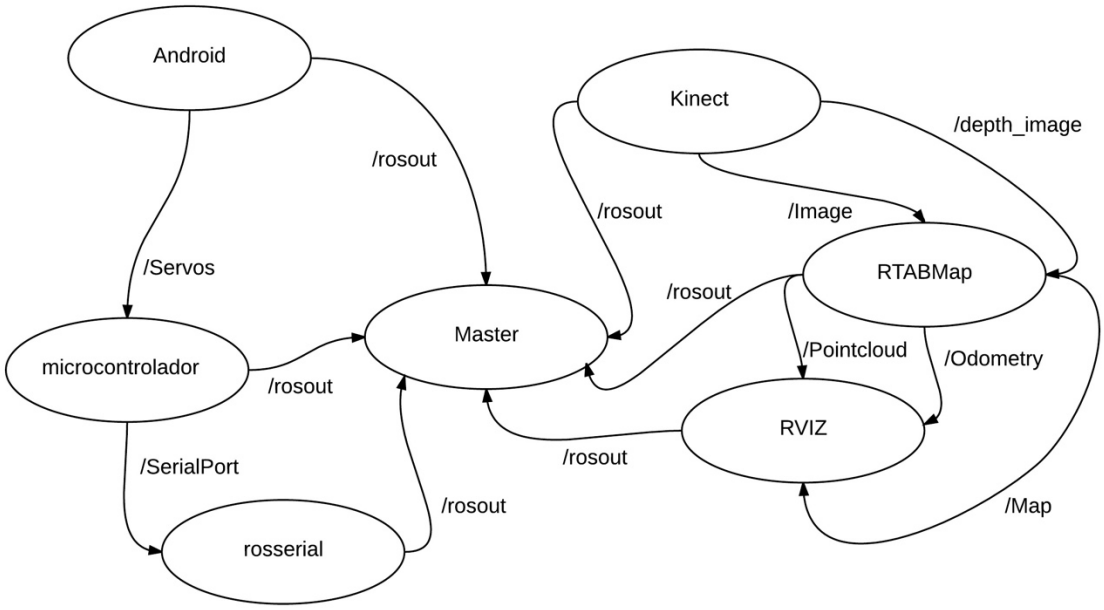
Como se ha mencionado a lo largo del documento, ROS es el framework sobre el cual todo el desarrollo de software y la integración con el respectivo hardware ha sido diseñado, éste framework cuenta con diferentes versiones las cuales son denominadas por los creadores como *distribuciones*, éstas se reconocen con un nombre clave, es importante resaltar que cada versión cuenta con sus características y el soporte es diferente para cada una, de igual manera se encuentran dirigidas hacia una variante específica del sistema operativo Ubuntu. Ver anexo D para conocer más detalles.

La distribución lanzada en el año 2014 (Indigo) es la que se implementa en el proyecto ya que es, hasta el momento, la única versión LTS *(del inglés Long Term Support),* teniendo soporte mucho mayor y completo por parte de los creadores, desarrolladores y la comunidad en general. El proceso de instalación se encuentra claramente descrito en el wiki respectivo[13] así mismo el proceso de configuración del entorno de trabajo para el desarrollo de paquetes propios.

El esquema de distribución general del framework se observa en la Figura 28. Al ser un sistema tan flexible, ROS permite una configuración muy personalizada, por lo tanto, solo es necesario comprender el concepto general de funcionamiento y diseñar el software de acuerdo a las necesidades de la plataforma. En el caso de VItour, dentro del esquema de ROS se destacan cinco partes fundamentales que soportan la columna vertebral de la plataforma; Kinect, SLAM, puerto serial, micro procesamiento y aplicación.

[13] http://wiki.ros.org/indigo

Figura 29. Esquema general de ROS para VItour



Fuente: Propia

Los nodos que conforman la parte del Kinect son los encargados de brindar los controladores que permitan al sistema reconocer la conexión del hardware y el manejo del sensor, los drivers son de libre distribución bajo licencia Apache20 entregados por el proyecto OpenKinect[14]. En el SLAM se encuentran todos los requisitos necesarios para permitir a la plataforma realizar el proceso de mapeo, anteriormente en la sección 2.2 se analizaron tres de los algoritmos más comunes de SLAM basado en la plataforma ROS. Tras el análisis correspondiente a cada uno de los algoritmos, se encontró que las técnicas de Gmapping y Hector SLAM inicialmente trabajan mucho mejor cuando se tienen datos correspondientes a la información captada por algún tipo de sensor láser, sin embargo, es posible implementar estos métodos aplicando una transformación de datos para emular la información de un láser. Por otra parte, la propuesta de los autores de RTAB-Map permite una implementación simple con tan sólo un sensor de tipo RGB-D como el Kinect, obteniendo buenos resultados sin necesidad de tener algún sensor o realizar transformaciones de algún tipo para suplir el requisito impuesto tanto en Hector SLAM como Gmapping. Ahora bien, la mayoría de robots cuentan con la posibilidad de tener información de odometría, bien sea por encoders conectados a las ruedas para calcular el desplazamiento del robot o por unidades de medición inercial (IMU), lo que incrementa el costo de la construcción de la plataforma robótica, de igual manera es posible transformar datos y pasarlos al sistema como información de

[14] https://openkinect.org/wiki/Main_Page

odometría, sin embargo la información tiene que ser proveniente de algún láser para el caso Gmapping, mientras que los métodos de Hector SLAM y RTAB-Map pueden trabajar sin odometría en el caso del primero mientras que el segundo implementa una aproximación visual para obtener información de odometría. A continuación se presenta una tabla comparativa entre los tres métodos mencionados.

Tabla 2. Cuadro comparativo entre los métodos Gmapping, Hector SLAM y RTAB-Map

	GMAPPING	HECTOR SLAM	RTAB-MAP
Compatibilidad con el sensor Kinect	SI	SI	SI
Rendimiento usando únicamente el sensor Kinect	40% ya que requiere odometría para su funcionamiento	70% Puesto que está diseñado para sensores LIDAR con mayor campo de visión	90% Sin embargo no se tiene el comportamiento esperado cuando hay espacios sin características sobresalientes
Requiere una fuente de odometría en la plataforma	SI	NO	NO
Requiere implementación de hardware LIDAR	Brinda mejores resultados	SI	Trabaj de manera correcta sin un sensor láser 2D

	GMAPPING	HECTOR SLAM	RTAB-MAP
Personalización de los métodos	Permite configuración básica siempre y cuando se garanticen parámetros como la odometría	Configurable	Muy configurable
Aspectos generales	Método de procesamiento rápido. Ideal para robots con hardware robusto Resultados óptimos cuando hay UMI o encoder más hardware LIDAR Comúnmente usado en la comunidad de ROS Precisa recreación de espacios provenientes de información de un archivo *bag	Diseñado para hardware de baja potencia. No requiere odometría Gran comportamiento con cualquier tipo de sensor LIDAR. Se puede implementar sin necesidad de láser con las transformaciones necesarias. Permite hasta seis grados de libertad. Ágil toma de información y procesamiento de la misma	Compatibilidad con diferente tipo de hardware Orientado a SLAM 3D Permite creación de mapas por multi-sesiones Implementa odometría visual No requiere hardware de tipo LIDAR Cuenta con una GUI propia para observar el comportamiento y los resultados. Ideal en plataformas que solo monten cámaras estéreo o sensores RGB-D (Kinect) Algoritmo con mejoras para el procesamiento en línea. SLAM en grandes entornos. Preciso en ambientes indoor.

	GMAPPING	HECTOR SLAM	RTAB-MAP
Aspectos generales			No presenta el problema de identificación de formas presente en los métodos de SLAM 2D con sensores LIDAR.

Fuente: Propia.

De acuerdo a lo descrito y a lo que se puede observar en la tabla comparativa entre los tres métodos, RTAB-Map tiene las características necesarias para la plataforma VItour cumpliendo los requisitos de diseño como bajo costo (al no tener que implementar UMI, encoders y/o hardware de tipo LIDAR), método de gran rendimiento en entornos cerrados y una estimación de posición precisa basado en la odometría visual, adicionalmente el rasgo más importante es que es un método diseñado totalmente para sistemas que usen sensores de tipo RGB-D como lo es el Kinect.

Por otra parte el puerto serial tiene las configuraciones necesarias que permiten la comunicación serial entre el computador donde se ejecuta el máster de ROS y la unidad de micro procesamiento. La unidad de micro procesamiento actúa como un nodo dentro del entorno de ROS, la cual permite el control de los motores del robot VItour al estar suscrita al tema donde son publicados los mensajes de control.

Finalmente la aplicación Android como se describió en la sección 3.1 interactúa como un nodo de tipo publicador, el cual interpreta los movimientos según la pulsación de cada uno de los botones y transmite los mensajes de control a la unidad de micro procesamiento.

3.3.2 KINECT

Éste dispositivo contiene una serie de sensores incluidos el sensor RGB y un sensor de profundidad 3D. Las imágenes captadas por el sensor de profundidad son de tamaño VGA (640x480 píxeles), mientras que la resolución del sensor RGB es 4xVGA (1280x960). Para mayor información de las especificaciones técnicas ver anexo E. Éste sensor es un gran recurso para implementaciones de SLAM a un costo moderado. En la tabla 3 se muestra un cuadro comparativo entre el sensor de Microsoft y los láseres LIDAR comúnmente usados en robótica, extraído del artículo de la revista *Sensors*(Kamarudin, Mamduh, Md Shakaff, & Zakaria, 2014).

Tabla 3. Comparación técnica entre el sensor Kinect y los láseres 2D

Componentes	***Kinect***	***Láseres 2D***
Rango de operación (m)	0.4-0.8 hasta 3.5-6.0	~0 hasta 4-250
Ángulo de apertura horizontal (°)	57	180-360
Ángulo de apertura vertical	43	-
Número de puntos de medición	640x480=307,200	Mayor a 6,000
Precio (CP)	473,808	3M-47M

Fuente: K. Kamarudin, S. M. Mamduh, A. Y. Md Shakaff, and A. Zakaria, "Performance analysis of the microsoft kinect sensor for 2D simultaneous localization and mapping (SLAM) techniques," *Sensors (Switzerland)*, vol. 14, no. 12, pp. 23365–23387, 2014

El sensor Kinect se ubica en la parte superior de la plataforma para brindar un mayor campo visual. Se sujeta a los soportes verticales de la plataforma de manera tal que la posición de la base quede alineada en paralelo al cuerpo del robot y al suelo.

3.3.3 COMPUTADOR (LAPTOP)

La unidad central de procesamiento del VItour es un computador portátil o *laptop*, en el cual se ejecuta el framework de ROS y donde se realiza todo el procesamiento de imágenes. El computador usado cuenta con un procesador Intel core i5 a 2.5GHz y con una memoria RAM de 6GB. Adicionalmente usa Ubuntu Mate 14.04 como sistema operativo lo que permite ejecutar el Robot Operating System sin ningún inconveniente. De igual manera las librerías de libfreenect se encuentran instaladas para tener los controladores necesarios para el reconocimiento e interacción con la Kinect.

3.3.4 DISPOSITIVO ANDROID

Cualquier terminal con Android xx o superior tiene la capacidad de correr la aplicación y puede interactuar con la plataforma VItour de manera remota, sin importar si es *smartphone* o *tablet*.

3.3.5 ESTACIÓN DE MOVIMIENTO

La estación de movimiento es donde se encuentran los mecanismos que permiten que el robot VItour pueda desplazarse, está diseñada bajo el modelo de tracción de tipo triciclo con acople de tres motores, permitiendo que el robot pueda tener desplazamientos transversales (adelante y atrás) y movimientos de rotación sobre su eje central en ambos sentidos. Para las labores de movimiento se implementan tres servo motores TowerPro MG995 que ofrecen un torque de 8.5Kgf-cm. Dos modificados para tener una rotación continua (360°), el anexo F ofrece información técnica más detallada. El control de los mismos se establece por una unidad de micro procesamiento que establece un puente de comunicación serial con el computador a través del protocolo de comunicación serial de ROS “rosserial”.

Una consideración muy importante en el diseño es el equilibrio que debe tener el robot, por esto se buscó el apoyo de una rueda adicional en la parte trasera que no cuenta con un motor propio, pero ayuda a que el robot mantenga su estabilidad. De esta forma logramos que el robot avance en línea recta cuando se fijan los motores de las ruedas de tracción a la misma velocidad, también se puede hacer girar en sobre su propio eje cuando las velocidades tienen igual magnitud con sentidos opuestos.

El sistema se encuentra alimentado por una batería de 12 voltios a 7.8 amperios-hora, por ende se implementa un circuito reductor de voltaje para la alimentación de los motores que maximiza la corriente de salida para garantizar la corriente necesaria exigida por cada uno. La figura 29 muestra el esquema eléctrico usado en la estación de movimiento. Las ecuaciones que describen el circuito usado para la regulac2ión de voltaje se muestran a continuación.

Ecuación 19. Cálculo de la resistencia de protección R1

$$R1 = \frac{V_{BEQ1}}{I_{REG} - I_{Q1}/B_{Q1}}$$

Fuente: Hoja de especificaciones del LM78XX Fairchild semiconductor, 2006.

Una vez estimada la resistencia de protección del transistor Q_1 se determina la relación para la corriente de salida.

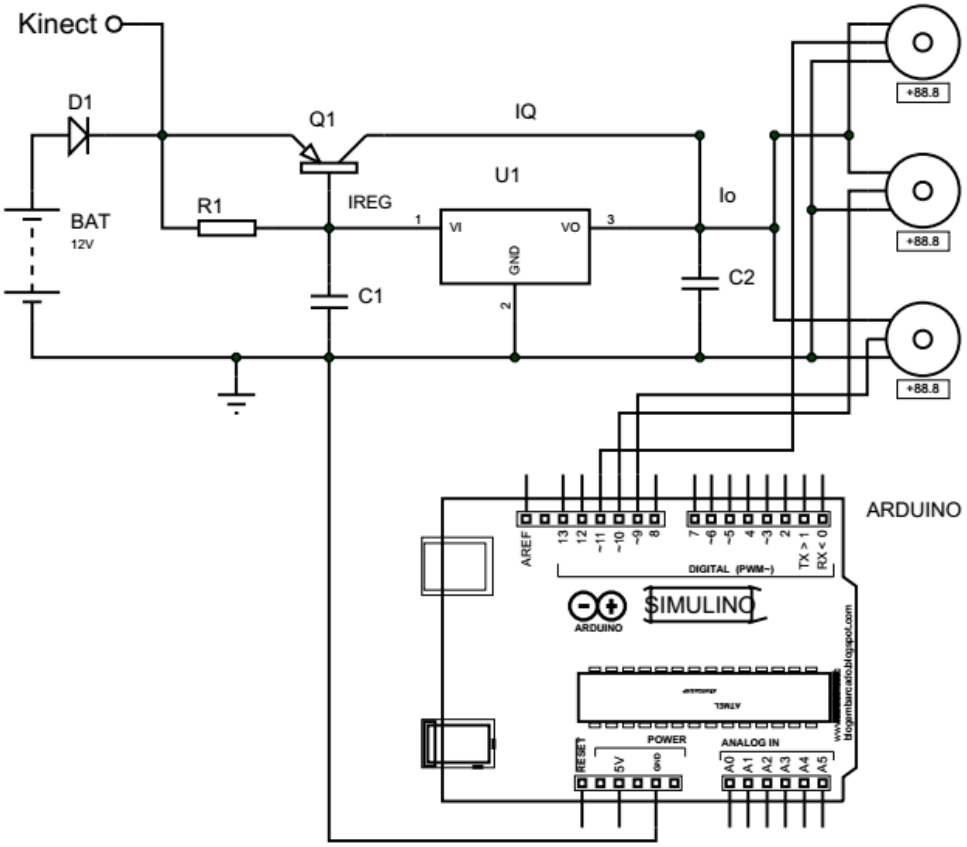
Ecuación 20. Estimación de la corriente de salida I_O

$$I_O = I_{REG} + B_{Q1}(I_{REG} - V_{BEQ1}/R1)$$

Fuente: Hoja de especificaciones del LM78XX Fairchild semiconductor, 2006.

El movimiento rotacional se logra girando las ruedas laterales en sentidos opuestos y ajustando el sentido del motor central o volante en la dirección que se desea que se produzca el movimiento. La sección 4 se pueden observar los resultados de precisión en los diferentes desplazamientos realizados por la plataforma VItour.

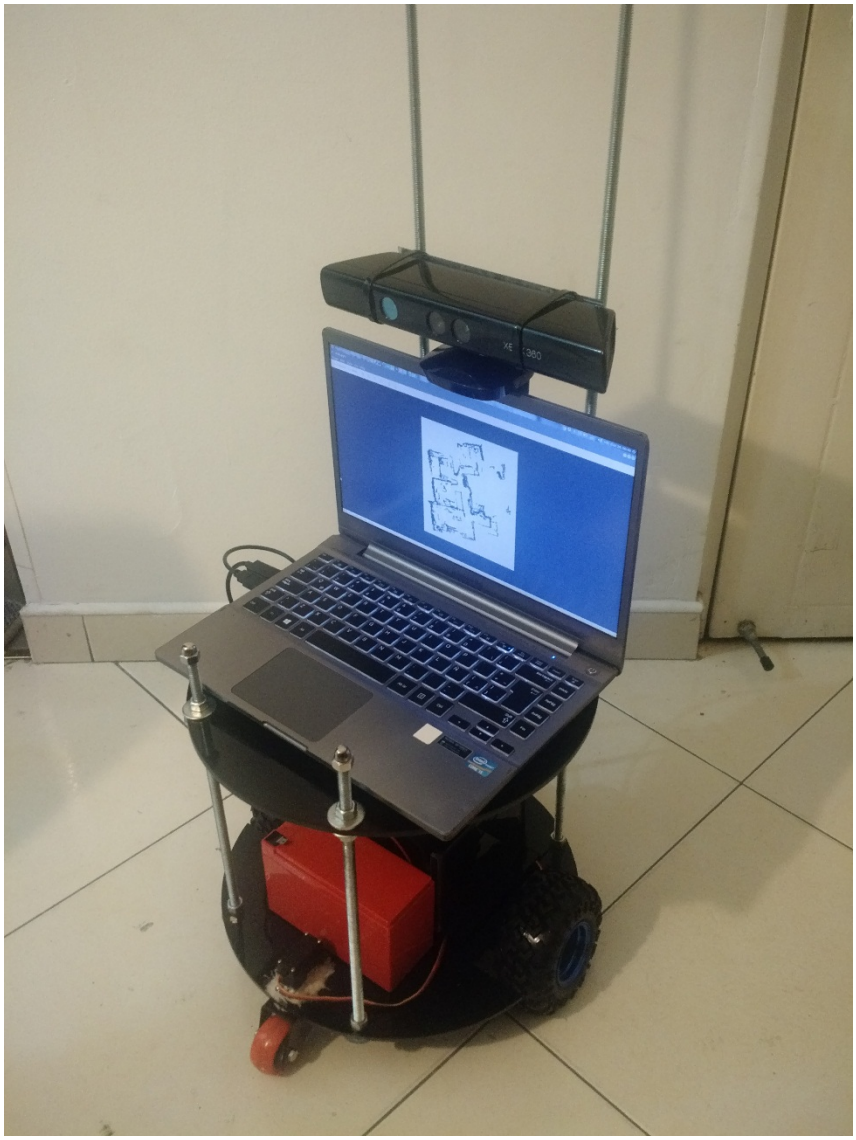
Figura 30. Diagrama eléctrico de la plataforma VItour



Fuente: Esquema diseñado con un simulador de circuitos electrónicos.

En la imagen de la figura 30 se observa la plataforma VItour, producto del presente proyecto construido con materiales de bajo costo y con implementación de software bajo licencia de distribución libre y de código abierto *(open source)*.

Figura 31. VItour



Fuente: Fotografía propia de VItour

4 RESULTADOS

4.1 CONTROL Y MOVIMIENTO

A continuación, presentamos las pruebas realizadas del desplazamiento del robot *VItour;* inicialmente se realizaron pruebas de traslación en línea recta a una distancia de recorrido máxima de 4m con intervalos de 0.8m; para de esta forma determinar el ángulo de desviación en trayectorias rectilíneas.

Con los resultados obtenidos, se halló el porcentaje de error para cada uno de los casos y con esto, podemos analizar el comportamiento y optimizar el control de *VItour*.

Tabla 4. Dirección hacia adelante (La desviación fue hacia el lado derecho)

Distancia (m)	**Ángulo De Desviación (°)**	**Porcentaje De Error (%)**
0.8	0.4	0.44
1.6	0.57	0.63
2.4	1.31	1.45
3.2	1.557	1.73
4	2.51	2.78

Tabla 5. Dirección hacia atrás (La desviación se produjo hacia el lado izquierdo)

Distancia (m)	**Ángulo De Desviación (°)**	**Porcentaje De Error (%)**
0.8	0.35	0.38
1.6	0.358	0.397
2.4	0.716	0.79
3.2	1.342	1.49
4	2.219	2.46

MOVIMIENTO ROTACIONAL

Posteriormente se realizar pruebas correspondientes a la rotación sobre el propio eje del robot, obteniendo el desplazamiento presentado por la plataforma al rotar 360º en sentido a las manecillas del reloj y en sentido antihorario.

En las siguientes tablas observaremos que al realizar el giro en diferentes sentidos (horario y antihorario) el desplazamiento sobre el eje *X* e *Y* varían dependiendo el sentido de giro.

Tabla 6. Giro - Sentido Antihorario

	Eje X (m)	Eje Y (m)
Posición 1	0	0
Posición 2	0.045	0.008

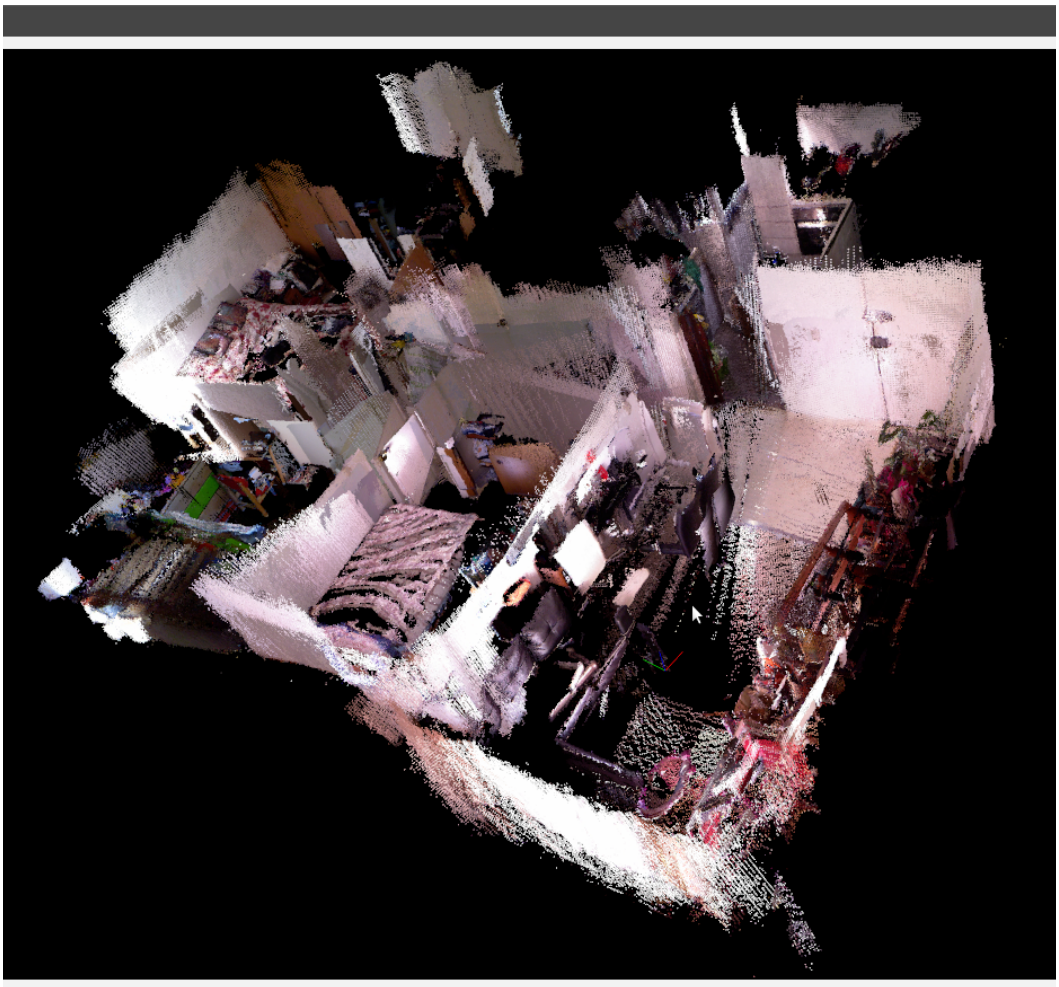
Tabla 7. Giro - Sentido Horario

	Eje X (m)	Eje Y (m)
Posición 1	0	0
Posición 2	0.040	0.006

4.2 MAPAS OBTENIDOS

Los mapas presentados a continuación se obtuvieron de dos entornos diferentes, uno con espacios más reducidos que el otro. De igual manera por cada entorno se pueden observar dos mapas, uno corresponde a la representación tridimensional dada por la construcción del mapa haciendo uso de matrices de nubes de puntos, mientras que el otro, es la extracción en dos dimensiones de aquel mapa. Cabe resaltar que el mapa tridimensional es producto de la técnica implementada ya que su proceso de SLAM está basado en la percepción de imágenes de tipo RGB-D, sin embargo el mapa que se utiliza generalmente para los procesos de navegación autónoma, son los mapas de 2D.

Figura 31. Mapa del entorno 1 en tres dimensiones

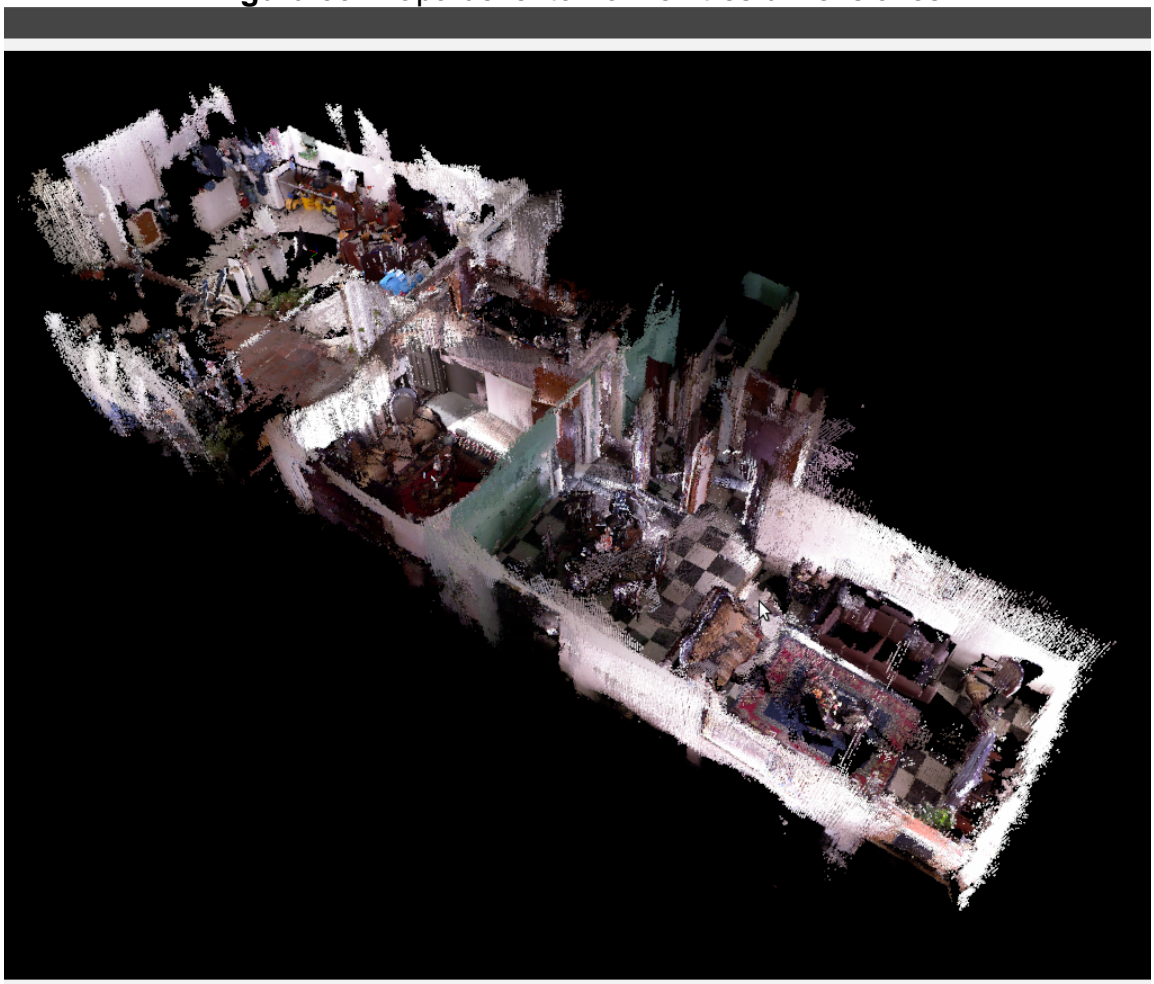


Fuente: Reconstrucción 3D de la nube de puntos obtenida del entorno 1 usando el software rtabmap.

Figura 32. Mapa en 2D del entorno 1

Fuente: Mapa generado del entorno 1 haciendo uso del paquete map_server de ROS.

Figura 33. Mapa del entorno 1 en tres dimensiones



Fuente: Fuente: Reconstrucción 3D de la nube de puntos obtenida del entorno 2 usando el software rtabmap.

Figura 34. Mapa en 2D del entorno 1

Fuente: Mapa del entorno 2 generado haciendo uso del paquete map_server de ROS.

5 TRABAJOS FUTUROS

Como se describió a lo largo del presente documento, aquí se presenta la primera etapa de desarrollo de la plataforma VItour, la cual únicamente describe la dinámica del sistema y el desarrollo del proceso de SLAM para ambientes cerrados. Como trabajos futuros existen otras fases que incluyen distintos procedimientos descritos a continuación para entregar un producto final que brinde el soporte adecuado a la navegación de personas con discapacidad o pérdida total de la vista.

5.1 ETAPA DE NAVEGACIÓN AUTÓNOMA

Actualmente la plataforma robótica VItour tiene las herramientas de hardware y software necesarias para realizar el proceso de SLAM. VItour al ser un asistente robótico que mejora el desplazamiento de las personas con pérdida parcial o total de la vista en entornos cerrados y no conocidos, debe tener la capacidad de guiar a dichas personas durante todo el recorrido desde un punto de partida A hacia un punto de destino B; para ello, es necesario proveer a la plataforma el algoritmo que otorgue la habilidad de realizar navegación autónoma dado un entorno conocido por la misma plataforma (dicho entorno ya lo brinda el proceso de SLAM), donde se

deben considerar puntos clave como lo son la planificación óptima de rutas, el seguimiento de una trayectoria definida y la evasión de obstáculos.

Del mismo modo se realizó el procedimiento para implementar el desarrollo de SLAM, es necesario testear diversos algoritmos para los procesos mencionados anteriormente, teniendo en cuenta que el eje central de la operación sigue siendo el framework de ROS. Una vez se defina el algoritmo más adecuado para la aplicación sobre VItour, serán evaluados los resultados de su implementación en diferentes entornos reales de tipo estático y dinámico y de esta manera realizar una comparación del comportamiento de la plataforma tomando en cuenta éste parámetro.

Con lo anterior, se completaría las tareas básicas que la plataforma debe tener para ser un asistente robótico de navegación en entornos cerrados. Ya que inicialmente VItour es capaz de construir el mapa de cualquier espacio, obteniendo resultados fiables según lo mostrado en la sección 4.2. Posteriormente el mapa se carga al sistema y una vez sean entregados los puntos de partida y destino, el robot puede no solo planificar la trayectoria sino que será capaz de seguirla de manera autónoma evadiendo los obstáculos que sean encontrados durante el recorrido.

5.2 ETAPA DE LOCALIZACIÓN DE USUARIO

Con el fin de brindar una mejor experiencia de usuario, se propone otorgar la habilidad de localización del beneficiario para ir en búsqueda del mismo. En otras palabras VItour será capaz de estimar la ubicación, dentro del mapa de trabajo, del dispositivo Android desde el cual recibe la alerta. Una vez localizado el usuario VItour planifica de manera correcta la ruta adecuada para llegar al lugar donde se encuentra la persona y podrá identificarla a través de algún gesto captado por procesamiento de imágenes computacionales apoyado en las librería de OpenCV.

VItour se ubicará frente al usuario simulando ser un perro guía, y el sujeto a través de la aplicación Android indicará cual es el lugar de destino y la plataforma lo guiará a través del entorno hasta llegar al lugar indicado.

Para la localización se evaluarán diferentes sistemas de localización interna o IPS, algoritmos de triangulación de redes o integraciones del método de Monte Carlo y filtros de partículas, con el fin de determinar cuál es el método que mejor se adapta a cualquier tipo de entorno sin necesidad de realizar grandes alteraciones al mismo y que ofrezca resultados fiables para el procedimiento descrito.

5.3 IMPLEMENTACIÓN DE UN SISTEMA EMBEBIDO

Los sistemas embebidos han permito el desarrollo de diversas tecnologías con gran procesamiento computacional sobre hardware de dimensiones contenidas, la implementación de tarjetas FPGA u otras plataformas de desarrollo son ideales para economizar costos y desarrollar tareas específicas donde la implementación de un ordenador de escritorio o portátil es de cierta manera exagerado.

En el caso de la plataforma VItour, mejoraría considerablemente la respuesta en velocidad de los movimientos y el esfuerzo de los motores, puesto que el portátil junto a la batería son los elementos que mayor peso agregan a la plataforma.
Hay que tener en cuenta que el procesamiento de imágenes consume muchos recursos de sistema, por ende es necesario escoger adecuadamente la tarjeta de desarrollo que reemplazará al portátil para que el rendimiento de la plataforma en general no se vea diezmado. Existen innumerables tarjetas o sistemas embebidos en la actualidad y su selección depende directamente de la compatibilidad que dichas tarjetas tengan con el framework ROS y el otro criterio de selección será directamente sus especificaciones técnicas; tipo y velocidad del procesador, memoria RAM y procesamiento gráfico.

6 CONCLUSIONES

Se entrega la primera etapa del asistente robótico planteado en los objetivos del presente proyecto. VItour es la plataforma de bajo costo y *open source* que asiste a las personas con discapacidad visual en la tarea de desplazamiento por ambientes cerrados desconocidos construyendo mapas precisos del lugar. Para ello fueron evaluados diferentes algoritmos de segmentación de imágenes y métodos de mapeo y localización simultánea (SLAM) para implementar el más adecuado según las especificaciones técnicas de la plataforma.

Dentro de los algoritmos evaluados encontramos que RTAB-Map es ideal para el hardware implementado en VItour, puesto que para la extracción de imágenes tan sólo se cuenta con el sensor Kinect. Por otra parte, en el caso de Gmapping es indispensable tener información de odometría, bien sea a través de encoders montados en los ejes de las ruedas o por medio de una Unidad de Medición Inercial (IMU), haciendo que éste método no sea el ideal para la implementación del presente proyecto. A pesar que la técnica de Hector SLAM si trabaja de manera adecuada sin requerir datos de odometría, no se puede considerar su implementación puesto que VItour no cuenta con ningún dispositivo laser de tipo LIDAR, por lo que la respuesta de Hector SLAM no será precisa y por con siguiente su mapa tendrá un alto porcentaje de error.

El centro de operaciones de VItour se basa en el framework ROS de tipo *open source* y cuyo núcleo se distribuye bajo licencia de tres cláusulas BSD[15], fomentando de esta manera a quienes lean el presente documento el uso de software alternativo de licencia libre y de código abierto ya que como se evidenció tiene grandes ventajas. Especialmente a la comunidad de la Fundación Universitaria los Libertadores se impulsa la implementación del framework de ROS en sus cursos de robótica. Así mismo la aplicación desarrollada en Android para el control de la plataforma se podrá distribuir libremente y su código fuente es abierto para otros desarrolladores que se animen a aplicar la integración de estas dos plataformas en sus proyectos de investigaciónes.

La aplicación cumple con su rol adecuadamente, establece la comunicación remota entre la plataforma móvil y el usuario, permitiendo dar las órdenes del control de la misma desde el terminal Android.

La estación de movimiento diseñada tiene en cuenta el modelo de la estructura física de la plataforma, lo que permite establecer el tipo de motores necesarios para mover el peso de la plataforma y el diseño del control dinámico de la misma. Se tiene una plataforma móvil tipo triciclo con motores independientes para cada rueda. De esta manera se logran movimientos sobre el eje con mayor precisión.

La primera etapa de diseño del robot asistencial VItour tiene un muy buen rendimiento, una arquitectura de diseño simple, de integración multiplataforma y con unos resultados en la construcción de mapas de gran acierto a bajo costo, en comparación con otros robots comerciales que realizan la misma tarea. VItour fue puesto a prueba en diferentes tipos de ambientes cerrados, con espacios más abiertos, corredores más amplios o espacios reducidos y a pesar que el comportamiento no es el mismo, los resultados no dejan de tener un margen alto de calidad.

[15] https://opensource.org/licenses/BSD-3-Clause

BIBLIOGRAFIA

Aguerrevere, D., Maroof, C., & Barreto, A. (2004). Portable 3D sound / sonar navigation system for blind individuals. *Laccet'2004*, (June).

Barrientos Sotelo, V. R., Silva Ortigoza, R., Hernandez Guzman, V. M., Silva Ortigoza, G., Garcia Sanchez, J. R., & Molina Vilchis, M. A. (2007). Disponible en: http://www.redalyc.org/articulo.oa?id=78460301. *Télématique*, *6(3)*, 4–5.

Chaccour, K., & Badr, G. (n.d.). Novel indoor navigation system for Visually Impaired and blind people.

Dakopoulos, D., & Bourbakis, N. G. (2010). Wearable Obstacle Avoidance Electronic Travel Aids for Blind: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *40*(1), 25–35. http://doi.org/10.1109/TSMCC.2009.2021255

Doucet, A., Freitas, N. De, Murphy, K., & Russell, S. (2000). Rao-Blackwellised particle filtering for dynamic Bayesian networks. *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 176–183. http://doi.org/10.1049/iet-spr:20070075

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping (SLAM): part I The Essential Algorithms. *Robotics & Automation Magazine*, *2*, 99–110. http://doi.org/10.1109/MRA.2006.1638022

Durrant-Whyte, H. F. (1988). Uncertain geometry in robotics. *IEEE Journal on Robotics and Automation*, *4*(1), 23–31. http://doi.org/10.1109/56.768

Enrique, L., Alejandro, M., & Leonardo, E. (2014). Seguimiento de Trayectorias con un Robot Móvil de Configuración Diferencial. *Revista Ingenierías USBMed*, *5*(1), 26–34.

Fusiello, A., Panuccio, A., Fontana, F., Murino, V., & Rocchesso. (2002). A multimodal electronic travel aid device. *Proceedings of ICMI 2002*, 39–44. http://doi.org/10.1109/ICMI.2002.1166966

Grisetti, G., Stachniss, C., & Burgard, W. (2005). Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. *Proceedings - IEEE International Conference on Robotics and Automation*, *2005*, 2432–2437. http://doi.org/10.1109/ROBOT.2005.1570477

Ito, K., Okamoto, M., Akita, J., Ono, T., Gyobu, I., Takagi, T., … Mishima, Y. (2005). CyARM. *CHI '05 Extended Abstracts on Human Factors in Computing Systems - CHI '05*, 1483. http://doi.org/10.1145/1056808.1056947

Johnson, L. a., & Higgins, C. M. (2006). A navigation aid for the blind using tactile-visual sensory substitution. *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, 6289–6292. http://doi.org/10.1109/IEMBS.2006.259473

Kamarudin, K., Mamduh, S. M., Md Shakaff, A. Y., & Zakaria, A. (2014). Performance analysis of the microsoft kinect sensor for 2D simultaneous localization and mapping (SLAM) techniques. *Sensors (Switzerland)*, *14*(12), 23365–23387. http://doi.org/10.3390/s141223365

Kohlbrecher, S., Von Stryk, O., Meyer, J., & Klingauf, U. (2011). A flexible and

scalable SLAM system with full 3D motion estimation. *9th IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2011*, 155–160. http://doi.org/10.1109/SSRR.2011.6106777

Kulyukin, V., Gharpure, C., Nicholson, J., & Pavithran, S. (n.d.). RFID in Robot-Assisted Indoor Navigation for the Visually Impaired.

Labb, M., & François, M. (n.d.). Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation.

Labb??, M., & Michaud, F. (2014). Online global loop closure detection for large-scale multi-session graph-based SLAM. *IEEE International Conference on Intelligent Robots and Systems*, 2661–2666. http://doi.org/10.1109/IROS.2014.6942926

Labbe, M., & Michaud, F. (2011). Memory management for real-time appearance based loop closure detection. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1271–1276. http://doi.org/http://dx.doi.org/10.1109/IROS.2011.6094602

Labbe, M., & Michaud, F. (2013). Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, *29*(3), 734–745. http://doi.org/10.1109/TRO.2013.2242375

Muja, M., & Lowe, D. G. (2009). Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *International Conference on Computer Vision Theory and Applications (VISAPP '09)*, 1–10. http://doi.org/10.1.1.160.1721

Murphy, K. (2000). Bayesian map learning in dynamic environments. *Advances in Neural Information Processing Systems*, *12*, 1015–1021. http://doi.org/10.1.1.21.3240

Pons, J. V. (2012). *Localización y generación de mapas del entorno (SLAM) de un robot por medio de una Kinect .*

Quigley, M., Conley, K., Gerkey, B., FAust, J., Foote, T., Leibs, J., … Mg, A. (2009). ROS: an open-source Robot Operating System. *Icra*, *3*(Figure 1), 5. http://doi.org/http://www.willowgarage.com/papers/ros-open-source-robot-operating-system

Riisgaard, S., & Blas, M. R. (2004). SLAM for Dummies. *A Tutorial Approach to Simultaneous Localization and Mapping*, *22*(June), 1–127. http://doi.org/10.1017/S0025315400002526

Sainarayanan, G., Nagarajan, R., & Yaacob, S. (2007). Fuzzy image processing scheme for autonomous navigation of human blind. *Applied Soft Computing*, *7*(1), 257–264. http://doi.org/10.1016/j.asoc.2005.06.005

Sheikh Sadi, M., Mahmud, S., Mostafa Kamal, M., & Ibne Bayazid, A. (2014). Automated Walk-in Assistant for the Blinds.

Shiizu, Y., Hirahara, Y., Yanashima, K., & Magatani, K. (2007). The development of a white cane which navigates the visually impaired. *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, 5005–5008. http://doi.org/10.1109/IEMBS.2007.4353464

Sivic, J., & Zisserman, A. (2003). Video Google: a text retrieval approach to object

matching in videos. *Proceedings Ninth IEEE International Conference on Computer Vision*, (Iccv), 2–9. http://doi.org/10.1109/ICCV.2003.1238663

Smith, R. C., & Cheeseman, P. (1986). On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, *5*(4), 56–68. http://doi.org/10.1177/027836498600500404

SOLAR, J. R. DEL, & SALAZAR, R. (n.d.). Robots Móviles. *Introducción a Las Robótica*, 1–30.

Van Der Merwe, R., Doucet, a, De Freitas, N., & Wan, E. (2001). The Unscented Particle Filter. *Advances in Neural Information Processing Systems*, *96*(6080), 584–590. http://doi.org/10.1152/jn.00126.2006

WHO (World Health Organization); DE BODE, Chris. 10 Facts on visual impaired and blindness [en línea]. <http://www.who.int/features/factfiles/blindness/blindness_facts/en/>. [citado Junio 26, 2015].

K-SONAR; Bay Advanced Technologies Ltd; [en línea] <http://www.ksonar.com/spanish/how-ksonar-works.php>. [citado, Enero 15, 2016]

Miniguide; GPD Research; [en línea] <http://www.gdp-research.com.au/minig_1.htm>. [citado, Enero 15, 2016]

Pioneer 3-DX

Pioneer 3-DX is a small lightweight two-wheel two-motor differential drive robot ideal for indoor laboratory or classroom use. The robot comes complete with front SONAR, one battery, wheel encoders, a microcontroller with ARCOS firmware, and the Pioneer SDK advanced mobile robotics software development package.

Pioneer research robots are the world's most popular intelligent mobile robots for education and research. Their versatility, reliability and durability have made them the preferred platform for advanced intelligent robotics. Pioneers are pre-assembled, customizable, upgradeable, and rugged enough to last through years of laboratory and classroom use.

Product Features and Benefits

- **Easy to Use** - Comes assembled and integrated with its accessory packages.
- **Reliable** - Construction is durable and rugged. Easily handles the small gaps, minor bumping, jarring, or other obstacles that hinder other robotic platforms. Some Pioneer robots have been in service for over 15 years.
- **Pioneer Software Development Kit** - All Adept MobileRobots platforms include Pioneer SDK, a complete set of robotics applications and libraries that accelerate the development of robotics projects. Pioneer SDK is backed by our product support team.
- **Customizable** - Easily accessorize by choosing from dozens of supported and tested accessories that integrate with the robotic platform. Additional help is available for future upgrades or added accessories.
- **Reference Platform** - Pioneer robots are a standard in intelligent mobile platforms. Search your preferred robotics journal or conference listings to find many examples of Pioneer platforms in research applications.
- **Technical Support** - Pioneer software and hardware comes fully documented with additional help available through our product support team.

Specifications

Construction
- Body: 1.6 mm aluminum (powder-coated)
- Tires: Foam-filled rubber

Operation
- Robot Weight: 9 kg
- Operating Payload: 17 kg

Differential Drive Movement
- Turn Radius: 0 cm
- Swing Radius: 26.7 cm
- Max. Forward/Backward Speed: 1.2 m/s
- Rotation Speed: 300°/s
- Max. Traversable Step: 2.5 cm
- Max. Traversable Gap: 5 cm
- Max. Traversable Grade: 25%
- Traversable Terrain: Indoor, wheelchair accessible

Power
- Run Time: 8-10 hours w/3 batteries (with no accessories)
- Charge Time: 12 hours (standard) or 2.4 hrs (optional high-capacity charger)
- Available Power Supplies:
  - 5 V @ 1.5 A switched
  - 12 V @ 2.5 A switched

Batteries
- Supports up to 3 at a time
- Voltage: 12 V
- Capacity: 7.2 Ah (each)
- Chemistry: lead acid
- Hot-swappable Batteries: Yes

Available Recharge Options:
- Direct plug-in
- Docking station
- Powercube (3-battery charging bay)

* Batteries are accessible through hinged latched access panel for hot-swapping (continuous operation)

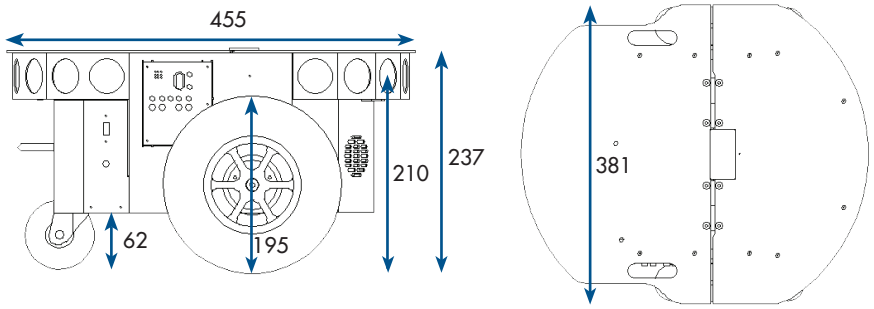
Microcontroller I/O
- System Serial
- 32 digital inputs
- 8 digital outputs
- 7 analog inputs
- 3 serial expansion ports

*Some ports may not be available if certain accessories are included with the robot

User Control Panel
- MIDI programmable piezo buzzer
- Main power indicator
- Battery charge indicator
- 2 AUX power switches
- System reset
- Motor enable pushbutton

Pioneer 3-DX

Dimensions (mm)



Core Software - included with all research platforms

ARIA provides a framework for controlling and receiving data from all MobileRobots platforms, as well as most accessories. Includes open source infrastructures and utilities useful for writing robot control software, support for network sockets, and an extensible framework for client-server network programming.

MobileSim open-source simulator which includes all MobileRobots platforms and many accessories.

MobileEyes graphical user interface client for remote operation and monitoring of the robot.

Mapper 3-Basic tool for creating and editing map files for use with ARIA, MobileSim, and navigation software.

SONARNL provides sonar-based approximate localization and navigation.

Accessory Support Software - bundled with purchase of robotic accessory

ARNL enables robust, laser-based autonomous localization and navigation.

Robotic Arm Support Pioneer arms are packaged with integrated software support.

Speech Recognition and Synthesis Library: Easy-to-use C++ development library for speech recognition based on the open source Sphinx2 system. Speech synthesis (text-to-speech) based on Cepstral synthesizer.

ACTS Color Tracking System: Software application which reads images from a camera and tracks the positions and sizes of multiple color regions. Information can be incorporated into your own software via ARIA.

Optional Industrial Grade Internally Mounted Computers

Mamba EBX-37 (Dual Core 2.26 GHz - 2-8 GB RAM)
6 X USB2.0 Ports
2 X PC/104+ Slots
4 X RS-232 Serial Ports
2 X 10/100/1000 Ethernet Ports
Onboard Audio & Video
Solid State Drive
Optional Wireless Ethernet

Optional Accessories:

- Laser-range finders
- Mono- and stereo-vision cameras
- Rear SONAR
- Wireless serial to Ethernet for remote operation
- Robotic arms and grippers
- Gyroscope
- Segmented bumper arrays
- Speakers and microphones
- Joystick
- Many more...

Include our integrated & supported accessories with your Pioneer 3-DX.

Here are some popular configurations to choose from:

Mapping & Vision

Gripping & Manipulation

Audio & Speech

More Information:

See our website www.mobilerobots.com for a full range of supported accessories or contact our sales department to discuss your application.

Adept Technology, Inc. 10 Columbia Drive, Amherst, NH 03031
Tel: 603-881-7960 Email: sales@mobilerobots.com
www.mobilerobots.com

Specifications subject to change without notice.

©2011 Adept Technology, Inc. ALL RIGHTS RESERVED. The information provided in this communication or document is the property of Adept Technology, Inc. and is protected by copyright and other intellectual property laws. In addition, any references to any Adept Technology, Inc. products are the marks and property of Adept Technology, Inc. [and may be registered trademarks]. All other trademarks or tradenames are the property of their respective holders.

09366-P3DX Rev. A

Search ...

TEAM HECTOR

TECHNISCHE UNIVERSITÄT DARMSTADT

Hector UGV

Specifications	
Weight	10kg
Height	0.5 m
Sensors	Wheel encoders, laser scanner, inertial measurement unit, infrared camera, webcam

Hector UGV has been developed since 2010. It was the basis for the development of Hector SLAM and the achievements in the RoboCup Rescue Robot League from 2012 - 2014.

It is based on a R/C model car Kyosho Twin Force as mobility platform (refered to as ”Hector GV”). It is modified for better autonomous vehicle handling and enhanced with an onboard computer and a laser range finder. For victim identification a vision box was developed, including a visual and a thermal camera.

Robot Locomotion:

The Hector GV is based on a Kyosho Twin Force RC model with a powerful and fast drive train. For indoor navigation we modified the drive train, the steering and the suspension because of the much higher weight.

4-wheel-drive: The 4-wheel-drive of this vehicle has one differential gear per axis and no middle differential gear. This ensures that if only three wheels have grip the vehicle is moving. To reduce the maximum speed and increase the strength we added a 1:5 gear.

4-wheel-steering: The front and rear wheels can be controlled independently to have three advantages over a normal 2-wheel-steering:

- a smaller minimum turn radius (half of 2-wheel-steering),
- the possibility that the rear wheels use the same trajectory as the front wheels (if both steering angles are the same)
- the possibility to move sidewards (up to 35 degrees to the longitudinal axis of the vehicle).

Normally the rear wheels are set to the same steering angle as the front wheels, so that the resulting trajectories are identical and the risk of obstacle contact is reduced. With this vehicle we have a very flexible, mobile and powerful platform which additionally has the advantage of precise odometry.

Sensors for Navigation and Localization:

Wheel Encoders: To measure the translational and rotational speed of the vehicle, all four wheels are equipped with incremental optical encoders.

Laser Scanner: The Hokuyo URG04-LX laser scanner covers an arc of 240° with 0.36◦ resolution per scan. It has a maximum range of 4m and a maximum sample rate of 10Hz.

Inertial Measurement Unit: To get the attitude, the vehicle contains a 6DoFinertial sensor ADIS16350 by Analog Devices which measures

accelerations and angular rates.

Navigation filter: All sensor information is fused to an overall estimation of position, velocity and attitude of the vehicle by using an extended Kalman filter. Our approach is to combine this two sources of information in a loosely-coupled way in order to gain a robust navigation solution.

Sensors for Victim Identification:

Victim detection will be approached from several complementary directons. With team members working in computer vision we plan to leverage their extensive experience and prior work. Significant progress in visual object recognition and scene understanding now allows us to apply these methods to real-life conditions. The victim detection will be supported by the integration of other sensor types, like a thermal camera, CO2 -sensor or microphone.

Vision-Based Recognition of Victims and Hazmat Symbols: The recognition of the objects is performed by using a combination of visual cues based on the gradients of image intensity. Such cues can be efficiently captured by a descriptor based on the histograms of oriented gradients (see figure for illustration). First, the gradient magnitude and orientation are computed densely in the image. The local distributions of the gradient orientation are then captured by the histogram. Such histograms are then grouped with their neighbors and jointly normalized. The normalization and local pooling of gradient information significantly improves the stability of the description to viewpoint changes, noise and changes in illumination.

The onboard computer with an nVidia graphic card allows real-time feature computation and recognition. We plan to use the recognition system for detection of hazmat symbols at the victim sites. The same system, but trained on the images of human body parts, will be used to recognize victims parts.

Multi-Cue Victim Detection: In addition to visual victim detection we will use a thermal camera as our secondary sensor. Thermal images often contain not only victims but also other warm objects, such as radiators or fire, so that thermal and visual recognition systems will deliver complimentary information. For the final victim verification we will also use acoustic and CO2 sensors.

©2016 Team Hector | Legal Notice | Website by Christian Rose, Martin Oehler | Template by Hurricane Media

Date: 2012.12.3

Scanning Laser Range Finder
UTM-30LX-EW
Specification

Symbol	Amendment Details	Amendment	Date	Amended by	Number
④×1	Correction of External Dimension representation	3	2012.12.3	KAMON	RS-0159
③×1	The description of the Multiecho function was added.	6,7	2012.11.5	KAMON	RS-0147
②×1	Electric cable connection is modified	4	2012.3.28	TAMAKI	RS-0052
①×2	Error code table is added.	5,6	2011.11.25	TAMAKI	RS-0006

Approved by	Checked by	Drawn by	Designed by	Title	UTM-30LX-EW Specification	
KAMITANI	UTSUGI	KAMON	KAMON	Drawing No.	C-42-3785	1/7

1. Introduction

1.1 Operation principles

The UTM-30LX-EW uses a laser source (λ=905nm) to scan a 270º semicircular field (Figure 1). It measures the distance for each angular step to objects in its range. The measurement data along with its angular step are transmitted via a communication channel. The laser safety is class 1.

2. Diagram of Scanned Area

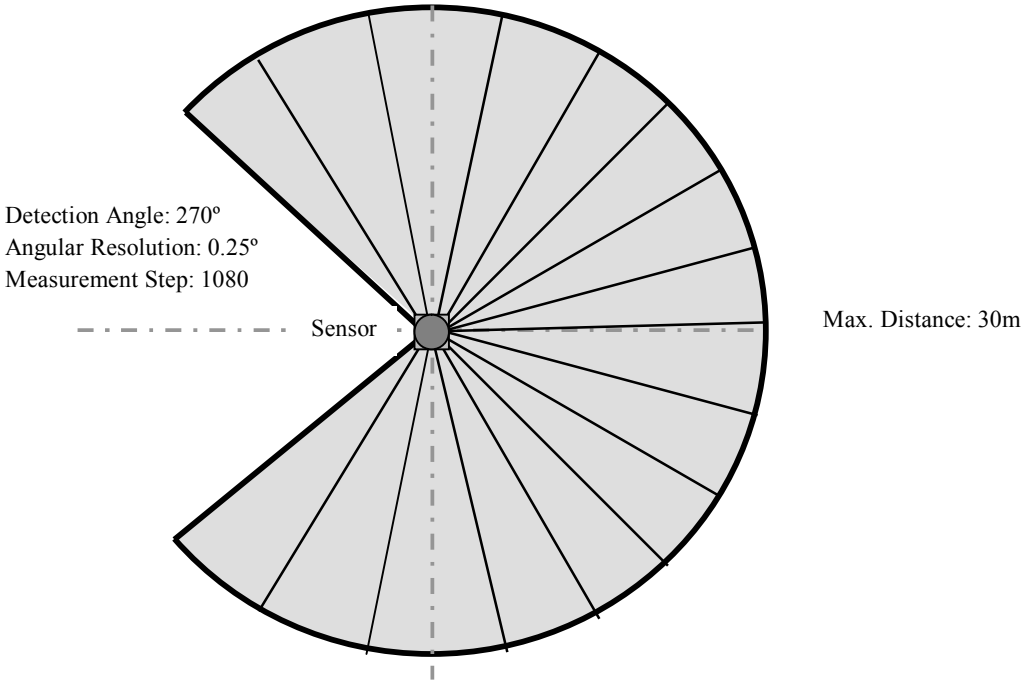


Figure 1

3. Important Notes

- This sensor is not a safety device/tool.
- This sensor is not for use in human detection.
- Hokuyo products are not developed and manufactured for use in weapons, equipment, or related technologies intended for destroying human lives or creating mass destruction. If such possibilities or usages are revealed, the sales of Hokuyo products to those customers might be halted by the laws of Japan such as Foreign Exchange Law, Foreign Trade Law or Export Trade Control Order. In addition, we will export Hokuyo products for the purpose of maintaining the global peace and security in accordance with the above laws of Japan
- Read specifications carefully before use.

4. Specifications

Product Name	Scanning Laser Range Finder
Model	**UTM-30LX-EW**
Light Source	Laser Semiconductor λ = 905nm Laser Class 1
Supply Voltage	12VDC ± 10%
Supply Current	Max: 1A, Normal : 0.7A
Power Consumption	Less than 8W
Detection Range and Detection Object	Guaranteed Range: 0.1 ~ 30m (White Kent Sheet) *2 Maximum Range : 0.1 ~ 60m Minimum detectable width at 10m : 130mm (Vary with distance)
Accuracy	0.1 – 10m : ±30mm, 10 – 30m : ±50mm (White Kent Sheet) *2 Under 3000lx : White Kent Sheet: ±30mm*1 (0.1m to 10m) Under 100000lx : White Kent Sheet: ±50mm*1 (0.1m to 10m)
Measurement Resolution and Repeated Accuracy	1mm 0.1 – 10m : σ <10mm, 10 – 30m : σ <30mm (White Kent Sheet) *2 Under 3000lx : σ = 10mm*1 (White Kent Sheet up to 10m) Under 100000lx : σ = 30mm*1 (White Kent Sheet up to 10m)
Scan Angle	270°
Angular Resolution	0.25° (360°/1440)
Scan Speed	25ms (Motor speed : 2400rpm)
Interface	Ethernet 100BASE-TX(Auto-negotiation)
Output	Synchronous Output 1- Point
LED Display	Green: Power supply. Red: Normal Operation (Continuous), Malfunction (Blink)
Ambient Condition (Temperature, Humidity)	-10°C ~ +50°C Less than 85%RH (Without Dew, Frost)
Storage Temperature	-25~75°C
Environmental Effect	Measured distance will be shorter than the actual distance under rain, snow and direct sunlight*2.
Vibration Resistance	10 ~ 55Hz Double amplitude 1.5mm in each X, Y, Z axis for 2hrs. 55 ~ 200Hz 98m/s^2 sweep of 2min in each X, Y, Z axis for 1hrs.
Impact Resistance	196m/s^2 In each X, Y, Z axis 10 times.
Protective Structure	Optics: IP67 (Except Ethernet connector)
Insulation Resistance	10MΩ DC500V Megger
Weight	210g (Without cable)
Case	Polycarbonate
External Dimension (W×D×H)	62mm×62mm×87.5mm ④ MC-40-3240

*1 Under Standard Test Condition (Accuracy can not be guaranteed under direct sunlight.)

*2 Indoor environment with less than 1000Lx.
Please perform necessary tests with the actual device in the working environment.
Use data filtering techniques to reduce the effect of water droplets when detecting objects under the rain.

5. Quality Reference Value

Vibration resistance during operation	10~150Hz 19.6m/s^2 Sweep of 2min in each X,Y,Z axis for 30min
Impact resistance during operation	49m/s^2 X, Y,Z axis 10 times
Angular Speed	2π/s (1Hz)
Angular Acceleration	π/2rad/ s^2
Life-span	5 Years (Varies with operating conditions)
Noise Level	Less than 25dB at 300 mm
Certification	FDA Approval (21 CFR part 1040.10 and 1040.11)

6. Interface

6.1 Robot Cable 4 Pin

Color	Function
Brown	+12 V
Blue	0 V
Green	Synchronous Output

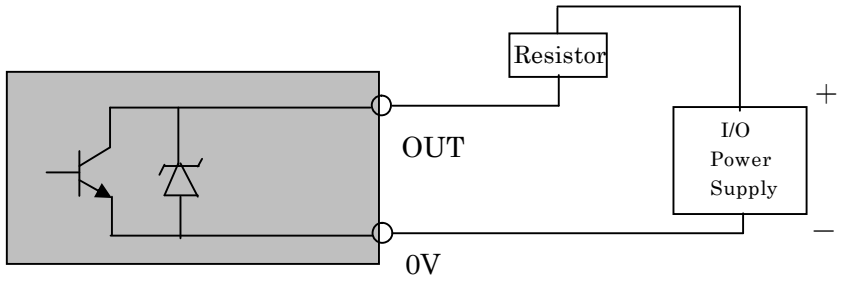
Note: 0 V of the power supply (Blue) and COM Output (0V) (White) are internally connected.

6.2 Ethernet Cable

RJ-45 plug is attached to the cable. (Length: 300mm)

This sensor is compatible with SCIP2.2 communication protocol standard.

6.3 Output Circuit Diagram



Rated power: 30V, 30mA (or less)

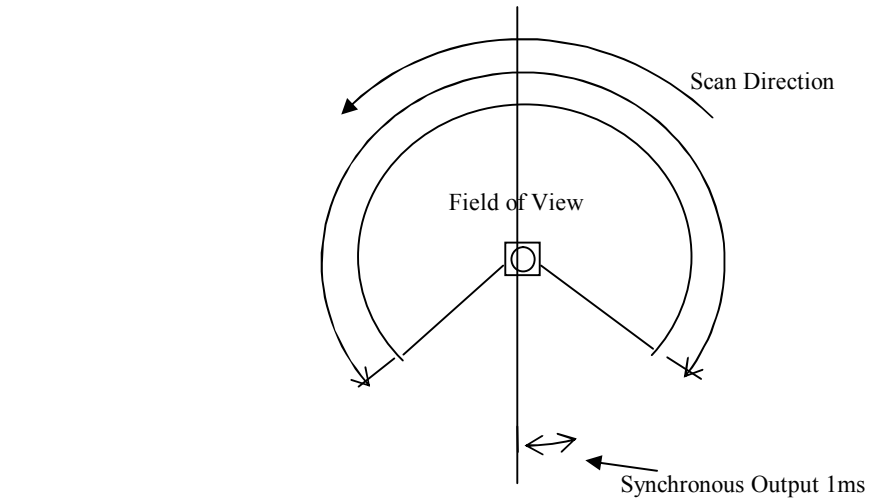
Note: Rated resistor should be used for the output.

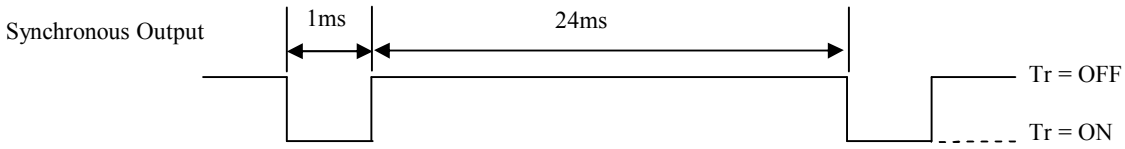
Figure 2

7. Control Signal

Synchronous Output (UTM-30LX)

1 pulse is approximately 1 ms. Output signal Synchronization timing chart is shown below (Figure 3).





Tr is OFF during Malfunction

Figure 3

8. Malfunction Output:

1. Laser malfunction: When the laser does not emit or exceeds safety class 1.
2. Motor malfunction: When the rotation speed differs from the default value (> 25 ms).

Synchronous/Warning signal will be turned OFF when these malfunctions are detected. The motor and laser will also stop. The details of error can be obtained via communication.

[Error code]

The cause of an error can be acquired from a "STAT" line of the "II" command response of the SCIP communications protocol. An error code and a solution acquired from a "STAT" line are as follows.

ID	Message	Meaning	Solution
000	no error.	Normal	No action is required
050	internal chip access failed.	Abnormal sensor processing system	Sensor has failed and needs to be repaired
100	Internal chip access failed.	Abnormal sensor processing system	
150	internal chip access failed.	Abnormal sensor processing system	
151	internal chip initialize failed.	Sensor processing system failed to initialize	
200	encoder error.	Encoder error	
250	motor startup failed.	Abnormality of the motor	
251	motor rotation error.	Motor rotation is not stable	Reduce the vibration and noise to the sensor
300	laser too high.	Abnormality of the laser light	Reduce the ambient light and noise to the sensor
301	laser too low.	Abnormality of the laser light	
302	laser no echo	Abnormality of the laser light	
303	measurement error.	The control process for measuring distance failed	Reduce the vibration and ambient light and noise to the sensor

△1

[The meaning of the distance value]

The meaning of “x” distance value of each step is as follows.

Distance value “x”	Meaning
x ＜ 23	Measurement error. The distance cannot be measured due to light interference or noise.
23 ≦ x ＜ 60000	Valid distance value [mm]
60000 ≦ x	Object does not exist or the object has low reflectivity.

△1

9. Multiecho Function △3

The sensor masures up to three echoes of reflection for each step (direction). Distance and intensity values of every echoes are obtained

Multiple echoes are produced by reflection on surface of transparent objects, reflection on objects' boundary and reflection from small particles such as rain drops, mist, dusts and fog.

This sensor feature of getting distance and intensity values of multiple reflections at the same direction is called Multiecho Function.

※ Two closely positioned objects or low reflectance objects may not produce multiple echoes, so that they are not detectable as separate ones.

9. Ethernet Settings

① Initial value

IP address: 192.168.0.10
Port number: 10940

② IP initialization

Remove the rubber cap located at the side of the bottom cover of the sensor. Press and hold the switch inside this hole for more than two seconds in order to start the IP initialization process. Release the switch after the LED flashes in orange color. This indicates the restart of the sensor. Finally, please insert the rubber cap to its original position.

10. Cautions

The heat is generated as the internal circuit of the sensor runs at a very high speed. The generated heat is concentrated at the bottom of the sensor. Please mount a heat sink or any appropriate component to release the heat. An aluminum plate (200mm x 200mm x 2mm) is recommended as the heat sink.

Mutual Interference could occur when two or more identical sensors are mounted at the same detection plane. This is because the sensor could not identify the origin of the received laser pulses. It causes measurement error for one or two steps. Performing data filtering could overcome this problem.

ANEXO D.
DESCRIPCIÓN E INSTALACIÓN DE ROS

ROS es un framework que permite escribir software orientado a robots el cual incluye un conjunto de herramientas, librerías y paquetes de software que brindan una experiencia de desarrollo completa y modular de manera tal que se puedan crear comportamientos robóticos complejos de una manera más sencilla. ROS nace como un proyecto de robótica propio de *Willow Garage* en el año 2007 aunque después el desarrollo y mantenimiento del framework paso a *Open Source Robotics Foundation*, a partir de su creación el proyecto creció basado en el desarrollo de software en conjunto o colaborativo, es decir el proyecto de ROS al ser construido bajo licencia BSD (open-source) cuenta con la colaboración de muchos laboratorios y comunidades enfocadas a la robótica, por ende un laboratorio especializado en la construcción de mapas en entornos cerrados puede crear un sistema robusto en este campo, mientras que otro grupo de desarrolladores es experto en la navegación y desarrolla los paquetes de software esenciales para cumplir a cabalidad dicha tarea, creando una red que contribuye a un sistema global y general, ROS.

Éste framework cuenta con los siguientes criterios de construcción y diseño:

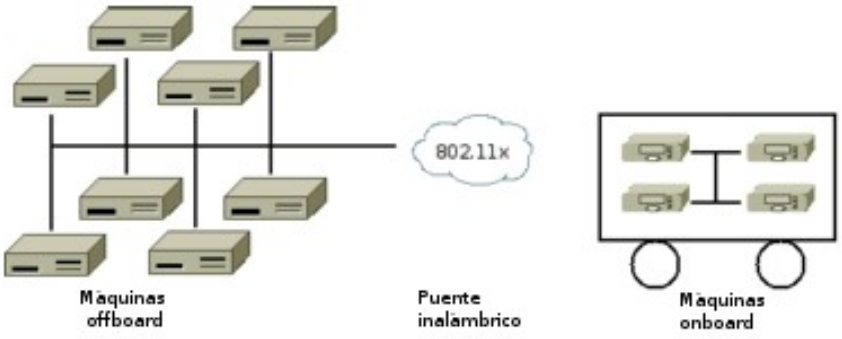
- ✓ Peer-to-peer
- ✓ Basado en herramientas
- ✓ Multilenguaje
- ✓ Ligero
- ✓ Libre distribución (Open-source).

Los criterios mencionados anteriormente hacen de ROS uno de los frameworks más poderosos para escribir código orientado a la robótica, una de las grandes ventajas es que cualquier sistema construido bajo ROS consiste en diversos *hosts* conectados en tiempo de ejecución bajo topología P2P (Ver figura 1) y de esta manera se evita la congestión de tráfico de datos innecesarios en los sistemas comunes basados en un servidor central, cuando se tienen diferentes centros de procesamiento conectados entre sí a través de una conexión LAN.

Adicionalmente los creadores diseñaron un micro kernel donde se alojan varias herramientas que ayudan a que se puedan construir, usar y ejecutar diferentes componentes de ROS con mayor facilidad a pesar de ser una arquitectura tan robusta. Las herramientas cumplen diversas funciones de gran utilidad en el desarrollo de cualquier sistema, la navegación por el árbol de código fuente, la visualización de las conexiones P2P, medición del ancho de banda en uso, auto

generación de documentación, entre otras muchas funciones que optimizan el proceso de desarrollo de código.

Figura 1. Conexión típica de un sistema de ROS



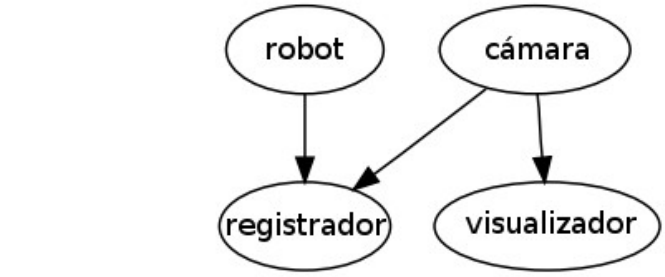
Fuente: M. Quigley, K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," *Icra*, vol. 3, no. Figure 1, p. 5, 2009.

De los aspectos más destacados en la implementación de ROS es que el desarrollo de drivers y algoritmos por parte de los colaboradores son librerías independientes que no tienen alguna dependencia propia del sistema o construcción de ROS, de ésta manera los creadores aseguran que se siga la filosofía de construcción de módulos ligeros. El paquete o librería se construye en el entorno de ROS el cual crea los ejecutables necesarios que permiten que el sistema pueda hacer uso de las funcionalidades de la librería lo que posibilita con mayor simpleza la reutilización y adecuación de las librerías en función de las necesidades del desarrollador. Por ésta característica es que ROS reutiliza numerosos proyectos de tipo open-source como sistemas de navegación, simuladores, algoritmos de visión y planeación provenientes de fuentes como Player Project, OpenCV u OpenRAVE. Como beneficio adicional es que cada paquete o librería se puede actualizar de manera independiente a través de repositorios externos.

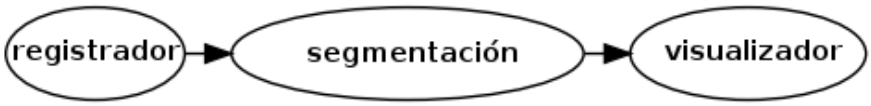
ROS posee la capacidad de grabar y reproducir datos de un sensor y otro tipo de información lo que significa que se pueden realizar diferentes pruebas sin necesidad de tener el robot de manera física, al grabar los datos obtenidos por el sensor éstos se pueden reproducir cuantas veces sean necesarios ya que ROS incluye la herramienta necesaria (rosbag) que cumple con dicha tarea y de una forma óptima lo que se traduce en una gran fiabilidad ya que los datos almacenados son prácticamente idénticos a los sensados por el dispositivo y

gracias a la arquitectura modular con la que se diseñó ROS no es necesario realizar algún tipo de modificación en el software para el correcto funcionamiento en cualquiera de los escenarios (robot real, simulador o reproducción de datos). Por ejemplo, al observar el diagrama de la figura 2(a) se ve una red que recoge información visual a través de una cámara, los datos son almacenados para su posterior reproducción. Los datos se almacenan en archivos".*bag" que se pueden reproducir en condiciones diferentes a como fueron grabados para realizar pruebas experimentales sobre un sensor, parte, comportamiento, etc. específico del robot que se está desarrollando, para le ejemplo se reproducen los datos para hacer un proceso de segmentación de la imagen.

Figura 2. Esquema de red que graba y reproduce datos en ROS



(a)



(b)

ROS ofrece muchas más bondades en beneficio del desarrollo de software pero las características mencionadas anteriormente son las que más sobresalen y las razones principales por las cuales se ha usado ésta plataforma como eje central de la operación algorítmica del presente proyecto. Adicional a ello encontramos que ROS es una de las comunidades más grandes de robótica y cuenta con un soporte en red a nivel mundial a través de distintos medios listados en su sitio web[1] en donde desarrolladores de todo el mundo ofrecen una asistencia fiable a cualquier particular que tenga inquietudes en el desarrollo de su proyecto. Además de un wiki que contiene la información necesaria para guiar a los novatos en su proceso de aprendizaje también se cuenta con un gran foro como lo es ROS Answers[2]. El framework ya cuenta con nueve versiones diferentes las cuales son denominadas por los creadores como *distribuciones*, éstas se reconocen con un nombre clave, es importante resaltar que cada versión cuenta con sus

1 http://www.ros.org/support/
2 http://answers.ros.org/questions/

características y el soporte es diferente para cada una, de igual manera se encuentran dirigidas hacia una variante específica del sistema operativo Ubuntu.

Distribuciones de ROS:

- Box Turtle (Marzo, 2010)
- C Turtle (Agosto, 2010)
- Diamondblack (Marzo, 2011)
- Electric Emys (Agosto, 2011)
- Fuerte (Abril, 2012)
- Groovy Galapagos (Diciembre, 2012)
- Hydro Medusa (Septiembre, 2013)
- **Indigo Igloo** (Julio, 2014)
- Jade Turtle (Mayo, 2015)

La distribución lanzada en el año 2014 (Indigo) es la que se implementa en el proyecto ya que es, hasta el momento, la única versión LTS *(Long Term Support),* lo que infiere que el soporte por parte de los creadores, desarrolladores y la comunidad en general es mucho más amplio y generoso. El proceso de instalación se encuentra claramente descrito en el wiki respectivo[3] así mismo el proceso de configuración del entorno de trabajo para el desarrollo de paquetes propios.

ESTRUCTURA BÁSICA

EL MASTER

Como se mencionó anteriormente, ROS está construido bajo la filosofía de "complejidad bajo composición", haciendo referencia a la manera en que ROS trabaja. Una colección de programas de software pequeños, denominados **nodos**, que al ejecutarse al mismo tiempo cumplen una función específica, lo que implica que la comunicación entre dichos programas debe ser garantizada para un correcto funcionamiento. El master es el encargado de intercomunicar cada pieza de software que se ejecuta en el entorno de ROS habilitando un intercambio multidireccional de información. Para utilizar ROS o cualquier programa implementado en el framework es obligatorio ejecutar y mantener ejecutado el master durante todo el tiempo de uso.

NODOS

Cualquier programa ejecutable en ROS se denomina *Nodo*, un sistema está compuesto de diferentes nodos, describiendo perfectamente la composición modular a la que se ha hecho referencia, si por ejemplo en un sistema robótico complejo se requieren realizar pruebas de un módulo específico (reconocimiento

3 http://wiki.ros.org/indigo

de objetos, desplazamiento, SLAM, etc.) solo es necesario ejecutar el o los nodos que se encargan de cumplir con la tarea deseada sin necesidad de ejecutar el sistema completamente. Los nodos se conectan directamente al master que se ejecuta inicialmente, una vez se tiene la comunicación entre los nodos, éstos transmiten información a través de **mensajes**. Básicamente los nodos son la célula o componente básico de cualquier sistema basado en ROS y se identifican dos roles para ellos *publicar* y *subscribir.* Un nodo transmite datos publicando la información en un **tópico** dado, mientras que el nodo que recibe los datos se suscribe a dicho tópico. Más adelante se da un ejemplo que esclarece el funcionamiento de dichos roles en el sistema general y la importancia de los mismos.

MENSAJES

Los mensajes contienen la información que es intercambiada a través de los nodos, en sí un mensaje es estrictamente un tipo de estructura de datos, ROS soporta los tipos primitivos tales como entero, float o punto flotante, boolean, etc. Pero adicionalmente se permite al desarrollador crear su propia estructura de datos conforme a los criterios y necesidades propias del proyecto. Los mensajes se alojan en los tópicos para que puedan ser accesibles a los nodos que los requieran.

TOPICOS

Como se indicó anteriormente los mensajes se transmiten haciendo uso de los tópicos, un tópico en esencia no es más que un dato de tipo String (por ejemplo odometry, map, cmd_vel) que hace la función de ser un identificador o etiqueta para los mensajes.

SERVICIOS

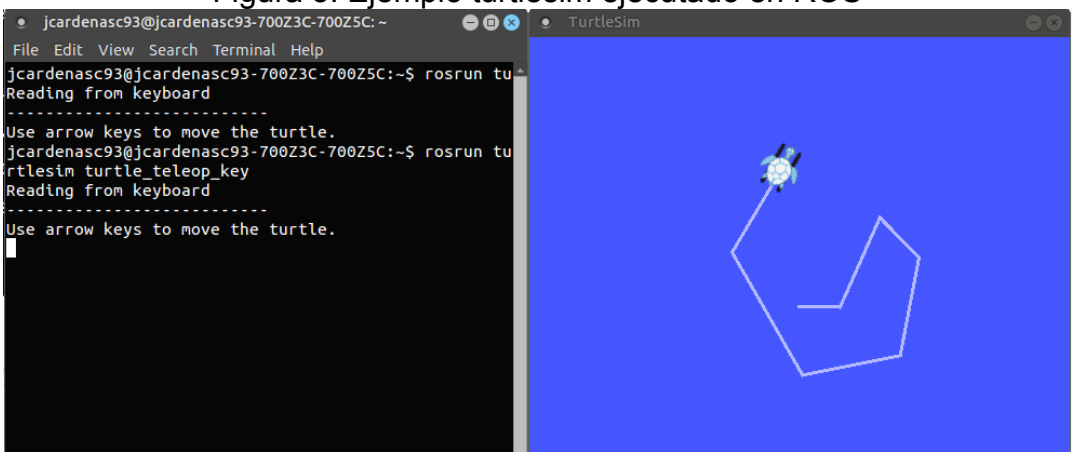
De forma paralela el framework ROS cuenta con otro método de comunicación entre los nodos activos, existen tres diferencias entre el método de tipo broadcast (a través de mensajes publicados en tópicos) y la implementación de servicios de comunicación.

- Existe un único nodo que cumple un servicio con un nombre particular y exclusivo, a diferencia del método de implementación de tópicos donde pueden existir diferentes nodos con el mismo nombre que publiquen o se suscriban a un mismo tópico.
- Haciendo una analogía a los servicios web, un nodo cliente realiza una solicitud a un nodo servidor el cual envía una respuesta a dicha solicitud devuelta al nodo cliente lo que implica que la información fluye de manera bidireccional entre los nodos.

- La comunicación de los servicios es uno a uno, el nodo cliente se comunica con el nodo servidor y la respuesta retorna exclusivamente al nodo cliente, a diferencia de la comunicación broadcast que se tiene con el otro método donde la información se encuentra en un tópico dado donde se pueden tener diversos publicadores y subscriptores.

Estos son los cinco componentes elementales de un proyecto de ROS, a continuación se proporciona un ejemplo sencillo de cómo interactúan entre sí en el entorno de ROS para poder tener un concepto general del funcionamiento de un sistema simple. El ejemplo es extraído del tutorial que se puede encontrar en el sitio web de ROS[4], consta de la ejecución de una serie de nodos que permiten tener control sobre una tortuga animada, haciendo uso del teclado, que se visualiza en una ventana dedicada como se puede observar en la figura 3.
La figura 4 muestra la relación entre los nodos y la interacción con el tópico donde se encuentran los mensajes que se transmiten. En este caso se puede observar que existen dos nodos uno identificado con el nombre de *teleop_turtle* mientras que el otro como *turtlesim*. El nodo *teleop_turtle* continuamente "publica" mensajes con la información captada del teclado en el tópico "turtle1/cmd_vel", a éste tópico se encuentra suscrito el nodo *turtlesim* quien se encarga de visualizar los movimientos en pantalla. Es de esta manera que la información se transmite entre los diferentes nodos conectados a un mismo máster.

Figura 3. Ejemplo turtlesim ejecutado en ROS



Fuente: Imagen tomada al ejecutar el ejemplo turtlesim bajo la plataforma ROS.

Figura 4. Esquema de interacción entre nodos.

Fuente: Gráfica creada con la herramienta rqt_graph del framework ROS.

4 http://wiki.ros.org/ROS/Tutorials/

Por otra parte la plataforma ROS cuenta con una estable integración con otras plataformas y librerías tales como los son Gazebo, un completo simulador de robots que a través de un plugin integra el framework ROS sin tener alguna complicación, de igual manera la librería OpenCV, muy común en el desarrollo de la robótica por sus múltiples librerías con algoritmos de visión computacional, muchas de las librerías encontradas en OpenCV son implementadas en diferentes paquetes de ROS para el uso y manejo de cámaras sin importar el hardware usado, adicionalmente la librería PCL (Point Cloud Library), la cual provee diversos algoritmos de nubes de puntos para la manipulación y procesamiento de imágenes tridimensionales y datos de profundidad, también se destaca la compatibilidad con la plataforma de Google, Android por medio de la implementación de su paquete rosjava (el cual se describirá en la siguiente sección) del mismo modo ROS incluye librerías para hacer uso de microcontroladores de la familia AVR estableciendo su comunicación a través de la UART con la que cuentan los mismos, por consecuencia las tarjetas Arduino tienen una gran compatibilidad con el framework.

En la página oficial de ROS hay un listado de diversas tarjetas de desarrollo, procesadores y dispositivos sobre los cuales ROS funciona correctamente, entre los cuales se destacan la Raspberry Pi 2, la tarjeta Jetson TK1 fabricada por Nvidia, los sistemas embebidos de Qualcom Dragonboard 410c e Inforce IFC6410 e incluso el Smartphone Nexus 5 fabricado por Google. Es de aclarar que por las especificaciones técnicas de rendimiento y procesamiento de los dispositivos mencionados, la instalación de ROS en los mismos no incluye todos los paquetes que se encuentran en la instalación *full-desktop* en ordenadores de escritorio o portátiles, sin embargo es posible instalar de manera individual algún paquete según lo desee el desarrollador.

El manual de instalación de la plataforma ROS (Robot Operating System) se encuentra en la página web oficial de la comunidad ROS [1]. A continuación se describe de manera resumida el proceso de instalación y la configuración del entorno de trabajo para el desarrollo y uso de programas que trabajen sobre esta plataforma.

Nota: Es importante resaltar que este proceso de instalación es para la versión de ROS "*indigo"* la cual solo esta soportada para versiones de Ubuntu Saucy (13.10) y Trusty(14.04), para instalar otras versiones es necesario leer la documentación encontrada en la página oficial de ROS

1) Instalación

Los comandos de instalación tienen que ser ejecutados en el terminal del sistema, al cual se puede acceder directamente presionando las teclas *Ctrl+Alt+t.*

1.1) Inicialmente es necesario permitir al sistema instalar repositorios no provenientes de la comunidad de Ubuntu, “restringidos”, “universales” y “multiversales”.
1.2) Configurar las listas de recursos “sources.list” para permitir la instalación de software proveniente de packages.ros.org, para ello se ejecuta desde el terminal

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu trusty main" > /etc/apt/sources.list.d/ros-latest.list'
```

1.3) Instalar las llaves de autenticación, ya que antes de instalar paquetes de ROS se deben adquirir estas llaves y agregarlas al administrador de instalación de paquetes del sistema.

```
$ wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -O - | sudo apt-key add -
```

1.4) Una vez configurados los repositorios, se actualizan los paquetes disponibles de instalación y se procede con la instalación de los paquetes de ROS, la versión full incluye: ROS, rqt, rviz, librerías de robots genéricos, simuladores 2D/3D, navegación y percepción en 2D/3D.

```
$ sudo apt-get update
$ sudo apt-get install ros-indigo-desktop-full
```

1.5) Se debe iniciar rosdep ya que esto permite instalar dependencias necesarias para ejecutar los componentes de ROS.

```
$ sudo rosdep init
$ rosdep update
```

Una vez se termine el proceso de instalación, se procede a configurar el entorno de trabajo en el sistema para ejecutar los paquetes de ROS.

2) Configuración del entorno

2.1) ROS utiliza ciertas variables de entorno para localizar los archivos que necesita para su ejecución. Para establecer estas variables de entorno, se debe ejecutar el script setup.bash que ROS ofrece mediante este comando

$ source /opt/ros/indigo/setup.bash

Cada vez que se lance un nuevo shell es necesario ejecutar este script, para agregar de forma automática las variables de entorno a cada shell que se ejecute es necesario agregar dicho script en el bash general del sistema

$ echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
$ source ~/.bashrc

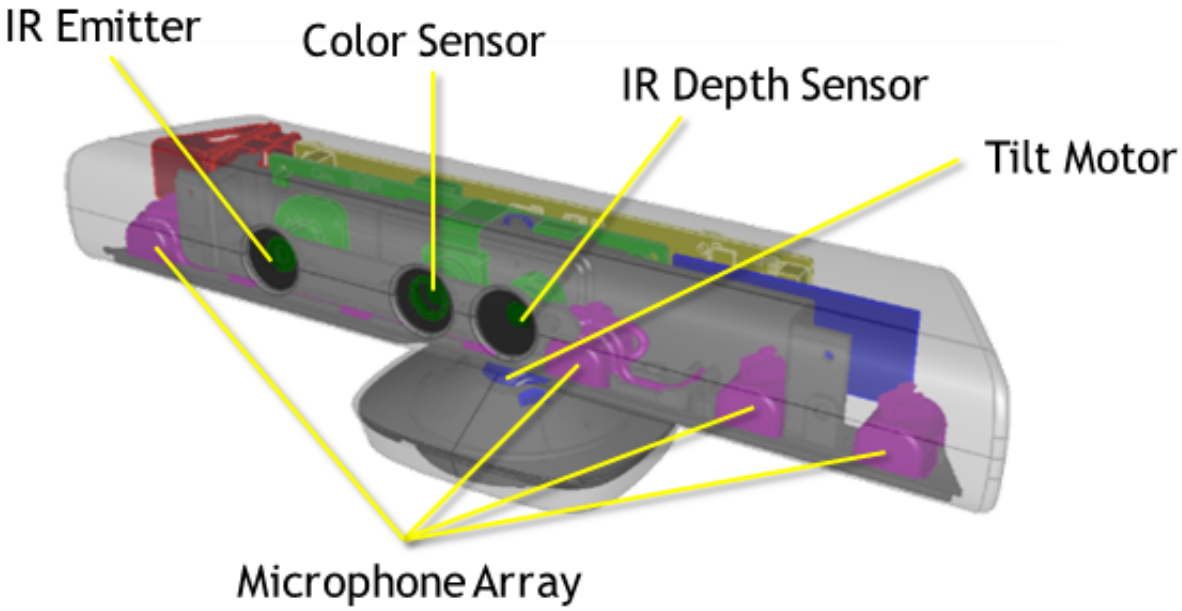
De esta manera se tiene la instalación completa del framework y se encuentra listo para hacer uso del mismo en cualquier proyecto que se desee.

Kinect for Windows Sensor Components and Specifications

Kinect for Windows 1.5, 1.6, 1.7, 1.8

Inside the sensor case, a Kinect for Windows sensor contains:

- An RGB camera that stores three channel data in a 1280x960 resolution. This makes capturing a color image possible.
- An infrared (IR) emitter and an IR depth sensor. The emitter emits infrared light beams and the depth sensor reads the IR beams reflected back to the sensor. The reflected beams are converted into depth information measuring the distance between an object and the sensor. This makes capturing a depth image possible.
- A multi-array microphone, which contains four microphones for capturing sound. Because there are four microphones, it is possible to record audio as well as find the location of the sound source and the direction of the audio wave.
- A 3-axis accelerometer configured for a 2G range, where G is the acceleration due to gravity. It is possible to use the accelerometer to determine the current orientation of the Kinect.



Specifications for the Kinect

Kinect	Array Specifications
Viewing angle	43° vertical by 57° horizontal field of view
Vertical tilt range	±27°

Frame rate (depth and color stream)	30 frames per second (FPS)
Audio format	16-kHz, 24-bit mono pulse code modulation (PCM)
Audio input characteristics	A four-microphone array with 24-bit analog-to-digital converter (ADC) and Kinect-resident signal processing including acoustic echo cancellation and noise suppression
Accelerometer characteristics	A 2G/4G/8G accelerometer configured for the 2G range, with a 1° accuracy upper limit.

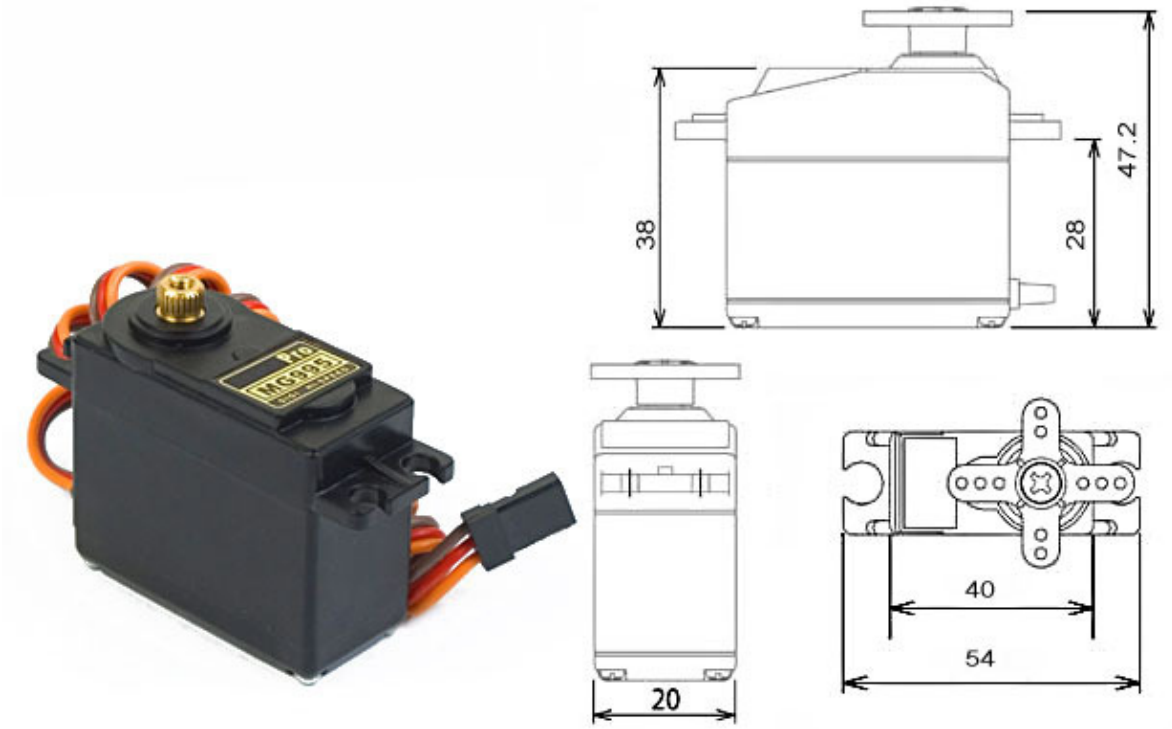
The resolution of the depth stream is dependent on the frame rate, and is specified by the DepthImageFormat Enumeration enumeration. Similarly, the resolution of the color stream is specified by the ColorImageFormat Enumeration enumeration.

See the depth space range section to see the depth data ranges as well as values for out-of-range data.

Community Additions

© 2016 Microsoft

MG995 High Speed
Metal Gear Dual Ball Bearing Servo

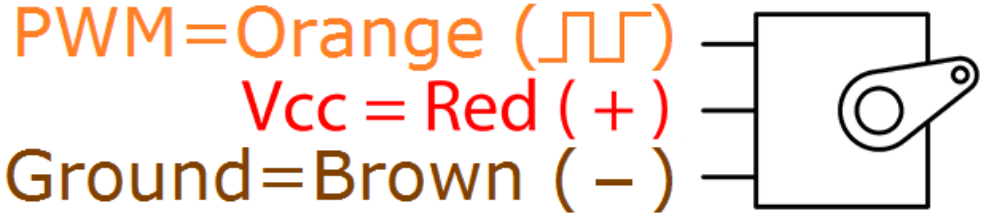


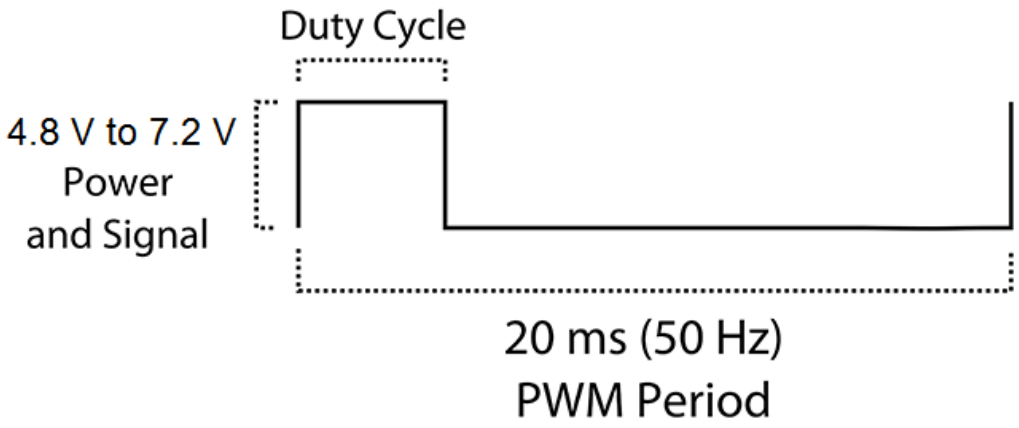
The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-speed standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG995 Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC


PWM=Orange
Vcc = Red (+)
Ground=Brown (–)


Duty Cycle
4.8 V to 7.2 V
Power
and Signal
20 ms (50 Hz)
PWM Period