UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO EN INGENIERÍA EN MECATRÓNICA

TEMA:

"PLATAFORMA DE ROBÓTICA MÓVIL PARA EXPERIMENTO A TRAVÉS DEL SISTEMA OPERATIVO DE ROBOTS (ROS): DESARROLLO DEL META-SISTEMA OPERATIVO"

(MOBILE ROBOTICS PLATFORM FOR EXPERIMENTS USING ROS: DEVELOPMENT OF THE META-OPERATING SYSTEM)

AUTOR: ANGEL ANIBAL OÑA PUJOTA

DIRECTOR: CARLOS XAVIER ROSERO CHANDI

IBARRA – ECUADOR

2017

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN

IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del Proyecto Repositorio Digital Institucional determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual se pone a disposición la siguiente información:

DATOS DEL AUTOR	
CEDULA DE IDENTIDAD	172258439 6
APELLIDOS Y NOMBRES	ANGEL ANIBAL OÑA PUJOTA
DIRECCIÓN	CAYAMBE, BARRIO "LA CRUZ", CALLES: JUAN MONTALVO S3-57 Y CALDERON
E-MAIL	aaoniap@utn.edu.ec
TELÉFONO MÓVIL	0986064059
DATOS DE LA OBRA	
TÍTULO	**"PLATAFORMA DE ROBÓTICA MÓVIL PARA EXPERIMENTO A TRAVÉS DEL SISTEMA OPERATIVO DE ROBOTS (ROS): DESARROLLO DEL META-SISTEMA OPERATIVO"** (MOBILE ROBOTICS PLATFORM FOR EXPERIMENTS USING ROS: DEVELOPMENT OF THE META-OPERATING SYSTEM)
AUTOR	ANGEL ANIBAL OÑA PUJOTA
FECHA	2017
PROGRAMA	PREGRADO
TÍTULO POR EL QUE OPTA	INGENIERO EN MECATRÓNICA
ASESOR	CARLOS XAVIER ROSERO CHANDI

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Angel Aníbal Oña Pujota, con cédula de identidad Nro. 172258439-6, en calidad de autor y titular de los derechos patrimoniales del trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Artículo 144.

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrollo sin violar derechos de autor de terceros, por lo tanto, la obra es original, y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

.....

Angel Aníbal Oña Pujota

C.I: 172258439-6

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DECLARACIÓN

Yo, Angel Aníbal Oña Pujota, con cédula de identidad N°. 172258439-6, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; y que éste no ha sido previamente presentado para ningún grado o calificación profesional.

A través de la presente declaración cedo los derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Técnica del Norte, según lo establecido por las Leyes de la Propiedad Intelectual, Reglamentos y Normativa vigente de la Universidad Técnica del Norte.

.....

Angel Aníbal Oña Pujota

C.I : 172258439-6

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Angel Aníbal Oña Pujota, con cédula de identidad Nro. 172258439-6, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5, 6, en calidad de autor del trabajo de grado denominado: "MOBILE ROBOTICS PLATFORM FOR EXPERIMENTS USING ROS: DEVELOPMENT OF THE META-OPERATING SYSTEM" , que ha sido desarrollado para optar por el título de Ingeniero en Mecatrónica en la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte.

.....

Angel Aníbal Oña Pujota

C.I: 040180021-4

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICO

Que la tesis previa a la obtención del título de Ingeniero en Mecatrónica con el tema: "MOBILE ROBOTICS PLATFORM FOR EXPERIMENTS USING ROS: DEVELOPMENT OF THE META-OPERATING SYSTEM", ha sido desarrollado y terminado en su totalidad por el Sr.Angel Anibal Oña Pujota, con cédula de identidad 172258439-6, bajo mi supervisión para lo cual firmo en constancia.

.....

Carlos Xavier Rosero Chandi

DIRECTOR

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AGRADECIMIENTO

A mi familia porque a pesar de muchas adversidades y duros momentos siempre han estado junto a mí brindándome su apoyo incondicional, gracias por enseñarme a luchar por mis sueños y jamás abandonarme en los momentos que más lo necesitaba.

Especial reconocimiento a Carlos Xavier Rosero por su apoyo y guía en el desarrollo del presente trabajo.

A la Universidad Técnica del Norte, la Facultad de Ingeniería en Ciencias Aplicadas, y de manera especial al personal docente quienes me ilustraron con sus conocimientos, siempre útiles en la vida profesional.

A todos los amigos y amigas que influyeron de manera directa o indirecta en la elaboración del proyecto, gracias a las personas que me han demostrado el verdadero significado de una amistad.

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DEDICATORIA

A:

Dios, por darme la oportunidad de vivir y por estar conmigo en cada paso que doy por fortalecer mi corazón e iluminar mi mente y por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante todo el periodo de estudio.

A mi mamá Guadalupe Pujota y mi padre Aníbal Oña, por darme la vida, quererme mucho, creer en mí y porque siempre me apoyan. Mamá y Papá gracias por darme una carrera para mi futuro, todo esto se lo debo a ustedes.

A mis hermanos, por darme el apoyo para culminar esta etapa en mi vida.

A todos mis amigos que tuve la oportunidad de conocerlos en esta etapa de estudio en la que pasamos muy buenos momentos compartiendo día a día las aulas de estudio. Gracias por esas amistades que serán duraderas.

RESUMEN

En la actualidad existen ambientes automatizados para facilitar actividades a la humanidad. Esto ha originado la necesidad de mejorar las tecnologías existentes, que generen mayor producción en el ámbito industrial y faciliten las actividades cotidianas del ser humano. Es así que aparecen los robots móviles terrestres, acuáticos y aéreos. Estos muestran su gran avance en la implementación de sensores y actuadores ya que pueden adquirir perspectivas del entorno que los rodea.

En la Universidad Técnica del Norte existe un laboratorio ampliamente implementado para la experimentación de diferentes cátedras netamente de ingeniería. Además en la carrera de Ingeniería Mecatrónica existe la necesidad de una plataforma de robótica que facilitaría la realización de prácticas de robótica tanto para estudiantes como para docentes. El presente proyecto se encuentra aplicado para el apoyo académico en la asignatura de Robótica.

Para la ejecución de este proyecto se abordó la literatura concerniente a robótica móvil, sistemas operativos y meta-sistemas operativos de robótica. Además se realizó la implementación del robot móvil a través de la plataforma robótica ROS (Robot Operating System) que permitió crear nodos para manejo de comunicación, visión artificial, movimientos, planeación de movimientos y autonomía. Se lo ejecutó en un ordenador con sistema operativo de código abierto basado en Linux. En este se implementó un meta-sistema operativo para un robot móvil. Haciendo uso de sus facilidades tales como librerías, aplicaciones sensoriales de software y herramientas que conjugadas con el hardware, se generó un automatismo de alto rango.

Como resultado se obtuvo una plataforma robótica móvil con aplicaciones para demostración de navegación autónoma, planificada y tele-operada. Esta plataforma servirá como base para la implementación de algoritmos de robótica.

La presente, es una investigación inicial que se usará como base para posteriores estudios con sistemas robóticos móviles. Los mismos podrán ejecutar nuevas aplicaciones robóticas y de inteligencia artificial. Para efectos de demostración, cumplimiento de alcances y

objetivos planteados en el presente proyecto, se ha realizado experimentaciones documentadas en este escrito.

ABSTRACT

At present there are automated environments to facilitate activities for humanity. This has led to the need to improve existing technologies, which generate greater production in the industrial field and facilitate the daily activities of the human being. So that look like terrestrial, aquatic and aerial mobile robots. These show their great advance in the implementation of sensors and actuators since they can acquire perspectives of the environment that surrounds them.

At the Technical University of North there is a laboratory widely implemented for the experimentation of different chairs of engineering. Also in the race of Engineering Mechatronics there is need for a robotic platform that would facilitate the realization of practical robotics for both students and teachers. This project is implemented for academic support in the course of Robotics.

For the execution of this project literature concerning mobile robotics, operating systems and meta-systems s robotic operation it was discussed. In addition , the implementation of mobile robot was made through robotic platform ROS (Robot Operating System) which created nodes for communication management, computer vision, movement, motion planning and autonomy. He was executed on a computer operating system open source based on Linux. In this meta-operating system for a mobile robot it was implemented. Making use of its facilities such as library s, sensory software applications and tools conjugated with hardware, automatism senior generated.

As a result a mobile robotic platform was obtained with applications for demonstration autonomous, planned and tele-operated navigation. This platform will serve as the basis for implementing robotics algorithms.

This, and s initial research will be used as a basis for further studies with mobile robotic systems. They can run new robotic and artificial intelligence applications. P ara

demonstration purposes, compliance is fingertips.Registering and objectives in this project, there has been documented experiments in this paper.

ÍNDICE DE CONTENIDOS

CONTENIDO **Pág.**

ÍNDICE DE FIGURAS

Figura **Pág.**

ÍNDICE DE TABLAS

Tabla **Pág.**

CAPÍTULO I

GENERALIDADES Y MOTIVACIÓN

1.1 Introducción

Una de las áreas de conocimiento que se ha desarrollado de manera acelerada en los la últimos años ha sido la robótica [1], es así que se han generado frameworks de robótica para ayudar a desarrollar aplicaciones sin mucha complejidad.

Para el desarrollo del presente proyecto se partirá de la necesidad de un robot móvil para el buen desempeño de la asignatura de Robótica e inteligencia artificial de la carrera de Ingeniería en Mecatrónica de la Universidad Técnica del Norte. Es así que se implementó una meta-sistema operativo robótico en un robot móvil enfocado para aspectos netamente didácticos, de desarrollo y estudio de la mencionada asignatura.

Existen soluciones tales como el implementado en la Escuela Politécnica Nacional (EPN), desarrollada por estudiantes a titularse en carrearas a fines a la Robótica. Que desarrollan un prototipo para evasión de obstáculos y aplicaciones básicas pero no adecuadas o exactas y de bajo coste como demostrará a la culminación del presente proyecto. Aquí un fragmento del alcance: "Para cumplir con este objetivo, se implementaron aplicaciones desarrolladas en ROS para la tele-operación, evasión de obstáculos y desplazamiento por distancia para la plataforma móvil Robotino® desarrollada por Festo Didactic." [2] Además, se cita un prototipo ejecutado por la Escuela Superior Politécnica del Ejercito (ESPE) el cual expone lo siguiente en su resumen: "En el presente trabajo se diseña y construye un prototipo de robot móvil cuyo objetivo es incorporar un control que pueda darle autonomía en el desplazamiento. Así, se integra la localización y mapeo simultáneos a las funciones operativas de la plataforma robótica móvil."[3]. Que de igual manera como el anterior proyecto realizado son prototipos basados en experimentación a diferencia del robot móvil que se expone en el presente proyecto. El cual es más versátil,

con más funcionalidades móviles, una perfecta manufacturación, capacidad sensorial y un adecuado control en su autonomía.

Es así que se implementó un meta-sistema operativo para el control del robot móvil en ROS (Robot Operating System). Esté se ejecuta en una distribución de Linux (Ubuntu 14.04.1) que es un sistema operativo de código abierto. El cual se acoplo a los requerimientos de un robot para navegación autónoma y planificada. Para el desarrollo y demostración de este proyecto, el meta-sistema operativo se ha implementado en la base de un robot móvil (iClebo Kobuki de firma coreana Yujin Robot) con un hardware ya manufacturado, versátil y se adecua para nuestra investigación. El desarrollo de este sistema se basa en la investigación y los conocimientos adquiridos a lo largo de la preparación académica e investigación de campo realizada. Se establece un producto real y ampliamente adecuado para el estudio, experimentación y diseño de innumerables aplicaciones que mejorarían la funcionalidad de sistemas robóticos móviles en nuestro entorno.

El documento se encuentra expuesto en cuatro capítulos y su contenido se resumirá a continuación:

Capítulo I: Muestra la problemática, los antecedentes he introducción de lo desarrollado.

Capítulo II: Se desarrolla el estado del arte es decir la fundamentación y justificación teórica de lo que se ha realizado.

Capítulo III: Se desarrolla la parte teórica del robot kobuki, características y funcionalidades

Capítulo IV: Muestra el desarrollo de lo implementado es decir los procedimientos y fundamentación teórica más detallada, paso a paso tal como se ha de presentar el trabajo ya finalizado.

Capítulo III: Muestra las conclusiones, análisis de resultados de la problemática y alcances obtenidos, además muestra trabajos futuros a realizar médiate esta investigación.

1.2 Problemática

Con el transcurrir del tiempo, el control y la automatización pasaron a ser parte de los procesos relacionados con el hombre. Hoy en día muchas de las necesidades humanas han sido solventadas a través de aplicaciones robóticas complejas. Así, existen innumerables aplicaciones basadas en plataformas de desarrollo en robótica (RSF) que día a día mejoran el diario vivir de la sociedad mediante aplicaciones de diferente topología.

En la actualidad existen varios entornos de desarrollo para robótica tales como Player/Stage/Gazebo, Yarp, Orocos, OpenRave, entre otros, todos ellos de código abierto y de desarrollo dependiente. Sin embargo, hasta hace pocos años atrás, no existía una plataforma que logre agrupar las mejores características de los proyectos mencionados en una solución integral y uniforme.

Con la aparición de ROS (Robot Operating System) se ha logrado resolver el problema de desarrollo de sistemas robóticos, proporcionando una plataforma integral, multi-lenguaje, orientada a herramientas, ligera y de código abierto. Un sistema robótico que usa ROS, posee servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes [4].

ROS está basado en una arquitectura de punto a punto donde el procesamiento toma lugar en los nodos, los cuales pueden recibir, entregar y multiplexar mensajes de sensores, controladores, actuadores, así como mensajes de estado y planificación.

En lo concerniente a la realidad de la Universidad Técnica del Norte y particularmente a la Facultad de Ingeniería en Ciencias Aplicadas, no existen robots móviles que se utilicen como equipo para enseñanza e investigación, situación que repercute negativamente en las asignaturas relacionadas con robótica y control, convirtiéndolas en netamente teóricas.

Es así que se considera necesario el desarrollo de una plataforma robótica basada en ROS para ser usada como base para la realización de innumerables investigaciones orientadas al mejoramiento del propio robot. Luego de adquirir experiencia en el tema, se puede usar la misma plataforma para el desarrollo de sistemas con propósitos industriales y/o de producción.

1.3 Justificación

Desde siempre, el ser humano ha utilizado los recursos a su disposición herramientas que colaboren a realización de tareas de forma más efectiva, rápida y segura. A medida que la tecnología avanza, estas herramientas se convierten en ingenios cada vez más complejos y capaz de realizar trabajos complicados de manera precisa y mucho mejor de lo que podría haberlo hecho un humano.

Para realizar el proyecto se implementará el sistema operativo robótico (ROS) por todas sus ventajas como: procesamiento de imágenes, transformación de coordenadas, navegación (Path Planning), control distribuido, bajo costo, escalabilidad, repetitividad, etc. Bajo este concepto, se precisa la necesidad de un autómata que muestre una interfaz sencilla, fraterna con el operador y posea suficiente versatilidad para que con ligeras modificaciones de hardware y firmware consienta su empleo en cualquier tipo de aplicación de robótica móvil. Se requiere de un equipo con índice costo – beneficio aceptable, vida útil larga, mantenimiento barato y repuestos accesibles en nuestro medio.

La relevancia del dispositivo propuesto consiste en una investigación aplicativa que servirá como base para el posterior diseño de sistemas robóticos móviles, y como indicador para exponer el alto nivel de conocimientos y el gran potencial innovador de los estudiantes de la UTN.

La implementación de este sistema se basa en nociones, habilidades, capacidades, destrezas, aptitudes, vinculadas a la competencia profesional y a los conocimientos adquiridos por el Ingeniero Mecatrónico.

1.4 Objetivos

1.4.1 Objetivo General

- ✓ Implementar el meta-sistema operativo de un robot móvil a través de la plataforma ROS.

1.4.2 Objetivos Específicos

- ✓ Determinar los requerimientos de la infraestructura del meta-sistema operativo (funciones).
- ✓ Desarrollar los nodos de ROS para el meta-sistema operativo a implementarse en el computador del robot.
- ✓ Integrar el meta-sistema para el robot móvil con la parte cinética y el sistema de tiempo real.
- ✓ Realizar pruebas y análisis de resultados de programación, funcionalidades y requerimientos para aplicaciones.

1.5 Alcance

El proyecto a implementarse, consistirá en el desarrollo de una investigación con nueva tecnología de aplicaciones y/o procesos en el ámbito académico e investigativo. Así, se usará la infraestructura de desarrollo de robótica ROS (Robot Operating System), que se implementará sobre un sistema operativo libre y abierto. El meta-sistema operativo se orientará a controlar el funcionamiento del robot.

Mediante nodos basados en ROS que básicamente se implementarán sobre la computadora del robot móvil, se pondrán en marcha el control remoto, visión artificial y el nodo serial para comunicación con la placa de tiempo real. Se pondrá énfasis en el desarrollo de este último nodo. La *Figura 1*, muestra un primer esquema sobre el meta-sistema operativo.

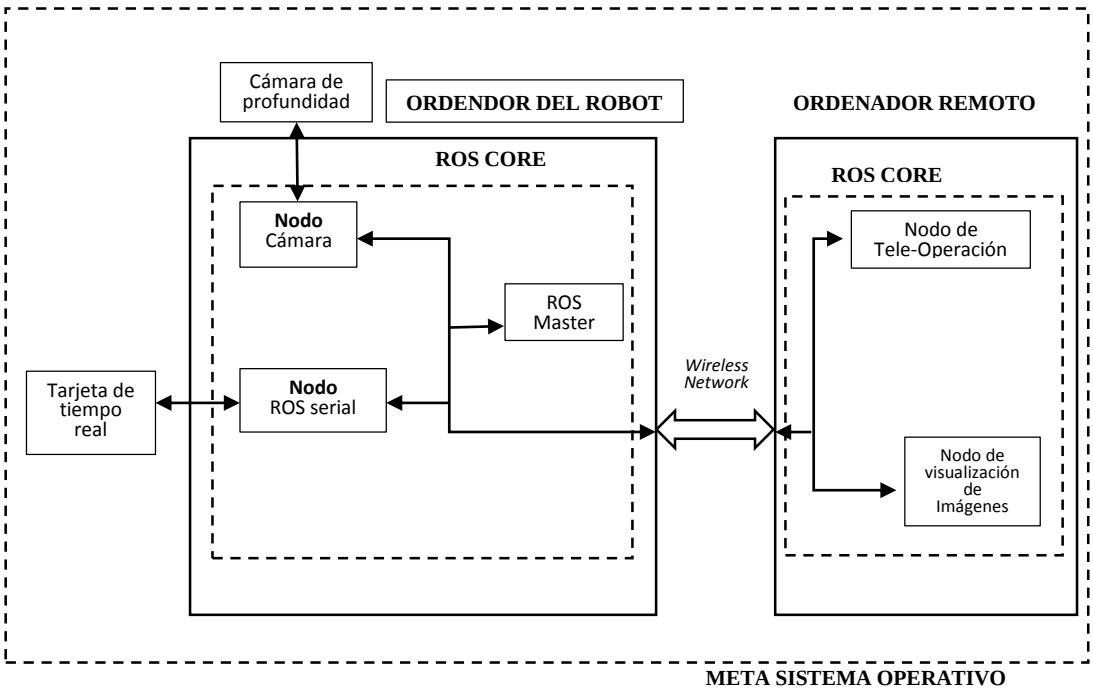


Figura 1: Meta-sistema operativo

Autor: Oña Angel

Además, con el meta-sistema a implementar se establecerá un nodo maestro permitiendo que a futuro se realicen innumerables aplicaciones en procesos robóticos basados en ROS, ya sean éstos industriales o de aporte a los problemas sociales.

CAPÍTULO II

REVISIÓN BIBLIOGRÁFICA

En esta sección establece directrices generales y el sustento para el desarrollo del presente trabajo de investigación. Se ha tomado en cuenta un punto de partida en las investigaciones de un complejo e inexplorado campo de la robótica, mostrando fragmentos adquiridos de trabajos realizados hasta la actualidad que se asemejan o utilizan la plataforma robótica (ROS) en el País.

2.1 Estado del Arte

La robótica en nuestro entorno ha ido ganando terreno en el campo de la investigación puesto que existen aficionados, adeptos, catedráticos y científicos que han hecho énfasis en inquirir, implementar robots que muestran la esencia o las generalidades principales de lo que es la robótica. En el norte del país y básicamente en la Universidad Técnica del Norte no se ha logrado profundizar el estudio de automatismos, es así que nuestro trabajo trata de mostrar y demostrar tecnologías diferentes a las ya manejadas en nuestro medio.

2.1.1 Robots Móviles

El poder manipular un prototipo robótico real que muestre características principales y funcionalidades exactas de robótica, es nuestro principal objetivo y es así que se han creado infinidad de estos prototipos en nuestro entorno académico y de investigación.

En base a la investigación realizada se destaca los siguientes proyectos de titulación: Diseño y Construcción de una Plataforma Robótica Móvil para Interiores Capaz de Realizar Slam (Simultaneous Localization And Mapping) ejecutado por estudiantes de la ESPE y un “Robot de evasión de obstáculos mediante el entorno ROS”, implementado por un estudiante de la Escuela Politécnica Nacional en noviembre del 2014, los cuales utilizan el sistema planteado ROS para su funcionamiento.

En este proyecto se construye un prototipo Robot que se muestra en la F*igura 2* con varias características. Aquí su resumen: "En el presente trabajo se diseña y construye un prototipo de robot móvil cuyo objetivo es incorporar un control que pueda darle autonomía en el desplazamiento. Se integra la localización y mapeo simultáneos a las funciones operativas de la plataforma robótica móvil."**[3]**. En la *figura 3* se muestra un fragmento de un mapa adquirido por este prototipo.

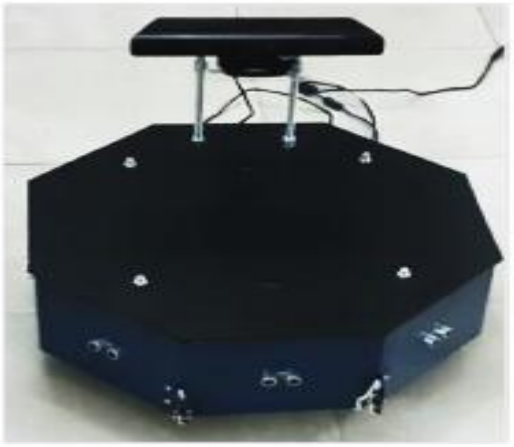


Figura 2: Prototipo de robot móvil para interiores (ESPE)[3]

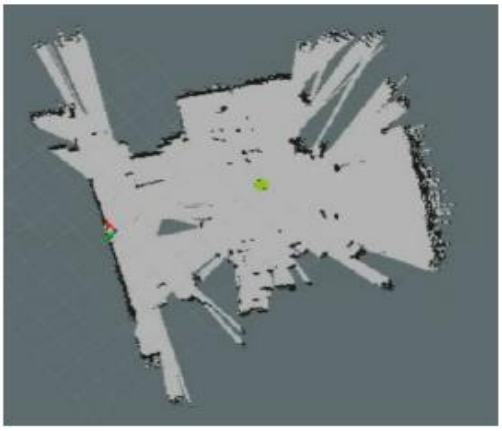


Figura 3: Mapa realizado adquirido por el prototipo [3]

Además citaremos un prototipo implementado por un estudiante de la Escuela Politécnica Nacional en noviembre del 2014, el cual según su resumen al igual que nuestro prototipo tiene la finalidad didáctica, la extensión de conocimientos y ampliar nuevos horizontes en el campo de la Robótica en todas las instancias donde así lo tengan previsto.

Para este proyecto los desarrolladores han adquirido un prototipo robótico de FESTO dedicado al desarrollo y didáctica. "Para cumplir con este objetivo, se implementaron aplicaciones desarrolladas en ROS para la teleoperación, evasión de obstáculos y desplazamiento por distancia para la plataforma móvil Robotino® desarrollada por Festo Didactic." **[2]**

Y se ha implementado una interfaz gráfica para la teleoperación la cual servirá para el mejor desempeño de la movilidad del prototipo que se muestra en la *Figura 4* y aquí un bosquejo del resumen realizado: "En el proyecto se implementó una Interfaz Gráfica de Usuario (GUI) en una PC, la cual fue desarrollada en el software Qt e incorporada a ROS. La GUI se encarga de realizar las acciones de teleoperación, evasión de obstáculos y desplazamiento por distancia de Robotino®".**[2]**

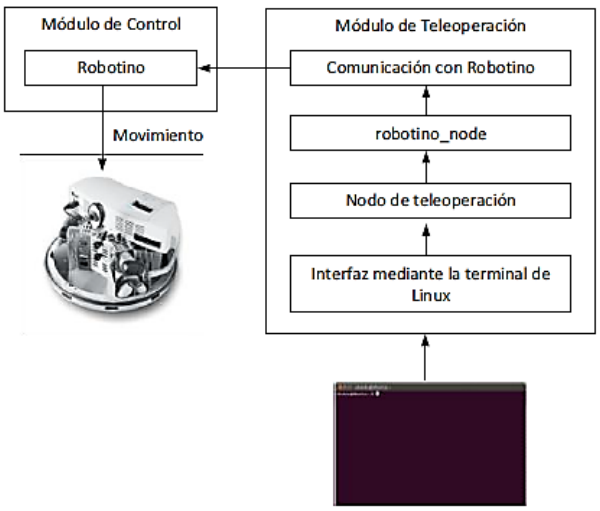


Figura 4: Módulo de control y Sistema de tele-operación.

2.1.2 Otras aplicaciones desarrolladas con ROS

Se ha destacado cronológicamente y acorde a mi investigación a los siguientes proyectos: un estudio de ROS para la extensión de capacidades y funcionalidad de legos NXT aplicado por una estudiante de la ESPOL en el 2013. Tomamos en cuenta el desarrollo de un robot Delta que mediante la aplicación de ROS se le ha implementado un cortador laser CNC para la elaboración de artículos publicitarios aplicado por estudiantes a titularse de la ESPE en diciembre del 2014. Y se muestra en la *Figura 5* la estructura del funcionamiento de los nodos en ROS.

Aquí mostramos en la imagen 5 una fotografía del prototipo y a continuación citamos una parte del resumen del "Estudio de ROS para la extensión de capacidades y funcionalidad de legos NXT" [5] aplicado por estudiantes a titularse de la ESPOL, con la colaboración del PhD. Daniel Ochoa D. "El objetivo de esta investigación nace de la necesidad de fomentar el uso de ROS ya que cabe recalcar que es de libre distribución y de código abierto. ROS es un framework de desarrollo de algoritmos de robótica, el cual permitió implementar la aplicación "LASER SCANNER". Para esta aplicación se usó Legos NXT, una cámara web USB y un láser de línea. El objetivo de la aplicación es escanear un objeto el cual se encuentra en cierta posición para que un láser de línea lo ilumine de arriba hacia abajo y así poder tomar una foto del mismo, cada cierta posición tomara una foto, que se guardara en una ruta especificada y así obtener la forma del objeto" **[5]**

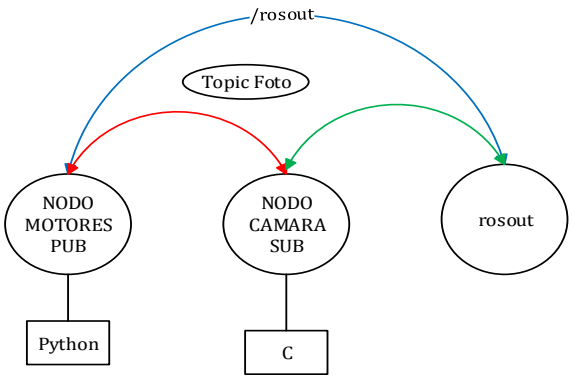


Figura 5: Estructuración de nodos

Figura 6: Fotografía del prototipo en funcionamiento

De este proyecto vale destacar la utilización de la aplicación en la plataforma ROS que expande la posibilidad de innumerables aplicaciones y funcionalidades ya sean estas de investigación y aplicación en la cotidianidad de la sociedad.

Este estudio utiliza las funcionalidades de ROS, implementado en legos NXT como se muestra en la *Figura 6,* además se muestran en la *Figura 5* la estructura de nodos para la funcionalidad de los motores y la cámara, el objetivo principal de esta aplicación es que se implementa ROS en varios lenguajes de programación, como Python y C++ que la plataforma admite como válidos para la funcionalidad de todas sus bondades.

Este trabajo al parecer fue el primero que se realizó y se presentó de una forma explícitamente académica en nuestro país, las ventaja que nos ofrece ROS como plataforma robótica, que encaja en las aplicaciones que nosotros queramos realizar, de esta aplicación, este estudio está enfocado en otros objetivos, nuestro proyecto e investigación de más alto nivel con un aporte más humanitario y aplicado a la didáctica que necesita o requiere una catedra de robótica en nuestro entorno y a su vez se deja una pauta para la generación de conocimiento que el prototipo y el meta-sistema operativo implementado nos permite.

Además se ha tomado en cuenta el desarrollo de un robot delta al cual mediante la aplicación de ROS se ha implementado un cortador laser CNC para la elaboración de artículos publicitarios elaborado por estudiantes a titularse de la ESPE publicado Diciembre del 2014. Y se muestra en la *Figura 7* una fotografía del robot.

Citamos un fragmento del resumen del trabajo realizado que tiene como finalidad la construcción, programación, simulación de un robot delta, para imprimir con láser sobre materiales suaves, imágenes prediseñadas o diseñadas por el usuario en ROS.

“La modelación se realizó mediante la cinemática inversa, la obtención de las ecuaciones del robot se utilizó el método geométrico, estas ecuaciones son utilizadas para el desarrollo de la programación en Python. En la programación, se utiliza librerías como OpenCV para la vectorización de la o las imágenes a dibujar por el robot. El visualizador de la simulación es el Rviz propio del software ROS.” **[6]**

Figura 7: Esquema del robot Delta

Es así como las funcionalidad de ROS nos demuestra su largo alcance como plataforma base, para innumerables proyectos "En esta ocasión se ha realizado un prototipo específico mas no dirigido a desarrollo del conocimiento, puesto que es aplicado a una microempresa de publicidad o con fines de lucro a comparación de nuestro robot que se va integrando de manera categórica en el campo de la investigación y el desarrollo de conocimiento". **[8]**

2.2 Breve comparación con el presente proyecto

Se ha puesto a consideración estos trabajos realizado en nuestro país y palpamos que cada uno tiene un aporte importante en el desarrollo e investigación de tecnologías en el campo de la Robótica, pues bien nuestro trabajo de campo e investigación muestra más de los trabajos citados en la importancia del estudio a profundidad y detallado de la materia.

Este trabajo se enfoca facilitar en él era de la didáctica, experimentación y desarrollo de aplicaciones netamente robóticas, con facilidad y accesibilidad de manipulación directa con un robot en este caso móvil, real funcional y con gran aceptación para el manejo de innumerables aplicaciones, puesto que esta echo o implementado en Software libre es decir sin fines de lucro en su gran mayoría, factibilidad y lo más importante con un robot altamente resistente y con proyecciones industriales como es el robot móvil Kobuki.

2.3 Introducción a ROS

El desarrollo de sistemas o plataformas para el control de automatismos, ha sido un eje fundamental para poder sobrellevar actividades del diario vivir de la sociedad en todos sus entornos ya sean estos: industrial, académico y/o doméstico, buscando brindar al ser

humano accesibilidad para ejecutar aplicaciones de diferente topología y en ambientes adversos o inaccesibles es así, que existen innumerables aplicaciones basadas en plataformas de desarrollo en robótica (RSF) que día a día mejoran su productividad.

Los entornos de desarrollo para robótica se han ido implementando en la actualidad tales como Player, Yarp, Open Robot Control Software (Orocos), OpenRave, Microsoft Robots Studio, entre otros, la gran mayoría de código abierto y de desarrollo dependiente[7]. Es así que hace pocos años atrás, se ha estado desarrollando una plataforma que agrupe las mejores características para el desarrollo de soluciones integrales y uniformes a actividades específicas en el entorno industrial y complejidad de ejecución humana, etc.

ROS (Robot Operating System), ha logrado resolver el problema de desarrollo de sistemas robóticos, proporcionando una plataforma integral, multi-lenguaje, orientada a herramientas, ligera y de código abierto.

Un sistema robótico que usa ROS, posee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. ROS está basado en una arquitectura de punto a punto donde el procesamiento toma lugar en los nodos, los cuales pueden recibir, entregar y multiplexar mensajes de sensores, controladores, actuadores, así como mensajes de estado y planificación**[4]**.

2.3.1 ¿Qué es ROS?

El Sistema Operativo de Robot (ROS) es un framework flexible para desarrollar algoritmos para softwares de robot. Es una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento robótico complejo y robusto a través de una amplia variedad de plataformas robóticas**[8]**.

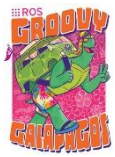
2.3.2 Distribuciones de ROS hasta la fecha

ROS es una plataforma en constante desarrollo ya que cuenta con desarrolladores y colaboradores al ser una plataforma de código abierto, ha impulsado a mejorar las prestaciones, es así que cada año presenta nuevas distribuciones ya sean estas una mejora

de la anterior o desarrollada para algo en específico. Lo esencial es el aporte de todos los que utilizamos ROS, ya sean en su repositorio que es un intercambio de conocimientos o en su gran mayoría aportes desinteresados. Aquí todas distribuciones hasta la actualidad en la *Tabla 1*.

Tabla 1: Lista de distribuciones de ROS hasta la actualidad [9]

<table>
<tr><td rowspan="11">LISTA DE DISTRIBUCIONES DE ROS</td><td>NOMBRE</td><td>FECHADE LANZAMIENTO</td><td>LOGOS</td></tr>
<tr><td>ROS Lunar Loggerhead</td><td>Mayo 23 del 2017</td><td rowspan="2">  </td></tr>
<tr><td colspan="2">Sitio web para más información:
http://wiki.ros.org/lunar</td></tr>
<tr><td>ROS Kinetic Kame</td><td>Mayo 23 del 2016</td><td rowspan="2">  </td></tr>
<tr><td colspan="2">Sitio web para más información:
http://wiki.ros.org/kinetic</td></tr>
<tr><td>ROS Jade Turtle</td><td>Mayo 23 del 2015</td><td rowspan="2">  </td></tr>
<tr><td colspan="2">Sitio web para más información:
http://wiki.ros.org/jade</td></tr>
<tr><td>ROS Indigo Igloo</td><td>Julio 22nd, 2014</td><td rowspan="2">  </td></tr>
<tr><td colspan="2">Sitio web para más información:
http://wiki.ros.org/indigo</td></tr>
<tr><td>ROS Hydro Medusa</td><td>Septiembre 4th, 2013</td><td rowspan="2">  </td></tr>
<tr><td colspan="2">Sitio web para más información:
http://wiki.ros.org/hydro</td></tr>
</table>

	ROS Groovy Galapagos	Diciembre 31, 2012	
	Sitio web para más información: http://wiki.ros.org/groovy		
	ROS Fuerte Turtle	Abril 23, 2012	
	Sitio web para más información: http://wiki.ros.org/fuerte		
	ROS Electric Emys	Agosto 30, 2011	
	Sitio web para más información: http://wiki.ros.org/electric		
	ROS Diamondback	Marzo 2, 2011	
	Sitio web para mas information: http://wiki.ros.org/diamondback		
	ROS C Turtle	Agosto 2, 2010	
	Sitio web para más información: http://wiki.ros.org/cturtle		
	ROS Box Turtle	Marzo 2, 2010	
	Sitio web para más información: http://wiki.ros.org/boxturtle		

2.3.3 Robots que trabajan con ROS

Por la versatilidad del sistema operativo que brinda ROS existen robots que han sido desarrollados para diferentes prestaciones tanto en el campo de la investigación,

académica y desarrollo de nuevas aplicaciones, tales como los expuestos a continuación. Aquí se muestran una tabla con los principales prototipos que se han desarrollado, ya que es un número extenso para citarlos a la totalidad.

Se han dividido por categorías, tales como aéreos, por sus componentes, terrestres, marinos y por la categoría de sensores, puesto que esta tabla fue extraída de la página principal de ROS la cual es desarrollada por la colaboración de innumerables desarrolladores en la última categoría no se ha podido mostrar actualmente ningún prototipo, no porque no se han desarrollado sí que no se puede mostrar porque se encuentra en proceso de publicación. En la *Tabla 2* se cita una muestra de la diversidad de robots, prototipos que utilizan ROS para su funcionamiento. Esta tabla muestra varios ejemplares divididos en categorías.

Tabla 2: Lista de robots desarrollados por ROS[9]

<table>
<tr><th colspan="3">ROBOTS DESAROLLADOS POR ROS</th></tr>
<tr><td>CATEGORIA</td><td colspan="2">http://robots.ros.org/category/aerial/</td></tr>
<tr><td rowspan="4">AEREOS
  </td><td>Gapter</td><td rowspan="2">  </td></tr>
<tr><td>Gapter EDU es una plataforma perfecta para aprender la programación de drones con ROS. Un paquete ROS completo con varias demostraciones está disponible para Gapter
http://robots.ros.org/gapter/</td></tr>
<tr><td>Erle-Plane</td><td rowspan="2">  </td></tr>
<tr><td>Es un avión de ala fija basado en el ganador del premio APM: Plane platform. Nuestro piloto automático de hardware, Erle-brain, utiliza el piloto automático APM para poder ejecutar misiones siguiendo una lista de puntos registrados por un GPS.
http://robots.ros.org/erle-plane/</td></tr>
</table>

<table>
<tr><td rowspan="2"></td><td>Erle-Copter</td><td></td></tr>
<tr><td>Erle-copter es un zumbido basado en Linux que utiliza el galardonado APM: Copter piloto automático del software. Es capaz de los diferentes modos de vuelo y es ideal para operaciones al aire libre. Ha sido diseñado para un tiempo de vuelo extendido y puede llevar un peso de despegue de aproximadamente 2 kilogramos.
http://robots.ros.org/erlecopter/</td><td>  </td></tr>
<tr><td rowspan="4">CONPONENTE
  </td><td colspan="2">http://robots.ros.org/category/component/</td></tr>
<tr><td>ABB Manipulators</td><td></td></tr>
<tr><td>Este repositorio forma parte del programa ROS-Industrial. En la actualidad contiene paquetes que proporcionan nodos para la comunicación con controladores de robots industriales ABB, modelos URDF para manipuladores soportados y paquetes MoveIt asociados.
http://robots.ros.org/abb-manipulators/</td><td>  </td></tr>
<tr><td>Lego NXT</td><td></td></tr>
<tr><td></td><td>NXT es un kit robótico modular fabricado por Lego. Hay interfaces ROS para el firmware de Lego por defecto (v1.28) basado en la biblioteca NXT-python de Douglas Lau.
http://robots.ros.org/lego-nxt/</td><td>  </td></tr>
</table>

<table>
<tr><td rowspan="2"></td><td>Intel Edison</td><td></td></tr>
<tr><td>El Intel® Edison es una plataforma de computación ultra pequeña que le permitirá a ROS ejecutarse como con un ordenador de tamaño normal.
http://robots.ros.org/intel-edison/</td><td>  </td></tr>
<tr><td rowspan="5">SUELO
  </td><td colspan="2">http://robots.ros.org/category/ground/</td></tr>
<tr><td>iRobot Roomba</td><td></td></tr>
<tr><td>El Roomba es una aspiradora autónoma vendida por iRobot. Fue diseñado para navegar por un espacio habitable y evitar obstáculos comunes como muebles mientras aspiraba el piso. Fue introducido en 2002.
http://robots.ros.org/irobot-roomba/</td><td>  </td></tr>
<tr><td>Kobuki</td><td></td></tr>
<tr><td>Kobuki proporciona datos altamente fiables de odometría, larga duración de la batería y proporciona potencia externa para sensores y actuadores. Con la estructura personalizada por la demanda del usuario. Establecido para el desarrollo y la investigación de varios robots.
http://en.yujinrobot.com/archives/portfolio-items/kobuki
http://robots.ros.org/kobuki/</td><td>  </td></tr>
</table>

	TurtleBot2	
	TurtleBot es un kit de robot personal de bajo costo con software de código abierto. Con TurtleBot, podrás construir un robot que pueda manejar tu casa, ver en 3D, y tener suficiente potencia para crear aplicaciones emocionantes http://robots.ros.org/turtlebot/	
	TIAGo	
	TIAGo es un robot de servicio diseñado para trabajar en ambientes interiores. Las características de TIAGo la convierten en la plataforma ideal para la investigación, especialmente en la vida asistida ambiente o la industria ligera. Combina la movilidad, la percepción, la manipulación y las capacidades de interacción humano-robot para un objetivo específico: poder ayudar en la investigación. http://robots.ros.org/tiago/	
	Robotino	
	Robotino es un popular sistema de robot móvil desarrollado por Festo Didactic. Robotino viene con sensores, actuadores e interfaces de software que usted esperaría de un moderno y moderno sistema de robot móvil. http://robots.ros.org/robotino/	

<table>
<tr><td rowspan="6">MARINO
  </td><td colspan="2">http://robots.ros.org/category/marine/</td></tr>
<tr><td>Kingfisher</td><td></td></tr>
<tr><td>El buque de superficie sin tripulación Kingfisher (USV) es un barco ágil y con pilas diseñado para investigación y prototipado rápido. Equipado con una estación de sensores, un Atom PC integrado para ejecutar controladores de hardware e inteligencia, propulsores eléctricos, GPS, radio wifi y cascos semi-planar, Kingfisher sirve como una plataforma de investigación marina, así como un sistema de reconocimiento remoto para sistemas batimétricos e higrométricos recopilación de datos.
http://robots.ros.org/kingfisher/</td><td>  </td></tr>
<tr><td>BlueROV</td><td></td></tr>
<tr><td>El BlueROV es un vehículo submarino operado remotamente de Blue Robotics. Cuenta con propulsores Blue Robotics, un controlador de vuelo con software ArduSub , y un Raspberry`Pi ejecutando ROS.
http://robots.ros.org/bluerov/</td><td>  </td></tr>
<tr><td></td><td></td></tr>
<tr><td>SENSOR
  </td><td>No existe información detallada ya que se encuentra esta sección en desarrollo.</td><td></td></tr>
</table>

2.3.4 Estructura, comandos y conceptos de ROS

Para ejecutar ROS en cualquier prototipo, este sistema operativo de robots necesita de parámetros y varios componentes que se comunican entre sí. Estos se conjugan de forma estructural, de acuerdo al desarrollo de la aplicación y requerimientos del sistema. A continuación se conceptualiza los parámetros para una mejor comprensión de lo expuesto.

Repositorio.- Son ficheros cuales están formados por uno o varios paquetes, pilas los cuales están a nuestra disposición y son parte fundamental en el desarrollo ya que tomaremos contacto con este cada vez y cuando necesitemos realizar nuevas aplicaciones**[10]**.

Pilas (Stack).- Es una colección de paquetes que comparten una misma funcionalidad, por ende son un conjunto de nodos los cuales juntos proporcionan una variedad de funcionalidades (por ejemplo una pila de navegación). Se los puede comparar o son equivalentes a una librería en otros sistemas de programación, dentro de estos podemos encontrar unos ficheros con metadatos que proporcionan la información para que estos funcionen de forma adecuada (dependencias, compilaciones etc.)**[10]**.

Paquete (package).- El software en ROS está organizado en paquetes, es así que un paquete puede contener un nodo, una librería, conjunto de datos, o cualquier información que pueda ser funcional para construir un módulo y es análogo a un paquete en C. Los paquetes pueden organizarse en pilas (stacks)**[4]**.

Nodo.- Los procesos ejecutables que están incluidos dentro de los paquetes. Un nodo es un proceso que realiza algún tipo de computación en el sistema. Los nodos se combinan dentro de un grafo, compartiendo información entre ellos, para crear ejecuciones complejas. Un nodo puede controlar los motores, otro un láser y otro la creación de mapas, etc. y básicamente procesos que están incluidos dentro del paquete**[4]**.

ROS tiene herramientas para manejar los nodos y darles información sobre ellos como rosnode. La herramienta rosnode es una herramienta de línea de comando para mostrar información sobre los nodos y se muestra con más detalle cada uno en la T*abla 3*.

Servicios.- Este tipo de arquitectura está encargada de realizar la comunicación entre nodos, utiliza dos tipos de mensajes, uno para la solicitud, y otro que es la respuesta dada

por el otro nodo a esa petición. En los programas se pueden ejecutar nodos servidores y nodos clientes para la comunicación entre nodos como se muestra en la *Figura 8***.**

Figura 8: Comunicación entre nodos

Tópicos.- "Tópicos o temas, son los nombres que identifican el contenido de un mensaje; y estos se enrutan de dos formas, una publicador y otra suscriptor"**[4]**. Un nodo que está interesado en un determinado tipo de datos se suscribe al tema correspondiente. En la *Figura 9* se aprecia cómo se enlazan nodos con diferentes temas o tópicos para él una aplicación determinada.

Puede haber varios editores y suscriptores concurrentes a un mismo tema, y un único nodo puede publicar y / o suscribirse a múltiples temas. En general, los editores y suscriptores no son conscientes de la existencia de los demás.

Se puede pensar en un tema como un Bus de mensajes. Cada Bus tiene un nombre, y cualquier persona puede conectarse al bus para enviar o recibir mensajes, siempre y cuando sean del tipo correcto.

ROS tiene una herramienta para trabajar con tópicos llamada rostopic. Es una herramienta de línea de comandos que proporciona información sobre el tópico o publica datos directamente sobre la red y se los muestra con más detalle en la *Tabla 3***.**

Figura 9: Esquema de funcionamiento nodos (node) y temas (topics)

Mensajes.- Los nodos se comunican entre sí pasando mensajes. Un mensaje es simplemente una estructura de datos, que comprende los tipos de campos. Los mensajes pueden incluir estructuras arbitrariamente anidadas y matrices (al igual que las estructuras de C)**[10]**.

Maestro.- El Maestro proporciona registro de nombres y la búsqueda para el resto de los nodos. Sin el Maestro, estos no serían capaces de encontrar mensajes entre sí, intercambiar, o invocar los servicios, lo que hace que sea totalmente indispensable a la hora de ejecutar cualquier tipo de programa**[10]**.

Tabla 3.- Comandos básicos de ROS [4]

COMANDOS	DESCRIPCIÓN
roscore	Ejecuta todo lo necesario para que dar soporte de ejecución al sistema completo de ROS. Siempre tiene que estar ejecutándose para permitir que se comuniquen los nodos. Permite ejecutarse en un determinado puerto (ej. roscore o roscore -p 1234)
roscd	Cambia a un directorio de paquete o pila (ej. roscd stage)
roscreate-pkg	Crea e inicializa un paquete. Se tiene que ejecutar desde uno de los directorios válidos para que contengan paquetes. El formato de ejecución es: roscreate-pkg paquete [depen1 ...] donde depen1 es una dependencia. Por ejemplo, si el paquete que estamos creando va a usar los mensajes estándar y va a usar código c++, debemos indicar las dependencias std_msgs y roscpp.
rosnode	Proporciona información sobre un nodo. Disponemos de las siguientes opciones:
	rosnode info: nodo (muestra información sobre el nodo)
	rosnode kill: nodo (mata ese proceso)
	rosnode list: (muestra los nodos ejecutándose)
	rosnode machine: maquina (muestra los nodos que se están ejecutando en la máquina).
	rosnode ping nodo: (comprueba la conectividad del nodo).
rosrun	Permite ejecutar cualquier aplicación de un paquete sin necesidad de cambiar a su directorio. Podemos pasarle parámetros con

	_my_param:=value (ej. rosrun stage stageros) stage es el paquete y stageros es la aplicación que ejecutamos.
rostopic	Permite obtener información sobre un tópico.
	rostopic bw: (muestra el ancho de banda consumido por un tópico)
	rostopic echo: (imprime datos del tópico por la salida estándar)
	rostopic find: (encuentra un tópico)
	rostopic info: (imprime información de un tópico)
	rostopic list: (imprime información sobre los tópicos activos)
	rostopic pub: (publica datos a un tópico activo)
	rostopic type: (imprime el tipo de información de un tópico)
roswtf	Permite chequear si algo va mal. Ejecutamos roscd y después roswtf.

2.3.5 ¿Cómo funciona ROS?

Pues bien una vez ya conocido los parámetros de funcionamiento de ROS explicaremos la lógica del funcionamiento, mediante un pequeño ejemplo:

Supongamos que tenemos dos nodos; Un nodo “Cámara” y un nodo Image_viewer. Una secuencia normal de eventos comienza con el nodo cámara notificando al maestro que quiere publicar imágenes sobre el tema o tópico "imágenes". Tal cual se muestra en la *figura 10***.**

Figura 10: Solicitud del nodo A para publicar

Ahora, “Cámara” publica imágenes en el tema de "imágenes", pero aun nadie se ha suscrito a ese tema sin embargo, no se envían datos. Ahora, Image_viewer quiere

suscribirse al tema "imágenes" para ver si quizás allí hay algunas imágenes. Como se ilustra en la *figura 11*.

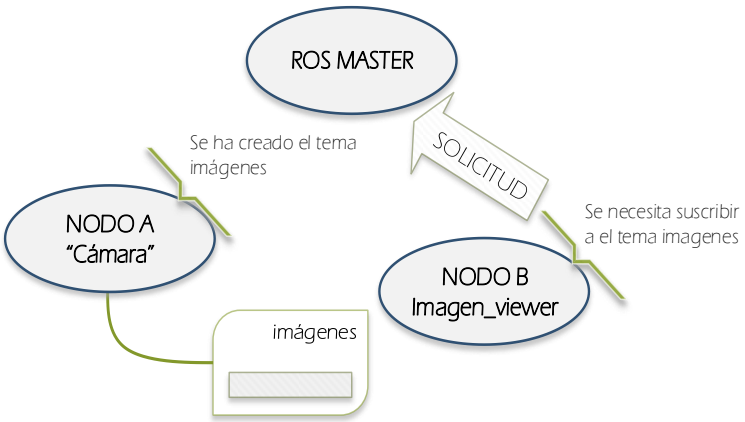


Figura 11: Solicitud del nodo B para suscribirse al tópico del nodo A

Ahora que el tema "imágenes" tiene tanto un editor como un suscriptor, el nodo maestro notifica a Cámara y Image_viewer sobre la existencia del tema imágenes para que puedan empezar a transferir imágenes entre sí. Como se ilustra en la *figura 12*.

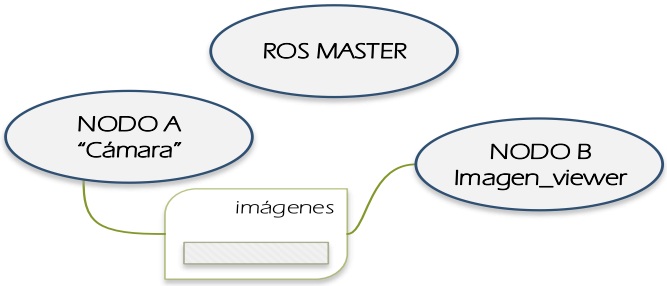


Figura 12: Comunicación entre el nodo A y nodo B

CAPITULO III

ROBOT KOBUKI

3.1 Generalidades

Kobuki es una base móvil de una plataforma de hardware de código abierto denominado TurtleBot. Ésta tiene la capacidad de manejar visión, localización comunicación y movilidad. Pudiéndose mover de forma autónoma con cualquier cosa encima de ella a donde a cualquier sitio propuesto, evitando los obstáculos en el camino**[11]**.

La base móvil es la parte principal del módulo y la interoperabilidad de cualquier prototipo robótico.

El robot móvil Kobuki es catalogado como la primera tortuga robótica de Corea ya que fue desarrollada en este país, diseñada para durar mucho tiempo con alto rendimiento de baterías, implementado con la más alta gama de sensores, además tiene una odometría altamente confiable, desarrollada específicamente para funcionar con ROS**[12]**.

Las características funcionales como tiempo de carga, velocidad máxima, escaladas en diferentes superficies etc. se resumen en la *Tabla 4*.

Tabla 4: Características funcionales de la base móvil kobuki[13]

ESPECIFICACIÓN FUNCIONALES	
DESCRIPCIÓN	**ESPECIFICACIONES**
Velocidad máxima:	70 cm/s
Diámetro	351.5mm
Altura:	124.8mm
Peso:	2.35kg (4S1P - pequeño)
Velocidad de rotación máxima	180 grados/s (> 110 grados/s rendimiento de giro se degradará)
Carga útil	5 kg (suelo duro), 4 kg (alfombra)
Acantilado	no va a conducir a un precipicio con una profundidad superior a 5 cm
Escalada Umbral	sube umbrales de 12 mm o más bajas
Escalada alfombra	sube alfombras de 12 mm o menores

Carga útil	3/7 horas (batería pequeña / grande)
Tiempo de espera para carga	1,5 / 2,6 horas (batería pequeña / grande)
Auto carga	dentro de un área 2mx5m en frente de la estación de acoplamiento

3.1.1 Hardware

Es el conjunto que integra la estructura física de un equipo. En general son los componentes básicos de tipo eléctrico, electrónico, electromecánico y mecánico necesarios para lograr una funcionalidad mínima; sin embargo, también se encuentran componentes complementarios utilizados para realizar funciones específicas que van más allá de las básicas, pero que no son requeridos para un correcto funcionamiento.

3.1.2 Características y Especificaciones

Las características del hardware de la base móvil se muestran en la *Tabla 5*, la cual hace referencia a sensores, actuadores, sobrecargas, alimentación y conexiones. En la *Figura 13* se muestra la ubicación de las características del hardware.

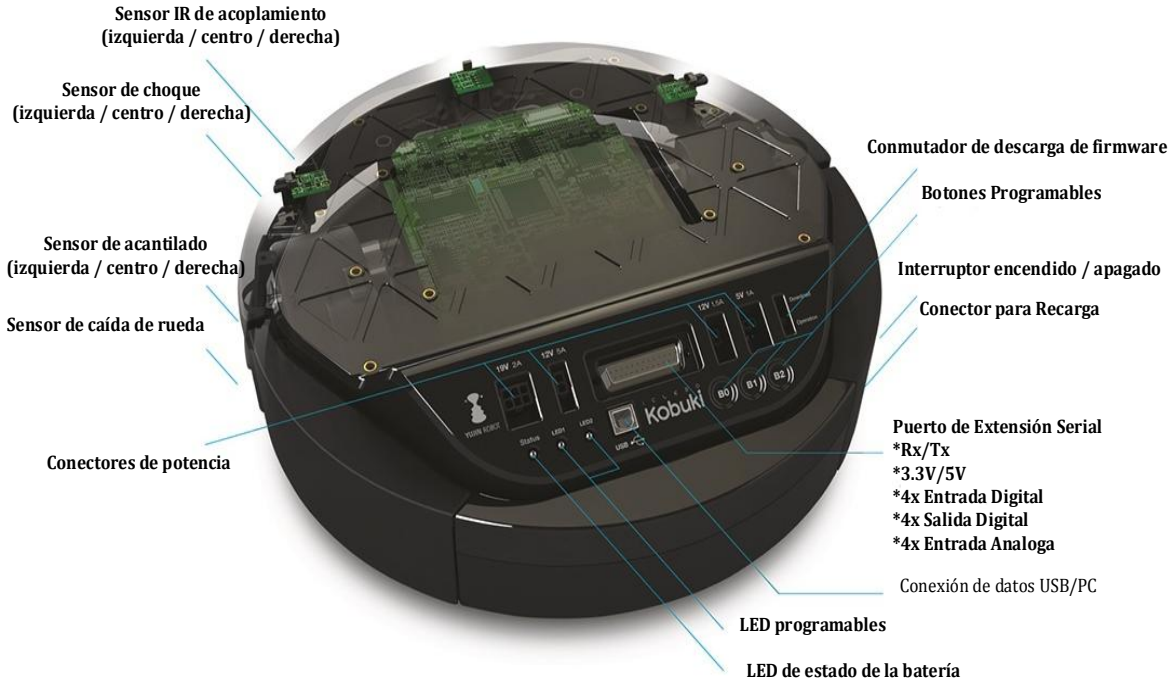


Figura 13: Componentes y partes específicas de la base móvil [14]

Adema de las características expuestas cabe recalcar que se suministra una odometría precisa a través de 3 ejes de giro y un encoder de alta resolución en las ruedas. Para instalación de módulos adicionales es posible a través del puerto serie Rx / Tx y el pin

USB. Además se facilita con el suministro de potencia adicional para nuevos sensores y actuadores con conexiones a 5V / 1A, 12V / 1.5A y 12V / 5A[14]. Algunos componentes adicionales como los nombrados anteriormente se los puede apreciar en la *Figura 13.*

Cabe recalcar la autonomía de las baterías son de 3 horas con una batería básica y más de 7 horas de funcionamiento con batería de gran tamaño que se exponen sus características en la *Tabla 5*, la carga simultánea del robot y de una notebook portátil (carga opcional) es posible a través de una toma de corriente de 19V / 2.1A. o al suministro doméstico con el cargador o estación de carga de la base del robot.

Tabla 5: Características del Hardware[15]

ESPECIFICACIONES DE HARDWARE	
DESCRIPCIÓN	**ESPECIFICACIÓN**
Conexión a ordenador	USB o por medio de pines RX / TX en el puerto paralelo
Sobrecarga del Motor de detección	desactiva potencia en la detección de corriente elevada (> 3A)
Giro	calibrado de fábrica, 1 eje (110 grados /s)
Parachoques	izquierda, centro, derecha
Los sensores de desnivel	izquierda, centro, derecha
sensor de caída de la rueda	izquierda, derecha
conectores de alimentación	5V / 1A, 12V / 1.5A, 12V / 5ª
Puerto de extensión serial	3,3 V / 1A, 5V / 1A, 4 x analógicas en, 4 x digitales en, 4 x salida digital
Audio	varias secuencias de pitidos programables
LED programable	2 x dos de color LED
Estado del LED	1 x LED bicolor [Verde - alta, Naranja - baja, verde y parpadeante - carga]
Botones	3 x botones táctiles
Batería	Litio-Ion, 14,8 V, 2200 mAh (4S1P - pequeño), 4400 mAh (4S2P - grande)
Sensor de Velocidad de datos	50Hz
Recarga de adaptador	Entrada: 100-240V AC, 50 / 60Hz, 1.5A máx.; Salida: 19V DC, 3.16ª
Netbook conector de recarga (sólo permitido cuando se está recargando robot)	19V / 2.1A DC
Soporte para receptor IR	izquierda, centro, derecha

3.3 Software

El software necesitan varias características para su funcionamiento como el firmware, lenguaje de programación etc. que se detallan en la *Tabla 6*. El software muestra el control, protocolos de comunicación, Además en esta sección se toma en cuenta las especificaciones de los nodos de la base móvil a utilizar y que se detallan en la *Tabla 6* y se ilustra en la *Figura 14* con un ejemplo como se integran para generar una aplicación.

Tabla 6: Características y especificaciones del Software

ESPECIFICACIÓN DE SOFTWARE	
Firmware:	actualizable a través de USB
Lenguajes de programación soportados:	C ++ para Linux y Windows
Módulos de simulación	Rviz, Gazebo y herramientas ROS
Nodos de ROS:	
Odometría	/Odom
Diagnostico	/Diagnostic
Control de sensores, motores actuadores	/sensors /divices /controller
Entro otros nodos, tópicos y comandos que se detallan en la siguiente página: http://wiki.ros.org/kobuki_node	

3.3.1 Nodos de KOBUKI

Un nodo se lo puede denominar un programa o algoritmo corto ejecutable dentro del paquete de ROS, usa la librería cliente de ROS para comunicarse con otros nodos, y estos pueden publicar o subscribirse a un tópico, además de usar cualquier servicio. ROS nos permitirá usar nodos creados con otros lenguajes de programación (phyton y c++).

En la *Figura 14* se muestra un ejemplo que muestra cómo trabajan los nodos del Kobuki en ROS en un proceso navegación. En la parte de la plataforma de inicio y el robot se muestra el Launchpad_node que se suscribe al tópico pid_velocity y como siguiente tenemos al nodo twist_to_motors el cual hace el control de giro de los motores para guiar la dirección y como último tenemos a navigatio stack este nodo hace las veces de ntrelazar todo el sistema con el diff_tf que es un controlador diferencial.

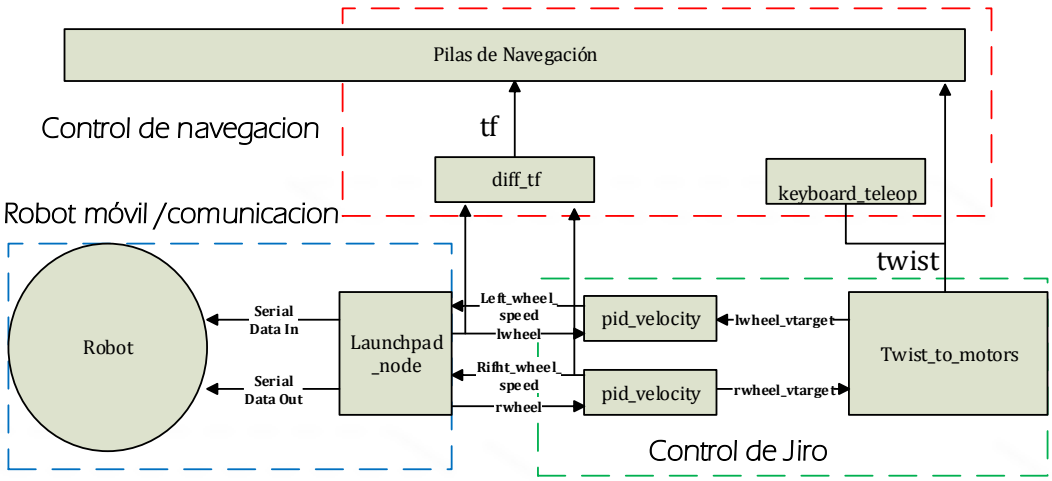


Figura 14.- Jerarquía del funcionamiento, control de nodos y tópicos en el robot Kobuki [16]

CAPÍTULO IV

IMPLEMENTACIÓN, PRUEBAS Y ANÁLISIS DE RESULTADOS

En esta parte del presente trabajo escrito se muestra cómo se conjuga la revisión bibliográfica y la experimentación. Al implementar el meta-sistema operativo en la plataforma ROS, el funcionamiento y demostración de movilidad del robot móvil. Es así como se conjuga el software y el hardware para mostrar funcionalidades físicas del robot móvil implementado en el presente proyecto.

4.1 Requerimientos

- ✓ Para el funcionamiento del proyecto se necesita los siguientes requerimientos.
- ✓ Una base de un robot móvil Kobuki.
- ✓ Una netbook de menos de 21cm de ancho para encajar en la estructura del robot.
- ✓ Estación de carga: la base del robot móvil se acopla autónomamente a esta estación en la cual puede cargarse cuando detecte que necesita energía.
- ✓ Estación de trabajo: Un segundo computador con una tarjeta gráfica y RAM suficiente para simulaciones 3D (4 GB +). Este ordenador hace las veces de maestro para tener control sobre el robot móvil en caso de manejarlo remotamente (se lo puede tele-operar o comandar remotamente de varios computadores).
- ✓ Router: Un enrutador inalámbrico permite que la estación de trabajo y el robot móvil se comuniquen a través de IPs locales.

4.2 Configuración del Hardware

La configuración del hardware para poner el robot en marcha se ilustra en la *Figura 15***.** La base del robot móvil kobuki esta pre-construido y viene con dos cables: un cable USB y un cable de poder. El cable USB conecta el ordenador con la base móvil del robot. El cable de poder alimenta al robot es decir recarga una batería que se incorpora a la base del robot móvil, el cable de poder también se lo acopla a la estación de carga que de igual

manera alimentara a la batería del robot mediante un algoritmo programado en el robot o manualmente. Además la configuración del hardware para la puesta en marcha utilizando dos ordenadores necesita de un router para la comunicación de los mismos mediante el protocolo de comunicación ssh.

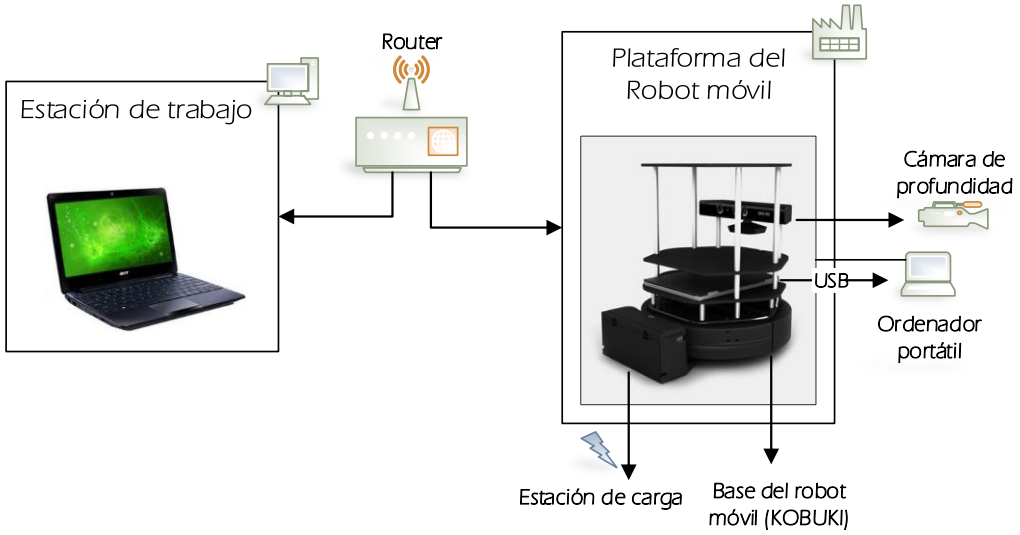


Figura 15.- Esquema del Hardware del robot móvil

En la configuración se añade una cámara de profundidad Microsoft Kinect la cual se ha de utilizar en aplicaciones especiales como los reconocimientos de objetos 3D y la evasión de obstáculos entre otras opciones. Se detalla los componentes de esta cámara en la *Figura 16.*

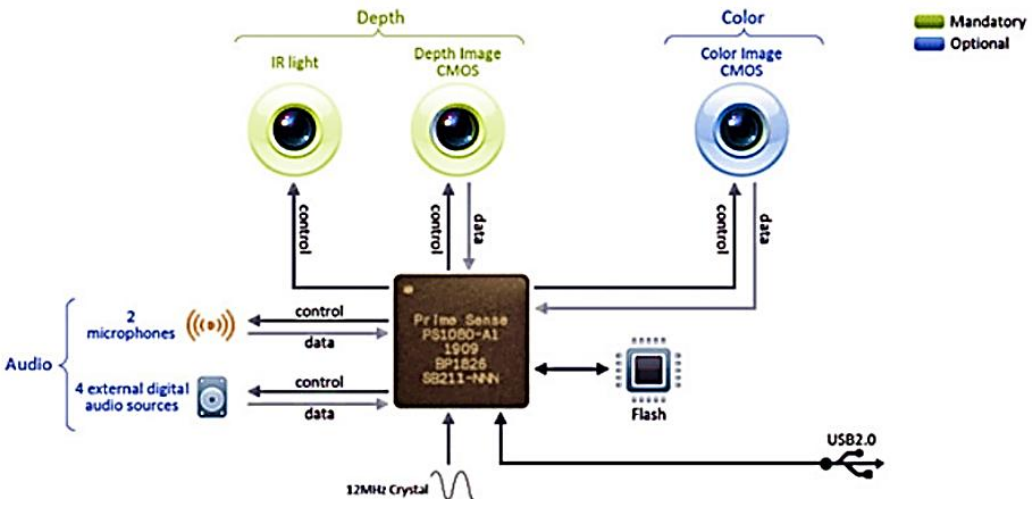


Figura 16.- Partes de la cámara Microsoft Kinect [17]

4.2.1 Comunicación Ordenadores

Para la navegación con tele-operacion o remota es necesario tomar en cuenta los siguientes aspectos.

Se necesita dos ordenadores ya que la netbook de TurtleBot o la base móvil viaja cerrada junto con él. Además, el netbook del robot móvil generalmente no tiene suficientes recursos para manejar los requisitos gráficos para RViz, que es la principal herramienta de visualización 3D para ROS. O a su vez dependiendo de la capacidad del ordenador que se le incorpore al robot móvil puede mostrar todo lo implementado por las herramientas grafica de ROS.

Generalmente un ordenador trabaja como parte de la base kobuki y el otro puede funcionar como una estación de trabajo con comunicación remota. Es decir los dos ordenadores estarán conectados uno como master y el otro como esclavo mediante comunicación ssh.

4.2.1.1 Comunicación SSH

SSH es un protocolo que facilita las conexiones entre dos sistemas. A diferencia de otros protocolos de comunicación, SSH encripta los datos que envía para hacer imposible que alguien pueda acceder a la información que se está enviando y recibiendo, con una encriptación robusta de 128 bits**[18].**

Para la comunicación mediante una red inalámbrica (wi-fi) se lo hace realiza utilizando un router al cual se conectaran los dos ordenadores para transferencia de datos. Para que se comuniquen simultáneamente cada uno de los nodos a utilizar. Y así suscriban y publiquen de forma libre a tópicos o nodos del otro ordenador y viceversa.

"Un sistema ROS en ejecución puede comprenden decenas, incluso cientos de nodos, repartidos entre múltiples máquinas. Dependiendo de cómo esté configurado el sistema, es posible que cualquier nodo necesite comunicarse con cualquier otro nodo, en cualquier momento" **[19]**.

4.3 Puesta en marcha

Conéctese el lado A del cable USB en su computadora portátil y el lado B en la base Kobuki. A continuación, conecte el lado hembra del divisor USB en el cable que sale del Kinect. Conecte el USB macho del divisor en el ordenador portátil y la otra mitad del divisor en el enchufe de 12V 1.5amp en la base de Kobuki. Finalmente, ubique el botón On / Off en el lado de la base Kobuki para encenderlo.

Una vez encendido en robot móvil se establece una serie de comandos para el funcionamiento. Los cuales han de generar el movimiento y las órdenes establecidas por el operario. Se tiene que tomar en cuenta que el nodo principal es el roscore. Se invoca en la consola de Ubuntu el nodo principal el cual se suscribe al tópico roslaunch. El cual da el preámbulo para generar el movimiento.

```
> ~$roslaunch turtlebot_bringup minimal.launch
```

4.3.1 Tele-operación

Si bien el robot Kobuki, tiene aplicaciones más detalladas como es realizar una movilidad autónoma mediante o con el uso de una cámara de profundidad. Empezaremos por la aplicación más básica que es la tele-operación. Es decir se lo maneja y ordena movimientos dirigidos por consola. Para generar el movimiento tenemos que subscribirnos a un nodo de tele-operación que se lo genera con el siguiente comando:

```
> ~$ roslaunch kobuki_keyop keyop.launch
```

Esté nos muestra en la consola de Ubuntu varios comandos a digitar para el movimiento y varios parámetros como la reducción e incremento de la velocidad del robot entre otros que se muestran en la *Figura 17*.

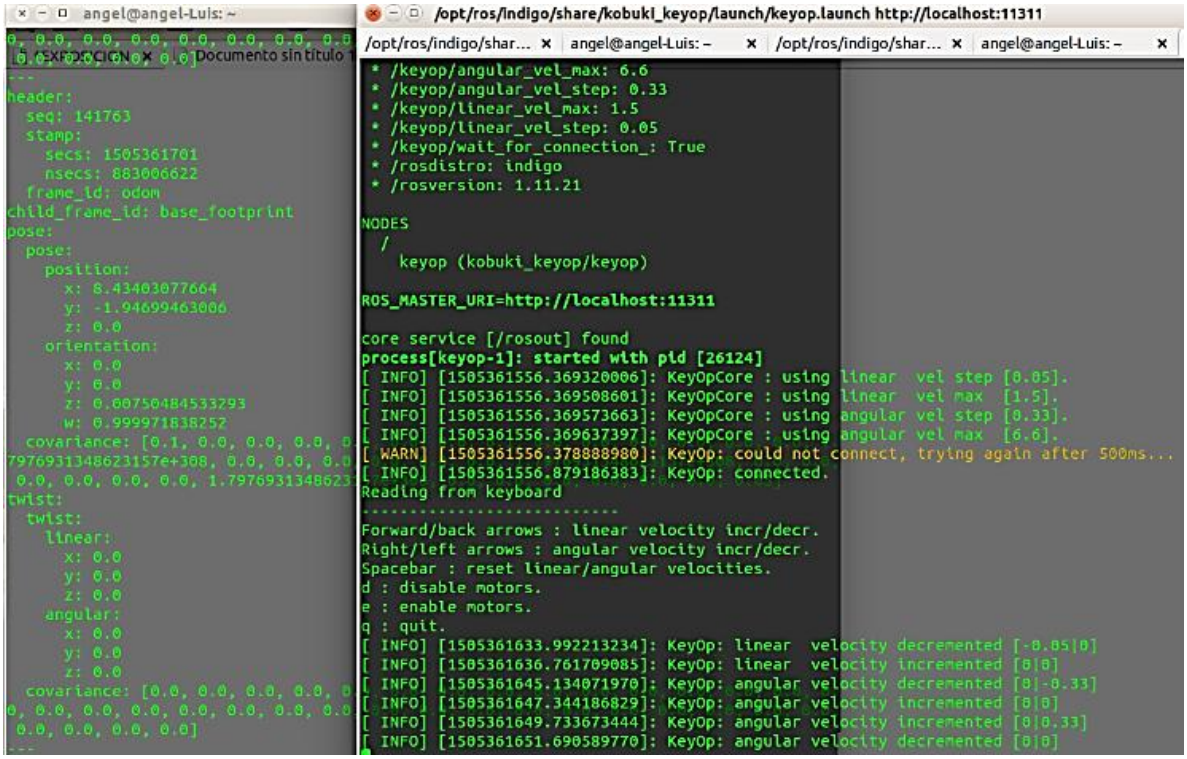


Figura 17.- Imagen generada para la tele-operación

4.4 Pruebas y análisis de resultados

Se implementó un meta sistema operativo en un robot móvil, y se demuestra mediante pruebas el avance tanto como el desarrollo del meta-sistema y aplicaciones utilizándolo. Pues así, se puede evaluar y realizar análisis del funcionamiento del robot en todas sus etapas. Como pruebas se realizara l correcto funcionamiento del meta-sistema operativo incorporado en el robot, tele-operación, odometría y navegación autónoma y mapeo simultáneo.

4.4.1 Prueba 1: Funcionalidad del meta-sistema operativo

El comando rqt_graph de ROS muestra la comunicación y funcionamiento de los nodos que se encuentran activos, la aplicación rqt_monitor muestra en una ventana los módulos activados en la puesta en marcha del robot móvil. Para demostración del funcionamiento de esta prueba se expone el diagrama obtenido en la *Figura 18*. Con estos diagramas se puede verificar correcto funcionamiento y nos muestran una comprobación fiable al momento de manipular el robot con sus diferentes aplicaciones. Para generar el estatus y la funcionalidad de los nodos se ha aplicado los siguientes comandos:

```
> ~$ rqt_graph
```

```
> ~$rosrunrqt_robot_monitor rqt_robot_monitor
```

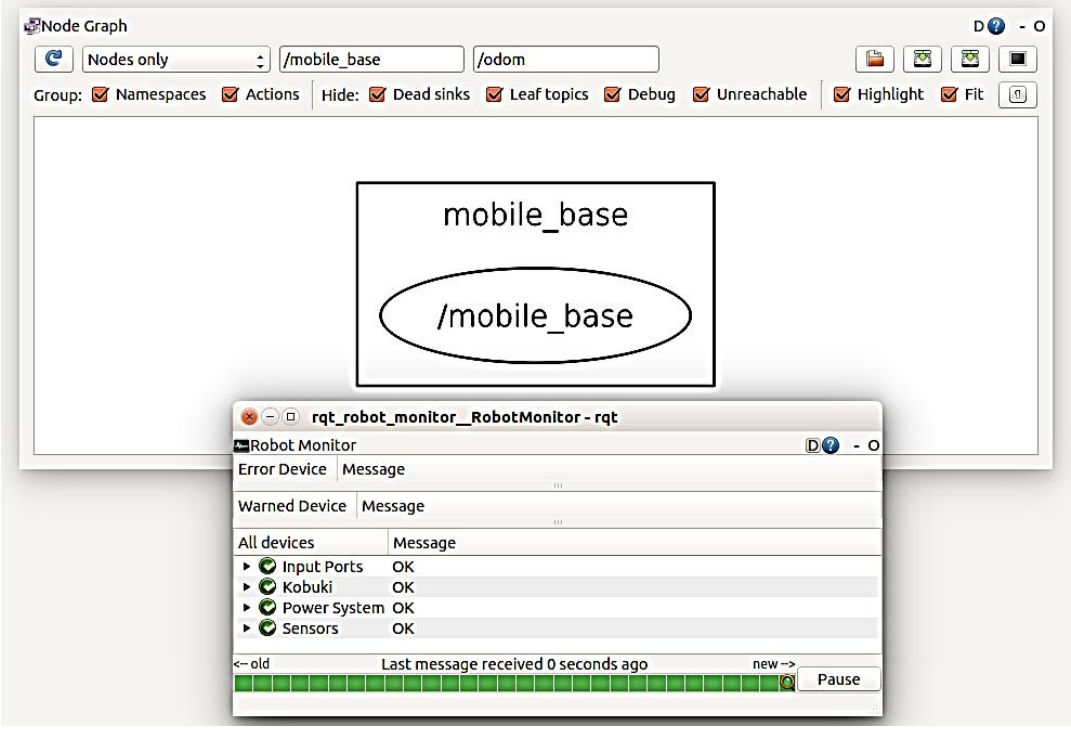


Figura 18.- Cuadros de diálogo generados por aplicaciones se ROS (rqt_graph y robot monitor)

El meta-sistema fue implementado adecuadamente. El soporte que ROS proporciona con la herramienta rqt facilita en cierto modo el análisis ya que se genera resultados de la efectividad de todos los componentes físicos y parámetros de software en el cuadro de dialogo mostrado en la *Figura 18* muestran los elementos conectados tales como sensores y actuadores. Mostrando la efectividad del sistema.

4.4.2 Prueba 2: Odometria

La odometría se basa en ecuaciones simples que se pueden implementar fácilmente y que utilizan datos de encoders situados en las ruedas del robot. Sin embargo, la odometría también está basada en la suposición de que las revoluciones de las ruedas pueden ser traducidas en un desplazamiento lineal relativo al suelo.

ROS muestra la odometria mediante un nodo. Esté se lo ha de llamar cuando el robot esté en funcionamiento con un comando lanzado en Ubuntu mediante consola.

```
> ~$ rostopic echo /odom
```

El correcto funcionamiento del nodo de odometria (/odom) se visualiza en consola ejecutando el comando ***rostopic echo /odom***. El mismo que muestra datos en coordenadas de x y z. Y mostrados únicamente en coordenadas numéricas (x,y) ya que el movimiento es lineal. Los datos fluctúan dependiendo de la velocidad aplicada por el operador y son adquiridos en un rango de milisegundos y mostrados en pantalla como se expone en la *Figura 19.*

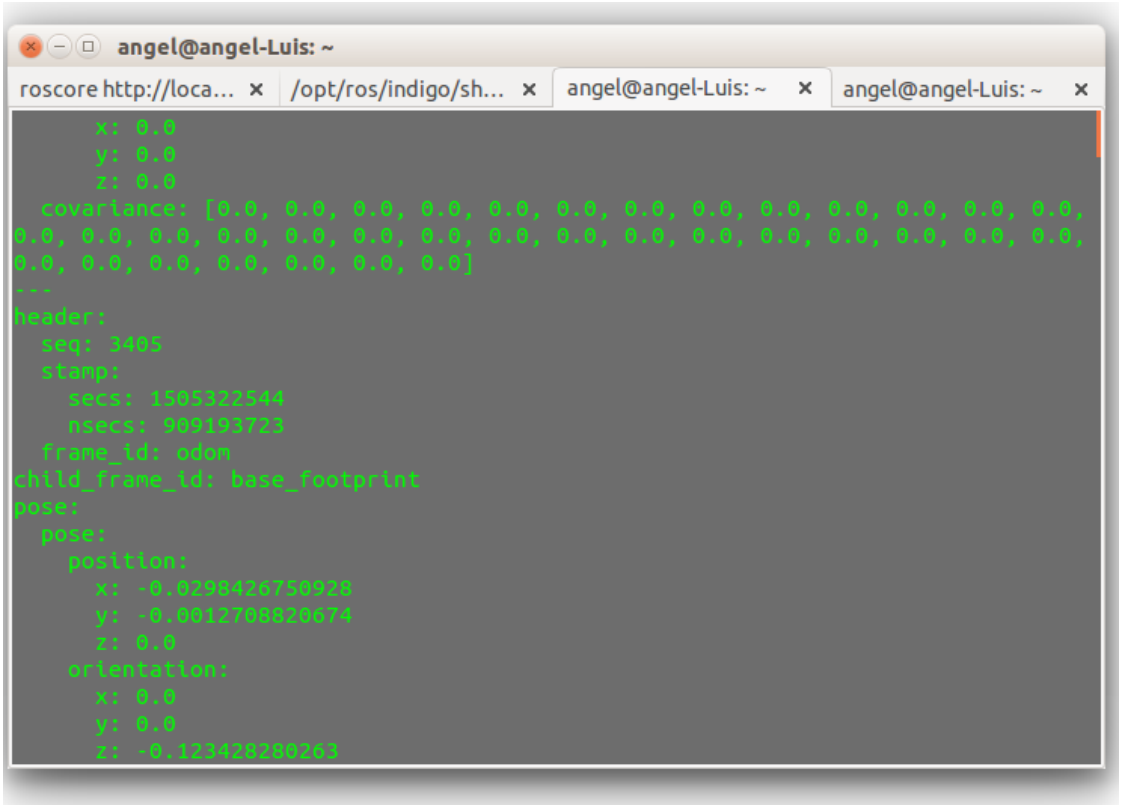


Figura 19. Datos de edometría del robot móvil.

La velocidad de tele operación da una variación de datos. En esta prueba el movimiento es únicamente de incremento o decremento y empieza en las coordenadas [0|0]. Estas coordenadas varían dependiendo del recorrido ya sea angular o lineal son utilizadas para análisis de ubicación en un ambiente desconocido. Estás se muestran en la *Figura 20.*

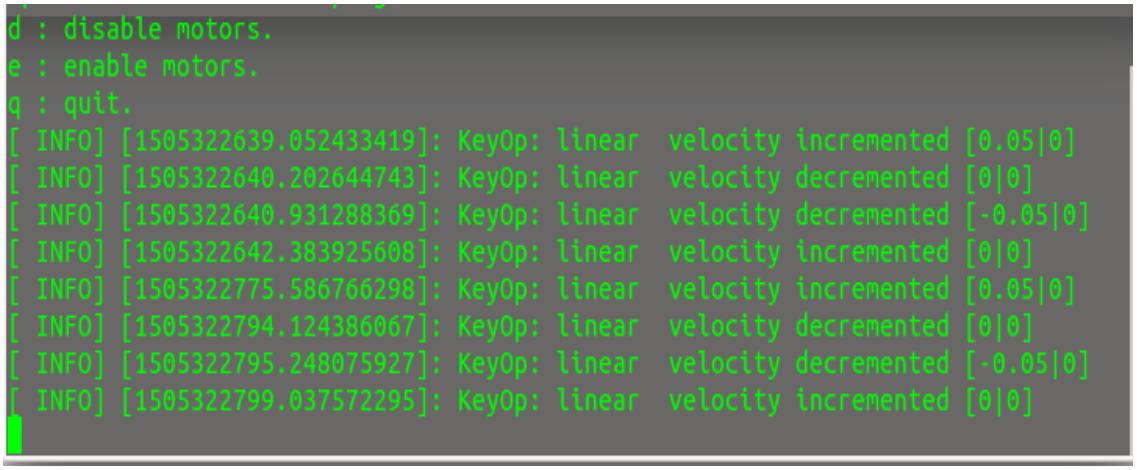


Figura 20. Velocidad en tele-operación.

4.4.3 Prueba 3: Test cámara de profundidad

Para realizar esta prueba se ha añadido una cama de profundidad Microsoft Kinect la cual consta de cámaras de profundidad y color para reconocimiento y evasión de obstáculos como aplicaciones prácticas.

Esta prueba se basa en probar la comunicación y el correcto funcionamiento de los lentes que esta cama posee. Para lo cual generaremos varios terminales en Ubuntu con la finalidad de aplicar varios comandos para establecer tanto la comunicación con el robot kobuki como la generación de imágenes. Es así que para empezar se ha de inicializar de manera normal y luego colocaremos el comando:

```
> ~$ roslaunch openni_launch openni.launch
```

Este comando se genera para la comunicación con los paquetes controladores de la cámara. Después se generara las imágenes de color y profundidad con los siguientes comandos respectivamente:

```
> ~$ rosrun image_view image_view image:=/camera/rgb/image_color
> ~$ rosrun image_view image_view image:=/camera/depth/image
```

En la *Figura 21* se muestra las imágenes obtenidas por los lentes de la cámara que se incorporó en el robot.

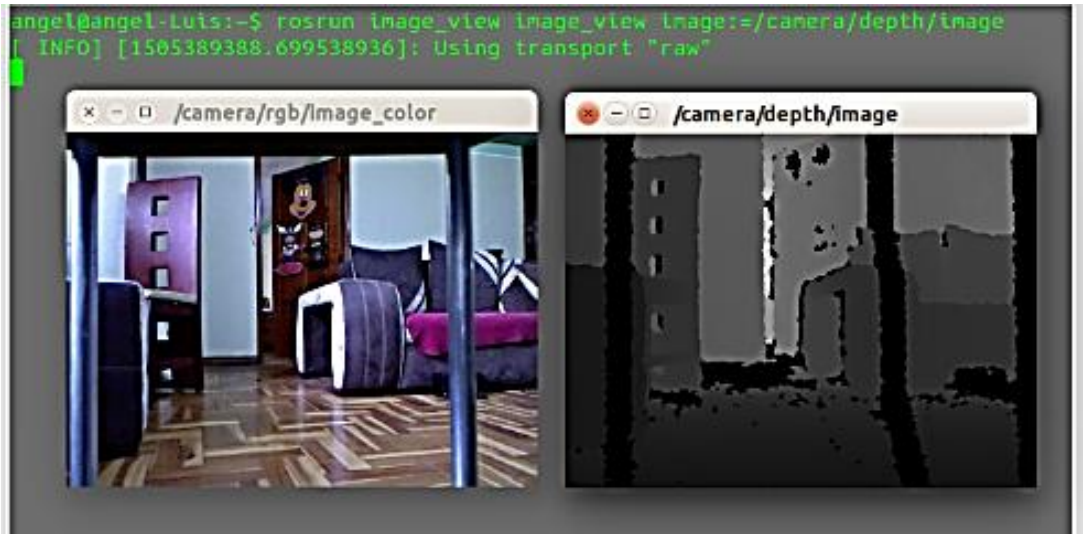


Figura 21.- Imágenes generadas por la cámara Kinect, color y profundidad respectivamente.

4.4.4 Prueba 4: Mapeo y movilidad Autónoma

Para la realización de esta prueba se ha de generar varios comandos adicionales y la utilización de módulos adjuntos a ROS como es Rviz. Primeramente verificaremos que se haya inicializado el robot y crearemos un mapa con el módulo Rviz que se encarga de comunicar el robot y la cámara. Este mapa se lo guarda, para posteriormente ser llamado. Primeramente se ha de poner en marcha el robot móvil para lo cual aplicamos el siguiente comando:

```
> ~$ roslaunch turtlebot_bringup minimal.launch
```

ROS puede trabajar en entornos desconocidos, comenzaremos con un mapa del entorno en el que estamos trabajando usando la tele-operación. Esto nos proporcionara de un mapa y podemos hacer referencia en otros scripts dando al robot una navegación completamente autónoma. Para lo cual introducimos le siguiente comando el cual virtualmente nos proporciona un espacio para generar nuestro mapa.

```
> ~$ roslaunch turtlebot_navigation gmapping_demo.launch
```

Se hará uso del módulo Rviz para crear el mapa de forma real ya que entrelaza el robot con la cámara de profundidad. Para lo cual aplicaremos el siguiente comando:

```
> ~$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Para lo movilidad se utiliza el comando de tele-operación que realiza el siguiente comando:

```
> ~$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

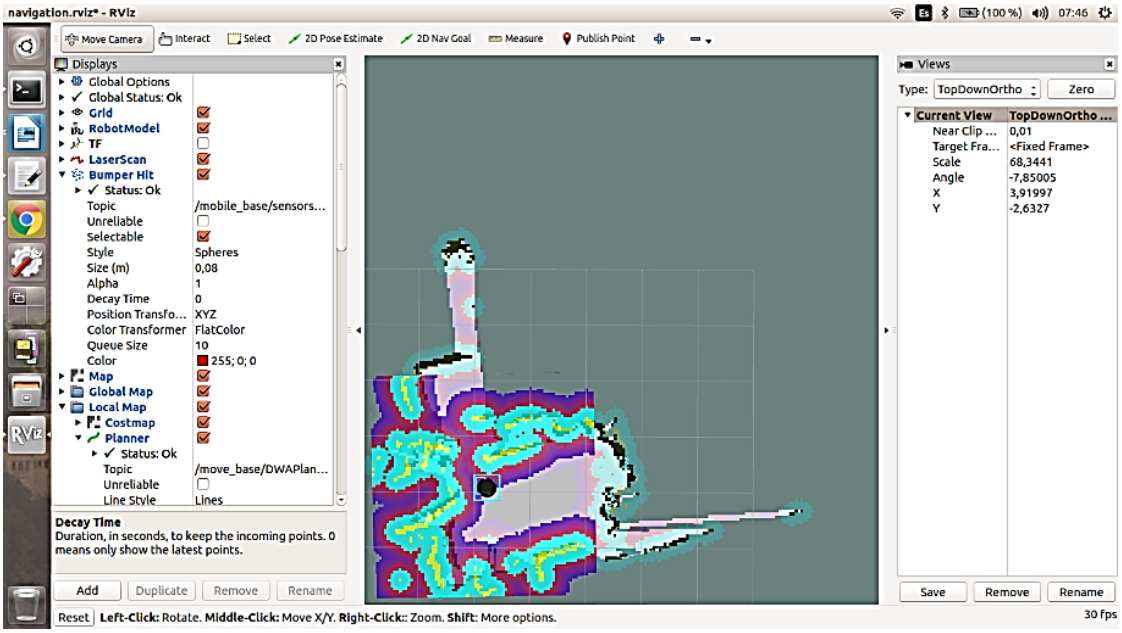


Figura 22.- Mapa generado para el movimiento autónomo

Como se puede visualizar en la *Figura 22* se genera un mapa mediante los movimientos y dirección que el operario crea conveniente. Se guardara el mapa generado en un espacio temporal por motivos de demostración, aplicando el siguiente comando:

```
> ~$ rosrun map_server map_saver -f /tmp/my_map ls /tmp/
```

Ahora ROS tiene almacenado un mapa en el cual con ROS tenemos la capacidad de mover al robot kobuki (o cualquier otro robot) de un lugar a otro, evitando tanto los obstáculos estáticos y dinámicos todos con unas pocas líneas de código. Con el siguiente comando se ha de llamar al mapa guardado en la dirección especificada anteriormente.

```
> ~$ roslaunchturtlebot_navigationamcl_demo.launch
map_file:=/tmp/my_map.yaml
```

Se ejecuta el comando para navegar por nuestro mapa con Rviz y navegaremos u ordenaremos al robot que se dirija de manera autónoma al lugar que deseemos.

```
> ~$ roslaunch turtlebot_rviz_launchers view_navigation.launch --screen
```

Para demostración de movimiento autónomo seleccionaremos "2D Pose Estimate" que se visualiza en el Rviz y se hace clic se mantiene presionada la ubicación donde el robot kobuki se encuentra según el mapa aproximadamente. Una flecha aparecerá debajo del puntero del ratón mientras mantiene pulsado el botón del ratón - utilícelo para estimar su orientación.

Después de configurar la posición estimada, seleccione "2D Nav Goal" y haga clic en la ubicación donde desea TurtleBot ir. Estimate". El robot Kobuki ahora debe conducir alrededor autónomamente basado en sus metas. Trate de especificar un objetivo y caminar delante de él para ver cómo reacciona a obstáculos dinámicos.

En la *figura 23* se muestra como en el mapa se ha seleccionado un área en la cual se desenvuelve el movimiento autónomo.

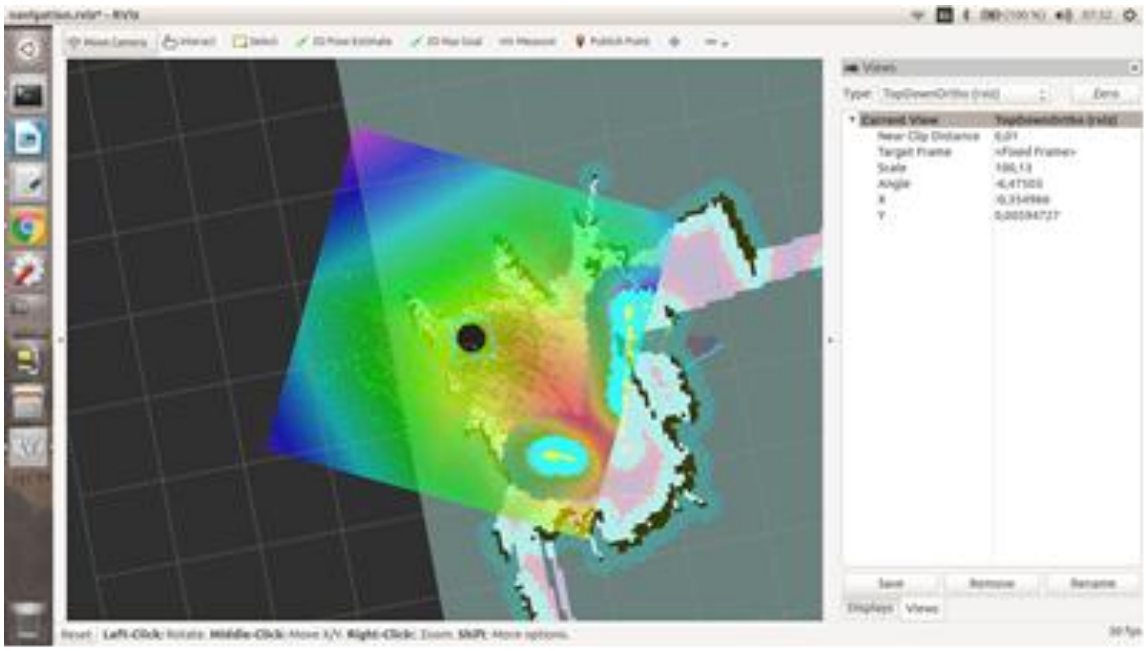


Figura 23.- Mapa para navegación autónoma.

CAPÍTULO V

CONCLUSIONES RECOMENDACIONES Y TRABAJOS FUTUROS

Este capítulo aborda conclusiones y recomendaciones que se han obtenido a partir de las demostraciones realizadas. También, se expone futuras investigaciones y aplicaciones que se le puede dar a trabajo realizado ya que es un prototipo presto a facilitar innumerables necesidades.

5.1 Conclusiones

Es necesario recalcar que el objetivo de desarrollar un meta-sistema operativo para un robot móvil usando ROS y que sea funcional para cualquier aplicación futura para navegación tele-operada y autónoma se ha cumplido.

El desarrollo del meta-sistema operativo en un robot que realice trayectorias autónomamente permite generar ideas para nuevas aplicaciones de alto rango en el ámbito de la robótica.

Los resultados que se obtienen con el uso cámara de profundidad 3D del sensor Kinect, controlados desde un mando a distancia y cuya comunicación se logra a través de una red Wi-Fi local, muestra que este método es propicio para que el usuario logre tener un panorama claro del entorno que está explorando.

Se ha integrado el meta-sistema operativo a la parte cinética del robot móvil. Y se ha establecido la plataforma y el meta-sistema operativo para la creación de nodos que controlen sensores y actuadores en tiempo real.

5.2 Recomendaciones y Trabajos Futuros

Para la implementación y desarrollo de trabajos futuros se recomienda realizar la tele-operación desde internet para así poder realizar un control desde partes geográficamente alejadas de la plataforma móvil.

La implementación de un meta-sistema operativo en la plataforma robótica móvil. Permite la incorporación de nuevos elementos para variar el funcionamiento. Así, en futuros trabajos el módulo de visión permitiría desarrollar odometría visual, mecanismo de percepción que reemplazaría a los encoders y giroscopios como método para determinar valores de desplazamiento del robot.

Este trabajo da bases para el desarrollo de aplicaciones de seguimiento de personas por identificación de movimiento, al usar el Kinect ya que este reconoce objetos y patrones de movimiento.

ANEXOS

Figura 24.- Robot Kobuki realizando pruebas de tele-operación

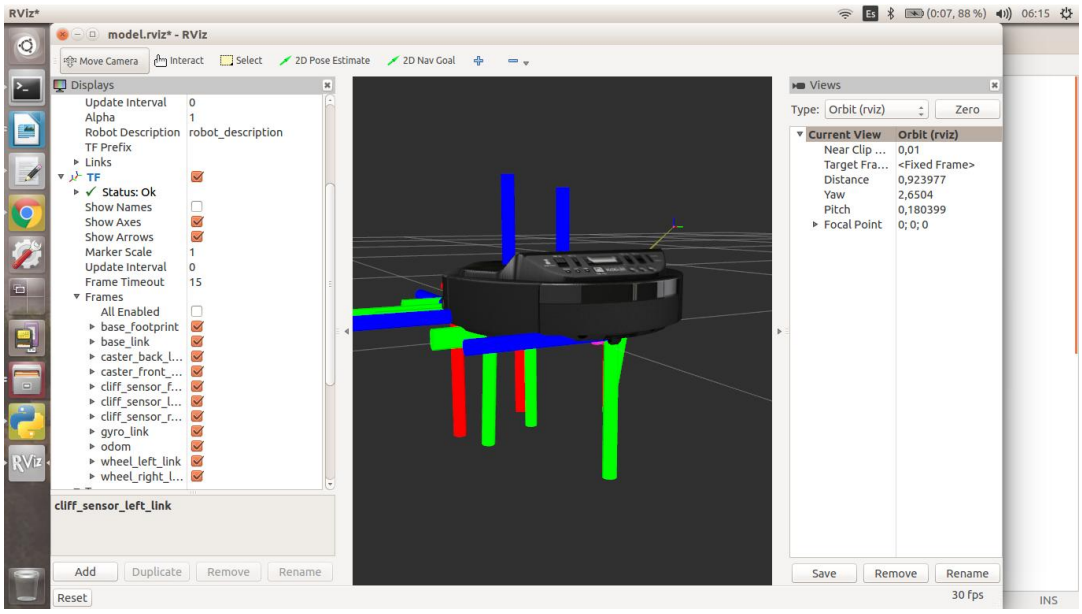


Figura 25.- Sensores activados virtualmente con la herramienta Rviz

BIBLIOGRAFÍA

[1] B. Siciliano y O. Khatib, Eds., *Springer handbook of robotics*. Berlin: Springer, 2008.

[2] Harold Díaz, «CD-5884.pdf». nov-2014.

[3] J. A. Gallardo Andrade y A. D. Gaona Román, «Diseño y construcción de una plataforma robótica móvil para interiores capaz de realizar SLAM (Simultaneous Localization and Mapping).», B.S. thesis, Universidad de las Fuerzas Armadas. Carrera de Ingeniería en Mecatrónica., 2015.

[4] «Manual de ROS». [En línea]. Disponible en: https://moodle2015-16.ua.es/moodle/mod/book/tool/print/index.php?id=82546. [Accedido: 29-jul-2017].

[5] María Utreras y Diana Decimavilla, «Resumen de tesis MUtreras y DDecimavilla, director de tesis Ph.D. Daniel Ochoa D. 17 octubre 2013.pdf», ESPOL, Guayaquil, 2013.

[6] D. C. Tumbaco Mendoza, Q. Zapata, y W. Enrique, «Artículo Científico-Diseño y construcción de un prototipo de robot Delta con implementación de un cortador lasér CNC utilizando la Plataforma Robotic Operating System (ROS) para la elaboración de artículos publicitarios.», 2014.

[7] «ROS | Erle Robotics Gitbook». [En línea]. Disponible en: https://erlerobotics.gitbooks.io/erlerobot/es/ros/ROS.html. [Accedido: 16-sep-2017].

[8] «ROS.org | About ROS». .

[9] «ROS.org | Powering the world's robots». [En línea]. Disponible en: http://www.ros.org/. [Accedido: 21-mar-2017].

[10] «ROS Introduction ES | Erle Robotics». .

[11] E. F. Cepeda Castellanos y others, «Desarrollo de móvil teledirigido basado en ROS», B.S. thesis, Universidad Militar Nueva Granada, 2015.

[12] «About | KOBUKI». [En línea]. Disponible en: http://kobuki.yujinrobot.com/about2/. [Accedido: 16-sep-2017].

[13] «Connectors – KOBUKI». [En línea]. Disponible en: https://kobuki.yujinrobot.com/wiki/connectors/. [Accedido: 21-mar-2017].

[14] «Kobuki», *Yujinrobot | Robot Vacuums*. .

[15] «Online User Guide – KOBUKI». [En línea]. Disponible en: http://kobuki.yujinrobot.com/wiki/online-user-guide/. [Accedido: 16-sep-2017].

[16] «Instructions | DIY Autonomous Mobile Robot | Hackaday.io». [En línea]. Disponible en: https://hackaday.io/project/8588/instructions. [Accedido: 16-sep-2017].

[17] A. M. Ronchetti Marco, «Avancini_Mattia_138793_Thesis.pdf». Diciembre-2011.

[18] «Protocolo SSH». [En línea]. Disponible en: http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ssh.html. [Accedido: 16-sep-2017].

[19] «ROS/NetworkSetup - ROS Wiki». [En línea]. Disponible en: http://wiki.ros.org/ROS/NetworkSetup. [Accedido: 14-sep-2017].