



UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE ELECTROTECNIA Y COMPUTACIÓN

INFORME DE TRABAJO MONOGRÁFICO

**SISTEMA OPERATIVO PARA ROBOTS(ROS): APLICACIÓN EN EL
DESARROLLO DE UN LABORATORIO VIRTUAL PARA EL ESTUDIO
DE LOS FUNDAMENTOS DE LA ROBÓTICA INDUSTRIAL**

Presentado por: Br. Yeser Alfredo Morales Calero

Tutor: MSc. Alejandro Alberto Méndez Talavera

Diciembre 12, 2018

DEDICATORIA

RESUMEN

ABSTRACT

INDICE

INTRODUCCIÓN.....	1
OBJETIVOS	4
JUSTIFICACIÓN	5
CAPÍTULO I: MARCO TEÓRICO	7
1.1 Tipos de laboratorios	7
1.1.1 Laboratorio Físico	7
1.1.2 Laboratorio Virtual	8
1.1.3 Laboratorio Remoto online	8
1.1.4 Realidad virtual 3D	8
1.2 Sistema operativo para robot (ROS)	10
1.2.1 Conceptos básicos de ROS	10
1.2.1.1 Nodos (Nodes)	11
1.2.1.2 Mensajes (Messages).....	11
1.2.1.3 Tópico (Topic)	11
1.2.1.4 Maestro (master)	11
1.2.1.5 Servicios (services)	11
1.2.1.6 Bolsas (bags)	11
1.2.2 Librerías de ROS.....	14
1.2.3 Herramientas de visualización de datos en ROS	14
1.2.3.1 RVIZ.....	14
1.2.3.2 RQT	15
1.2.3.3 MoveIt!.....	16
1.2.4 Lenguajes de programación soportados por ROS	16
1.3 Laboratorio virtual para robótica industrial	18
1.3.1 Interfaz gráfica de usuario (GUI).....	19
1.3.2 Fundamentos de la robótica industrial.....	20
1.3.2.1 Morfología del robot	20
1.3.2.2 Herramientas matemáticas.....	21
1.3.2.3 Cinemática directa e inversa.....	21
1.3.2.4 Programación y simulación del robot	22
1.4 Uso del laboratorio virtual de robótica industrial	25
2.1 Diseño del LVR	26
2.2 Recursos de software	27
2.2.1 Sistema operativo Ubuntu.....	28

2.2.2	ROS	29
2.2.2.1	Instalación y Puesta en marcha de ROS.....	29
2.2.2.2	Estructura de ROS	32
2.2.2.3	Compilación de Paquetes de ROS.	35
2.2.3	IDE QT-Creator.	37
2.2.3.1	Plugin de ROS para Qt-Creator	38
2.2.3.2	Widgets QT	39
2.2.3.3	Widgets QML	39
2.2.4	Librería Orococos.....	39
2.3	Modelos de los robots.....	40
2.3.1	Modelos de los Fabricantes de robots.....	41
2.3.2	Creado por el usuario	41
2.3.2.1	URDF	41
2.3.3	Soporte de ROS – Industrial	42
2.4	Programación de los Robots.	44
2.4.1	Diseño de la interfaz interprete de comandos.	44
CAPÍTULO III: ANÁLISIS DE RESULTADOS		48
CAPÍTULO IV: CONCLUSIONES Y RECOMENDACIONES		49
BIBLIOGRAFÍA.....		50
ANEXOS		53
A: Instalación de Recursos de Software para el Laboratorio Virtual.		54
A.1	Instalación de UBUNTU 16.04 Xenial	54
A.2	Instalación de ROS Kinetic.....	56
A.3	Espacio de trabajo Catkin.....	57
A.4	Instalación de QT – Creator.	58
A.5	Instalación del Plugin ROS para QT – Creator.	58

Lista de Figuras

Capítulo I

Figura 1.1: Esquema de Funcionamiento de la Arquitectura de ROS	12
Figura 1.2: Comunicación utilizando Tópicos. En Topics por ROBOTIS Co., Ltd 2017.....	13
Figura 1.3: Nodo robot_localization recibiendo tópicos de un robot físico y mostrando su localización en el ambiente virtual. En robot_localization por Charles River Analytics Inc 2017.	13
Figura 1.4: RVIZ y visualización de modelo de robot y datos provenientes de mensajes 3D.	15
Figura 1.5: Gráfico de línea XY de datos provenientes de nodos	16
Figura 1.6: Modelo de robot, sistemas de referencia. En “Robótica” (P.6), por J. J. Craig, 2006	21
Figura 1.7: Cinemática directa e inversa En “Fundamentos de robótica” (P.119), por A. Barrientos et al, 2007	22
Figura 1.8: Modelo de utilización del LVR.	25

Capítulo II

Figura 2.1: Modelo de laboratorio virtual físico.....	26
Figura 2.2: Modelo del laboratorio Virtual de Robótica Industrial.	27
Figura 2.3: Logo del sistema operativo para robots.....	29
<i>Figura 2.4: Logo de la distribución de ROS, Kinetic Kame.</i>	<i>30</i>
<i>Figura 2.5: roscore servidor maestro</i>	<i>31</i>
<i>Figura 2.6: Organización del software en ROS.....</i>	<i>32</i>
<i>Figura 2.7: Organización de un paquete de software en ROS.....</i>	<i>33</i>
Figura 2.8: Espacio de Trabajo Con Catkin.....	34
Figura 2.9: Archivo CMakeLists con parámetros para compilación.....	35
Figura 2.101: Archivo package.xml con etiquetas de referencia de paquetes.	36
Figura 2.11: Logo de QT-Creator.....	37
Figura 2.12: Recursos de QT para el Desarrollador.	37
Figura 2.13: Captura de pantalla de Software ABB RobotStudio.....	41
Figura 2.14: Representación de Links y Joints en un URDF.	42
Figura 2.15: Estructura del lenguaje de marcado URDF.....	42
Figura 2.16: Modelos de CAD en 3D	44

Anexos

Anexo A

Figura A.1: IDE QT-Creator Activación del Plugin Catkin, para el espacio de Trabajo.	59
Figura A.2: Ubicación del espacio de Trabajo.	60
Figura A.3: Espacio de trabajo Creado.	60

Lista de Tablas

Capítulo I

Tabla 1.1: Requerimientos de la fuente de alimentación a usar	11
----------------------------------------------------------------------------	-----------

Capitulo II

Tabla 2.1: Comandos ROS	30
Tabla 2.2: Estructura de ROS	32
Tabla 2.3: Contenido de un Package	33
Tabla 2 4: Comandos Catkin.	34

Anexos

Anexo A

Tabla A.1: Instalación de UBUNTU 16.04 Xenial.....	54
Tabla A.2: Instalación de ROS Kinetic	56
Tabla A. 3 Creación del Espacio de Trabajo Catkin.....	57
Tabla A.4: Instalación del IDE QT- Creator.....	58
Tabla A.5: Instalación de Plugin ROS QT.	58

INTRODUCCIÓN

La robótica es uno de los campos de la tecnología que se desarrolla velozmente y encuentra cada día nuevas áreas de aplicación de tal forma que la clasificación de los robots debe ser ajustada constantemente. En la actualidad encontramos aplicaciones de la robótica en el campo personal y doméstico, servicios profesionales, investigación y desarrollo. Unos de los primeros campos donde los robots entraron en acción fue la industria en la cual destaca el uso de los mismos en el sector automotriz (soldadura y/o manejo de materiales). A pesar de que el uso de los robots en la industria data de la década de los 60s, en Nicaragua el uso de los mismos es relativamente bajo. El hecho mencionado tiene un impacto negativo en el desarrollo de la industria nicaragüense. Son pocas las industrias que poseen robots industriales y, de igual forma, son pocas las instituciones de educación superior que cuentan con la infraestructura adecuada, y el recurso humano, para la enseñanza e investigación de la robótica. Una de las razones es que un laboratorio físico, para experimentar en los diferentes aspectos asociados con los robots industriales, requiere de una inversión relativamente alta dado que es necesario garantizar el espacio físico, adquirir los robots y su mantenimiento.

La falta de infraestructura para el desarrollo e investigación de la robótica ha sido experimentada en la mayoría de las instituciones de educación superior latinoamericana muchas de las cuales han recurrido a la utilización laboratorios virtuales, diseñados e implementados por dichas instituciones o adquiridos en el mercado internacional. MATLAB, software ampliamente utilizado en el campo de las ingenierías, cuenta con un toolbox para robótica (Corke, 1996) mediante el cual se puede modelar y simular el comportamiento de un manipulador, sin embargo, el mismo tiene limitaciones. Simscape Multibody (antes llamado SimMechanics), de MATLAB y Simulink, permite la simulación de sistemas mecánicos en 3D, tales como robots. El software mencionado requiere de una licencia y el costo de la misma aumenta en dependencia del número de usuarios. También se puede utilizar el software de programación de robots, versión evaluación, de algunos fabricantes, tales como Robot Studio de ABB. Dicho

software, además de limitaciones temporales, solo puede ser utilizado en los robots fabricados por la empresa. Para trabajar con diferentes robots, es necesario adquirir y aprender los lenguajes de programación de cada uno de ellos.

En la actualidad se busca estandarizar el trabajo con los robots y una de las herramientas disponibles para tal fin es el middleware “Sistema Operativo para Robot (ROS, por sus siglas en inglés)” el cual se ha convertido en el estándar de facto a nivel industrial tanto así que ya existe una versión denominada ROS Industrial (2017). ROS (s.f.) es un middleware, open-source, para el desarrollo a gran escala de complejos sistemas robóticos. Se trata de una colección de herramientas, librerías y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento complejo y robusto en una amplia variedad de plataformas robóticas.

Para dar a conocer ROS y mostrar la importancia de este, para el desarrollo de la robótica, se desarrollo un laboratorio virtual de robótica (LVR) para el estudio de los fundamentos de la robótica industrial.

Para el desarrollo del laboratorio, además de ROS, se utilizaron herramientas de simulación tales como GAZEBO, visualizadores como RVIZ y RQT. La integración de los diferentes nodos se logró utilizando el lenguaje de programación C++, para la modelación de los robots se utilizó el formato de descripción unificado de robots (URDF, por sus siglas en inglés), y para la creación de ambientes y escenarios fue empleado el formato de descripción de la simulación (SDF, por sus siglas en inglés).

El laboratorio virtual cuenta con los recursos necesarios para que los estudiantes estudien y realicen prácticas relacionadas a la morfología de los robots, herramientas matemáticas necesarias para el estudio de los robots, cinemática de los robots y programación de manipuladores industriales. Un elemento de mucha importancia en el LVR, es el simulador que incorporará el cual permitirá manipular el modelo de un robot en un escenario particular, en respuesta a un programa utilizando instrucciones parecidas a las que incorporan los lenguajes de programación de fabricantes tales como ABB.

El laboratorio virtual de robótica cuenta con un manual en el cual es presentada información de los recursos disponibles en el LVR, de forma tal que los docentes puedan diseñar prácticas que permitan a los estudiantes afianzar los fundamentos de la robótica industrial, e información general sobre ROS y herramientas asociadas, a un nivel tal que el usuario podrá implementar un ambiente de desarrollo básico. El profesor podrá elaborar prácticas de laboratorio relacionadas con los cuatro aspectos mencionados utilizando para ello los modelos de robots presentes en el LVR. Por ejemplo, el profesor podrá diseñar una práctica para que los estudiantes apliquen el procedimiento de Denavit y Hartenberg para determinar la matriz de transformación homogénea. En su versión inicial, el LVR cuenta con dos guías para realizar algunas actividades que permitan a los futuros usuarios apropiarse del procedimiento necesario para utilizar efectivamente los recursos del LVR.

En el presente documento son presentados los principales aspectos relacionados con el desarrollo del laboratorio virtual, así como los resultados obtenidos.

OBJETIVOS

Objetivo general

- Desarrollar un laboratorio virtual para el estudio de los fundamentos de la robótica industrial utilizando, para su implementación, el sistema operativo para robots (ROS).

Objetivos específicos

- Identificar las principales características/recursos/estructura de ROS enfatizando aquellas aplicables directamente a la robótica industrial.
- Identificar las herramientas de software requeridas, además de ROS, para el desarrollo del laboratorio virtual para el estudio de los fundamentos de la robótica industrial.
- Desarrollar la interfaz gráfica de usuario (GUI) del laboratorio virtual, según los requerimientos establecidos.
- Verificar la efectividad de los recursos del laboratorio virtual mediante la realización de dos prácticas de laboratorio.
- Elaborar un manual para el instructor donde se describa los recursos, y las características, del laboratorio virtual

JUSTIFICACIÓN

El campo de la robótica ha tenido, principalmente en las últimas décadas, un desarrollo vertiginoso y está impactando de forma positiva, principalmente en países desarrollados, en áreas como la medicina, la industria, y la investigación, entre otras. En Nicaragua, país en vías de desarrollo, el uso de la robótica es relativamente bajo, sin embargo, dicha tecnología tendrá mayor presencia en el futuro cercano y por consiguiente es necesario contar con profesionales con las competencias requeridas para realizar efectiva y eficientemente las diferentes tareas asociadas con la aplicación, el desarrollo y la investigación de la robótica.

La formación de profesionales en el campo de la robótica requiere de recursos humanos e infraestructura apropiada para tal fin (Jara, 2011a). En Nicaragua pocas instituciones de educación superior cuentan con recursos básicos mínimos para la formación de los estudiantes en el campo de la robótica (Universidad Tecnológica La Salle, 2017), siendo uno de los principales obstáculos para tener mejores condiciones la imposibilidad de garantizar la alta inversión requerida para la adquisición de los robots, software, accesorios asociados, instalación y mantenimiento de acuerdo a un estudio de Castellanos, & Martínez (2010). Compañías como FESTO ofrecen Kits denominados modulares, para prácticas de un proceso automatizado utilizando un robot Mitsubishi, que cuesta alrededor de 41,500 euros.

En el campo industrial, muchos fabricantes ofrecen manipuladores industriales los cuales tienen precios altos y, además, cada uno ofrece su propio software para la programación de los robots (Owen, 2016). Lo anterior significa que los estudiantes o especialistas en robótica industrial tendrían que aprender a utilizar el software correspondiente al manipulador en turno lo cual es imposible lograr si no se cuenta con la infraestructura adecuada. Se han realizado intentos para estandarizar los aspectos relacionados con la programación de los robots, hacerla independiente del fabricante, y uno de los resultados más destacado en la última década es el middleware “Sistema Operativo para Robot” (ROS, por sus siglas en inglés). En

la actualidad es considerado el estándar de facto para la aplicación, desarrollo e investigación de la robótica. Una muestra de la importancia de ROS es que en el campo industrial crearon el ROS-Industrial.

El desarrollo de un laboratorio virtual en robótica contribuiría en la mejora de las condiciones requeridas para el estudio de los fundamentos básicos de esta ya que brindaría a los estudiantes, y docentes, una herramienta que podría ser utilizada en cualquier lugar y en cualquier momento.

El laboratorio virtual sería desarrollado tomando como elemento principal el middleware ROS lo cual, entre otras cosas, permitiría a los estudiantes experimentar con modelos de robots de diferentes fabricantes tales como ABB, KUKA, FANUC, entre otros.

La inversión en el desarrollo e implementación del laboratorio virtual siempre será inferior a la requerida para tener un laboratorio físico ya que la mayoría de las herramientas utilizadas, tales como C++, ROS, GAZEBO, Qt creator son de código abierto (Open Source).

Los principales beneficiarios de los resultados de este proyecto son los docentes y los estudiantes. Los primeros dispondrán de una herramienta a partir de las cuales podrán diseñar experimentos, con su guía apropiada, relacionados con los fundamentos básicos de los robots industriales. Los estudiantes, por su parte, contarán con una herramienta flexible, que podrán utilizar en cualquier momento y en cualquier lugar, la cual les posibilitará verificar los fundamentos teóricos relacionados con los robots industriales abordados en programas de asignatura relacionados así programar los robots para que realicen tareas básicas.

El laboratorio virtual de robótica desarrollado debe ser considerado como el primer paso hacia un laboratorio virtual con mayores recursos en el cual se pueda experimentar con diversos tipos de robots, no solo los antropomórficos. Lo anterior contribuiría considerablemente en el estudio del funcionamiento de los robots y sus aplicaciones y presentaría un espacio para el desarrollo e investigación en el campo de la robótica.

CAPÍTULO I: MARCO TEÓRICO

Las instituciones académicas, principalmente las de educación superior (IES), buscan constantemente mejorar la calidad de la educación y una forma de hacerlo es integrando nuevas técnicas y herramientas orientadas a mejorar los resultados del proceso enseñanza-aprendizaje. En la actualidad, en muchas IES, a la tradicional clase presencial se suman otras formas de enseñanza-aprendizaje las cuales se fundamentan en las TIC, permitiendo de esta manera nuevas formas de enseñar, aprender, generar y compartir conocimiento.

Un recurso de mucho impacto en la formación de los futuros profesionales y cuya presencia crece día a día en los procesos de enseñanza aprendizaje es el laboratorio virtual (LV), el cual puede presentar altos niveles de flexibilidad y la inversión es muy inferior a la requerida para instalar un laboratorio real.

Uno de los campos en el cual el desarrollo y utilización de un laboratorio virtual reviste mucha importancia, dada la inversión alta requerida para tener un laboratorio físico, es el de la robótica en general y en particular, la robótica industrial.

En la introducción del presente documento se estableció que el trabajo de monografía propuesto pretende dar a conocer el middleware ROS, su importancia en el desarrollo de la robótica industrial, y dar muestra de sus posibilidades mediante el diseño y construcción de un laboratorio virtual que sirva de base para el estudio de los fundamentos de los manipuladores industriales tipo serie.

En los siguientes apartados se presentan los elementos teóricos-tecnológicos requeridos para el desarrollo del laboratorio virtual.

1.1 Tipos de laboratorios

La experimentación de un fenómeno físico, o mediante la simulación, requiere de entornos de trabajos que reúnan los equipos o herramientas necesarias para el desarrollo efectivo de una práctica. En los ambientes académicos podemos encontrar, en la actualidad, diferentes tipos de laboratorios

1.1.1 Laboratorio Físico

Es ampliamente utilizado en universidades con modelos clásicos de enseñanza convirtiendo estos sitios de trabajo como el único sitio concebido para elaborar un experimento el cual involucra la presencia física tanto del tutor como del alumno. Cabe destacar que la interacción directa con los equipos apropiados de un laboratorio físico aporta una experiencia difícil de igualar debido a que los alumnos perciben los resultados de carácter palpable entrando en juego los cinco sentidos (Vista, tacto, audición, olfato e incluso, a veces el gusto) Calvo, (2009a).

1.1.2 Laboratorio Virtual

Es una alternativa al laboratorio físico y utiliza recursos computacionales haciendo uso de modelos matemáticos y recursos de visualización como modelos CAD y animaciones gráficas. Se pretende, mediante de este tipo de laboratorio, que el estudiante pueda estudiar el comportamiento de la realidad mediante el uso de modelos, sencillos o complejos, de la misma. Los recursos de un laboratorio virtual pueden ser ampliados en el tiempo permitiendo la realización de prácticas asociadas a temas de mayor complejidad.

1.1.3 Laboratorio Remoto online

Nacen bajo la necesidad de dar acceso a los estudiantes a un espacio de trabajo, de forma online, que combina los recursos de un laboratorio físico con recursos de software. El estudiante accede al laboratorio por medio de una página web pudiendo de esta forma controlar los recursos disponibles y llevar a cabo el experimento de interés. Su aprovechamiento está basado en la accesibilidad por parte de un usuario en horarios flexibles.

1.1.4 Realidad virtual 3D

Es la integración de recursos de hardware y software para acercar aún más al usuario a la experimentación de una teoría estimulando los sentidos tanto de la vista al exponerlos a entornos virtuales 3D y audición bajo altavoces con sonido envolvente.

En la actualidad se puede encontrar diferentes recursos para la enseñanza de la robótica Industrial tales como kits de robótica industrial, laboratorios virtuales de robótica de categoría libre o de pago para ejecutar cierta práctica. El desarrollo de un laboratorio remoto requiere contar con acceso a un laboratorio físico y enlazarlo a un servidor para que el estudiante pueda acceder vía remota garantizándole una plataforma amigable y multiplataforma como pueden ser los navegadores WEB. Candelas, F. et al. (2004) analizan las ventajas del uso de un laboratorio remoto ROBOLAB, proyecto desarrollado por ellos mismos bajo el nombre de grupo de investigación AUROVA. El proyecto utiliza herramientas gráficas para modelado y visualización de objetos 3D de un brazo robótico que coincide con el mismo modelo de un robot físico, permitiendo al estudiante realizar sus practica al observar y configurar el modelo virtual y una vez alcanzado un nivel de aprendizaje aceptable, proceder a interactuar con el robot físico utilizando la misma plataforma web. Ellos concluyen que al utilizar el laboratorio Remoto ROBOLAB *“La mayoría de alumnos prefieren disponer de un laboratorio en la universidad dónde trabajar con la ayuda de los compañeros y el apoyo didáctico del profesor, pero también hay muchos alumnos que reciben con agrado la opción*

de un laboratorio virtual que les ofrezca unos horarios flexibles en los que realizar los experimentos”. (Candelas, et al, 2004b).

En resumen, un laboratorio virtual es un elemento importante en el proceso enseñanza-aprendizaje ya que destacan, entre otras, las siguientes ventajas según Calvo, et al (2009b):

1. El estudiante se familiariza con el experimento evitando acudir al aula sin conocimiento previo.
2. Comparación del comportamiento de modelos matemáticos ante una simulación permitiendo extraer sus propias conclusiones de cierta práctica.
3. Manejo de herramientas informáticas contemporáneas para la formación integral de un estudiante.
4. Repetitividad de los experimentos realizados por el estudiante que permitirá reproducir cuantas veces desee hasta consolidar el conocimiento.
5. Disminución de riesgos y accidentes que pueden ocasionar una mala práctica o configuración de un equipo físico.
6. Multiplicidad de experimentos simultáneos realizados ya que cada estudiante podrá ejecutar la práctica indicada en su computador asignado, además de esta forma se favorecen los procesos colaborativos como el de “Lluvia de ideas” al opinar cada alumno sobre su percepción adquirida al ejecutar la práctica.

Considerando lo anterior, se decidió, para dar a conocer la importancia y posibilidades de ROS, desarrollar un laboratorio virtual el cual tendrá a ROS como elemento integrador. Es importante destacar la inversión relativamente baja requerida para implementar el laboratorio virtual, la posibilidad de ampliación del mismo y la posibilidad que el mismo ofrecerá a los docentes para diseñar nuevas prácticas a partir de los recursos disponibles.

1.2 Sistema operativo para robot (ROS)

El sistema operativo para robot (ROS), elemento fundamental del presente trabajo, es un middleware ampliamente utilizado en el mundo creciente de la robótica. Originalmente fue desarrollado en el 2007 en los laboratorios de Inteligencia Artificial de Stanford y en el 2008, cedieron el derecho de desarrollo al instituto de investigación de robótica Willow Garage (2010), donde la filosofía es proporcionar bibliotecas y herramientas libres para ayudar a los desarrolladores de software a crear aplicaciones para robots

En [este trabajo monográfico](#), se promueve la importancia de ROS en el campo de la robótica y se hace mediante el diseño y construcción, utilizando los recursos proporcionados por este, de un Laboratorio Virtual de Robótica, en el cual será posible realizar experimentos sobre los fundamentos de la robótica industrial específicamente de los manipuladores industriales tipo serie.

Una característica importante de ROS es su código abierto (Open Source) lo que ha llevado al desarrollo, por muchos colaboradores a nivel mundial, de una amplia colección de herramientas, librerías y convenciones. Los recursos que ofrece ROS permiten el desarrollo, a gran escala, de sistemas robóticos complejos, tanto físicos como simulados.

Los desarrolladores de ROS han establecido un conjunto de convenciones y buenas prácticas en el desarrollo de software de forma tal que se fomente la reutilización de código para robots, trabajando sobre una arquitectura robótica totalmente sólida y funcional. De esta forma se garantiza, entre otras cosas, que el proyecto no sea fallido y que sus resultados satisfagan la calidad requerida en este tipo de trabajos.

1.2.1 Conceptos básicos de ROS

El sistema operativo para robot (ROS) fue diseñado bajo una estructura distribuida y modular con el propósito de que los usuarios puedan usar los recursos requeridos según la aplicación a desarrollar. Es decir, el usuario puede seleccionar los recursos de software necesarios para implementar la solución.

En su operación, ROS presenta una red de **procesos** que se ejecutan simultáneamente en una estructura **peer-to-peer** donde cada nodo o elemento del sistema puede actuar al mismo tiempo como cliente y como servidor. Para una aplicación distribuida de ROS se requiere que los nodos peer to peer estén sincronizados bajo un middleware el cual genera un orden temporal para los eventos generados en el sistema, además de utilizar la comunicación XML/RPC que define la información de registro de cada nodo al ROSMaster, una vez registrado los nodos se procede a ejecutar la comunicación peer to peer entre los nodos utilizando el protocolo TCP/IP y, entre los muchos conceptos básicos relacionados en la misma, destacan los siguientes:

1.2.1.1 Nodos (Nodes)

Son procesos que realizan cálculos. Por ejemplo, en un sistema robótico, un nodo realiza la planificación de la trayectoria y otro puede suministrar una vista grafica del sistema.

1.2.1.2 Mensajes (Messages)

Los nodos se comunican entre ellos enviándose mensajes. Un mensaje es simplemente una estructura de datos, que contiene campos de un solo tipo de datos. Tipos primitivos de datos (enteros, booleanos, punto flotante, etc.) son soportados, así como lo son arreglos de tipos primitivos.

Tabla 2.1: Tipos de datos en mensajes

Tipos	Descripción del tipo
int8	Entero de 8 bit
int64	Entero de 64 bit
float32	Flotante de 32bit
string	Cadena
time	Variable de tiempo

1.2.1.3 Tópico (Topic)

El tópico es un nombre usado para identificar el contenido de un mensaje. Un nodo envía un mensaje publicándolo en un tópico dado. Los mensajes son enrutados vía un sistema de transporte con una semántica de publicación/suscripción. es un nombre utilizado para identificar el contenido de un mensaje. Un nodo que está interesado en cierta clase de datos debe suscribirse al tópico apropiado.

1.2.1.4 Maestro (master)

El master proporciona los elementos necesarios para que los nodos se encuentren los unos a los otros, intercambien mensajes o invoquen servicios.

1.2.1.5 Servicios (services)

Es un modelo de comunicación cliente-servidor, es una manera en el que los nodos se pueden comunicar. Estos permiten que se envíen solicitudes y se reciban respuestas

1.2.1.6 Bolsas (bags)

Formato para guardar y reproducir datos de un mensaje proveniente de un nodo de ROS. Son un importante mecanismo para almacenar datos, tales como datos

de sensores, que son difíciles de obtener pero que son necesarios para el desarrollo y prueba de algoritmos.

La arquitectura básica de comunicación de ROS, usando 3 nodos como ejemplo simplificado es mostrada en la figura 2.1. Para establecer comunicación con diferentes nodos es necesario la utilización de registros, cuyos nombres son suministrados por el maestro.

Como se muestra en la Figura 2.1 el Nodo 1, el Nodo 2 y el Nodo 3 se registran al Máster usando comunicación XML/RPC el cual, luego de conocer cada nodo registrado, permite a los diferentes nodos intercambiar la información entre ellos utilizando para esto el protocolo TCP/IP.

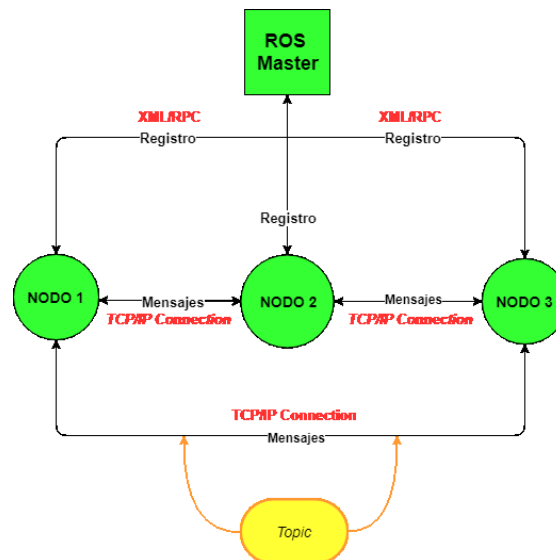


Figura 1.1: Esquema de Funcionamiento de la Arquitectura de ROS

Dado que cada nodo puede compartir múltiples mensajes, a estos se les clasifica como tópicos (topics) lo cual permite la centralización de información proveniente de varios procesos de un sistema robótico ya sea este físico o simulado. En la figura 2.2 podemos observar un comportamiento básico de un nodo y sus tópicos.

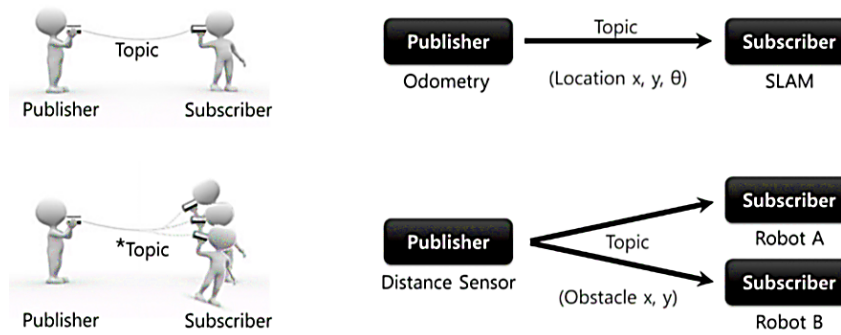


Figura 1.2: Comunicación utilizando Tópicos. En Topics por ROBOTIS Co., Ltd 2017..

Un ejemplo del funcionamiento de la arquitectura de ROS es mostrado en la figura 2.3 en la cual se identifica el **nodo** “robot_localization.” El nodo mencionado recibe mensajes desde varios nodos bajo los **tópicos**: IMU, Camera, GPS, Detección 3D (Sensing 3D) y odometría (odometry), los datos suministrados en los **mensajes** son procesados por el nodo y como resultado comparte bajo un único tópico, y varios mensajes, al visualizador 3D RVIZ las coordenadas de localización del objeto físico para que este pueda ser representado de manera virtual.

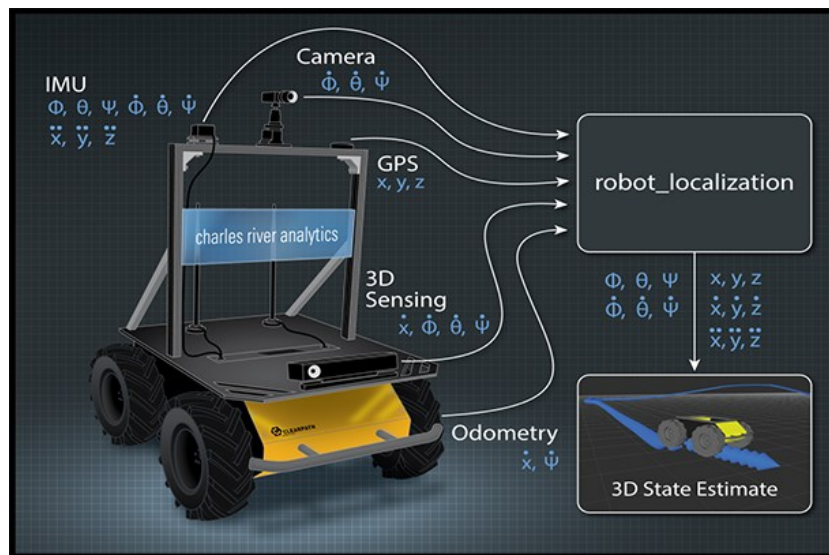


Figura 1.3: Nodo robot_localization recibiendo tópicos de un robot físico y mostrando su localización en el ambiente virtual. En robot_localization por Charles River Analytics Inc 2017.

Un nodo podemos considerarlo como un ejecutable dentro del paquete de ROS y utiliza la librería cliente, lista de nodos disponibles de ROS, para comunicarse con otros nodos, y estos a su vez se les configura para publicar o suscribirse a un tópico, o algunos procesos de ROS tales como servicios o almacenamiento de datos (Bags).

1.2.2 Librerías de ROS

El sistema operativo para robots (ROS) presenta una serie de características que lo hacen atractivo, para las personas dedicadas a la robótica, a la hora de diseñar e implementar sistemas basados en robots. Una de las características principales es que puede acceder a una amplia colección de librerías las cuales varían de acuerdo con la distribución de ROS (turtle, kinetic, etc.) con la que se desee trabajar. Las librerías de las distribuciones más recientes cuentan con librerías con códigos más estables e incorporan nuevas librerías probadas y recomendadas por la comunidad ROS.

Para implementar un nodo en ROS se pueden utilizar diferentes librerías tales como roscpp para C++ o rospy para Phyton.

ROS dispone de las librerías necesarias para la implementación de las actividades que deben ser desarrolladas en un sistema robótico tales como el intercambio de mensajes entre nodos o la estructuración de datos para descripción de trayectorias.

En el **desarrollo del trabajo monográfico se utilizó la distribución** (versión) Kinetic dado que es una de las más estables hasta la fecha.

1.2.3 Herramientas de visualización de datos en ROS

Un elemento importante en cualquier laboratorio virtual de robótica es la posibilidad de visualizar el comportamiento del modelo de un robot en un escenario dado. También es fundamental que el mismo cuente con los recursos necesarios para facilitar el análisis de los datos generados por el sistema robótico bajo estudio. El sistema operativo para robots (ROS) dispone de una serie de herramientas para la visualización de los mensajes de los sistemas robóticos ya sean estos físicos o virtuales. Entre el sinnúmero de herramientas disponibles mencionamos RVIZ, RQT, y MoveIt, nodos en el sistema de suscripción de ROS con sus tópicos y servicios para permitir la entrada y salida de los datos a procesar. Las mismas herramientas **mencionadas son utilizadas** en el desarrollo del laboratorio virtual propuesto.

1.2.3.1 RVIZ

Es una herramienta de visualización 3D que permite combinar en una misma pantalla modelos de robots, datos de sensores (cámara, láser, etc.) y otros datos provenientes de mensajes 3D. **RVIZ es utilizado** en el desarrollo del laboratorio virtual principalmente para visualizar la morfología y estudiar la cinemática de los robots. La figura 2.4 muestra una imagen generada utilizando RVIZ.

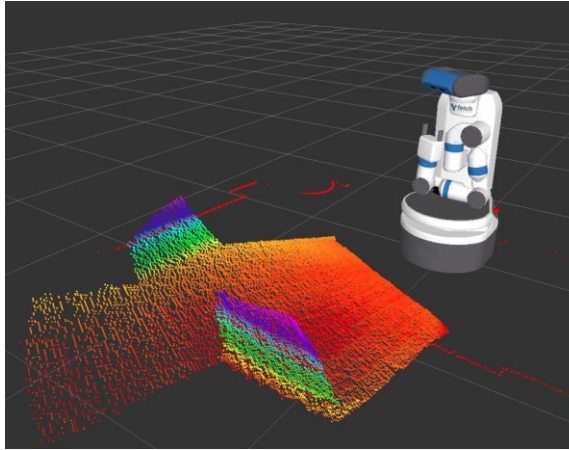


Figura 1.4: RVIZ y visualización de modelo de robot y datos provenientes de mensajes 3D.

1.2.3.2 RQT

La herramienta `rqt` ofrece varios plugins, entre los cuales destacan `'rqt_image_view'`, `'rqt_graph'`, `'rqt_plot'` y `'rqt_bag'`, los cuales pueden ser utilizados para la introspección y visualización de datos provenientes de procesos de ROS. Por ejemplo, si es necesario visualizar los mensajes provenientes de una cámara en un sistema robótico, podríamos utilizar el plugin `rqt-image_view`. `rqt` expone los nodos, y las conexiones entre ellos, lo que permite entender la estructura del sistema, su funcionamiento, así como su fácil depuración. En la figura 2.5 se muestra el resultado de utilizar `rqt_plot` para visualizar los datos provenientes de un nodo.

En el laboratorio virtual de robótica RQT se utilizó para graficar la estructura de datos correspondiente a la posición del robot durante su movimiento.

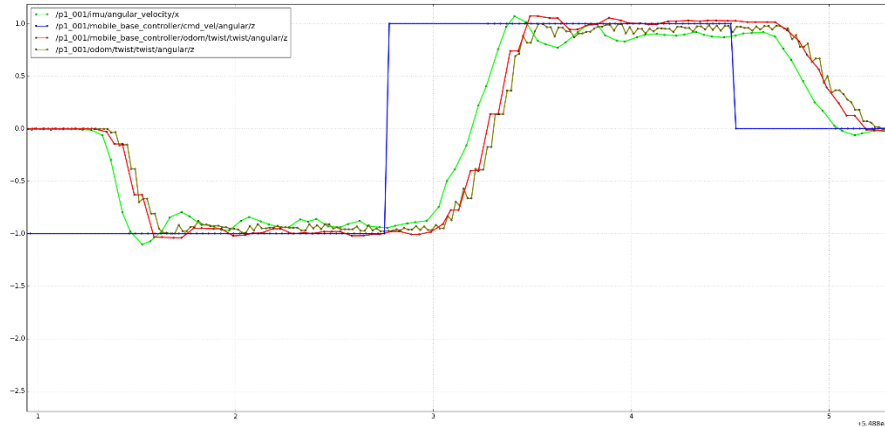


Figura 1.5: Gráfico de línea XY de datos provenientes de nodos

1.2.3.3 MoveIt!

Es una herramienta de software, escrita en C++, que permite la planeación de trayectorias para un robot industrial, así como la visualización de los datos y del modelo 3D del mismo, valiéndose de un plugin que es incorporado a la aplicación de visualización de datos RVIZ. Actualmente, MoveIt (s. f.) es utilizado en diversas industrias para hacer la planeación de las trayectorias para diferentes tipos de manipuladores industriales, adoptando la visión de ROS de facilitar la programación de diferentes tipos de robots con los mismos recursos de Software.

Al sistema en desarrollo se le puede suministrar mayor funcionalidad al utilizar ROS ya que el mismo permite la integración y utiliza la información generada por aplicaciones externas de código abierto muy populares tales como GAZEBO y OpenCV. Por ejemplo, es posible utilizar OpenCV para hacer reconocimiento de imágenes y tomar decisiones, a partir de los datos suministrados. Más adelante en el documento se presentarán las características de GAZEBO dado que es el **simulador que se incorpora en el LVR en la practicas** de programación del robot.

1.2.4 Lenguajes de programación soportados por ROS

En el desarrollo del software para una aplicación de robótica, ROS permite el uso de distintos lenguajes de programación. De manera oficial soporta Python, C++ y LISP. Java podría ser soportado en el futuro, pero en la actualidad se encuentran en una fase experimental, apoyada por Google. De igual forma C# se encuentra en una fase experimental utilizando UNITY 3D, el cual es un motor de video juegos multiplataforma que permite simular robots importando el URDF como un

GameObject en UNITY3D y obtener de este una simulación más realista con el renderizado de UNITY3D.

Con ROS es posible utilizar nodos creados en diferentes lenguajes de programación tales como Python y C++ y garantizar el intercambio de información entre estos.

En el desarrollo de la GUI y el intercambio de información entre los nodos ROS se utilizará el lenguaje C++ y el compilador CMake el cual fue diseñado para construir, probar y empaquetar software. CMake es el compilador oficial soportado por la comunidad ROS y el mismo desplazó al anterior compilador llamado rosbld.

El ambiente integrado de desarrollo Qt-Creator posee un plugin para ROS (ROS Industrial, 2016) que cuenta con los recursos necesarios (por ejemplo, templates) para crear nodos, servicios, y archivos URDF, entre otros. Qt-Creator permite al desarrollador construir los recursos de software necesarios, desde el IDE, para un sistema robótico.

1.3 Laboratorio virtual para robótica industrial

En una institución de educación superior, que dispone de un laboratorio físico para la enseñanza de la robótica, el robot es parte de un escenario que simula un ambiente industrial y las prácticas generalmente están relacionadas con la programación del robot para que realice una tarea determinada. El sistema no presenta la flexibilidad necesaria para que los estudiantes mejoren su comprensión acerca de los fundamentos de la robótica industrial. De hecho, aprenden a programar el robot utilizando el lenguaje específicamente diseñado para este.

El laboratorio virtual para robótica que se ha desarrollado contempla la incorporación de los recursos necesarios para que los estudiantes **estudien o realicen** experimentos relacionados con temas tales como la morfología del robot, la cinemática del robot, la programación de robots de diferentes fabricantes independientemente del lenguaje específico desarrollado para cada uno. Se incorporarán recursos que permitirán al estudiante entender las herramientas matemáticas utilizadas en el campo de la robótica industrial, específicamente manipuladores industriales tipo serie.

En el desarrollo de software es común el uso de los términos Backend y Frontend los cuales aportan una perspectiva de la clasificación de tipo de Software a utilizar en la construcción del propio laboratorio virtual, **se uso ROS** para manejo de datos en Backend y el uso de librerías de software libre para desarrollar GUI del lado de Frontend.

En el LVR el estudiante, para la realización de los experimentos, **puede utilizar** modelos de robots de fabricantes tales como ABB o KUKA. De igual forma, **cuenta** con las herramientas y la guía para desarrollar sus modelos propios.

Los principales elementos del laboratorio virtual son:

- Interfaz gráfica de usuario (GUI) para (interacción, programación, visualización, etc.)
- Modelos de robots industriales de ABB, KUKA, y DELTA
- Escenarios para la realización de tarea específicas
- Documentación sobre aspectos fundamentales sobre robótica industrial
- Otros

La interacción entre los diferentes elementos del laboratorio virtual es garantizada mediante el middleware ROS.

A continuación, son presentados algunos de los conocimientos o recursos tecnológicos requeridos para el desarrollo del LVR, específicamente para la implementación de la interfaz gráfica de usuario (GUI) y la construcción de algunos recursos para el estudio de los fundamentos de los manipuladores industriales.

1.3.1 Interfaz gráfica de usuario (GUI)

Una interfaz gráfica de usuario (GUI, por sus siglas en inglés) facilita la explotación de una aplicación mediante la incorporación de elementos destinados para tal fin. La GUI debe comportarse de una manera comprensible y predecible, de modo que un usuario sepa qué esperar cuando realiza una acción. La interfaz gráfica de usuario del **laboratorio virtual cuenta con los elementos** necesarios (botones, sliders, editor de texto, pantalla de visualización, etc.) para realizar las prácticas sobre los fundamentos de la robótica industrial, así como para acceder a documentos de interés tales como libros, guías de laboratorio, entre otros. **La GUI está diseñada de tal forma** que el estudiante podrá usar los recursos de manera transparente sin necesidad de tener que configurar aspectos básicos de ROS tales como su inicialización y el lanzamiento de nodos.

En la actualidad existen muchos lenguajes que pueden ser utilizados para desarrollar **una** interfaz de usuario y entre ellos destacan C++, C#, Java, Python, Visual Basic. Un criterio para utilizar un lenguaje en particular es que el mismo sea multi-plataforma ya que la interfaz podría ser utilizada bajo diferentes sistemas operativos.

Hoy día están disponibles un sinnúmero de ambientes integrados de desarrollo (IDE, por sus siglas en inglés) que facilitan la programación ya que aglomeran todas las herramientas necesarias tales como conjuntos de librerías, preprocesador, depurador, y compiladores. Entre los IDEs más conocidos destacan, entre otros, Visual Studio, Eclipse, NetBeans, CLion, y Qt Creator. Para la construcción de la GUI del laboratorio virtual de robótica industrial **se utilizó el IDE Qt Creator**, debido, principalmente, a que este posee un plugin de soporte para ROS.

Qt Creator se utilizó para crear las aplicaciones o nodos de ROS **y se exploró las capacidades** que posee Qt Creator para la creación de aplicaciones de escritorio. Qt Creator cuenta con QML, un estándar propio del IDE, que permite crear aplicaciones dinámicas y más elaboradas capaces de incorporar recursos WEB. Esa línea de **creación se utilizó para incorporar** un lector de PDF con el objetivo que los estudiantes dispongan en el LVR de documentación de interés tales como libros o guías de laboratorio.

En el desarrollo de la GUI del LVR se utilizaron los widgets de Qt Creator (slider, botones, etc.) los cuales están escritos en C++. Esta línea de creación es muy importante para lograr la visualización en el LVR ya que es muy fácil incorporar las librerías de ROS, escritas en C++, a una aplicación utilizando widgets. Un ejemplo es el desarrollo de una aplicación de escritorio personalizada que se acople a los requerimientos del LVR al incorporar las librerías de RVIZ para visualizar los modelos de los robots.

Es importante mencionar que en el laboratorio virtual se realizan múltiples procesos en el Backend, con el objetivo de garantizar la transparencia en el uso de los sus recursos por los usuarios, que darán inicio a la ejecución de las interacciones en la interfaz de usuario tales como la visualización de los robots y datos, programación y simulaciones de los robots. Dichos procesos están basados en nodos que son lanzados o eliminados de acuerdo con la lógica implementada en el propio LVR.

1.3.2 Fundamentos de la robótica industrial.

El laboratorio de robótica virtual que se desarrollo tiene un alcance limitado ya que de los temas considerados en un curso básico de robótica industrial el mismo solo contempla recursos para la experimentación en temas como herramientas matemáticas, morfología del robot, cinemática y programación de robots. El trabajo pretende mostrar la potencialidad del sistema operativo para robots (ROS) y lo hace poniendo en marcha el diseño, la programación y el uso del LVR. La información suministrada acerca de ROS y el laboratorio propuesto deben servir de base para ampliar los recursos de este.

1.3.2.1 Morfología del robot

El laboratorio virtual de robótica cuenta con modelos creados utilizando URDF y los mismos podrán ser visualizados con la herramienta RVIZ. El estudiante puede visualizar el espacio de trabajo, para una morfología dada, con la ayuda de plugins desarrollados para la visualización. Interactuar con diferentes tipos de articulaciones ya sean prismáticas o de revolución y robots de diferentes grados de libertad. El estudiante tendrá la posibilidad de experimentar con modelos construidos por él mismo utilizando el formato URDF.

RVIZ es una abreviación para ROS visualización y es una herramienta poderosa para la visualización en 3D. Permite al usuario ver el modelo simulado del robot, obtener información de los sensores del robot, y reproducir la información obtenida de los sensores. Mediante la visualización de lo que ve y hace el robot el usuario puede eliminar los errores de la aplicación robótica desde entradas de sensores hasta acciones planificadas o no planificadas.

1.3.2.2 Herramientas matemáticas

El modelado cinemático de un robot busca las relaciones entre las variables articulares y la posición (expresada normalmente en forma de coordenadas cartesianas) y orientación del extremo del robot (expresada como matrices de rotación, ángulos de Euler o algún otro de los métodos establecidos para tal fin).

Entender las herramientas matemáticas utilizadas para determinar la localización espacial de los diferentes elementos de un robot industrial, especialmente del efector final, presenta cierta dificultad a los estudiantes, principalmente cuando se utiliza solamente la pizarra para su explicación. Para contribuir al entendimiento y aplicación de dichas herramientas el laboratorio virtual de robótica incorpora los recursos necesarios para que los estudiantes puedan analizar la posición de un objeto utilizando coordenadas esféricas y cilíndricas, así como la orientación de este mediante la matriz de rotación, los ángulos de Euler, y los cuaternios. Las operaciones requeridas para el desempeño de los recursos estarán soportadas por la librería TF2, disponible para C++ y Python, propia de la distribución de ROS, que permite hacer conversiones de sistemas de coordenadas.

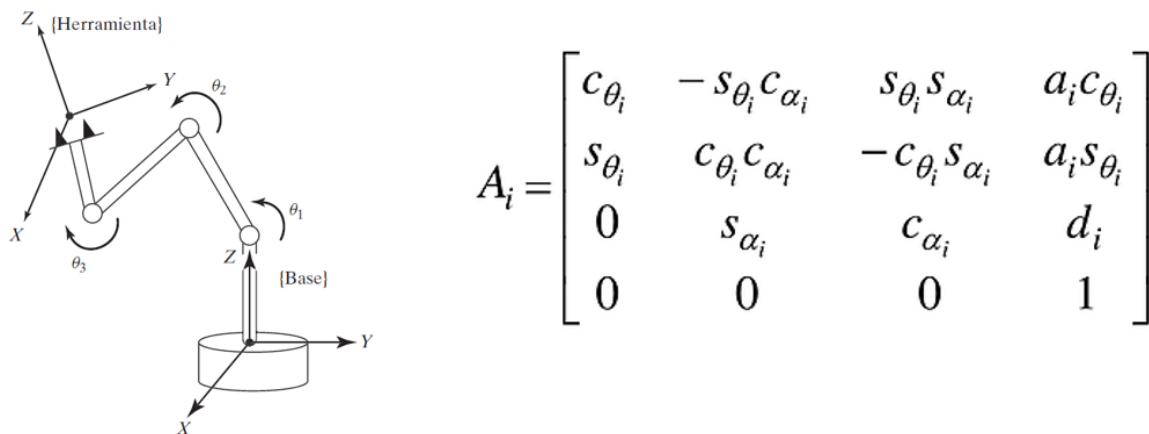


Figura 1.6: Modelo de robot, sistemas de referencia. En "Robótica" (P.6), por J. J. Craig, 2006

1.3.2.3 Cinemática directa e inversa

La cinemática del robot estudia el movimiento de este con respecto a un sistema de referencia sin considerar las fuerzas que intervienen. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares. Dentro de la cinemática se estudian la posición, velocidad, aceleración

y todas las derivadas de mayor orden de las variables de posición (respecto al tiempo o a cualquier otra variable).

En muchas ocasiones es necesario conocer en cuál posición se encontrará el efector final del robot dadas ciertas posiciones de las articulaciones y dicho problema es abordado por la cinemática directa. En otras ocasiones es necesario conocer cuál debe ser la posición de las articulaciones para que el efecto sea ubicado en una posición deseada. El último problema es abordado por la cinemática inversa. La *figura 2.7* muestra la relación entre las variables para cada uno de los tipos de cinemática.

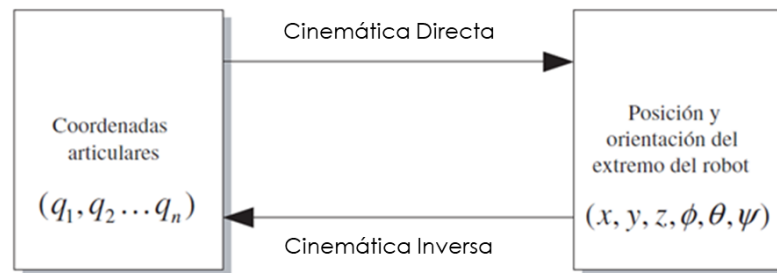


Figura 1.7: Cinemática directa e inversa En “Fundamentos de robótica” (P.119), por A. Barrientos et al, 2007

El laboratorio virtual de robótica que se propone cuenta con recursos que permitirán al estudiante realizar experimentos sencillos que le ayudarán a comprender las características de la cinemática directa e inversa.

A partir del modelo de un robot, construido utilizando el formato URDF, es posible obtener toda la información necesaria de la estructura de este utilizando la librería KDL. La información obtenida al utilizar el KDL en combinación con la información suministrada por el usuario (por ejemplo, los ángulos de los joints) mediante ciertos widgets de la GUI permiten manipular y/o determinar la posición del robot. La librería KDL también puede ser utilizada para determinar la matriz de transformación homogénea la cual es utilizada para determinar la posición y rotación del efector final del robot. Los datos obtenidos serán visualizados mediante widgets del GUI desarrollados para la representación de datos.

1.3.2.4 Programación y simulación del robot

En la robótica, en general, uno de los más aspectos de mayor importancia es el relacionado con la programación de los robots. Durante la ejecución de un programa se interacciona con la memoria del sistema, leyendo y actualizando el contenido de las variables utilizadas en el programa; con el sistema de control cinemático y dinámico del robot, encargados de dar la señal de mando a los accionamientos del robot a partir de las especificaciones del movimiento que se les proporciona; y con las entradas-salidas del sistema, consiguiéndose así la

sincronización del robot con el resto de las máquinas y elementos que componen su entorno.

El sistema de programación es, por tanto, la herramienta con que cuenta el usuario para acceder a las diversas prestaciones del robot, existiendo una relación directa entre las características y posibilidades del sistema de programación y las del robot en sí mismo.

La programación puede ser guiada o textual. Es importante destacar que en la actualidad es muy frecuente que los sistemas de programación de robots tiendan a combinar los dos modos básicos (guiado y textual), permitiéndose desarrollar el programa mediante la escritura de las instrucciones, y utilizando la posibilidad de guiado en línea en aquellos momentos en que sea necesario. Sistemas como RAPID de ABB, VAL II de Staübli y V+ de Adept Technology son ejemplos de esta ambivalencia.

Uno de los problemas relacionados con la programación de los robots es que no existe un estándar y el programador debe aprender el lenguaje específico para los robots de diferentes fabricantes. ROS brinda la posibilidad de programar robot de diferentes fabricantes desde el mismo entorno.

Supongamos que es necesario que dos robots de diferentes fabricantes, pero la misma morfología, deben seguir la misma trayectoria. En el ambiente de ROS, es posible escribir mensajes de ROS (que contiene la posición, velocidad, aceleración, etc., de cada articulación del robot), escritos en C++ o Python. El driver de cada robot, una vez recibido el mensaje, generará los comandos para que su robot siga la trayectoria indicada.

El laboratorio virtual de robótica permite al usuario escribir un programa (script, programación textual) que adoptará ciertas convenciones, utilizadas en el software de programación de robots de ABB Rapid, para que uno de los robots (modelos disponibles en el LVR) realice una tarea sencilla.

Una forma mediante la cual podemos verificar si el robot realiza la tarea indicada es la simulación. En la actualidad existen muchos simuladores, sencillos y complejos, para robots. También podemos clasificarlos en comerciales y de código libre.

Entre los comerciales destaca V-Rep el cual es un simulador dinámico con un entorno de desarrollo integrado que permite desde su interfaz lenguajes de programación tales como C, C++, Python, entre otros.

V-Rep es un simulador con entorno de desarrollo integrado donde cada objeto/modelo se puede controlar individualmente a través de un script de programación que puede ser escrito usando lenguajes de programación como

C/C++, Python, Java, Lua, Matlab or Octave, V-REP se utiliza para el desarrollo rápido de algoritmos, simulaciones de automatización de fábricas, prototipado rápido y verificación y educación relacionada con la robótica.

De los simuladores open-source uno de los más utilizados es GAZEBO, el cual provee simulaciones dinámicas donde los robots pueden interactuar con el entorno (pueden coger, empujar, rodar, deslizarse por el suelo), capacidad de manipular las características del entorno tal como la gravedad del mundo virtual.

Gazebo es simulador dinámico 3D que tiene la habilidad para simular poblaciones de robots, de forma exacta y eficiente, en ambientes complejos internos y externos. Aunque similar a máquinas de juego, Gazebo ofrece una simulación física a un grado de fidelidad mayor, un conjunto de sensores, e interfaces tanto para programas como para los usuarios. Usos típicos de Gazebo incluyen:

- Prueba de algoritmos robóticos
- Diseño de robots
- Realización de pruebas de regresión con escenarios realísticos.

Algunas características claves de Gazebo:

- Máquinas físicas múltiples
- Una librería de modelos de robots y ambientes.
- Una amplia variedad de sensores

Aunque ambos simuladores tienen la posibilidad de comunicarse utilizando ROS se ha seleccionado GAZEBO como el simulador a incorporar en el LVR debido a que este es open-source. El estudiante podrá observar en el simulador GAZEBO el comportamiento del modelo del robot en respuesta al programa (script) escrito por el usuario. El script incorporará el conjunto de instrucciones las cuales serán procesadas por un nodo el cual enviará a GAZEBO, utilizando el plugging de conexión correspondiente, los mensajes apropiados para lograr el movimiento de las articulaciones del robot en el mundo.

SDF es un formato XML utilizado para describir objetos y ambientes para simuladores de robots, visualización, y control. SDF tiene capacidad para describir todos los aspectos de un robot, objetos estáticos y dinámicos, iluminación, terreno y también la física. **En el laboratorio virtual el lenguaje SDF se utiliza para modelar los escenarios de trabajo en el ambiente de Gazebo.**

Podrán ser utilizados los siguientes parámetros:

- ✓ **Escena:** Iluminación ambiental, propiedades del cielo, sombras.
- ✓ **Física:** Gravedad, paso del tiempo, motor de la física.

- ✓ **Modelos:** Colección de enlaces, objetos de colisión, articulaciones y sensores.
- ✓ **Luces:** Punto, espacio y fuentes de luz direccionales.
- ✓ **Plugins:** plugins soportado por gazebo, plugins del mundo, del modelo, del sensor y del sistema.

1.4 Uso del laboratorio virtual de robótica industrial

Los diferentes elementos involucrados en el uso, y posible ampliación del laboratorio virtual, son mostrados en la figura 2.8. El laboratorio virtual incluye un manual de usuario cuyo objetivo es presentar al usuario los recursos disponibles y un par de guías de laboratorio para mostrar cómo utilizarlos. Como se aprecia en la figura 2.7, el docente podrá, partiendo de los temas relacionados con los fundamentos de la robótica industrial que desee desarrollar y de los recursos disponibles en el laboratorio, elaborar las guías que utilizarán los estudiantes para realizar las actividades de estudio o las prácticas de laboratorio correspondientes.

Se pretende, como parte del trabajo monográfico, desarrollar un taller, para los docentes interesados, con el objetivo de brindarles los detalles de diseño y herramientas utilizadas en la implementación del LVR para que tengan una base para la ampliación de los recursos de este.

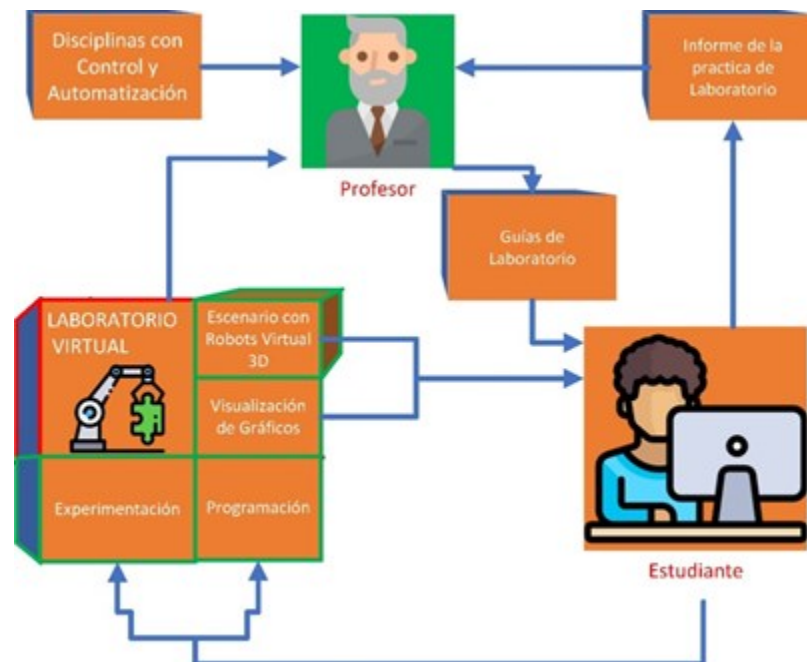


Figura 1.8: Modelo de utilización del LVR.

En la revisión de literatura y reconocimiento de los laboratorios en algunas instituciones de educación superior, encontramos que la principal actividad que se realiza en este tipo de laboratorios es la programación del robot, usando un lenguaje específico, para que ejecute una tarea determinada.

El laboratorio virtual desarrollado, ver Figura 2.2, cuenta con los recursos de un laboratorio básico de robótica e incorpora otros que permiten realizar otras actividades más allá de la simple programación del modelo del robot bajo consideración.

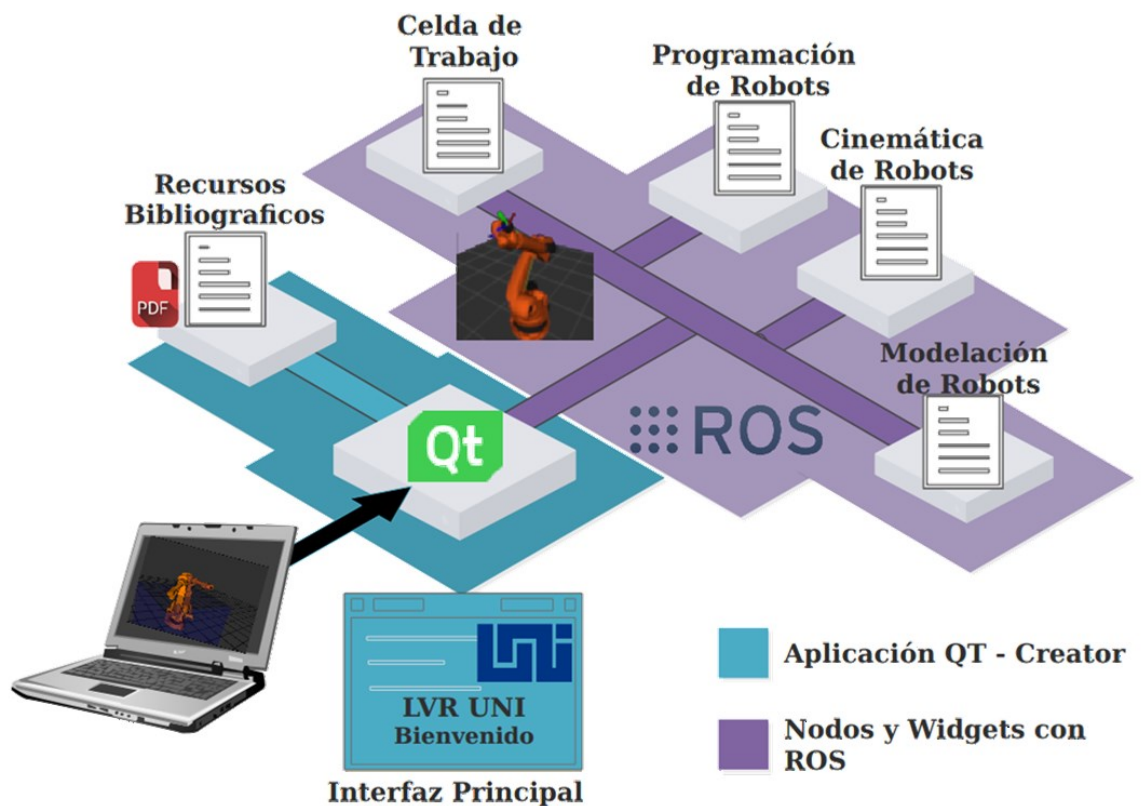


Figura 2.2: Modelo del laboratorio Virtual de Robótica Industrial.

2.2 Recursos de software

A continuación, se explicara el proceso de adquirir los recursos de software necesarios para lograr el desarrollo del laboratorio virtual, se detallara la importancia de estos y enfatizando en aspectos particulares que son necesarios dominar para trabajar con el sistema operativo Ubuntu, ROS, el IDE y librerías de importancia para este trabajo monográfico.

2.2.1 Sistema operativo Ubuntu

El Sistema operativo sobre el que se desarrolló este trabajo monográfico es una distribución de Linux, precisamente la distribución Ubuntu 16.04 LTS (Long Terminal Support), es libre y con compatibilidad con la versión más estable de ROS, la distribución Kinetic LTS cabe destacar que estas versiones se mantiene con actualizaciones hasta el 2020, actualmente está disponible la versión estable ROS Melodic pero exclusivamente no hace falta migrar a esta versión debido a que todos los recursos de software que se necesitaran para este trabajo están bajo la versión instalada Kinetic (Ver anexo x instalación de Ubuntu).

Debido a que el sistema operativo Linux no es ampliamente utilizado, representa una fragilidad para esta propuesta de trabajo que pretende sea usada (adoptada) en las aulas de clases de la Universidad Nacional de ingeniería, esto porque actualmente el sistema operativo dominante en las computadoras personales es Windows, es sabido que para acceder a los recursos de software en las plataformas Linux es necesario tener un cierto manejo y conocimiento de líneas de comandos (Shell) en Linux para movernos por el entorno, esto representa una falta de popularidad comparada con Windows, de tal forma es necesario dar a conocer que esto no representara un problema dentro de poco, en los trabajos desarrollados con ROS ya que Microsoft anuncio la colaboración con ROS en adoptar la compatibilidad del Middleware ROS a Windows (Microsoft 2018), de esta forma este trabajo monográfico posee la capacidad de ser actualizado a una versión en Windows.

Por lo tanto, dentro del desarrollo de este trabajo fue de vital importancia conocer el entorno Linux para entender cómo se realizan las compilaciones, ejecutar comandos con debida soltura por consola para manipular parámetros de manera rápida y como mostrar resultados de procesos por consola.

2.2.2 ROS

A como se mencionó en la sección 1.2 ROS se define como un framework para el desarrollo software de robot. ROS provee los servicios que ofrece cualquier sistema operativo: abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidades de uso común, paso de mensajes entre procesos y mantenimiento de mensajes. (Reordenar ideas)



Figura 2.3: Logo del sistema operativo para robots.

La curva de aprendizaje que toma aprender ROS es algo sinuosa al principio, pero algo que caracteriza a ROS de otros middleware como Orocos, playerStage, etc es que tenemos a nuestra disposición multitud de recursos que nos brinda la propia comunidad, de hecho ser parte de la comunidad complementa la filosofía que dio origen a ROS (Ver sección 2.2) de no reinventar la rueda al compartir ideas de cómo hacer robótica bajo el estándar de trabajo basado en un ecosistema ROS con más de 3,000 paquetes de software, una infraestructura de comunicación modular basada en nodos, tópicos, o mensajes de ROS, en su página cuenta con foros o la wiki de la propia página. En dicha Wiki nos muestran los pasos necesarios para comenzar en ROS en la sección Getting Started (Ros Wiki 2018)

2.2.2.1 Instalación y Puesta en marcha de ROS

Dentro del desarrollo de aplicaciones o recursos de software solventados o mantenidos por una comunidad es muy importante definir la versión con la cual se desea trabajar ya que estos recursos de software (Package) son actualizados o se agregan nuevas funcionalidades constantemente, si el trabajo investigativo lleva una extensión de tiempo considerable y se desea aprovechar todas las funcionalidades de cierta versión, se debe elegir una distribución LTS en nuestro caso desde el inicio del desarrollo de este trabajo se instaló la distribución Kinetic,

y se han aprovechado muchos de los recursos bajo esta distribución. Ver Anexo 1.2 Instalación de ROS.



Figura 2.4: Logo de la distribución de ROS, Kinetic Kame.

Al instalar todos los paquetes de ROS Kinetic con el comando

```
$ sudo apt-get install ros-Kinetic-desktop-full
```

Comando BASH

y entre otras herramientas como visualizadores de datos como, RQT, Rviz y el simulador Gazebo obtenemos los recursos de software necesarios para el desarrollo del Laboratorio Virtual de Robótica.

Es necesario conocer los comandos Shell para el desarrollo con ROS, los comandos más comunes son.

Tabla 2.1: Comandos ROS

Comando	Descripción
roscore	Es necesario ejecutar este comando ya que pone en marcha para que todos los demás funcionen. Es el encargado de gestionar toda la comunicación entre los nodos.
roscd:	Permite cambiar de paquete, directorio o stack. Sin necesidad que sean colindantes de forma jerárquica.
rostopic	[list] Muestra la lista de topics en ejecución
	[echo] Muestra información sobre el topic
	[type] Muestra el tipo del topic
	[pub] Publica un mensaje en un topic dado
roscd	[list] Muestra la lista de nodos en ejecución
	[info] Muestra información sobre el nodo
	[ping] Comprueba la conexión con el nodo

	[kill] Mata el nodo indicado
roslaunch	Ejecuta un nodo. <code>\$ roslaunch [PACKAGE_NAME] [NODO_NAME]</code>
rosmmsg:	Muestra información sobre los mensajes ROS
roslaunch	Permite lanzar varios nodos y parámetros simultáneamente \$ <code>ros launch [PACKAGE_NAME] [launch_FILE_NAME]</code>

Si ejecutamos el comando

```
$ roscore
```

Comando BASH

Podemos comprobar el estado de la instalación de ROS y por ende proseguir con el desarrollo de software.

```
roscore http://yesser-dell-system-inspiron-n411z:11311/
yesser@yesser-dell-system-inspiron-n411z:~$ roscore
... logging to /home/yesser/.ros/log/9de7fc22-1498-11e9-a202-b5ee938d8173/roslau
nch-yesser-dell-system-inspiron-n411z-14673.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
WARNING: disk usage in log directory [/home/yesser/.ros/log] is over 1GB.
It's recommended that you use the 'rosclean' command.

started roslaunch server http://yesser-dell-system-inspiron-n411z:34304/
ros_comm version 1.12.14

SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES

auto-starting new master
process[roscout-1]: started with pid [14696]
ROS_MASTER_URI=http://yesser-dell-system-inspiron-n411z:11311/

setting /run_id to 9de7fc22-1498-11e9-a202-b5ee938d8173
process[roscout-1]: started with pid [14696]
started core service [/roscout]
```

Figura 2.5: roscore servidor maestro

En la figura 2.5 se aprecia que está en excelente estado la instalación con el servidor maestro puesto en marcha.

2.2.2.2 Estructura de ROS

Como se especificó en la sección 1.2.1 Conceptos Básicos de ROS, los nodos en ROS se comunican entre sí por topics intercambiando mensajes siendo estos mensajes administrados y enrutados a su nodos correspondiente por el ROSMaster, podemos ver a estos nodos como ejecutables dentro de paquetes de software ya sea escritos en C++ o python para entender la estructura de ROS podemos ver la Figura 2.6 con su respectiva Tabla 2.2 que describe la clasificación de los elementos.

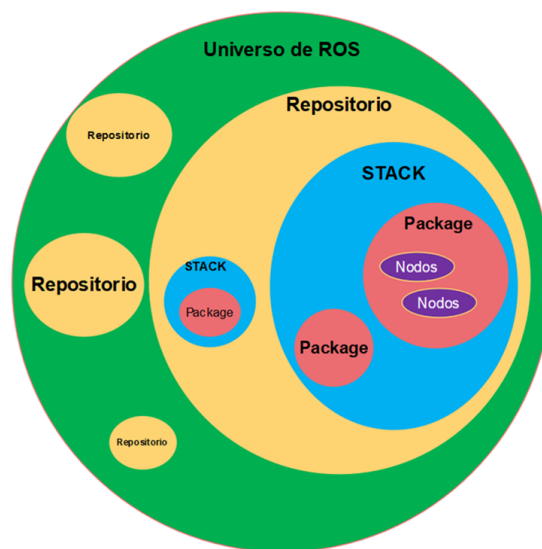


Figura 2.6: Organización del software en ROS

Tabla 2.2: Estructura de ROS

Clasificación	Descripción
Repositorio	Es en general un proyecto que puede tener diversos paquetes de software que interactúan intercambiando información bajo los nodos de ROS, El laboratorio Virtual de Robótica es considerado un Repositorio, en el van diferentes recursos de software como Widgets de Qt-Creator, QML con Qt Creator, librerías Boost, Librerías RVIZ y librerías ROS.
Stack	Las pilas o stacks contienen ciertos paquetes relacionados entre sí

Packages	Los paquetes contienen dentro la definición de los mensajes que utilizan los nodos asociados a estos, por ejemplo un paquete de un stack puede enviar información procesada de la cinemática de un robot vía mensajes a un nodo de otro paquete en otro Stack que no cuenta con este procesamiento y por ende los recursos de software.
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Para entender mejor la estructura de un paquete y su relación con un Stack podemos ver la estructura del mismo en la Figura 2.7, relacionando directamente a el tipo de recurso que posee entre los diferentes paquetes.

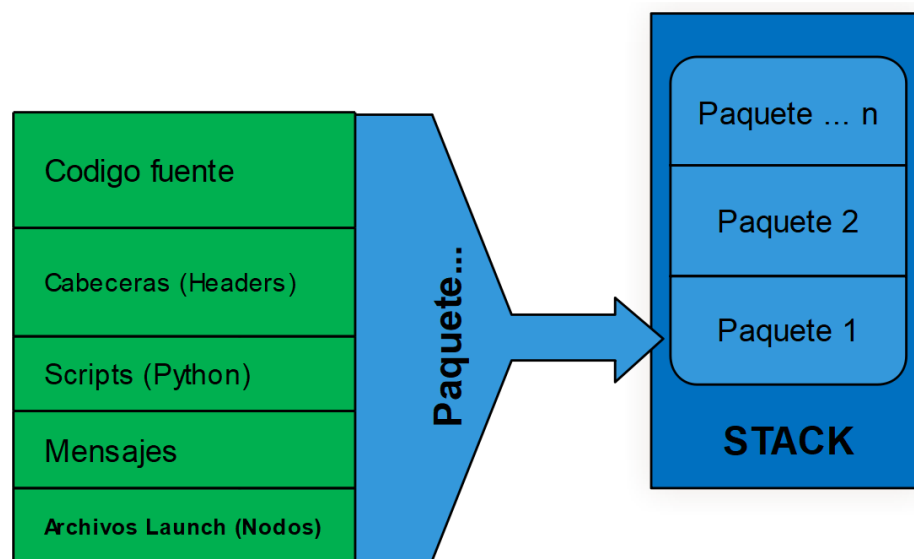


Figura 2.7: Organización de un paquete de software en ROS

Dentro de los paquetes encontramos los siguientes elementos que son fundamentales entender para el desarrollo de cualquier aplicación con ROS.

Tabla 2.3: Contenido de un Package

Elementos	Descripción
Package.xml	Este archivo de lenguaje descriptivo contiene todos los paquetes de los que depende el mismo paquete en cuestión. Se definen aquellos que van a ser construidos y los que van a ejecutarse bajo las etiquetas <build_depend/> y <run_depend/>

CMakeFile	Este archivo se encarga de compilar el paquete. Deben incluirse todas las dependencias y códigos que se hayan desarrollado si existiesen. Estos códigos tendrán librerías específicas que hacen referencia a paquetes de ROS por ello es necesario definir cada uno de estos paquetes para que el compilador lo tenga en cuenta
Carpetas	Normalmente se designa las siguientes carpetas: <i>src</i> , <i>launch</i> , <i>bag</i> . <i>src</i> se guarda el código que se desea compilar y ejecutar como nodo. <i>launch</i> se guardan los lanzadores de nodos.

Para crear un paquete utilizamos las instrucciones de línea de comando del compilador Catkin, para esto debemos crear un Espacio de trabajo para el compilador (Ver anexo A.4 Catkin) donde estarán almacenados todos los paquetes que vayamos creando, es necesario conocer 3 comandos importantes al compilar con Catkin.

Tabla 2 4: Comandos Catkin.

Comandos	Descripción
catkin_init_workspace	Inicializa el espacio de trabajo creado
catkin_create_pkg	Crea un paquete con la dependencia de una librería u otro paquete de ROS. [PACKAGE_NAME] [DEPENDENCY_PACKAGE1]
catkin_make	Compila todos los paquetes dentro del Espacio de Trabajo.

En la Figura 2.8 se muestra la estructura de carpetas que conforman nuestro espacio de trabajo con Catkin.

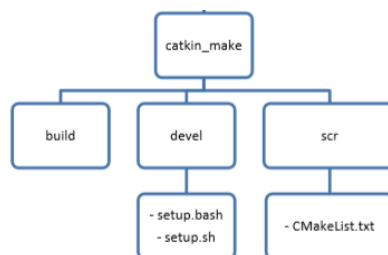


Figura 2.8: Espacio de Trabajo Con Catkin.

2.2.2.3 Compilación de Paquetes de ROS.

Todo paquete debe contener dos archivos los cuales son de importancia para el compilador CMakefile.txt y package.xml, En estos se definen los recursos de software de un paquete para que este sea compilado correctamente. Los archivos CMakefile y package.xml poseen varias funciones a tener en cuenta, pero en nuestro caso, tendremos en cuenta los siguientes parámetros.

find_package

CMakeLists.txt

Este atributo del archivo CMakeLists permite agregar los paquetes necesarios para la compilación.

Indicamos el mismo paquete en las dependencias de catkin.

catkin_package

CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(my_first_node_lab_UNI)
3
4 find_package(catkin REQUIRED COMPONENTS
5   roscpp
6 )
7
8 catkin_package(
9   # INCLUDE_DIRS include
10  # LIBRARIES myworkcell_core
11  CATKIN_DEPENDS
12    roscpp
13  # DEPENDS system_lib
14 )
```

Version de Catkin

Nombre del
paquete

Dependencia del
paquete o project

roscpp
dependencia
para creacion
de nodos

Dependencia del
paquete o project

Figura 2.9: Archivo CMakeLists con parámetros para compilación

Es necesario la correcta configuración de este archivo debido a que la compilación de un paquete a ser usado bajo ROS depende de lo indicado en el archivo CMakeLists, En la Figura 2.9 se muestra parte de los atributos necesarios a ser configurados.

El archivo package.xml es un complemento de CMakeLists.txt donde también se agregan las dependencias solo que en formato con etiquetas en un archivo .XML dentro de estos atributos tenemos los más destacados a analizar a continuación.

```
<buildtool_depend>catkin</buildtool_depend>
```

package.xml

```
<run_depend>roscpp</run_depend>
```

package.xml

Todas estas etiquetas en el archivo de marcado son atributos importantes para hacerle referencia al compilador de que recursos se ocuparan en el paquete y además como se utilizaran, por ejemplo el `<build_depend>` se utiliza para...

```
<build_depend>roscpp</build_depend>
```

package.xml

En cambio el `<run_depend>` del mismo paquete se utiliza para...

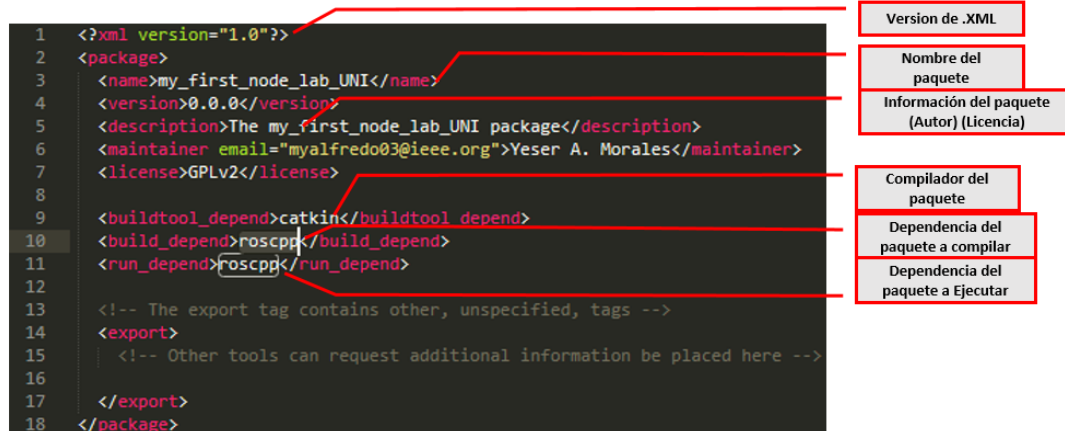


Figura 2.10: Archivo `package.xml` con etiquetas de referencia de paquetes.

Al crear un paquete siguiendo las indicaciones plasmadas en el anexo 1.4 Catkin, se garantiza que podemos crear lógica de programación dentro de un archivo en C++, crear un nodo, hacer traspasos de mensajes vía tópicos usando los recursos de ROS o en todo caso hacer uso de Widgets para la creación de Interfaces de usuario, donde estas interfaces a crearse para el laboratorio después de todo se puede definir como un paquete con recursos de ROS que puede mostrar información al usuario.

2.2.3 IDE QT-Creator.

En la Sección 1.3.1 se mencionan las diferentes tecnologías más conocidas para el diseño de una interfaz de usuario por lo tanto se da a conocer que QT-Creator posee varias ventajas que benefician este trabajo monográfico destacando la característica multiplataforma de este IDE, además, anteriormente en la sección 2.2.1 se dan a conocer los detalles del porque se usó el sistema operativo Ubuntu y la posibilidad de migrar este trabajo monográfico a un sistema operativo de uso común como lo es Windows, al usar QT - Creator junto con la futura versión de ROS (Microsoft ROS) para Windows es posible gracias a su característica multiplataforma.



Figura 2.11: Logo de QT-Creator

Qt Utiliza el Lenguaje de programación C++ como principal herramienta de desarrollo, pero adicionalmente puede ser utilizado en otros lenguajes de programación a través de “bindings” o Plugins Conocidos normalmente, los plugins mas conocidos son PyQt para el desarrollo de interfaz en Python, QtRuby, PHP-QT y entre otras.

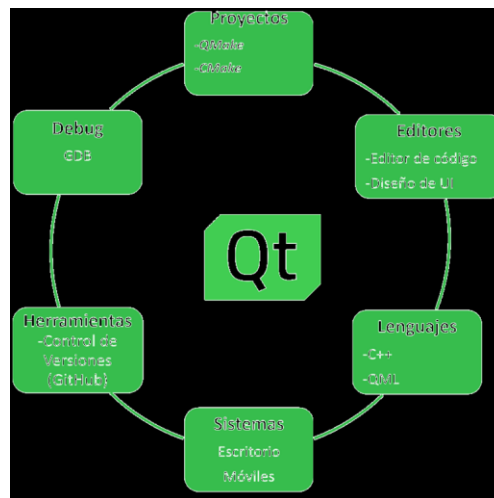


Figura 2.12: Recursos de QT para el Desarrollador.

Debido las características mostradas en la Figura 2.12 del IDE QT Como el uso, de proyectos compilados con CMake (Ver sección 2.2.2.3 Compilación de Paquetes de ROS.) Uso de lenguajes como C++ para trabajar con ROS y otras librerías como KDL escritas en este lenguaje y QML para el desarrollo de interfaces y entre otras bondades plasmadas en la Figura 2.12, por lo cual el IDE Qt Es el que adopta todos los recursos de software necesarios para un desarrollo ágil y centralizado.

QT provee de recursos a otros lenguajes de programación y posee la capacidad de adoptar muy fácilmente varios recursos de que poseen otros Lenguajes dentro del propio IDE, Por ejemplo, diseñar una interfaz de Usuario (GUI) con capacidad de cargar una pagina WEB o vincular recursos de base datos, tales recursos son utilizados dentro de este laboratorio Virtual.

Dentro del desarrollo de este trabajo monográfico se aprovecharen 3 ejes principales de Qt- Creator los cuales son el uso del plugin de ROS para el IDE, El uso de Widgets para aplicaciones de interfaz de Usuario nativas en C++ y el uso de Widgets QML para para aplicaciones de interfaz de usuario donde el lenguaje QML, presta ciertas bondades parecidas a usar CCS al diseñar paginas Web le aporta dinamismo y mayores prestaciones visuales al desarrollar una aplicación con QML.

2.2.3.1 Plugin de ROS para Qt-Creator

La Version de QT usada en este trabajo monográfico es la versión 5.7 compatible con el Plugin de ROS desarrollado por el Consorcio de ROS – Industrial, Este plugin Permite la agilización de creación de paquete de ROS, Mensajes, Servicios, Acciones e incorporar la estructura básica de programación en C++ de un nodo.

Al instalar el IDE (Ver anexo 1.4) se puede proceder a la instalación y configuración del plugin para el IDE QT (Ver anexo 1.5) de la misma forma que se agiliza la creación de un paquete podemos compilar y hacer debugeo del paquete de ROS, dentro del IDE, ver Figura 2.13.

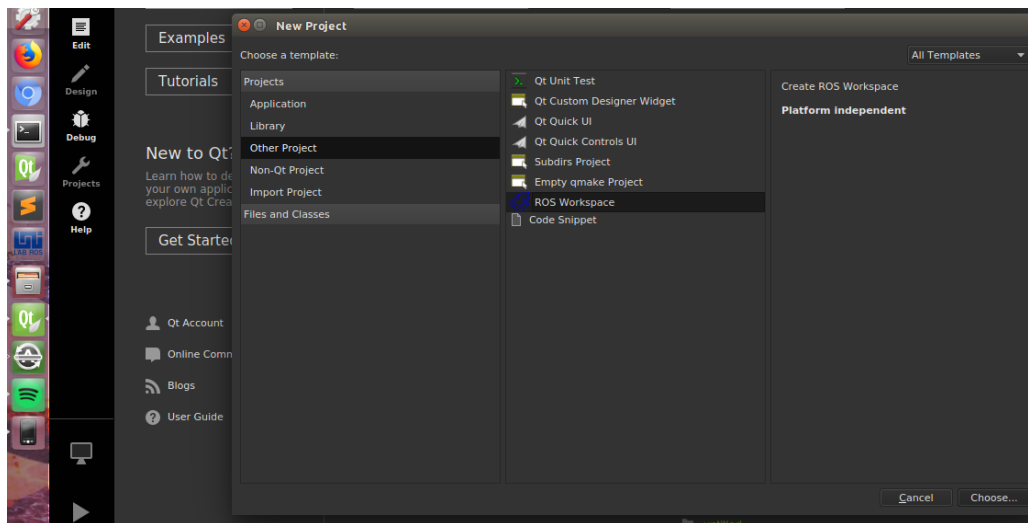


Figura 2.13: Plugin CATKIN con QT-Creator Creacion del Workspace de Trabajo.

2.2.3.2 Widgets QT

Widgets y formas creadas con Qt Designer se integran a la perfección con código programado bajo ROS, mediante el uso de las señales de Qt todo esto bajo el lenguaje C++,

2.2.3.3 Widgets QML

2.2.4 Librería Orocos

Orocos es el acrónimo de Open Robot Control Software. Su objetivo principal es desarrollar software libre de propósito general que se basa en un framework para el control de dispositivos robóticos. Soporta cuatro bibliotecas desarrolladas en C++:

- Real-Time Toolkit (RTT)
- Kinematics and Dynamics Library (KDL)
- Bayesian Filtering Library (BFL)
- Orocos Component Library (OCL)

La librería Kinematics and Dynamics Library (KDL) integrada en ROS [Cap 10.1.4] contiene clases y métodos implementados en su mayoría en C++, no obstante, también existen partes escritas en Python. Mediante la utilización de la librería somos capaces de calcular las soluciones al problema cinemático directo e

inverso. Estas soluciones se calculan de manera iterativa mediante el método de obtención de raíces Newton-Fourier /Rapson. Los métodos software utilizados son `CarToJnt()` para calcular la cinemática inversa y `JntToCart()` para calcular la directa. En la Ilustración 2 se puede observar un ejemplo de la API de KDL donde se aprecia el método `CartToJnt()`.

2.3 Modelos de los robots

El definir Simulación de robots puede significar varias cosas para diferentes roboticistas. Para ciertos roboticistas, la simulación de sistemas robóticos abarca la visualización de como un robot se mueve a través de su espacio. Estos simuladores están ampliamente basados en diseños CAD y herramientas de visualización grafica

Como se definió anteriormente es necesario contar con los modelos CAD de los robots para integrarlos a una simulación, de esta misma forma los fabricantes de Robots poseen sus propios simuladores (ver figura 2.8) de Robots, por lo tanto utilizan archivos CAD para visualizar el modelo del robot, estos mismos modelos están disponibles en las páginas WEB de los fabricantes más reconocidos como ABB, Fanuc, KUKA estos archivos están disponibles con mayor o menor Resolución como son STL, 3DS, VRML1, DXF, IGES, GoogleSketchUp, SLDASM, etc.

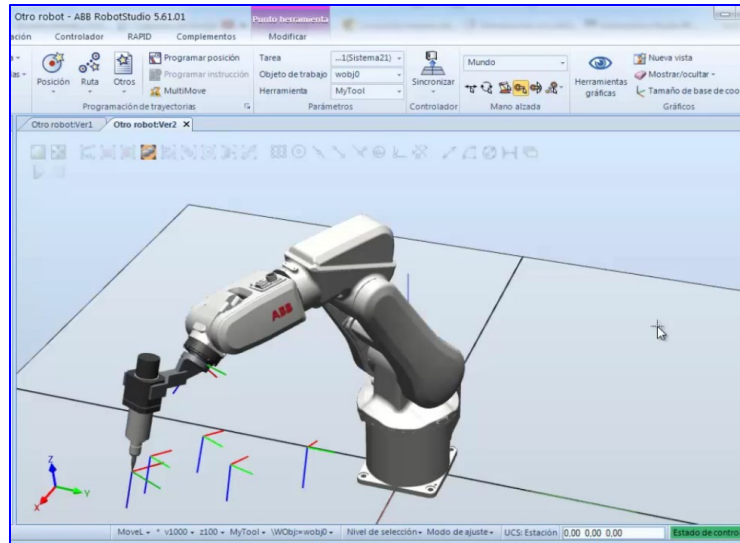


Figura 2 14: Captura de pantalla de Software ABB RobotStudio

2.3.1 Modelos de los Fabricantes de robots

Dentro del consorcio de industrias que posee ROS, llamado ROS-Industrial, se cuenta con la colaboración de empresas como ABB, FANUC, Yaskawa, KUKA, entre otras que colaboran proveyendo repositorios de libre acceso a la comunidad de roboticistas, dentro de esta colaboración se tiene acceso a los modelos de robots modelados por estas empresas con lenguaje descriptivo URDF que describen fielmente el modelo del robot tal como si se usara el modelo en el simulador del fabricante del robot.

2.3.2 Creado por el usuario

La creación de un nuevo modelo de robot es posible conociendo el lenguaje descriptivo URDF el cual consta de una serie de etiquetas XML que permiten especificar una serie de parámetros de cada eslabón y de cada articulación del robot, en los eslabones posible indicarle una etiqueta con referencia a un archivo CAD y proveer la información de su posición y orientación en el espacio.

2.3.2.1 URDF

Es necesario conocer que el modo descripción de etiquetas en un archivo URDF es prácticamente genérico para todo tipo de robots, desde un Vehiculó aéreo no

tripulado hasta un brazo manipulador, existen ciertas limitaciones que hacen imposible la descripción de ciertos robots. La principal limitación es que los robots deben ser robots series (estructura de árbol), descartando las cadenas cinemáticas cerradas.

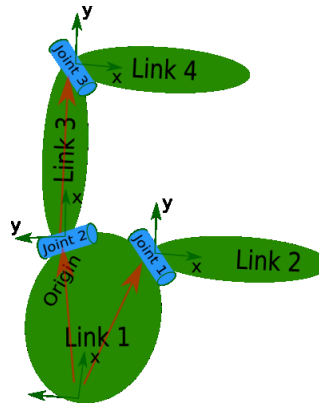


Figura 2.15: Representación de Links y Joints en un URDF.

El robot se interpreta como un conjunto de eslabones (links) unidos mediante un conjunto de articulaciones (joints) tal y como se muestra en la figura 2.9 de esta forma el archivo URDF poseerá etiquetas de lenguaje de marcado de la siguiente forma y clasificación a respetarse, a como se muestra en la figura 2.33, la etiqueta **robot** es la etiqueta general de un archivo URDF

```
<robot name="manipulador">
  <link name="link_1"> ... </link>
  <link name="link_n"> ... </link>
  <joint name="joint_1" type="revolute"> ... </joint>
  <joint name="joint_n" type="prismatic"> ... </joint>
</robot>
```

Figura 2.16: Estructura del lenguaje de marcado URDF

2.3.3 Soporte de ROS – Industrial

ROS – Industrial (ROS – I) es un consorcio de empresas que utilizan los recursos de ROS pero a nivel industrial, estas implementaciones de Software para aplicaciones Industriales a dado cabida de nuevos paquetes de software de ROS como MoveIt para la planeación de trayectorias para los robots industriales, además de la colaboración de los desarrolladores de las propias empresas como ABB, Fanuc, KUKA, Motoman, entre otras para proveer de paquetes que

interactúen con ROS por ejemplo, la capacidad de ROS – I de Comunicarse via un nodo de ROS con el controlador (CPU) de un Robot Industrial.

2.3.3.1 Estructura de Un paquete ROS-I

Dentro del desarrollo de este trabajo se utilizaron los paquetes de ROS Industrial estos poseen Archivos URDF de robots industriales, Archivos de Configuración de los Joints del robot, Archivos Launch para aplicaciones con Robots reales, Archivos CAD (meshes) y archivos de testeo de puesta en marcha de un robot.

```
abb_irb4600_support_package
├── urdf
├── config
├── launch
├── meshes
└── test
```

Los archivos URDF pueden ser configurados por nosotros mismo como se menciona en la sección 2.3.2 pero para LVR se usarán los Archivos URDF que han parametrizados las empresas unidas al consorcio de ROS tales como ABB, FANUC, Motoman etc. Garantizando de esta manera un robot simulado que cumple a cabalidad la hoja de datos proveída por el fabricante, en estos cumplimientos están la máxima apertura de los Joints en radianes, la inercia del robot en

Archivos CAD de los robots industriales

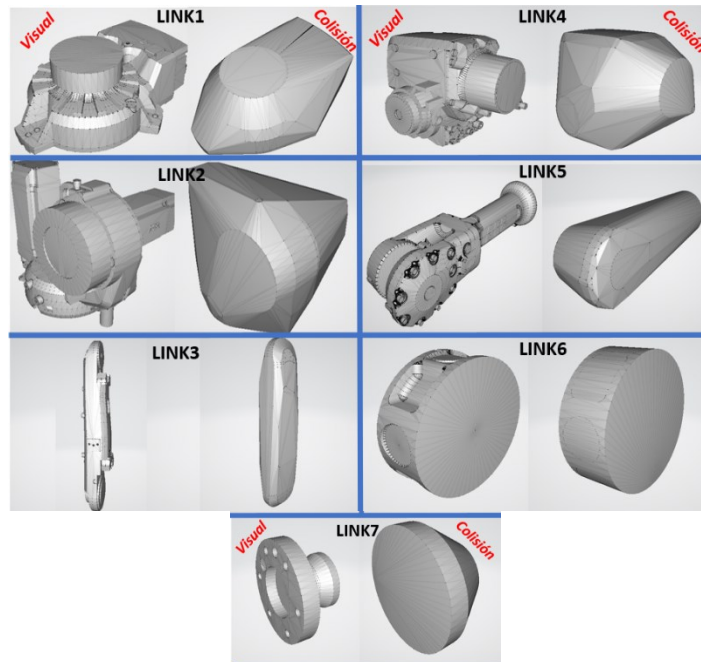


Figura 2.17: Modelos de CAD en 3D

2.4 Programación de los Robots.

2.4.1 Diseño de la interfaz interprete de comandos.

Cdas

Modelos de los robots

- Fabricantes
- Creado por el usuario

Programación de los robots

Documentación

Celda de trabajo

La Celda de Trabajo esta bajo Gazebo simulator, en este se puede incluir cualquier escenario diseñado en herramientas CAD, agregando los modelos a la carpeta de Gazebo

```
$ cd ./gazebo/models
```

Otros recursos más allá de los básicos

Se mencionó que en un laboratorio real la principal actividad es la programación del robot, para lo cual se utiliza el lenguaje del fabricante. En el laboratorio virtual se decidió incorporar otros recursos que le permitan al estudiante realizar otro tipo de actividad. Por ejemplo, el laboratorio cuenta con xxxx, lo cual permite al estudiantes visualizar el espacio de trabajo para el robot bajo estudio.

Interfaz gráfica de usuario (GUI)

Implementación del LVR

En el acápite anterior se mencionaron/describieron los diferentes recursos del laboratorio virtual. A continuación se explica el procedimiento y las herramientas utilizadas para la implementación de estos.

Modelos de los robots

En la actualidad se dispone de un sinnúmero de herramientas que pueden utilizarse para modelar un robot. En el mundo académico una de las herramientas es utilizadas es MATLAB, principalmente cuando el modelo es utilizado para la investigación en el campo de la robótica. Se pueden herramientas CAD, tal como SOLIDWORK, para la modelación del robot. (¿que ventajas tiene por ejemplo con respecto a AUTOCAD y MATLAB?).

Como se mencionó el laboratorio virtual incorpora modelos de robots de fabricantes tales como ABB, KUKA, etc. y por tal razón se utilizan los modelos suministrados por los fabricantes. (ventajas..) Dichos modelos son creados utilizando el formato de descripción de robot unificado (URDF, por sus siglas en inglés) lo cual facilita.....

¿Qué papel juega ROS en la implementación?

Programación de los robots

Documentación

Celda de trabajo

Otros recursos

Interfaz gráfica de usuario

- ✖ RVIZ es utilizado en el desarrollo del laboratorio virtual principalmente para visualizar la morfología y estudiar la cinemática de los robots.
- ✖ En el laboratorio virtual de robótica RQT se utilizó para graficar la estructura de datos correspondiente a la posición del robot durante su movimiento.
- ✖ el documento se presentarán las características de GAZEBO dado que es el simulador que se incorporo en el LVR en la practicas de programación del robot.
- ✖ De igual forma, cuenta con las herramientas y la guía para desarrollar sus modelos propios.
- ✖ La GUI está diseñada de tal forma que el estudiante podrá usar los recursos de manera transparente sin necesidad de tener que configurar aspectos básicos de ROS tales como su inicialización y el lanzamiento de nodos.

CAPÍTULO III: ANÁLISIS DE RESULTADOS

Qué experimentos podemos realizar para demostrar que los recursos del LVR permiten realizar, efectiva y eficientemente, ¿actividades relacionadas con los fundamentos de la robótica industrial?

¿Cuántos experimentos serían suficiente?

¿Cuáles serían los temas apropiados?

CAPÍTULO IV: CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

A continuación, son presentadas conclusiones sobre los objetivos específicos establecidos para lograr el objetivo general.

- .
- .
- .
- .
- .
- .
- .
- .

RECOMENDACIONES

A continuación, se presentan algunas recomendaciones que pueden contribuir a mejorar las prestaciones del prototipo desarrollado en este proyecto.

- .
- .
- .

Bibliografía

- Barrientos, A., Peñín, L. F., Balaguer, C., Aracil, R. (2007). Fundamentos de robótica. Madrid, España. McGraw-Hill/Interamericana de España, S. A. U.
- Craig, J. J. (2006). *Róbotica*, Mexico. Pearson Educación de México, S.A. de C.V.
- Jara, C., et al. (2011). Hands-on experiences of undergraduate students in Automatics and Robotics using a virtual and remote laboratory. *Computers & Education*, 57(4), 2451–2461. Obtenido de <https://doi.org/10.1016/j.compedu.2011.07.003>
- Candelas, F., Torres, F., Gil, P., Ortiz, F., Puente, S., & Pomares, J. (2004). Laboratorio virtual remoto para robótica y evaluación de su impacto en la docencia. *RIAI: Revista Iberoamericana de Automática e Informática Industrial*, 1(2), 49–57. Obtenido de <http://rua.ua.es/dspace/handle/10045/4609>
- Castellanos, F., & Martínez, O. (2010). Laboratorios virtuales (LV) como apoyo a las practicas a distancia y presenciales en Ingeniería. *INGE CUC*, 6(1), 267-280. Obtenido de <http://revistascientificas.cuc.edu.co/index.php/ingecuc/article/view/311>
- Ortega, J., Sánchez, R., González, J., & Reyes, G. (2016). Virtual laboratories for training in industrial robotics. *IEEE Latin America Transactions*, 14(2), 665-672. doi 10.1109/TLA.2016.7437208
- CNU (2018, abril) Repositorios Nicaragua. En Wellcome to the university repository of the CNU Obtenido de <http://repositorio.cnu.edu.ni/>
- Ramírez, K. (2018, abril) Material del Curso CI-2657. En Lista de Presentaciones Obtenido de <http://www.kramirez.net/ci-2657/materialci2657/>
- COSTARICAMAKERS. (2018, abril) Tag Archives: ROS: YA CONTAMOS PERSONAS... AHORA HAGAMOS ALGO CON PYTHON Obtenido de <http://costaricamakers.com/?tag=ros>
- Valverde, S. (2015). “Robótica inteligente: Implementación de sensores 3D para desenvolvimiento de robots móviles y vehículos autónomos” Obtenido de https://repositoriotec.tec.ac.cr/bitstream/handle/2238/6932/robotica_inteligente_implementaci%C3%B3n_sensores_desenvolvimiento.pdf?sequence=1&isAllowed=y.
- Rosas, L., Fuentes, M., Samaniego, C., Alvarado, R., & Valencia, J. (2013). *MODELADO Y CONTROL DEL ROBOT MÓVIL ROBOTNIK SUMMIT XL*.

- Obtenido de <http://cerescontrols.com/wp-content/uploads/2013/10/SUMMIT-XL-Informe-Final.pdf>
- Justina (2017, mayo) Justina. En user_manual Obtenido de https://github.com/RobotJustina/JUSTINA/blob/master/user_manual/user_manual.pdf
- Eagle X (2018, marzo) Un robot construido por alumnos del TEC. En Vinculación y prestigio Obtenido de <https://tec.mx/es/noticias/queretaro/vinculacion-y-prestigio/un-robot-construido-por-alumnos-del-tec>
- Juárez, G. (2008) IMPLEMENTACIÓN DE UN LABORATORIO VIRTUAL CON LA AYUDA DE LABVIEW, AL CURSO DE CIRCUITOS ELÉCTRICOS 1. Obtenido de http://biblioteca.usac.edu.gt/tesis/08/08_0148_ME.pdf
- Prieto, R., Zaldivar, U., & Bernal, R. (2010) Creación de un Laboratorio Virtual para Optimizar el uso de un Laboratorio de Robótica Real. Obtenido de https://www.academia.edu/374898/Creaci%C3%B3n_de_un_Laboratorio_Virtual_para_Optimizar_el_uso_de_un_Laboratorio_de_Rob%C3%B3tica_Real
- Ayala, J., Pupo, L., & Salazar, L. (2016) LABORATORIO VIRTUAL DE INSTRUMENTACIÓN Y CONTROL Obtenido de <http://www.informaticahabana.cu/sites/default/files/ponencias/EDU095.pdf>
- Owen, A. (14 de marzo de 2016). What is the Best Programming Language for Robotics? [Entrada en un blog] Robotiq. Recuperado de <https://blog.robotiq.com/what-is-the-best-programming-language-for-robotics>
- Tellez, R. (2017). A thousand robots for each student: Using cloud robot simulations to teach robotics. En Robotics in Education, 143-155. Springer, Cham.
- Movelt! (s. f.) Concepts. En Documentacion Obtenido de <http://moveit.ros.org/documentation/concepts/>
- Corke, P. I. (1996). A robotics toolbox for MATLAB. IEEE Robotics & Automation Magazine, 3(1), 24-32. doi 10.1109/100.486658
- ROS (s. f.) Core components. En Communications Infrastructure Obtenido de <http://www.ros.org/core-components/>
- Charles River Analytics Inc (2018). [fotografía]. robot_localization Recuperado de <https://www.cra.com/work/case-studies/robotlocalization>

- Pitzer, B., Osentoski, S., Jay, G., Crick, C., & Jenkins, O. C. (2012, May). Pr2 remote lab: An environment for remote development and experimentation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on* 3200-3205. IEEE.
- Casañ, G. A., Cervera, E., Moughlbay, A. A., Alemany, J., & Martinet, P. (2015). *ROS-based online robot programming for remote education and training. In Robotics and Automation (ICRA), 2015 IEEE International Conference on* 6101-6106.
- Universidad Tecnológica La Salle (2017) Ingeniería en Mecatrónica y Sistemas de Control. En Plan de estudio Obtenido de <http://www.ulsal.edu.ni/index.php/ingenieria-en-mecatronica-y-sistemas-de-control>
- Calvo, I., Zulueta, E., Gangoiti, U., López, J. M., Cartwright, H., & Valentine, K. (2009). *Laboratorios remotos y virtuales en enseñanzas técnicas y científicas* (Vol. 3, No. 3, pp. 1-21). Ikastorratza.
- Pérez, X., Cerda, A., & Incer, W. (2016) Desarrollar un laboratorio virtual para la realización de prácticas de laboratorio en la disciplina de control automático y automatización industrial
- ROS Industrial (2016, noviembre) ROS QTC Plugin. En Repositorio Obtenido de https://github.com/ros-industrial/ros_qtc_plugin
- ROS-Industrial (2017, Marzo) En ROS-Industrial Obtenido de <http://wiki.ros.org/Industrial/Tutorials>
- Willow Garage. (2010, Abril) En Comic: Reinventing the Wheel [Entrada en un blog] Obtenido de <http://www.willowgarage.com/blog/2010/04/27/reinventing-wheel>
- Merino, E. (2015) DISEÑO DE UN SIMULADOR DE COMPILADOR PARA PLATAFORMA MOODLE E IMPLEMENTACIÓN DE UN LABORATORIO VIRTUAL PARA LA ENSEÑANZA DE PROGRAMACIÓN.
- Guerrero, L., Gómez, D., Sandoval, E., Thomson, P., Marulanda, J. (2014). SISMILAB, UN LABORATORIO VIRTUAL DE INGENIERÍA SÍSMICA, Y SU IMPACTO EN LA EDUCACIÓN.
- ROBOTIS Co., Ltd. (2017). [fotografía]. *ROS Robot Programming*. Obtenido de <https://community.robotsource.org/t/download-the-ros-robot-programming-book-for-free/51>

ANEXOS

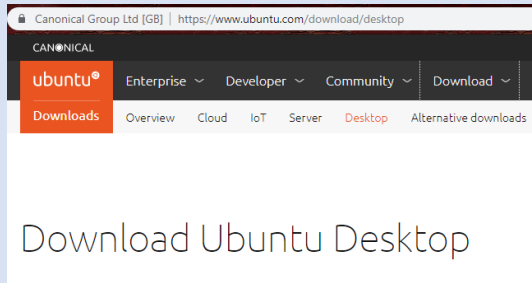
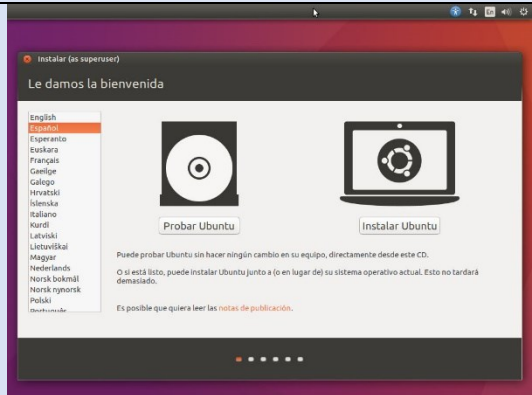
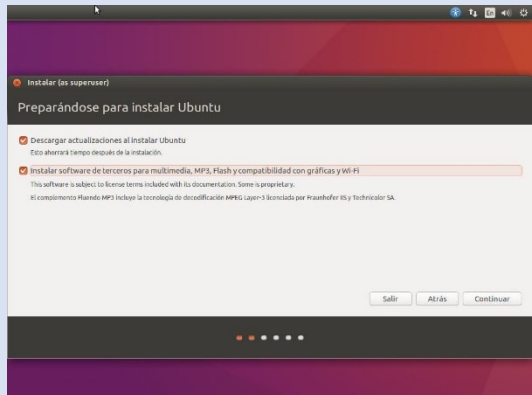
A: Instalación de Recursos de Software para el Laboratorio Virtual.

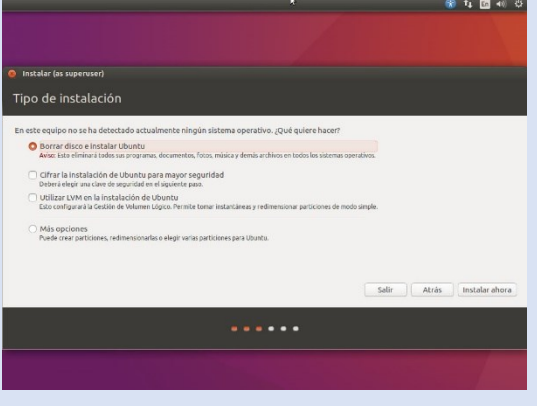
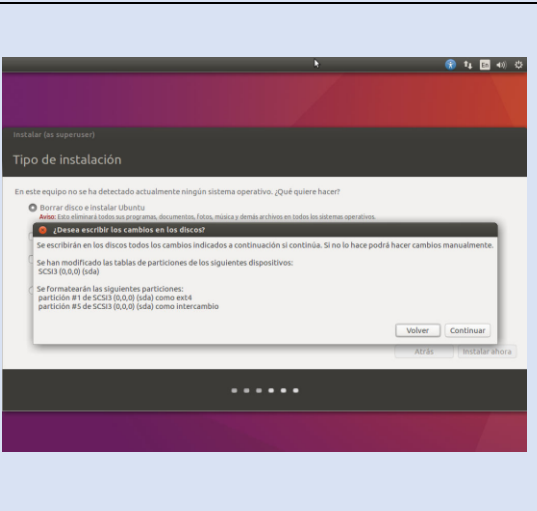
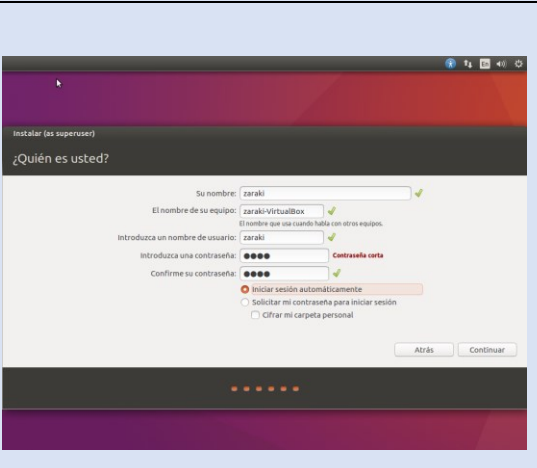

La instalación de todos los recursos de software expuestos fue utilizando una laptop Intel Core I3 con 8GB de RAM y 120 GB de espacio libre de almacenamiento en el disco duro.

A.1 Instalación de UBUNTU 16.04 Xenial

La tabla A.1 detalla el proceso de instalación:

Tabla A.1: Instalación de UBUNTU 16.04 Xenial

Paso 0	Requisitos Preliminares	
	<ul style="list-style-type: none"> ✓ Disponer de al menos 10 GB de espacio de almacenamiento libre. ✓ Tener acceso a un DVD o una unidad flash USB que contenga la versión de Ubuntu 16.04 ✓ Descargar ISO ingresando al siguiente link: https://www.ubuntu.com/download/desktop 	
Paso 1	Arranque desde una unidad flash USB	
	<p>Configurar la BIOS de la computadora para que se inicie automáticamente desde USB. Basta con insertar la unidad flash USB y encender el ordenador o reiniciarlo. Aparecerá la ventana de bienvenida, solicitando que elija el idioma e instale o pruebe Ubuntu.</p>	
Paso 2	Preparado para instalar Ubuntu	
	<p>Después de elegir instalar Ubuntu desde la ventana de bienvenida, se le preguntará sobre las actualizaciones y el software de terceros.</p> <p>Se aconseja activar la descarga de las actualizaciones y la instalación de software de terceros</p>	

Paso 3	<p align="center">Asignar Espacio</p> <p>Utilizar las casillas de verificación para elegir si desea instalar Ubuntu junto con otro sistema operativo.</p> <p>Se aconseja disponer de un espacio de almacenamiento mayor a 10GB esto por las herramientas de visualización de ROS que luego serán instaladas como Gazebo.</p>	
Paso 4	<p align="center">Inicio de Instalación de Ubuntu en la Unidad de Disco Asignada.</p> <p>Después de configurar el almacenamiento, haga clic en el botón "Instalar ahora". Aparecerá un pequeño panel con una visión general de las opciones de almacenamiento que ha elegido, con la posibilidad de volver si los detalles son incorrectos. Haga clic en Continuar para iniciar el proceso de instalación o en volver para modificar.</p>	
Paso 5	<p align="center">Detalles de registro.</p> <p>El nombre del equipo es la forma en que su computadora aparecerá en la red y ROS lo ocupará como identificador dentro del servidor ROSMaster, mientras que el nombre de usuario será su nombre de usuario y de cuenta. A continuación, ingrese una contraseña segura. El instalador le informará si es demasiado débil.</p>	
Paso 6	<p align="center">Instalación Completa.</p> <p>Después de que todo se ha instalado y configurado, una pequeña ventana aparecerá pidiendo que reinicie la máquina. Hacer clic en Reiniciar ahora y extraer el DVD o la unidad flash USB</p>	

A.2 Instalación de ROS Kinetic

La tabla A.2 detalla el proceso de instalación:

Tabla A.2: Instalación de ROS Kinetic

Paso 1	Permitir todas las Fuentes de ROS
	El paquete ROS no está soportado oficialmente en el conjunto de paquetes de UBUNTU para eso es necesario instalarlo desde un link externo, Se habilita el computador a que acepte paquetes de <i>packages.ros.org</i>
	<pre>\$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu \$(lsb_release -cs) main" > /etc/apt/sources.list.d/ros-latest.list'</pre>
Paso 2	Configuración del key
	Se crea el directorio ros-latest.list para el soporte de los paquetes que conforman ROS además que ciertos repositorios necesitan de instalación de Claves de autorización o .key
	<pre>\$ wget http://packages.ros.org/ros.key -O - sudo apt-key add -</pre>
Paso 3	Instalación general
	Primero se debe asegurar que Ubuntu este actualizado, mediante:
	<pre>\$ sudo apt-get update</pre>
	Luego que la actualización se complete, se puede instalar la versión completa de Kinetic, la cual es la más recomendada, mediante
Paso 4	<pre>\$ sudo apt-get install ros-kinetic-desktop-full</pre>
	Inicialización de ROS
	rosdep habilita la fácil instalación de dependencias en el código a compilar y además es requerido para correr algunas dependencias de los componentes de ROS, Ejecutar los comandos siguientes, uno a la vez:
	<pre>\$ sudo rosdep init \$ rosdep update</pre>
Paso 5	Configuración del entorno de trabajo de ROS para Ubuntu
	Existen usuarios que poseen varias distribuciones de ROS con estos comandos permites inicializar con cual se trabaja. Permitiendo que las variables del entorno ROS sean automáticamente añadidas a la sesión bash cada vez que abrimos una nueva ventana Shell, Ejecutar los comandos siguientes, uno a la vez:
	<pre>\$ source /opt/ros/kinetic/setup.bash \$ source ~/catkin_ws /devel/setup.sh</pre>

A.3 Espacio de trabajo Catkin

La tabla A.3 muestra el proceso de creación de un espacio de trabajo para compilar los paquetes de ROS:

Tabla A. 3 Creación del Espacio de Trabajo Catkin.

Paso 1	Creación del Espacio de trabajo ROS
	El compilador necesita que se le indique la ruta de acceso a los archivos a ser compilados para esto se construye un entorno de trabajo (catkin workspace), <i>Ejecutar en una terminal</i>
	<code>\$ mkdir -p ~/catkin_ws/src</code>
Paso 2	Inicialización del Espacio de trabajo
	Con las carpetas creadas accedemos a la carpeta donde estarán almacenados los archivos fuentes de C++ a ser compilados. Ejecutamos los dos comandos uno a la vez.
	<code>\$ cd ~/catkin_ws/src</code> <code>\$ catkin_init_workspace</code>
Paso 3	Construcción del Espacio de trabajo (Building)
	Ya que se ha inicializado el espacio de trabajo procedemos a compilar este mismo, debido a que no hemos agregado ningún paquete a compilar este no dilatara en crear todo el directorio necesario del espacio de trabajo.
	<code>\$ cd ~/catkin_ws/</code> <code>\$ catkin_make</code>
Paso 4	Creación de un paquete.
	Para la creación de un paquete nuevo utilizamos Catkin con el cual podemos hacer referencia a los recursos de software de otros paquetes necesarios para nuestros requerimientos, los paquetes más comunes son roscpp, sensor_msgs, geometry_msgs
	<code>\$ catkin_create_pkg -D "Descripción del paquete" -a Yeser --rostdistro KINETIC mi_robot roscpp sensor_msgs</code>

A.4 Instalación de QT – Creator.

La tabla A.4 muestra el proceso de Instalación del IDE QT – Creator.

Tabla A.4: Instalación del IDE QT- Creator.

Paso 1	Obtención del Instalador
	Ingresamos a la Pagina WEB www.QT57Download.com o via comando procedemos. <i>Ejecutar en una terminal</i>
	\$ wget http://download.qt.io/official_releases/qt/5.7/5.7.0/qt-opensource-linux-x64-5.7.0.run
Paso 2	Proveer Permisos al archivo.
	Procedemos a garantizarle los permisos al archivo descargado para la instalación. \$ chmod +x qt-opensource-linux-x64-5.7.0.run
Paso 3	Instalación
	Dentro de la instalación se abrirá una ventana en la cual se indicará si se desea una instalación completa o parcial del IDE, Preferiblemente indicarle instalación completa para no tener problemas con librerías de QT que puedan ser relevantes. \$./qt-opensource-linux-x64-5.7.0.run

A.5 Instalación del Plugin ROS para QT – Creator.

La Tabla A.5 muestra el proceso de Instalación del Plugin de ROS desarrollado por ROS – Industrial para el IDE QT – Creator.

Tabla A.5: Instalación de Plugin ROS QT.

Paso 1	Instalación de dependencias
	Instalamos via terminal las dependencias de el repositorio de Levi Armstrong para Qt versión Xenial. <i>Ejecutar en una terminal los comandos uno a la vez.</i>
	\$ sudo add-apt-repository ppa:levi-armstrong/qt-libraries-xenial \$ sudo add-apt-repository ppa:levi-armstrong/ppa
Paso 2	Instalación del Plugin
	Actualizamos e instalamos el plugin de ROS para la versión de Qt5.7 \$ sudo add-apt-repository ppa:levi-armstrong/qt-libraries-xenial

Paso 3	Configuración del sistema
	Se configura el IDE de QT a reconocer las librerías de la versión de Qt 5,7, cabe destacar que esto se hace debido a que nativamente Ubuntu posee librerías Qt para ejecutar ciertos procesos. Editamos via comando el archivo .conf de QT
	<pre>\$ sudo gedit /usr/lib/x86_64-linux-gnu/qt-default/qtchooser/default.conf</pre> Reemplazamos los parámetros: <pre>/usr/lib/x86_64-linux-gnu/qt4/bin</pre> <pre>/usr/lib/x86_64-linux-gnu</pre> Por: <pre>/opt/qt57/bin</pre> <pre>/opt/qt57/lib</pre>
Paso 4	Clonación del espacio de trabajo Recomendado para el Plugin
	Como se ha mencionado en la sección 2.2 de instalación de recursos es necesario el uso de un espacio de trabajo de compilación de ROS, el repositorio ros_qtc_plugin es recomendado para una buena instalación del plugin
	<pre>\$ git clone -b master https://github.com/ros-industrial/ros_qtc_plugin.git</pre>
Paso 5	Iniciación del workspace y de las librerías de ROS.
	Una Vez descargado el repositorio procedemos a inicializar El espacio de trabajo con el cual el IDE Qt Podrá compilar vía CMake los paquetes de ROS
	<pre>\$ cd ~/ros_qtc_plugin/devel</pre> <pre>\$ bash setup.sh -d</pre>

Con los pasos descrito en la tabla A.5 Podemos abrir el IDE Qt Creator y Inicializar el Plugin Tal como se muestra en las siguientes Imágenes.

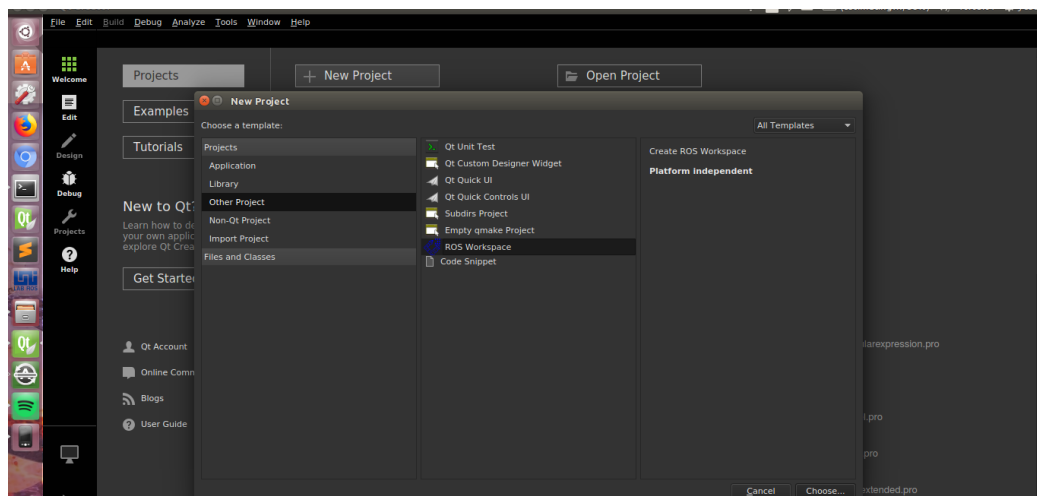


Figura A.1: IDE QT-Creator Activación del Plugin Catkin, para el espacio de Trabajo.

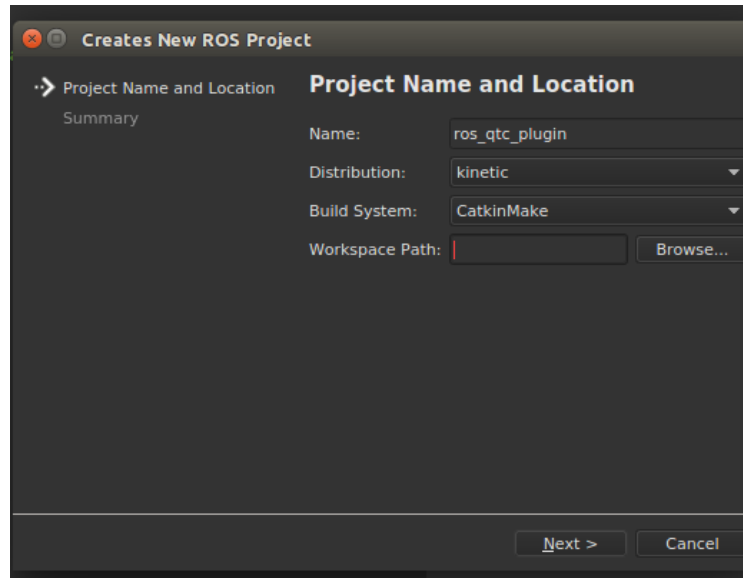


Figura A.2: Ubicación del espacio de Trabajo.

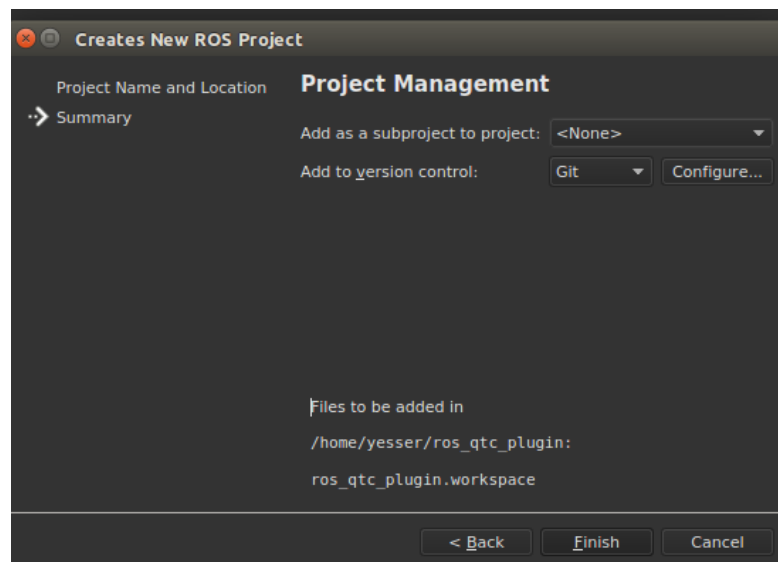


Figura A.3: Espacio de trabajo Creado.