



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Método para el desarrollo de aplicaciones en robótica de servicios basados en industria 4.0 y computación en la nube

Alejandro Serrate Hincapié

Universidad Nacional de Colombia

Facultad de Minas

Medellín, Colombia

2021

Método para el desarrollo de aplicaciones en robótica de servicios basados en industria 4.0 y computación en la nube

Alejandro Serrate Hincapié

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título
de:

Magister en Ingeniería de sistemas y ciencias de la computación

Director (a):

Jovani Alberto Jiménez Builes, Ph. D.

Codirector (a):

Gustavo Alonso Acosta Amaya, Ph.D.

Línea de Investigación: Robótica de servicios

Grupo de Investigación:

Grupo de Investigación y Desarrollo en Inteligencia Artificial, GIDIA

Universidad Nacional de Colombia

Facultad de Minas

Medellín, Colombia

2021

(Dedicatoria o lema)

A mis padres que me han brindado todo su amor y cariño, me han formado y brindado todo para superarme a lo largo de la vida. Gracias por enseñarme a nunca rendirme y a luchar siempre por mis sueños y metas.

A mi compañera de vida que ha estado apoyándome para ser mejor persona y ser la mejor versión de sí mismo cada día.

Agradecimientos

El autor expresa su agradecimiento a:

Gustavo Alonso Acosta Amaya, Ph. D. en Ingeniería - sistemas. Profesor del Politécnico Jaime Isaza Cadavid. Gracias por su apoyo incondicional durante tantos años de formación profesional. Su asesoría fue determinante para el desarrollo del trabajo.

Jovani Alberto Jiménez Builes Ph. D. en Ingeniería – sistemas. Profesor titular de la Universidad Nacional de Colombia. Gracias por su valioso acompañamiento en cada uno de los procesos formativos y de acompañamiento para la consecución del trabajo.

Yodi Pino Vargas, Ingeniera informática del Politécnico Jaime Isaza Cadavid por sus conocimientos en desarrollo de software para el desarrollo de este trabajo.

Resumen

En el presente trabajo se llevó a cabo el desarrollo de una metodología para el desarrollo de aplicaciones en robótica de servicios basado en industria 4.0 y computación en la nube. La primera parte del trabajo consiste en caracterizar los elementos de industria 4.0 aplicables a la robótica de servicios para interiores. Una vez definidos, se identificaron los modelos de computación en la nube aplicables a la robótica de servicios, y luego se incorporaron técnicas en inteligencia artificial para el control de la navegación de robots móviles en entornos interiores. Después se integraron los elementos de la industria 4.0 y computación en la nube, para la navegación autónoma de un sistema robótico. Con lo anterior se construyó una metodología, adicionándole elementos como sensorica, locomoción, manipulación, desarrollo web, fuentes de datos, entornos de desarrollo en la nube, simulación, arquitecturas de control, y marcadores fiduciales visuales. Finalmente se validó el método propuesto a través de la realización de un conjunto de pruebas en un escenario logístico, generando con ello resultados y conclusiones.

Palabras clave: Industria 4.0, computación en la nube, simulación, marcadores fiduciales, arquitectura de control, locomoción, manipulación.

Abstract

Method for the development of applications in robotics of services based on industry 4.0 and cloud computing

In the present work, the development of a methodology for the development of applications in robotics of services based on industry 4.0 and cloud computing was carried out.

The first part of the work consists of characterizing the elements of Industry 4.0 applicable to indoor service robotics. Once defined, the cloud computing models applicable to service robotics were identified, and then artificial intelligence techniques were incorporated to control the navigation of mobile robots in indoor environments. Afterwards, the elements of Industry 4.0 and cloud computing were integrated, for the autonomous navigation of a robotic system. With the above, a methodology was built, adding elements such as sensorics, locomotion, manipulation, web development, data sources, development environments in the cloud, simulation, control architectures, and visual fiducial markers. Finally, the proposed method was validated by carrying out a set of tests in a logistic setting, thereby generating results and conclusions.

Keywords: Industry 4.0, cloud computing, simulation, fiducial markers, control architecture, locomotion.

Contenido

	Pág.
Resumen	IX
Lista de figuras	XIII
Lista de tablas	XVI
Lista de Símbolos y abreviaturas	XVII
1. Introducción	19
1.1 Planteamiento del problema.....	21
1.2 Objetivos.....	23
1.2.1 Objetivo General.....	23
1.2.2 Objetivos Específicos.....	23
1.3 Marco Teórico	23
1.3.1 Robots autónomos.....	24
1.3.2 Simulaciones	25
1.3.3 IoT (Internet de las cosas)	26
1.3.4 Computación en la nube	26
1.4 Metodología	27
1.5 Alcances y limitantes.....	28
2. Elementos de la industria 4.0 aplicables a la robótica de servicios	30
2.1 Robótica avanzada, flexible y colaborativa	31
2.1.1 Controles	32
2.1.2 Condiciones de funcionamiento	32
2.1.3 Comunicación	32
2.1.4 Características Estructurales	32
2.1.5 Complejidad.....	32
2.2 Sensorica avanzada a nivel de fábrica	34
2.2.1 Robótica guiada por visión.....	34
2.3 Sistemas ciberfisicos (CPS)	37
2.4 Robótica en la nube	39
2.4.1 Arquitectura orientada a Servicios (SOA).....	40
2.4.2 Robótica en la nube a nivel de sistema (RaaS)	41
2.4.3 Robot Management System (RMS):	41
2.4.4 <i>Cloud Robotics Administrator</i>	42
2.4.5 <i>Virtual Robot System (VRS)</i>	42
3. Modelos de computación en la nube aplicables a la robótica de servicios ...	45
3.1 Arquitectura de comunicación M2M/M2C	46
3.2 Arquitectura de computación en la nube elástica	47
3.2.1 Modelo basado en pares	48
3.2.2 Modelo basado en proxy.....	48
3.3 Arquitectura de desarrollo de robots basado en la nube.....	49
3.4 Arquitectura <i>RoboWeb System</i>	50

4.	Desarrollo del método para la construcción de aplicaciones en robótica de servicios.	53
4.1	Pasos de la metodología.....	56
4.1.1	Levantamiento de requisitos.....	56
4.1.2	Selección de tecnologías.....	58
4.1.3	Sensórica Inmersa en el robot.....	58
4.1.4	Sensórica incorporada al entorno.....	59
4.1.5	Locomoción.....	60
4.1.6	Manipulación.....	62
4.1.7	Desarrollo de la aplicación.....	64
4.1.8	Simulación.....	65
4.1.9	Desarrollo web.....	66
4.1.10	Análisis jerárquico de tareas.....	68
4.1.11	Fuentes de Datos.....	69
4.1.12	Selección de Control y navegación.....	70
4.1.13	Análisis cinemático y dinámico para Sistema de Manipulación.....	76
4.1.14	Marcadores fiduciales visuales y sistemas de decodificación.....	78
5.	Validación experimental y resultados	81
5.1	Descripción del experimento.....	81
5.2	Establecimiento de requisitos.....	82
5.3	Selección de tecnologías.....	84
5.3.1	Sensórica.....	84
5.3.2	Locomoción y manipulación.....	85
5.3.3	Entorno de desarrollo.....	88
5.3.4	Simulación.....	90
5.3.5	Desarrollo web.....	91
5.3.6	Fuente de datos.....	95
5.3.7	Control de navegación y manipulación.....	97
5.3.8	Marcadores fiduciales.....	106
5.4	Resultados y análisis.....	108
5.4.1	Validación de comportamiento de aproximación para la toma y descarga de objetos.....	110
5.4.2	Validación de comportamiento evasión de obstáculos.....	111
5.4.3	Validación de comportamiento orientación al destino.....	114
5.4.4	Validación de comportamiento avanzar al destino.....	115
5.4.5	Prueba de validación para brazo manipulador en la toma de objeto.....	115
5.4.6	Prueba de validación para brazo manipulador en la entrega de objeto.....	117
5.4.7	Prueba de validación de conmutación de los comportamientos.....	118
5.4.8	Prueba de validación de tiempos de respuesta.....	119
5.4.9	Prueba de validación de entregas exitosas.....	121
6.	Conclusiones y recomendaciones	122
6.1	Conclusiones.....	122
6.2	Recomendaciones.....	124
	Bibliografía	127

Lista de figuras

	Pág.
Figura 2-1: Elementos principales de la industria 4.0.....	30
Figura 2-2: Comparación de robots no colaborativos y robots colaborativos	33
Figura 2-3: Gammas de robots colaborativos y móviles industriales Omron.....	33
Figura 2-4: Cámara RGB-D Kinect	37
Figura 2-5: Elementos principales del ROS-TMS	38
Figura 2-6: Componentes de SOA a nivel de computación en la nube	40
Figura 2-7: Descripción general de alto nivel de la comunicación entre RMS y ROS.....	41
Figura 2-8: Arquitectura VRS.....	42
Figura 3-1: Comunicación M2C y M2M en robótica cloud.....	46
Figura 3-2: Arquitectura de computación en la nube elástica: a) Modelo basado en pares b) Modelo basado en proxy c) Modelo basado en clones	48
Figura 3-3: Arquitectura del IDE basado en la nube	50
Figura 3-4: Arquitectura del sistema RoboWeb	51
Figura 4-1: Metodología propuesta para el diseño de aplicaciones robóticas basado en industria 4.0 y computación en la nube.	54
Figura 4-2: Ejemplo de requisitos técnicos de un proyecto de robótica.	57
Figura 4-3: Ejemplo de especificaciones meta u objetivos.....	58
Figura 4-4: Configuraciones de locomoción.....	61
Figura 4-5: Tipos de robots manipuladores	63
Figura 4-6: Arquitectura cliente servidor aplicado a robótica móvil en el uso de tareas jerárquicas por medio de una GUI.....	66
Figura 4-7: Interfaz gráfica para ejecución de tareas de una celda de ensamble por medio de un robot manipulador.....	67
Figura 4-8: Representación de un análisis jerárquico de tareas	68
Figura 4-9: Paradigma de arquitectura basada en comportamientos.....	70
Figura 4-10: Representación de mapas Geométricos.....	71
Figura 4-11: Representación de mapas Topológicos.....	72
Figura 4-12: Representación de mapas semánticos.....	73
Figura 4-13: Grafo de visibilidad en un entorno de dos obstáculos.....	74
Figura 4-14: Mapa de una casa aplicando diagramas de Voronoi.	75
Figura 4-15: Representación gráfica de método de descomposición de celdas.....	76
Figura 5-1: Entorno de diseño experimental en el simulador Gazebo.....	81
Figura 5-2: Relación de clientes con el ciclo de vida del robot.....	82
Figura 5-3: Apartado de metodología correspondiente a la sensorica	84

Figura 5-4: Apartado de metodología correspondiente a la locomoción y manipulación.	86
Figura 5-5: Dimensiones de plataforma robótica y brazo manipulador	87
Figura 5-6: Plataforma Turtlebot 3 integrada con OpenManipulator	87
Figura 5-7: Apartado de metodología correspondiente al entorno de desarrollo.....	88
Figura 5-8: Entorno de desarrollo ROS development Studio (RDS).	88
Figura 5-9: Interfaz gráfica de Azure DevOps	90
Figura 5-10: Apartado de metodología correspondiente a la simulación	90
Figura 5-11: Simulador Gazebo con robot <i>turtlebot 3 open manipulator</i>	91
Figura 5-12: Apartado de metodología correspondiente al desarrollo web	91
Figura 5-13: Arquitectura de <i>rosbridge</i> integrada con ROS	92
Figura 5-14: Análisis jerárquico de la aplicación web logística robótica.....	93
Figura 5-15: Interfaz de inicio	94
Figura 5-16: Interfaz de pedidos	94
Figura 5-17: Interfaz para ingresar datos	94
Figura 5-18: Interfaz de ejecución de pedidos.....	95
Figura 5-19: Interfaz de teleoperación.....	95
Figura 5-20: Apartado de metodología correspondiente a las fuentes de datos	96
Figura 5-21: Interfaz gráfica MySQL inmersa en el servidor XAMPP®.....	97
Figura 5-22: Apartado de metodología correspondiente al control de navegación y manipulación.....	98
Figura 5-23: Representación de distancia y error angular con respecto al Marcador	99
Figura 5-24: Conos de detección en sensor laser	100
Figura 5-25: Arquitectura de control reactiva.....	102
Figura 5-26: Etapas de configuración en el asistente de MoveIt!	103
Figura 5-27: Interfaz gráfica de MoveIt!.....	104
Figura 5-28: Interfaz gráfica de Rviz integrado con Moveit!.....	104
Figura 5-29: Uniones del brazo manipulador y efector final.....	105
Figura 5-30: Apartado de metodología correspondiente a los marcadores fiduciales ...	106
Figura 5-31: Procedimiento para el uso de la herramienta <i>ar_track_alvar</i>	107
Figura 5-32: Identificación de los marcadores fiduciales en la herramienta Rviz	108
Figura 5-33: Recorrido de robot móvil en una misión logística.	109
Figura 5-34: Error angular y velocidad angular en función de los periodos de control..	110
Figura 5-35: Velocidad lineal en función de la distancia recorrida	111
Figura 5-36: Prueba de validación de comportamiento de evasión de obstáculos para la distancia del cono derecho	112
Figura 5-37: Prueba de validación de comportamiento de evasión de obstáculos para la distancia del cono central.....	112
Figura 5-38: Prueba de validación de comportamiento de evasión de obstáculos para la distancia del cono izquierdo	113
Figura 5-39: Prueba de validación de comportamiento de orientación al destino	114
Figura 5-40: Prueba de validación de comportamiento avanzar al destino.....	115
Figura 5-41: Velocidades de articulaciones en brazo manipulador al tomar el objeto...	116
Figura 5-42: Esfuerzo de articulaciones en brazo manipulador al tomar el objeto.	116

Figura 5-43: Velocidades de articulaciones en brazo manipulador al entregar el objeto	117
Figura 5-44: Esfuerzo de articulaciones en brazo manipulador al entregar el objeto	118
Figura 5-45: Validación de conmutación de los comportamientos	119

Lista de tablas

	Pág.
Tabla 2-1: Comparación entre sensores pasivos y activos	36
Tabla 2-2: Acrónimo de investigaciones enfocadas en diferentes elementos de la industria 4.0	43
Tabla 4-1: Ventajas y desventajas de los diferentes tipos de sensores para robótica móvil	59
Tabla 4-2: Ventajas y desventajas de los diferentes tipos de locomoción para robótica móvil	61
Tabla 4-3: Ventajas y desventajas de los diferentes brazos manipuladores	63
Tabla 4-4: Comparación entre Servidor nube y servidor local.....	64
Tabla 4-5: Características del software de control de versiones	65
Tabla 4-6: Descripción de Cinemática directa e inversa	77
Tabla 4-7: Características de los métodos para el análisis dinámico	77
Tabla 4-8: Características de los marcadores fiduciales visuales	78
Tabla 5-1: Necesidades asociadas al robot manipulador logístico.....	83
Tabla 5-2: sensores empleados en la validación experimental	85
Tabla 5-3: Características técnicas turtlebot 3.....	86
Tabla 5-4: Características técnicas del OpenManipulator-X	87
Tabla 5-5: Características de las herramientas de XAMPP®.....	96
Tabla 5-6: Posiciones del brazo manipulador para la aplicación logística.....	105
Tabla 5-7: Tiempos de ejecución para destino con marcador fiducial AR tag 7	120
Tabla 5-8: Tiempos de ejecución para destino con marcador fiducial AR tag 9	120
Tabla 5-9: Tiempos de ejecución para destino con marcador fiducial AR tag 11	120

Lista de Símbolos y abreviaturas

Abreviatura	Término
ROS	<i>Robot Operating System</i>
SLAM	<i>Localization And Mapping</i>
OSRF	<i>Open Source Robotics</i>
SRC	<i>Foundation Source</i>
CAD	<i>Computer-aided desing</i>
RaaS	<i>Robotic as service</i>
PaaS	<i>Plataform as service</i>
IaaS	<i>Infraestructure as servie</i>
HRC	<i>Human-Robot Collaboration</i>

1. Introducción

La robótica de servicios se ha convertido en una herramienta indispensable en la industria y para el día a día del hombre. Con la llegada de la industria 4.0 los robots móviles han tomado gran protagonismo gracias los alcances que se han tenido a nivel tecnológico y computacional. En los ámbitos de aplicación de los robots móviles encontramos:

El hospitalario. En este ámbito que requiere gran cantidad de personal y disponibilidad los robots actuales están en la capacidad de transportar medicamentos y suministros propios para intervenciones quirúrgicas, realizar tomas detalladas de rayos x, preparación de medicamentos en dosis puntuales, de igual manera desenvolverse en un centro asistencial, interactuando con personas y ascensores. Investigadores han enfocado sus esfuerzos en desarrollar *frameworks* para administrar un grupo de robots en ambientes hospitalarios, con el fin de apoyar el personal médico, en actividades logísticas para entrega de medicamentos, la limpieza de derrames y patrullaje. Se plantea un marco de decisión para la gestión de la operación del sistema robótico lo cual comprende la configuración, ubicación, energía de los robots, asignación de misiones y control en tiempo real. Esto implica el diseño de diferentes arquetipos de robots que cubran las tareas y necesidades previamente mencionadas (Schuessler, & Strohaber, 2018) (Kuhner, Fiederer, & Aldingerad, 2019).

La asistencia personal. Los robots de servicio para la asistencia de personas se han trabajado desde años recientes y ha enfocado sus esfuerzos en apoyar personas con diferentes discapacidades. A pesar de que estos robots son cada vez son más asequibles se ha presentado una brecha importante en la interacción con personas y esto debido a que las interfaces de control generalmente se vuelven más dispendiosas. Esto debido a la complejidad misma de las tareas, las modalidades que se emplean para personas con dificultades como el tacto, él habla o los gestos no son más las adecuadas. Si bien varios usuarios pueden realizar el esfuerzo de familiarizarse con un sistema robótico, otros con discapacidades motoras pueden no llegar a ser capaces a controlar estos sistemas y requieren más asistencias robóticas. La robótica actual enfoca sus esfuerzos en asistentes

de servicio robótico de forma en que solo sea necesario usar sus pensamientos, se plantea una interfaz cerebro-computadora (BCI) que emplea la grabación de una señal neuronal no invasiva y aprendizaje profundo coadaptativo y planificación de tareas de alto nivel basadas en expresiones de referencia (Ozaki, Kagaya, & Satoshi Hirano, 2013). Otras investigaciones se han centrado en desarrollar y entrenar robots de asistencia de transporte personal (PTAR) para pacientes con trastornos del sistema nervioso central (Kousi, Koukas, Michalos, Makris, & Chryssolouris, 2016).

Estos avances han permitido cubrir brechas presentadas y garantizar una mejor calidad de vida para los humanos en su día a día. La logística, a través de los años se han realizado aplicaciones de robots de servicio para tareas logísticas a través de vehículos AGV (Vehículo de Guiado Automático) los cuales emplean técnicas de navegación básicas como seguimiento de línea, las cuales son muy limitadas y se vuelven invasivas para el área de trabajo, así mismo el limitante computacional compromete en gran medida una navegación suave y segura de los agentes. Se ha venido trabajando en optimizar estos sistemas, en el área automotriz se han desarrollado unidades auxiliares móviles (MAU), encargadas del transporte de los repuestos desde los almacenes hasta las estaciones de ensamble de vehículos, con su software web se proporcionan los servicios para el monitoreo del estado del taller y la programación dinámica del suministro de piezas en función del tiempo y el inventario (Yavasa, Deniz, & Ozenb, 2020). Otro ámbito importante en el que se han sumado esfuerzos es en la automatización de centros logísticos donde se proponen nuevos marcos de trabajo, y que por medio de lógica difusa (*fuzzy*) se optimicen los procesos según los requerimientos que requieran más demanda y enfocar a los robots o agentes a realizar dichas tareas prioritarias con el fin de cumplir con un alta demanda y competitividad en el mercado. Lo anterior supone una reestructuración de las metodologías tradicionales obteniendo resultados muy provechosos e interesantes (Jin, IlKwag, & DaeKo, 2020). Otra perspectiva de la aplicación de la logística robótica es en el área de hotelería en donde se busca automatizar tareas repetitivas por medio de robots móviles, tareas como acompañar a los huéspedes a ciertas áreas, entregar artículos específicos a las habitaciones y transportar algunos artículos que los huéspedes desean que se retiren de las habitaciones. Para lograr todo esto se propone un modelo matemático el cual se encarga de decidir la cantidad de robots a emplear con el objetivo de minimizar el costo total de la inversión y maximizar el total de trabajos cubiertos (Yan, Hua, Wang, Wei, & Imrand, 2017).

A la hora de contar con entornos logísticos óptimos, de precisión y despliegue de gran cantidad de robots, se ve comprometida la capacidad de procesamiento y se va volviendo complejo su respectivo escalamiento, debido a esto se ha introducido un nuevo término conocido como robótica *cloud*, y ha enfocado los esfuerzos en liberar de la carga computacional a los agentes robóticos. Esto supone un nuevo enfoque tecnológico para la ejecución de tareas y el intercambio de recursos en comparación con los robots industriales tradicionales. Se incluyen mecanismos de ajuste autoadaptativos para la calidad del servicio de una red de robot en la nube, mecanismos de asignación de carga informática para robótica en la nube y aprendizaje grupal basado en una plataforma en la nube (Velagic.J & Balta.H, 2010). En esta tesis se desarrolló una aplicación de robótica para logística de interiores por medio de técnicas de inteligencia artificial y haciendo uso de servicios *cloud* para su despliegue, administración y se construyó un servicio web en donde se gestionan las tareas del robot. Lo anterior con el fin de optimizar los procesos logísticos en la industria actual.

1.1 Planteamiento del problema

En años recientes la robótica de servicio ha encontrado muchos ambientes de aplicación buscando incrementar la productividad de las organizaciones e incorporando valor agregado. Una tendencia actual de la robótica móvil es trabajar con flotillas de robots que operan de manera coordinada. Estos equipos de robots deben operar o trabajar en entornos dinámicos y cohabitar con personas, con objetos estáticos y dinámicos por ejemplo otros robots de una manera segura y confiable. De este modo se incorpora valor agregado en los procesos productivos. Lo anterior permite lograr más agilidad y calidad en los mismos (Li & Savkin, 2018).

La robótica móvil ha sido empleada en procesos de patrullaje, acompañamiento a personas y el transporte de materiales industriales haciendo uso de técnicas de navegación básicas. Este método es conocido como “seguimiento de línea”, el cual ha cubierto necesidades y tareas elementales. Sin embargo, las necesidades actuales requieren robots más avanzados, con alta autonomía energética y que se puedan gestionar desde aplicaciones web y plataformas en la nube. Una tendencia que está tomando gran interés es que el

procesamiento de los robots se encuentre en la nube (cloud computing) (Zhu, 2018). Una buena cantidad de investigaciones en universidades y compañías no han dirigido sus esfuerzos en optimizar sistemas logísticos y de servicios para la industria. De esta manera se lograría una masificación de los sistemas de robótica inteligente que permita tomar decisiones de peso en las industrias a nivel general, aumentando la productividad y reduciendo considerablemente los costos (Hehua, Qingsong, Yingying, Wenguo, & Muhammad, 2017). El desarrollo de aplicaciones robóticas enfrenta diversos retos a nivel de recursos computacionales y de software. En cuanto a hardware, las aplicaciones son cada vez más demandantes y se requiere de más recursos en la línea de procesamiento, memoria, GPU's. El asunto se hace más complejo en escenarios de múltiples robots que deben trabajar de manera coordinada. A nivel de software las herramientas de diseño y desarrollo aparecen dispersas, no disponibles en un único ambiente. Se presentan muchas herramientas heterogéneas que no se integran y requieren una instalación de cada recurso por separado. En este entorno no se encuentra una cohesión, las herramientas son muy variadas y disímiles lo que implica gran esfuerzo y recursos de instalación, configuración y dominio de las herramientas y no en la aplicación, que constituye el foco. De igual manera, no se cuenta de primera mano con herramientas de simulación que sean ágiles y flexibles para colocar el código en ejecución. Esto con el fin de realizar pruebas continuas que proporcionen la seguridad a la hora de poner a prueba el robot real y saber si este es lo suficientemente capaz de cubrir la necesidad de la aplicación. Para que un proyecto de robótica a nivel industrial pueda tener un impacto significativo es indispensable contar con herramientas que permitan gestionar las tareas de los robots. Con esto se tiene la capacidad de monitorear el estado de los mismos y asignarles tareas de manera automática, en función de la demanda que se tenga en la producción.

En este trabajo se abordó el problema de la brecha existente entre el software y el hardware disponible para el desarrollo de aplicaciones robóticas y la aplicación misma. Se hizo énfasis en la interacción del usuario con la aplicación al momento de su implementación en entornos productivos, haciendo uso de los servicios propuestos por AWS Robo Maker y ROS (Robot Operating System) (Deuseab, 2014). Un reto al que se enfrentan los desarrolladores de proyectos de robótica de servicios, es que un robot móvil es considerado como un sistema complejo de ingeniería que involucra hardware, control, actuación y percepción, adicional a una arquitectura de control donde se incorporan

atributos de inteligencia artificial (Giusti Bravo, 2016). Debido a lo anterior es que se vuelve complejo el desarrollo de aplicaciones robóticas. Con este trabajo se buscó crear un método que facilita el desarrollo e implementación de estas aplicaciones, incorporando herramientas de computación en la nube.

1.2 Objetivos

1.2.1 Objetivo General

Diseñar un método para el desarrollo de aplicaciones en robótica de servicios, basado en industria 4.0 y computación en la nube.

1.2.2 Objetivos Específicos

- Caracterizar los elementos de industria 4.0 aplicables a la robótica de servicios para interiores.
- Identificar modelos de computación en la nube aplicables a la robótica de servicios.
- Incorporar técnicas de inteligencia artificial para el control de la navegación de robots móviles en entornos interiores.
- Integrar elementos de la industria 4.0 y computación en la nube para la navegación autónoma de un sistema robótico.
- Validar el método propuesto a través de la realización de un conjunto de pruebas en un escenario real.

1.3 Marco Teórico

La industria 4.0 es la tendencia actual en automatización, la cual comprende entendimiento entre máquinas, personas con máquinas, inteligencia artificial, mejoras tecnológicas continuas y comunicación continua con internet, cubriendo brechas presentadas en la industria en el pasado. En el modelo de Industria 4,0 se integran los siguientes elementos:

- Big Data and Analytics.
- Robots autónomos.
- Simulación.
- Sistemas para la integración vertical y horizontal.

- IoT (Internet de las cosas).
- Ciberseguridad.
- *Cloud computing*.
- Fabricación aditiva.
- Realidad aumentada.

1.3.1 Robots autónomos

Este concepto no es para nada nuevo en la industria, pero para esta nueva generación de robots, los robots autónomos incorporan nuevas capacidades a nivel de inteligencia artificial y coordinación para trabajar entre ellos, sin intervención humana y así lograr automatizar un buen número de tareas de logística y producción, y sin limitantes computacionales ya en la actualidad se implementan técnicas y *frameworks* de robótica *cloud*. Abordando una mirada más puntual en los robots autónomos de servicio, éstos han sido cada vez más requeridos y han cubierto necesidades como lo son mapeo de entornos, robots para el área de logística y transporte de materiales, robots para desenvolverse y apoyar entornos hospitalarios, pero siempre se ha tenido brechas con los humanos en la utilización de estos robots. En la actualidad se cuentan con herramientas capaces de reducir estas brechas con el fin de lograr una buena interacción de robots y humanos. Lo anterior se permite una importante aceleración de las compañías con industria 4.0, ya que es posible contar cada vez con más robots que se encarguen de realizar gran cantidad de tareas con un rendimiento y productividad realmente importante. Diferentes compañías de robótica móvil han desarrollado aplicaciones lo suficientemente robustas en el ámbito de la robótica de servicios para transportar importantes cantidades de materiales controlados desde aplicativos a la medida para gestión de tropas de robots y algoritmos de inteligencia artificial para un buen desempeño de los robots. Tomando como punto de partida la robótica de servicios y enfocándonos en la robótica logística para interiores, encontramos robots que pueden atender gran cantidad de pedidos y coordinarse con otros robots para interactuar en un mismo entorno, ya sean móviles o brazos manipuladores que les provean materiales o insumos a transportar, esto ha llegado a potenciar el rendimiento de grandes compañías logísticas como Amazon. Esto se logra gracias a que se ha incorporado a los robots móviles en la cadena de suministro en la industria logística, esto supone un gran

avance en este ámbito, una buena proyección en años futuros y un retorno de inversión muy interesante (Rivera, Simone, & Guida, 2019).

1.3.2 Simulaciones

Los simuladores han cobrado gran valor a través de los años ya que permiten acercarse a la realidad los proyectos en etapas tempranas, pero siempre han carecido de factor de realismo y rigor mecánico. En la actualidad contamos con nuevos simuladores que incorporan la parte mecánica excepcionalmente y se acercan a los resultados esperados al momento de implementar soluciones en la vida real y para tomar decisiones acertadas en los proyectos. Los simuladores de robótica móvil en la actualidad cuentan con capacidades interesantes que hacen muy provechoso su buen uso, como el “V-REP” el cual proporciona un entorno de simulación 3D e integra un entorno de desarrollo para modelado y programación de robots. Otro simulador muy empleado es “*The Stage Robot Simulator*” en donde su fuerte no es su parte gráfica ya que es 2D pero sí la interacción con el entorno y otros robots, su flexibilidad a la hora de colocar robots móviles a prueba en entornos realistas y validar el funcionamiento de controladores previamente desarrollados, lecturas de sensores entre otros. Por último, es importante mencionar un simulador de gran calado como lo es el Gazebo, un simulador el cual ha sido abstraído por Amazon en sus servicios de AWS Robo Maker. Gazebo ofrece la capacidad de simular con precisión y eficiencia poblaciones de robots en entornos complejos interiores y exteriores. Integra un motor de física robusto, gráficos de alta calidad e interfaces gráficas programáticas convenientes, permite cargar modelos mecánicos diseñados en software CAD. Adicionalmente permite realizar pruebas de regresión, entrenar sistemas de inteligencia artificial utilizando escenarios realistas. Gazebo cuenta con herramientas de integración disponibles con Solidworks y Matlab-Simulink, conocidas plataformas comerciales de modelado y control de robótica. Los requerimientos de la industria actual implican el desarrollo de modelos multicuerpo más detallados, los cuales deben ser puestos a prueba antes de su construcción de la mano de técnicas de autoaprendizaje, gracias a Gazebo se permite el desarrollo de una nueva clase de máquinas capaces de revolucionar la industria a como la conocemos (Dourado, y otros, 2019).

1.3.3 IoT (Internet de las cosas)

En el pasado los dispositivos como robots y sensores ricos en datos tenían la imposibilidad de conectarse a grandes distancias a una red para enviar datos en tiempo real. En la actualidad contamos con dispositivos que poseen la capacidad de conectarse a internet fácilmente y enviar gran cantidad de datos para su análisis y toma de decisiones de una manera rápida y eficiente dejando de lado altos costos y datos desviados. En la robótica móvil, se presentaban problemas al momento de enviar robots a grandes distancias debido a los temas de conectividad, dónde se desconocía la posición y el estado del robot, con la industria 4.0 los robots emplean activamente la capacidad de conectarse a internet en el caso que requieran, a nivel satelital, lo que permite en diferentes casos el robot alimentarse de un mapa preestablecido en un servidor para ubicarse correctamente, tomar tareas a realizar en tiempo real, enviar datos de sensores activamente o de tareas completadas. Se han alcanzado capacidades que gracias a los sistemas IoT se confiere la centralización del procesamiento, lo que reduce los costos y permite la reutilización de la potencia informática inactiva del robot, garantizando una mayor precisión y efectividad (Xiaa, Zhanga, Wang, Coleman, & Liua, 2018).

1.3.4 Computación en la nube

Las plataformas en la nube han permitido revolucionar y acelerar el mercado gracias a sus versátiles servidores y a las soluciones que ofrecen al momento de migrar arquitecturas locales con servidores propios, introduciendo términos como balanceador de cargas que permite mantener estable los servicios ante una gran demanda por parte de los clientes y evitar las pérdidas y demoras a la hora de intervenir servidores locales garantizando la disponibilidad. En el ámbito de la robótica móvil se ha empezado a hablar acerca de *cloud robotic*, el cual se ha vuelto un campo en desarrollo de robótica. Éste cercamiento resuelve varios paradigmas en el área de la robótica como lo son la limitación en términos de capacidad informática, memoria y almacenamiento de datos. La robótica *cloud* ofrece capacidad informática ilimitada, memoria, almacenamiento de gran cantidad de datos, esta tecnología se centra en los beneficios de integrar infraestructura y servicios compartidos a través de un entorno de computación en la nube. Actualmente se presentan arquitecturas monoméricas distribuidas tradicionales seguidos de un diseño de arriba abajo lo cual con el tiempo puede ocasionar diferentes problemas a la hora de escalarlo, en vista a esto

varios investigadores orientan la arquitectura al uso de microservicios con el fin de aumentar la eficiencia y la flexibilidad a nivel comercial-industrial se conocen los servicios y arquitectura por Amazon AWS robo-maker (Dröder, IBobka, Germann, Gabriel, & Dietrich, 2018).

Las técnicas computacionales en la robótica móvil que van de la mano con la industria 4.0 por su capacidad y alcance son las basadas en *machine learning*. Estas técnicas han cobrado gran importancia y relevancia en la robótica móvil y son empleadas con el fin de optimizar el rendimiento y el comportamiento de los robots en campo. Esto debido a que un robot que se desenvuelve en un entorno industrial requiere interactuar con personas que constantemente están en movimiento, entornos muy dinámicos con obstáculos fijos y móviles lo que implica que el robot móvil deba tener movimientos seguros. Esto radica principalmente en la variedad de posibles eventos diferentes que pueden ocurrir en un entorno no estructurado. Las técnicas de *machine learning* proporcionan las herramientas para cubrir estos paradigmas o problemas por medio de estrategias de control que combinan la planificación de las rutas más óptimas, análisis de agrupamiento y redes neuronales artificiales para la detección de obstáculos (Baalbaki & Xie, 2009). Todo lo anterior con el fin de garantizar una planificación adaptativa del camino para la protección del ser humano, la integridad del agente robótico y garantizar la seguridad del entorno en donde este se desenvuelve.

1.4 Metodología

La metodología para este trabajo inició con la identificación de los elementos de la industria 4.0 aplicables a la robótica de servicios y especialmente en la robótica logística. Luego se realizó un estudio sobre los modelos, *frameworks*, arquitecturas aplicadas en la robótica *cloud* y empleados en la robótica de servicios, y se adoptó uno de ellos, para que posteriormente fuera aplicado en el desarrollo de la aplicación.

Una vez definida la arquitectura, y los elementos de la industria 4.0, se dio paso a seleccionar el robot móvil que se empleó para este proyecto, caracterizando su sensoria interoceptiva y exteroceptiva. Luego se pasó a diseñar y desarrollar el algoritmo de inteligencia artificial para la navegación autónoma del robot en un ambiente logístico el cual debe estar en la capacidad de interactuar con personas en constante movimiento, contar con movimientos seguros y planificar las trayectorias óptimas para su

desplazamiento. Se desarrolló además una interfaz web con capacidad de asignarle tareas al robot bajo una metodología de colas.

Esta interfaz cuenta con la capacidad de brindar información del estado actual del pedido durante su ejecución. Se realizaron pruebas en el simulador Gazebo con el fin de garantizar un buen funcionamiento del robot en un escenario simulado. Se puso a prueba el funcionamiento del aplicativo web a la hora de gestionar los pedidos con el robot móvil. Se empleó herramientas como Git para el control de versiones, servicios para el manejo *cloud* y la plataforma ROS (Robot Operating System) para el desarrollo del algoritmo de inteligencia artificial y el simulador Gazebo para las pruebas.

1.5 Alcances y limitantes

Se contó con una aplicación de robótica móvil empleando técnicas de desarrollo de inteligencia artificial y computación en la nube en donde los agentes robóticos se desenvuelven en un escenario real a los cuales se le puede encomendar tareas de servicios como lo es el movimiento seguro de mercancías dentro de un entorno logístico industrial a diferentes lugares de un área de trabajo en específico con obstáculos y elementos móviles como las personas.

Todo lo anterior, a través de un aplicativo el cual tiene la capacidad de gestionar pedidos y administrar el comportamiento del robot en función de su asignación de pedidos. El robot tiene la capacidad de interactuar con otros agentes robóticos industriales manipuladores encargados de colocarles o retirarles la carga para su posterior transporte.

Con base en lo anterior se propuso un método para el desarrollo de aplicaciones en robótica de servicios basado en robótica *cloud* para emplearse en aplicaciones robóticas industriales y educativas.

Las limitantes para este trabajo se fundamentaron en base a las siguientes particularidades:

En el trabajo no tuvo en cuenta la comparación con otras metodologías de desarrollo de robótica de servicio en la nube.

El entorno de funcionamiento para el cual fue creada la plataforma robótica la limita a espacios internos o domésticos, ya que este tipo de plataforma se diseñó con fines de servicio en lugares con iluminación controlada, terrenos planos estructurados.

Dada la emergencia sanitaria generada por el COVID-19 las pruebas de validación experimental, se realizaron únicamente en escenarios simulados.

2. Elementos de la industria 4.0 aplicables a la robótica de servicios

La llegada de la industria 4.0 supone un gran reto en la actualidad en donde los procesos deben cerrar diferentes brechas que se han presentado en el pasado. Cómo contar con una capacidad de adaptación constante a la demanda, Diseñar, producir y vender productos en menos tiempo y añadir servicios a los productos físicos. La robótica móvil enfrenta varios retos adicionales en la industria 4.0 como lo es la implementación de algoritmos de control más avanzados y seguros, con mayor flexibilidad para realizar diferentes tipos de tareas y con una mayor facilidad en el proceso de programación y que pueda coexistir con las personas en un ambiente cerrado, qué ha sido uno de los grandes requisitos de las industrias actuales. En la Figura 2-1 se plantea una mirada general de los elementos más relevantes en la industria 4.0.

Figura 2-1: Elementos principales de la industria 4.0



Fuente: (Bhavsar, Patel, & Sobh, 2019)

A continuación, se mencionarán diferentes elementos propios de la robótica que están inmersos en la industria 4.0.

2.1 Robótica avanzada, flexible y colaborativa

Corresponde a la generación de robots que posee la facilidad de trabajar en el mismo entorno que el de los humanos, sin ser necesaria la instalación de elementos de seguridad realizando en todo momento trayectorias seguras habitando con otros agentes robóticos de manera inteligente y evitando obstáculos fijos y móviles. Estos robots deben ser fáciles de programar, sin la necesidad de contar con conocimientos en programación para su operación y parametrización. Todos los elementos unidos a la flexibilidad permiten realizar tareas muy diferentes como (*pick & place*, montaje, empaquetado, atornillado) por medio de plataformas robóticas de fácil acople mecánico, acercando la automatización a las pequeñas empresas y medianas empresas de manera impensable anteriormente. Estos robots son considerados como un compañero de trabajo más en la actualidad.

Para que esto sea una realidad se presentan diferentes retos en la industria 4.0 como lo son.

- Adquirir, procesar y fusionar datos diversificados para la clasificación de riesgos.
- Actualizar efectivamente el control para evitar cualquier interferencia en un modo de tiempo real.
- Desarrollar nuevas tecnologías para mejorar el rendimiento de la HMI, especialmente, cargas de trabajo y velocidades.
- Reducir el costo general de las funciones que garantizan la seguridad.
- Programación basada en tareas y en habilidades que incorporan la gestión de riesgos y las evaluaciones de la carga biomecánica y la distancia de frenado.
- Sensorica y algoritmos avanzados para la detección, procesamiento y fusión efectivos de datos diversificados, y aprendizaje automático para la complejidad y la incertidumbre de alto nivel.

Es ventajoso enfatizar las ventajas de un robot y un humano desarrollando un nuevo tipo de robot llamado robot colaborativo (también conocido como Cobot). En general un Cobot aprovecha la flexibilidad de los apoyos humanos para la toma de decisiones. El concepto de Cobot ha sido ampliamente aceptado para tratar interacciones directas humano-robot (HRC) en entornos abiertos y no estructurados con miras a aumentar la competitividad empresarial. La adopción de HRC aumentará el nivel de la automatización del sistema (D.

Bogataj, 2019). Describe la importancia del HRC para el envejecimiento de la sociedad, dónde los trabajadores de edad avanzada utilizaran los Cobots para compensar la pérdida de sus capacidades laborales.

Es importante mencionar los diferentes tipos de colaboración humanos-robot. Autores como (J. Guiochet, 2017) realizan una diferenciación entre los robots convencionales y los Cobots en los ámbitos de estructuras, condiciones de operación, movimientos, controles, comunicaciones, espacios de trabajo y nivel de complejidad.

2.1.1 Controles

Un programa de Cobot tiene la autonomía de decisión para adaptarse a los aportes humanos en tiempo real.

2.1.2 Condiciones de funcionamiento

Se eliminan las barreras para que el operador humano acceda al espacio de trabajo compartido.

2.1.3 Comunicación

Debe tener la interfaz para que el operador proporcione entradas de manera interactiva.

2.1.4 Características Estructurales

Un robot no colaborativo puede ser rígido, pesado y con una gran carga de trabajo, mientras que por el contrario un Cobot suele ser de servicio ligero y la estructura es compatible para diferentes procesos de la compañía.

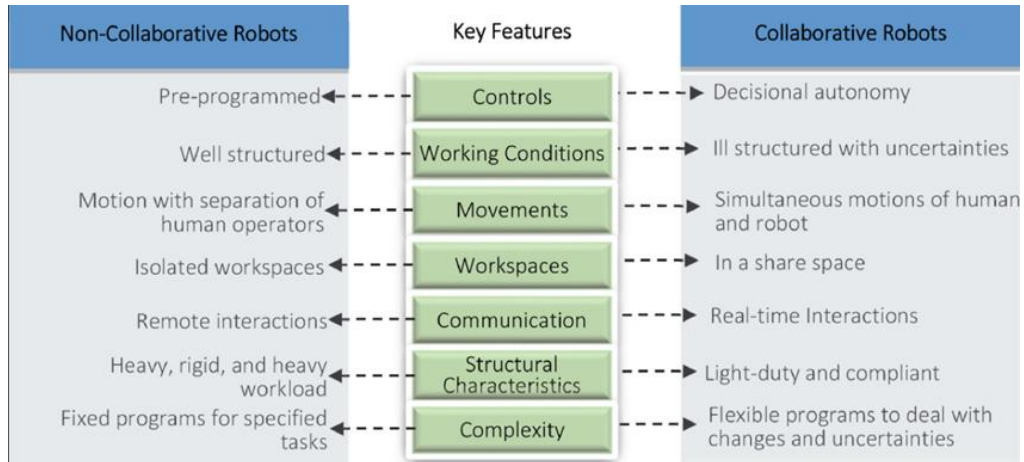
2.1.5 Complejidad

Un Cobot tiene más funcionalidades que un robot no colaborativo, ya que está diseñado para adaptarse a cambios e incertidumbres detectables cuando colabora con humanos. Tenga en cuenta que la complejidad podría estar relacionada con incertidumbres.

En la figura 2-2 se realiza una comparación detallada de los robots colaborativos y los no colaborativos en donde se parte de escenarios clave a los que se enfrentan los robots en

el ámbito industrial y se logra identificar las falencias de los robots no colaborativos frente a los colaborativos.

Figura 2-2: Comparación de robots no colaborativos y robots colaborativos



Fuente: (J. Guiochet, 2017).

El desarrollo de estos nuevos robots colaborativos es un campo relativamente nuevo en la robótica, sin embargo, las aplicaciones con robots colaborativos se expanden rápidamente. Empresas como Omron ha implementado sistemas de robots logísticos combinados con robots colaborativos como los que se ilustran en la figura 2-3. Donde es posible la gestión de flotas con todos los requerimientos de seguridad y facilidades propias de un Cobot, integrando la familia de robots móviles LD-250 y robots colaborativos TM abriendo un nuevo mercado de robots manipuladores móviles destinados a labores pesadas que puedan gestionar tareas de manipulación además del transporte de materiales de manera segura.

Figura 2-3: Gamas de robots colaborativos y móviles industriales Omron



Fuente: <https://industrial.omron.es/es/products/collaborative-robots.jpg>

2.2 Sensorica avanzada a nivel de fábrica

Esta definición corresponde a todas aquellas tecnologías que por medio de la utilización de sensores que por su naturaleza es posible analizar y obtener datos de manera no invasiva de diferentes objetos. Estos sensores corresponden a los campos de visión, ultrasonido, acústica y fotónica. Estos tipos de sensores cubren temáticas como lo son la visión artificial, escaneado 3D, espectrometría y termografía. La sensorica avanzada abre un gran panorama sobre las diversas áreas de aplicación sobre todo en la robótica móvil en donde la visión artificial ha tomado el protagonismo para garantizar trayectorias seguras y precisas, de igual manera permite identificar objetos complejos más fácilmente.

2.2.1 Robótica guiada por visión

Los sistemas VGR son básicamente cualquier sistema robótico equipado con una o más cámaras de visión donde es posible adicionar diferentes sensores como realimentación al sistema. Los sistemas VGR permiten que los robots cuenten con una gran adaptación a los cambios en los productos para lograr con esto una fabricación flexible.

La visión artificial por medio de sensores 2D ha tenido gran protagonismo durante décadas pasadas y se han empleado en sinnúmero de aplicaciones. Sin embargo, los requerimientos de la industria 4.0 son mucho más amplios y complejos para un sensor 2D y por tanto se requiere emplear sistemas de visión artificial basados en 3D que permitan desde la navegación autónoma hasta la lectura de un código de barras.

En el caso de los sistemas visión artificial 2D la imagen del cuerpo objetivo en cuestión es obtenida proporcionando datos únicamente en los ejes X y Y, pero se cuenta con la carencia de los datos de profundidad en el campo Z. Para estos casos la imagen objetivo se forma a partir de la luz reflejada. Estas variaciones presentadas en la iluminación ambiental o artificial pueden afectar notablemente la definición de las figuras convirtiendo esto en un problema importante por tanto para un óptimo rendimiento los sensores deben mantenerse en espacios con iluminaciones definidas.

Por otro lado, al profundizar los sistemas de visión artificial 3D se encuentra que cuentan con una nube de puntos tridimensional de coordenadas precisas donde se conoce la precisión de cada pixel en el espacio, datos de los planos X, Y y Z. Junto con la información de rotación respectiva. Es importante destacar que existen cuatro técnicas principales en los sistemas de visión artificial 3D (Schumann, 2018).

- Triangulación laser.
- Estéreo visión.
- Tiempo de vuelo
- Luz estructurada

En aplicaciones de robótica móvil encontramos con diferentes sensores para la percepción de 3D, estos se encuentran clasificados en sensores activos y pasivos. Los sensores activos hacen uso de una fuente de energía para sondear el medio ambiente y lo hacen emitiendo señales de sonido o luminosas para con esto medir el reflejo de la energía emitida. Por el contrario, los sensores pasivos hacen uso de la energía que se encuentra en el ambiente como en el caso de las cámaras o de otra fuente que es reflejada en los objetos destino y luego en la cámara. En la Tabla 2-1 se realiza una comparación entre los diferentes tipos de sensores.

Finalmente, en este apartado se describen las cámaras RGB-D (Figura 2-4) las cuales combinan las cualidades de los sensores activos y pasivos. Éste se compone de una cámara RGB (sensor pasivo) y un sensor de profundidad (sensor activo), de esta manera se logra obtener información de profundidad por cada pixel. Para este tipo de cámaras es tradicional encontrar como el sensor de profundidad un proyector infrarrojo (IR) y un receptor infrarrojo. De igual manera como el sensor de tiempo de vuelo, una cámara RGB-D calcula la profundidad analizando la luz reflejada, pero la modulación de onda incidente se realiza de manera espacial en vez de temporal ya que la luz infrarroja se emite en un patrón definido y la profundidad se calcula interpretando la deformación del patrón emitido causado por la superficie de los objetivos.

Tabla 2-1: Comparación entre sensores pasivos y activos

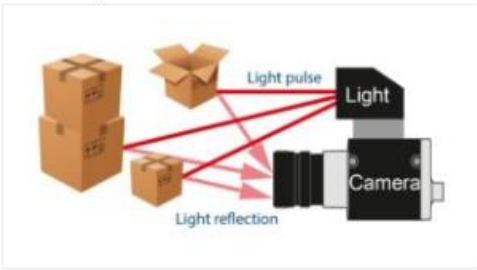
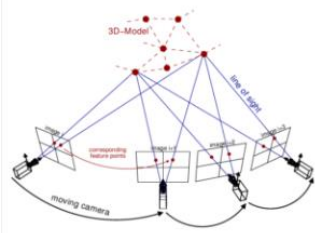
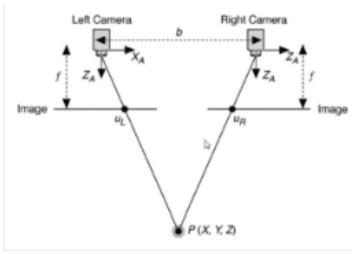
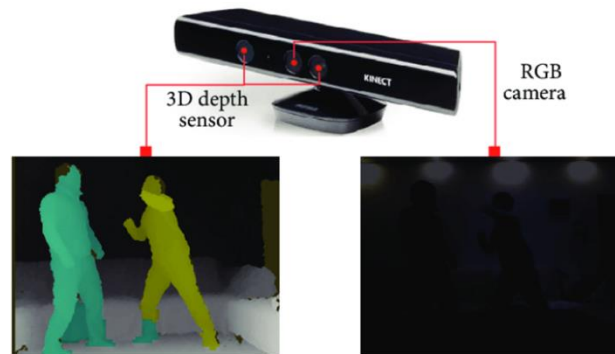
Sensores Activos	Sensores Pasivos
<p>LIDAR (Light Detection and Ranging o Laser Imaging Detection and Ranging)</p> <p>Iluminan el objetivo con un láser de pulsos y miden los pulsos reflejados, son bastante comunes en los robots más avanzados debido a su gran precisión, gran alcance, alta resolución en el plano horizontal</p>  <p>Cámara Tiempo de Vuelo</p> <p>Este tipo de sensor realiza medición de profundidad al iluminar un área con una fuerte luz infrarroja y determinar el tiempo que se toma viajar al objeto y volver de este encontramos dos tipos:</p> <ul style="list-style-type: none"> • sensor de tiempo de pulso. • sensor de onda continúa con desplazamiento de fase. 	<p>Cámaras monoculares</p> <p>Existe la posibilidad de medir la profundidad a través de una sola cámara usando técnicas de software para con esto obtener profundidad desde múltiples imágenes 2D.</p> <p>Técnicas como <i>Structure Motion</i> permiten reconstruir un modelo 3D a partir de la rotación de una cámara sobre un objeto fijo. Esto implica una gran precisión a lo largo de su movimiento para lograr realizar un cálculo correcto de la triangulación y así obtener la profundidad del objeto.</p>  <p>Cámaras estereoscópicas</p> <p>Este consiste en disponer dos cámaras monoculares separadas por una distancia precisa y determinada, bajo este escenario la profundidad es obtenida a través de la comparación obtenida de las dos cámaras viendo el mismo objeto o escenario.</p> <p>La diferencia de posición del objeto percibido se llama disparidad y de esta manera se tiene la profundidad del objeto como ocurre con el ojo humano.</p> 

Figura 2-4: Cámara RGB-D Kinect

Fuente:https://www.researchgate.net/figure/RGB-D-camera-a-Kinect-sensor-b-Depth-map_fig8_316815628

2.3 Sistemas ciberfísicos (CPS)

El empleo de sistemas ciberfísicos es uno de los elementos que posibilitan un cambio de paradigma en la industria 4.0. Se define como todo aquel equipo que integra capacidad de computación, almacenamiento y comunicación para controlar e interactuar con otros equipos o procesos en el mundo físico. Estos sistemas están normalmente conectados entre sí.

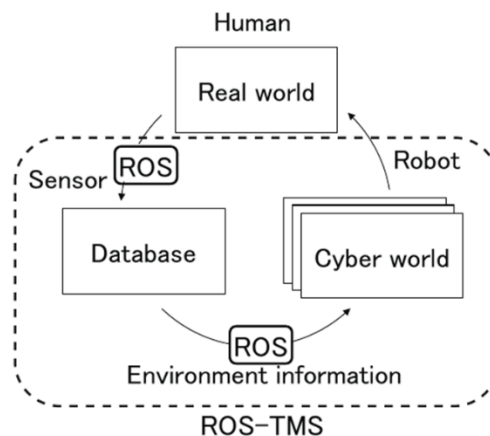
Las principales características de los sistemas CPS son por un lado que pueden utilizar la información proveniente del mundo virtual, de forma que puede tener la capacidad de aprender y evolucionar mediante técnicas de inteligencia artificial, y por otro lado que tiene la capacidad de interactuar con objetos físicos. El diseño y construcción de sistemas ciberfísicos requiere la aplicación transversal de conocimientos avanzados en diversas disciplinas los cuales agrupan áreas de conocimiento como son las siguientes. (Gonzales, Calvo, Etxberria, & Zulueta, 2015)

- Redes de comunicación.
- Computación embebida.
- Sistemas de tiempo real.
- Sistemas de control.

Para la implementación de sistemas ciberfísicos en robótica móvil es muy ventajoso disponer de arquitecturas de bajos costes estándar y libres. El software ROS (Sistema Operativo Robótico) que es en realidad un middleware creado para simplificar las comunicaciones de bajo nivel y facilitar la escritura de módulos en el control de robots.

Su estructura es modular, permite añadir más sensores y diferentes modos de locomoción de forma más flexible, todo esto permite para sistemas ciberfísicos, adquirir gran cantidad de datos, la ejecución de algoritmos de procesamiento, el envío de instrucciones a diferentes maquinarias, robots y cualquier tipo de actuadores. En vista al potencial que posee ROS para el desarrollo de aplicaciones de sistemas CPS, diferentes colaboradores de la comunidad han desarrollado un entorno estructurado informativamente (ISE) denominado ROS-TMS el cual es considerado como una implementación robótica de los sistemas ciberfísicos (CPS). La arquitectura propuesta para el ROS-TMS está compuesta de 3 componentes principales el del mundo real, de la base de datos y del mundo cibernético. Los cuales se detallan en la figura 2-5.

Figura 2-5: Elementos principales del ROS-TMS



Fuente: (Pyo, Nakashima, & Kuwahata, 2015)

Un sistema de detección distribuido detecta eventos que ocurren en el mundo real, como el comportamiento del usuario o las solicitudes del mismo y la situación actual del mundo real, como la posición de los objetos, humanos y robots.

La información recopilada se almacena en la base de datos. Los comandos de servicio apropiados se planifican utilizando la información ambiental en la base de datos y se simulan cuidadosamente en el mundo cibernético utilizando simuladores, como Choreonoid y Gazebo. Finalmente, las tareas de servicio se asignan a robots de servicio en el mundo real. (Sakamoto, Kiyoyama, Matsumoto, & Pyo, 2018), (Pyo, Nakashima, & Kuwahata, 2015).

La relevancia de los sistemas ciberfísicos en la industria 4.0 radica en la capacidad de generar análisis y auto-evaluación en las máquinas, y en los procesos de fabricación los sistemas ciberfísicos mejoran la productividad y la calidad gracias a los pronósticos y diagnósticos los cuales se pueden realizar por medio de big data en diferentes sensores, máquinas y sistemas de red.

Este tipo de pronósticos y diagnósticos permiten la auto-comprobación y auto-configuración para optimizar en todo momento el rendimiento de los agentes robóticos en campo impulsando de manera directa la rentabilidad de la empresa y mejorando los procesos en la industria.

2.4 Robótica en la nube

Uno de los elementos que ha tomado relevancia con la llegada de la industria 4.0 es la importancia de emplear servicios multi-roboticos para cubrir las grandes demandas de las compañías. Por tanto, es necesario que los agentes robóticos empleen tareas de control más complejas lo que requería emplear en cada robot un hardware más poderoso para satisfacer estas tareas de control. En vista a lo anterior es donde resulta de gran ayuda emplear las capacidades y recursos disponibles en la nube. Esta solución que se presenta de manera integral a nivel de recursos de hardware, software, redes y almacenamiento proporcionan de manera una alta disponibilidad y cantidad de recursos según se requiera. El concepto de desarrollo de servicios en la nube abarca tres tipos diferentes de desarrollo: Software como servicio (SaaS), Plataforma como servicio (PaaS), infraestructura como Servicio (IaaS), e incluso X como servicio (XaaS), donde X puede ser cualquier elemento como: Robot como Servicio (RaaS).

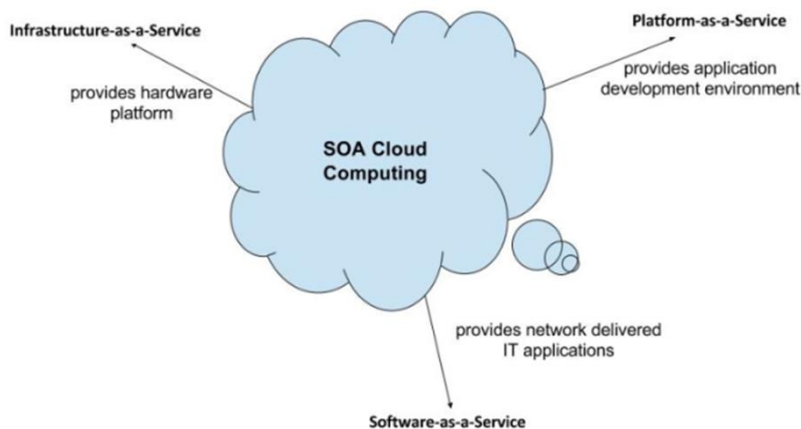
El concepto de Robot como servicio (RaaS) se basa en las funciones de una arquitectura orientada a servicios y esto implicar el uso de funciones de robots en la nube como servicio a demanda. Para entender a detalle el uso de la robótica como servicio a continuación se detallarán diferentes conceptos.

2.4.1 Arquitectura orientada a Servicios (SOA)

Esta arquitectura es una forma de diseñar, implementar y ensamblar servicios para la automatización de diferentes funcionalidades a nivel comercial e industrial. La construcción de esta arquitectura se basa en la comunicación de un grupo de servicios entre sí. Es importante mencionar que el proceso de comunicación compromete desde el paso de datos de manera simple hasta la coordinación de dos o más servicios para dar cumplimiento a alguna actividad en específico. La computación en la nube permite incrementar el alcance de SOA como se detalla en la figura 2-6 al incluir elementos como:

- **Infrastructure-as-a-Service (IaaS):** provee una plataforma de hardware como servicio.
- **Plataform-as-a-Service (PaaS):** Provee a los usuarios un entorno de desarrollo a través de internet.
- **Software-as-a-Service (SaaS):** Proporciona a los usuarios finales aplicaciones para su uso diario en la red.

Figura 2-6: Componentes de SOA a nivel de computación en la nube



Fuente: https://www.tutorialspoint.com/soa/images/soa_cloud_computing.jpg

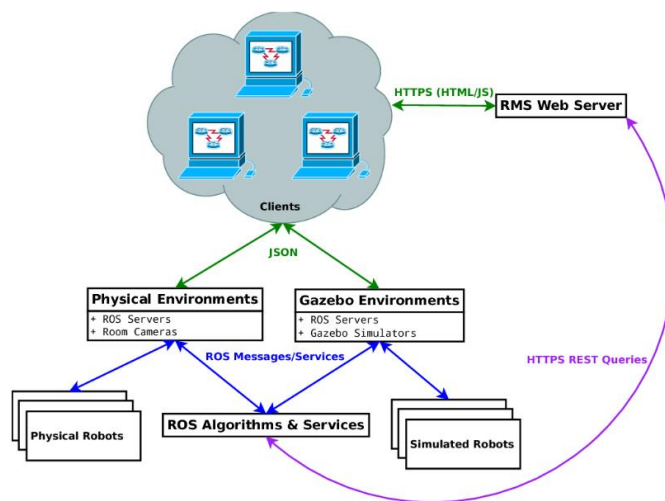
2.4.2 Robótica en la nube a nivel de sistema (RaaS)

Esta tecnología la cual se extiende rápidamente por el mercado industrial, esto gracias a la arquitectura orientada a servicios (SOA) implementada en la nube y añadiendo los robots en esta arquitectura, permitiendo con esto administrar y desarrollar robots de manera más eficiente a nivel industrial. Los componentes principales que intervienen en el (RaaS) son el Robot Management System (RMS), Cloud Robotics Administrator y el Virtual Robot System (VRS). A continuación, se detallarán cada uno de estos elementos.

2.4.3 Robot Management System (RMS):

Es un sistema de gestión remota, diseñada para controlar robots habilitados para middleware como ROS desde un aplicativo web. Éste sistema se encuentra desarrollado de manera independiente de la plataforma del robot lo que permite el control de una flota de robots si se requiere (Ver Figura 2-7). Los sistemas RMS son considerados el corazón de una arquitectura (RaaS) ya que permite gestionar toda la plataforma robótica en la nube, contiene módulos como lo son creación, comunicación, control y seguimiento y utiliza muchas tecnologías y protocolos (Php, Javascript, http, Jpegserver, WebSocket, Roslib).

Figura 2-7: Descripción general de alto nivel de la comunicación entre RMS y ROS



Fuente: (Toris, Kent, & Chernova, 2014)

2.4.4 Cloud Robotics Administrator

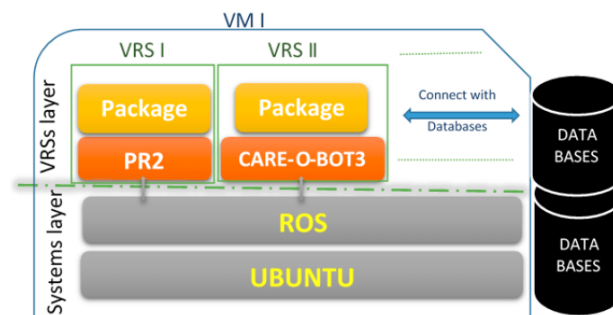
Este es el administrador involucrado como proveedor de los servicios de nube y es el encargado de realizar las siguientes tareas.

- **Gestión de Clientes.** Incorporación de clientes y asignación de recursos necesarios, así como eliminación de clientes y actualización de permisos-recursos si es necesarios.
- **Administración de VRS (*Virtual Robot System*).** Detener/iniciar VRS, aprovisionar recursos a un VRS y finalmente la monitorización de la plataforma *cloud* garantizando el monitoreo ayuda al administrador a prevenir fallas o necesidades de aprovisionamiento al analizar los *logs* generados durante la ejecución de VRS.

2.4.5 Virtual Robot System (VRS)

Es el sistema operativo del robot que es virtualmente ejecutado en el entorno en la nube. Los VRS son similares a las máquinas virtuales con las herramientas proporcionadas en la nube. Se puede considerar como plataformas robóticas donde cada uno corresponde a un robot en específico como se especifica en la figura 2-8. Estos sistemas resultan provechosos a la hora de tener rápidamente un ambiente que permita diseñar y planificar la trayectoria de un robot en un entorno industrial.

Figura 2-8: Arquitectura VRS



Fuente: (Sadek & Ayad, 2015)

Con lo anterior se evidencia que a medida que aumentan más y más robots en la industria de servicios, la comunicación entre robots es muy crítica. Esta comunicación se puede lograr mediante la administración en la nube y los robots se pueden monitorear de forma remota o local a través de RMS. Finalmente se encuentra la plataforma RaaS como una herramienta que cumple con todos los estándares de SOA (Arquitectura Orientada a Servicios) como la plataforma de desarrollo y la unidad de ejecución, creando así un proceso flexible y más amigable con el desarrollo (Bhavsar, Patel, & Sobh, 2019).

En la tabla 2-2 se detallan cada uno de los autores que han enfocado sus esfuerzos en cada uno de elementos que componen la industria 4.0 a en la robótica móvil.

Tabla 2-2: Acrónimo de investigaciones enfocadas en diferentes elementos de la industria 4.0

Autores	Robótica Flexible	Sensorica Avanzada	Sistemas Ciberfísicos (CPS)	Robótica Cloud	Visión Artificial	IOT	Simuladores
(Ullah, 2019)	N	Y	N	N	N	Y	N
(Bhavsar, Patel, & Sobh, 2019)	N	N	N	Y	N	N	Y
(Gonzales, Calvo, Etxberria, & Zulueta, 2015)	Y	N	Y	N	N	Y	N
(D. Bogataj, 2019)	Y	N	Y	N	N	Y	N
(J. Guiochet, 2017)	Y	N	Y	N	N	Y	N
(Schumann, 2018)	N	Y	N	N	Y	Y	N
(Sakamoto, Kiyoyama, Matsumoto, & Pyo, 2018)	Y	N	Y	N	Y	Y	Y
(Toris, Kent, & Chernova, 2014)	Y	N	Y	N	Y	Y	N
(Pyo, Nakashima, & Kuwahata, 2015)	Y	N	Y	N	N	Y	Y

3. Modelos de computación en la nube aplicables a la robótica de servicios

Como se ha mencionado, las aplicaciones robóticas presentan dificultades a la hora de expandirse y esto va directamente relacionado con la interacción entre los diferentes agentes robóticos en un área específica, lo anterior es por sus limitantes a nivel computacional, y por tanto la adquisición de su información en tiempo real. Lo anterior obliga a mantener las aplicaciones robóticas muy limitadas y con muy poca capacidad de expansión dejando de lado la capacidad de proporcionar herramientas web que permitan interactuar con los usuarios.

En las décadas pasadas se ha integrado algunas técnicas como el *Networkig*, en donde diferentes robots están conectados a una red inalámbrica interna y por medio de esta es posible obtener la información suministrada por los sensores y así permitirles a otros robots, conocer su ubicación entre ellos. Sin embargo, al analizar en detalle esta estrategia encontramos que la comunicación debe viajar entre cada robot constantemente lo que genera retrasos considerables debido al gasto computacional en cada agente robótico, para analizar, procesar y enviar la información a otros agentes que como se ha mencionado, es ajustado. Adicional a esto, se debe estar supeditado a entornos fijos ya previamente estudiados si el entorno es cambiante o completamente nuevo a lo previamente configurado, en este caso, no funcionará.

Gracias a estos servicios propios de la computación en la nube fue posible implementar aplicaciones robóticas bajo el concepto de robótica como servicio (RaaS), y a partir de este momento se ha comenzado a discutir el término de robótica Cloud. Éste es, un modelo que permite acceso de red conveniente y bajo demanda a un grupo compartido de recursos configurables que pueden ser aprovisionados y liberados con un mínimo esfuerzo de gestión o interacción del proveedor de servicios.

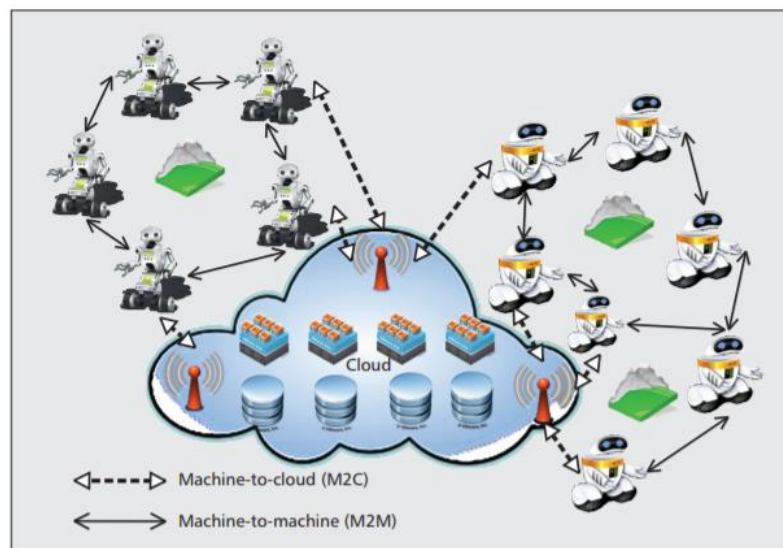
A continuación, se detallará las arquitecturas propuestas desde diferentes ámbitos y perspectivas en los niveles de red y a nivel de servicio.

3.1 Arquitectura de comunicación M2M/M2C

En la actualidad los agentes robóticos están en capacidad comunicarse entre sí y con los servidores dispuestos en la nube. Gracias a esto es posible formar una red de comunicación inalámbrica M2M para robots que trabajan de forma cooperativa entre sí, para enrutar y transmitir la información por medio de los diferentes protocolos de comunicación Zigbee, Bluetooth y WiFi Direct.

Esta arquitectura se define principalmente en dos capas de comunicación, a saber: M2C y M2M como se representa en la figura 3.1 en donde se especifica la interacción y transmisión de la información entre los agentes robóticos y estos, así mismo con la arquitectura dispuesta en la nube la cual se denomina nodo en la red M2M / M2C.

Figura 3-1: Comunicación M2C y M2M en robótica cloud



Referencia: (Hu, Peng, & Wen, 2012)

Una red de robots se forma de manera dinámica y no hay un controlador que coordine el flujo de comunicación en la red. De igual manera, la transmisión de los datos está supeditado por dos variantes. En la primera si dos nodos o robots están dentro del rango de comunicación, se establecen como vecinos. Cuando un robot quiere enviar un mensaje a un nodo de destino (ya sea otro robot o un punto de acceso a la nube), elige al azar uno de sus vecinos y transmite el mensaje junto con un encabezado que contiene el identificador del destino y el id del nodo o robot que lo envía, y un valor de tiempo de espera donde se valida sí el mensaje fue enviado con éxito. (Hu, Peng, & Wen, 2012).

En la segunda variante, el mensaje es transmitido a todos los vecinos de manera simultánea, lo que dependiendo del proyecto esto puede ser contraproducente ya que genera una alta carga de comunicación.

Esta arquitectura hace uso de la primera variante, en cada ciclo del programa, cada nodo elige aleatoriamente un vecino para retransmitir mensajes que no son destinados a sí mismos y siguen siendo válidos. Después de un número suficiente de ciclos de programa, todos los mensajes se retransmitirán a sus destinos con alta probabilidad.

Como aspecto importante esta arquitectura emplea protocolos “*gossip*” en el método de transmisión M2M y M2C. Este tipo de protocolos no requieren planificación o mantenimiento de rutas ya que su principio se basa en la forma en que se propagan las epidemias, por lo tanto, son adecuados para redes robóticas móviles altamente dinámicas. Estos protocolos también son muy simples de implementar y requieren un mínimo recursos adicionales de computación y memoria. (Mbunge, Fashoto, & Dlamini, 2020).

3.2 Arquitectura de computación en la nube elástica

Esta arquitectura está propuesta básicamente combinación de una nube *ad-hoc* formada por un grupo de robots en red y una infraestructura de nube compuesta por diferentes máquinas virtuales (VM), permitiendo una flexibilidad importante. Se proponen entonces tres modelos que son basados originalmente de la computación elástica.

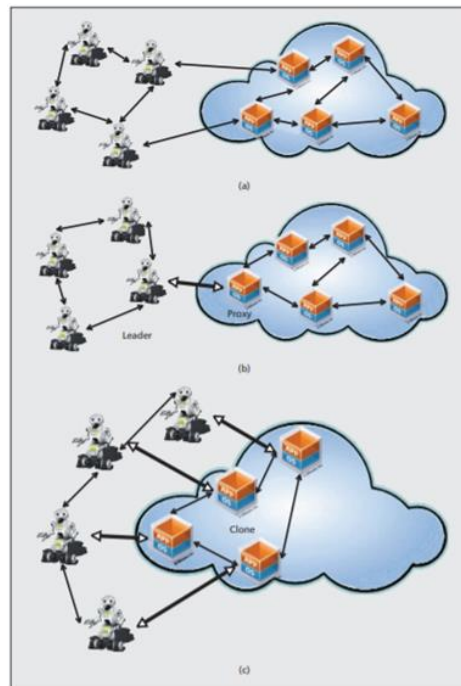
3.2.1 Modelo basado en pares

En este modelo cada robot o cada máquina virtual (VM) se considera como una unidad informática. Estos robots y máquinas virtuales forman una malla informática completamente distribuida permitiendo así dividir una tarea en módulos más pequeños o simples para su ejecución en un subconjunto de nodos en la malla informática. En la figura 3.2 en el apartado a, se representa esta arquitectura.

3.2.2 Modelo basado en proxy

En el grupo de robots en red, una unidad funciona como líder del grupo, comunicándose con una VM proxy en la infraestructura de la nube, para unir la interacción entre la red robótica y la red de internet. El conjunto de las unidades informáticas está organizado en una jerarquía de dos niveles. En la figura 3.2 en el apartado b se representa esta arquitectura.

Figura 3-2: Arquitectura de computación en la nube elástica: a) Modelo basado en pares b) Modelo basado en proxy c) Modelo basado en clones



Fuente: (Hu, Peng, & Wen, 2012).

3.3 Arquitectura de desarrollo de robots basado en la nube

En la actualidad, los servicios proporcionados en la nube son en gran medida los IaaS (Infraestructura como un servicio) que proporciona recursos computacionales y los denominados SaaS (Software como un servicio) que proporciona software de aplicación, los cuales se pueden emplear ampliamente en los servicios PaaS (Plataforma como un servicio).

En particular, PaaS puede pensarse como una forma de extender el concepto de SaaS a la plataforma de desarrollo. Actualmente, se proporcionan varias aplicaciones en los campos IaaS, SaaS y PaaS, pero hay pocos estudios de este tipo en el campo de los robots, y, de hecho, la mayor parte de la investigación de servicios en la nube en el campo de los robots se limita al campo de SaaS. (Yoon & Park, 2015).

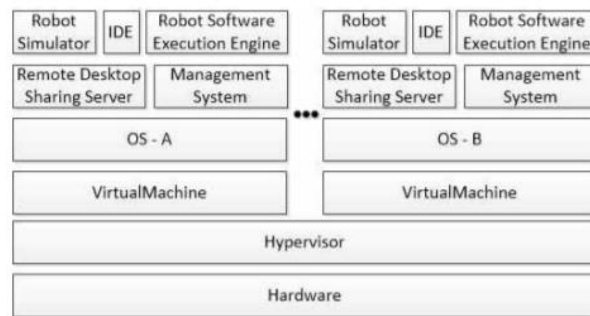
Concretamente las aplicaciones basadas en PaaS ofrecen un entorno de desarrollo integrado (IDE) que maneja tareas relacionadas con el desarrollo de programas como codificación, depuración, compilación y distribución, dentro de un solo software. La arquitectura base para un entorno de desarrollo en la nube, hace uso de una plataforma que incluye el cliente, el sistema de autenticación, gestión y el almacenamiento. Adicionalmente, se cuenta con un entorno de desarrollo integrado y un simulador para proporcionar a los usuarios dentro de la plataforma en la nube.

Para interactuar con estos servicios se hace uso de un sistema de uso compartido de escritorio remoto para conectar el nuevo cliente con recursos mínimos de cliente, el entorno de desarrollo integrado y el simulador del robot. De igual manera se provee de un repositorio el cual almacena los recursos y los productos de desarrollo del interesado. Los recursos para ayudar al desarrollo de los usuarios incluyen componentes de aplicativos informáticos para desarrollar software de robot, modelos de robot para usar en aplicaciones de robot completas y simuladores de robot, y modelos ambientales. Estos recursos se preparan con anticipación junto con la prestación de servicios a los usuarios.

La plataforma en la nube propuesta, está compuesta como se muestra en la Figura 3-3. Existen varias máquinas virtuales en un hipervisor para virtualizar hardware y existen

varios sistemas operativos en la máquina virtual. Los hipervisores incluyen el servidor ESX de VMware, el XenServer de Citrix y el Hyper-V de Microsoft (libro virtualizado a través de hipervisor). Para lograr proporcionar los servicios a los usuarios, se hacen uso de los denominados *Buds* los cuales son servidores que se conectan a los usuarios e incluyen varios sistemas operativos para proporcionar varios entornos de desarrollo y entornos de conducción de aplicaciones. (Yoon & Park, 2015).

Figura 3-3: Arquitectura del IDE basado en la nube



Fuente: (Yoon & Park, 2015)

f

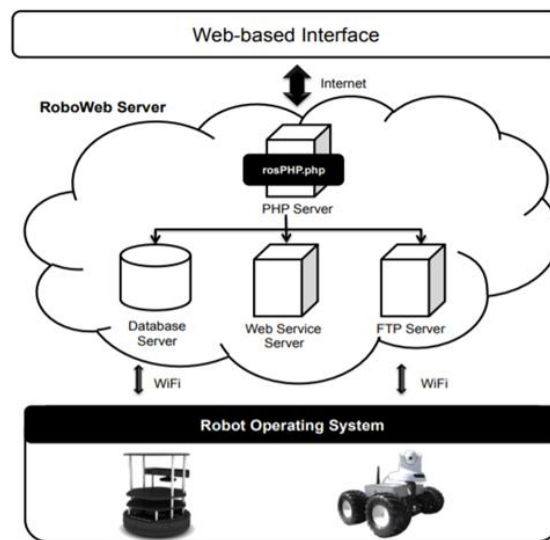
En este entorno, se configura un simulador de robot y un entorno de desarrollo integrado para la construcción de robots, y se puede desarrollar software de robot basado en varios tipos de plataformas según la elección del usuario. El software de robot desarrollado se maneja a través del motor de ejecución de software de robot, y se puede conectar un solo cliente y múltiples servidores de diferentes plataformas. Además, a través de la comunicación entre los motores de ejecución de cada servidor, múltiples sistemas operativos

3.4 Arquitectura *RoboWeb System*

Esta arquitectura tiene como punto de partida el uso de un *middleware* orientado a servicios bajo el marco de robótica como servicio (RaaS) el cual juega un papel vital en la capa de virtualización. Esta permite unir los recursos robóticos de software y hardware por medio de servicios web. Estas aplicaciones web permiten a los usuarios suscribirse a los servicios de los cuales se tengan interés, a través de los cuales podrá interactuar con los agentes robóticos dispuestos en campo. En la consecución de esta arquitectura se hace

uso de los servicios SOAP el cual presenta un modelo de transmisión con una estructuración sólida entre el cliente (servicio suscriptor) y el servidor (servicio publicador) que permite definir de manera clara un contrato entre ambos extremos. Por medio de los servicios web basados en SOAP es posible definir diferentes capas de servicios logrando una mayor abstracción lo cual permite lograr una virtualización de la arquitectura de manera más efectiva.

Figura 3-4: Arquitectura del sistema RoboWeb



Fuente: (Koubaa, 2014)

La figura 3-4 muestra una descripción general de alto nivel de la arquitectura del sistema. La capa inferior consiste en el ecosistema robótico que comprende robots habilitados con ROS (robot Operating System), cada uno de ellos ejecuta su propio nodo maestro ROS. Estos robots móviles están dotados de capacidades de comunicación inalámbrica que les permiten colaborar para realizar determinadas misiones bajo demanda. La plataforma ROS se utiliza para recopilación y transmisión de datos de sensores entre los agentes robóticos y los usuarios finales.

La capa superior define las interfaces web para que los usuarios accedan y manipulen los robots de forma remota. Se incorpora una biblioteca PHP, llamada rosPHP, para actuar como una capa de abstracción en la parte superior de ROS que proporciona las funcionalidades de ROS requeridas para interactuar con robots habilitados para ROS. La

capa rosPHP permite la interacción entre los usuarios finales y los robots a través de los servicios web SOAP y proporciona varias funcionalidades, incluida la conexión al servidor web, obteniendo la lista de robots habilitados para ROS disponibles, obteniendo nodos ROS y temas de robots seleccionados, obtener información sobre sensores de robots, publicar y suscribirse a un ROS tema, crear un nuevo paquete ROS, cargar, ejecutar y detener programas ROS (Koubaa, 2014)

4. Desarrollo del método para la construcción de aplicaciones en robótica de servicios.

Hasta este punto se ha caracterizado los elementos de la industria 4.0 que son aplicables a la robótica de servicios. Por otra parte, en el capítulo anterior se identificaron los modelos de computación en la nube aplicables a la robótica de servicios y la importancia de la robótica como Servicio (RaaS) en el marco de la industria 4.0. A partir de estas consideraciones se procede a formular una metodología que permita realizar el desarrollo de aplicaciones robóticas en el marco de la industria 4.0 empleando computación en la nube. Posteriormente se detallan cada una de las etapas de la metodología propuesta y por último se presenta un caso de estudio, en donde se implementan cada uno de los pasos propuestos. En la figura 4-1 se expone un diagrama de flujo el cual es la metodología misma.

Algunos autores proponen diferentes métodos para el desarrollo de aplicaciones robóticas. (Almasri, Elleithy, & Alajlan, 2016) proponen un método de desarrollo basado en el modelo de fusión de lógica difusa para la evasión de obstáculos en robots móviles, tomando como fundamento la combinación de múltiples sensores, una base de reglas de inferencia *fuzzy*, contemplando como única alternativa en la metodología un robot seguidor de línea. Esta metodología incorpora el uso de simuladores para los experimentos de validación. (Goergen, Maboni, & Gioppo de Souza, 2018) plantean un método de desarrollo de aplicaciones en robótica manipuladora, el cual involucra sistemas neumáticos en su accionar. La metodología utilizada se basa en fases como análisis de necesidades, diseño conceptual, diseño preliminar, diseño de detalle, construcción de prototipos, pruebas y evaluación.

Como se ha visto, estas metodologías se enfocan en aplicaciones específicas o desarrollo de algoritmos, (*aportando una perspectiva holística del diseño de aplicaciones robóticas*), lo cual es incorporado en la metodología propuesta como es la integración de robots móviles y manipuladores, aplicativos web y ambientes de desarrollo provisionados para aplicaciones robóticas.

Otros autores han propuesto metodologías de desarrollo desde la perspectiva del todo, por ejemplo. (Nielsen, Dang, Bocewicz, & Banaszak, 2017) proponen una metodología para la

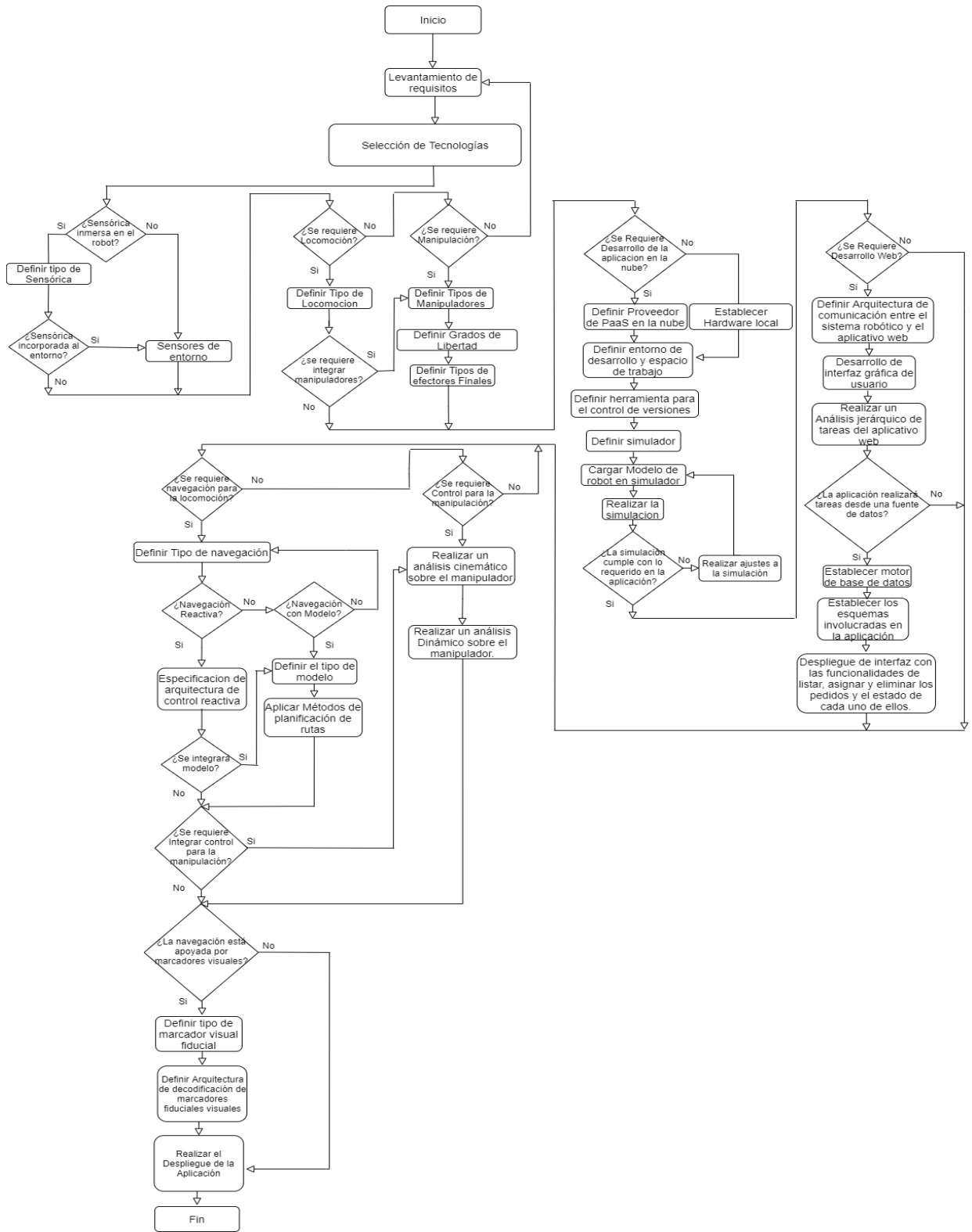
implementación de robots móviles en entornos de fabricación adaptativa. Esta metodología compone etapas como el diseño del sistema del robot móvil, una aplicación industrial adecuada, un concepto de implementación para la aplicación industrial, un modelo matemático y un algoritmo genético. Finalmente se abordan capas como los sistemas de comunicación y el intercambio de datos entre el robot móvil con otros sistemas de fabricación y los operadores de planta.

Este enfoque que, si bien es en gran medida completo, adolece de herramientas de computación en la nube para el desarrollo de aplicaciones y la integración de aplicativos webs que permitan mayor control de los sistemas robóticos en sitio, el enfoque del método esta dado en la arquitectura de desarrollo.

(Rossmann, Ruf, & Schlette, 2009) proponen un enfoque integrado de desarrollo de aplicaciones robóticas, el cual combina los métodos de desarrollo en línea y fuera de línea. Su objetivo es reducir el costo, esfuerzo y pendiente de la curva de aprendizaje a la hora de desarrollar una aplicación robótica que involucre un robot manipulador y una celda de trabajo. Esta arquitectura incorpora el uso de comandos intuitivos para el desarrollo de un flujo de trabajo, permitiendo definir trayectorias y modelar obstáculos en la celda de trabajo. Todas las tareas se coordinan de forma intuitiva a través de una interfaz gráfica de usuario, que incluye programación visual. Finalmente, se utiliza un sistema de simulación para la comprobación de colisiones, visualización y optimización de los programas.

Esta metodología de desarrollo en bloques, adolece de un alcance mayor a la hora de realizar una integración con robots móviles y elementos propios de la industria 4.0 como computación en la nube e interfaces web de control.

Figura 4-1: Metodología propuesta para el diseño de aplicaciones robóticas basado en industria 4.0 y computación en la nube.



4.1 Pasos de la metodología

La metodología propuesta incorpora diferentes etapas para lograr el desarrollo de un proyecto de robótica, como lo son la selección de sensores, locomoción, manipulación, computación en la nube, desarrollo web, navegación, control de manipulación y marcadores fiduciales visuales.

4.1.1 Levantamiento de requisitos

Un proyecto de robótica incorpora una variedad de componentes como lo son, la mecánica, la electrónica y el desarrollo de algoritmos de control. En las que intervienen variables críticas que pueden comprometer la obtención de los resultados esperados por parte del cliente. A partir de lo anterior es importante realizar un plan de acción a partir de las etapas que se mencionan a continuación, con el fin de realizar un correcto levantamiento de requisitos.

- **Ciclo de vida de la aplicación robótica e identificación de usuarios.** Para identificar las necesidades del usuario asociadas al proyecto, es necesario identificar los diferentes clientes o actores. Las necesidades se establecen para cada etapa a partir de un conjunto de reuniones con los clientes involucrados en cada una de las etapas. Definir de manera clara las necesidades de los clientes en cada una de las etapas de ciclo de vida es fundamental para aplicar cualquier metodología de diseño del proyecto. (Riaño, Peña, & Sánchez, 2018).
- **Requisitos del Proyecto.** A partir de la información proporcionada en el ciclo de vida, es necesario hacer un mapeo de cada una de ellas para transformarlas en requisitos del proyecto. El proceso de mapear las necesidades de los clientes a requisitos de proyecto tiene como propósito definir las propiedades físicas finales de la aplicación robótica. Lo anterior garantiza cumplimiento de las necesidades y especificaciones deseadas. Un lenguaje técnico es usado para definir cada uno de los requisitos del proyecto como se ejemplifica en la figura 4-2. Utilizando este lenguaje, cada requisito del proyecto será asociado a una característica técnica o expresión medible y verificable.

Figura 4-2: Ejemplo de requisitos técnicos de un proyecto de robótica.

Requisitos de proyecto	
Funciones	Requerimientos del proyecto
1	Fabricación de bajo costo.
2	Estructura resistente.
3	Herramienta intercambiable.
4	Fácil mantenimiento.
5	Área de trabajo apropiada.
6	Visualmente agradable.
7	Consumo de energía bajo.
8	Alta precisión.
9	Nivel de protección adecuado para el entorno de trabajo.
10	Control de posición y seguimiento de trayectoria.
11	Fácil fabricación.

Fuente: (Riaño, Peña, & Sánchez, 2018).

- **Diagramas de Mudge y Pareto.** El uso de diagramas de Mudge permite definir el grado de importancia o peso de cada requerimiento que fue levantado en la etapa anterior. A partir de los requerimientos del proyecto, los cuales representan necesidades claras, se debe identificar cuáles de estas demandan mayor atención. Por otra parte, los diagramas de Pareto permiten realizar un mejor análisis de los resultados obtenidos luego de aplicar diagramas de Mudge.
- **Especificaciones meta.** Estas se interpretan como acotaciones específicas del proyecto y representan las características de calidad que se quieren con la aplicación robótica como se representa en la figura 4-3 como un ejemplo. Para esto se contemplan los recursos técnicos y físicos que se disponen para el desarrollo del proyecto.

Figura 4-3: Ejemplo de especificaciones meta u objetivos

Especificaciones (Meta)	
Requerimientos	Especificaciones
Errores de Comunicación	0.3BRE (Tasa error en Bit)
Precisión de medición en sensor encoder	12-bit Encoder magnéticos (4096 Configuración, 0.1°/tick)
Complejidad del sistema de control	Parámetros de tiempo de respuesta (Mp < 4%, Tss=4)
Peso de las partes del Robot	Peso 5kg ±20%
Longitud de las partes del Robot	Longitud < 20cm ±10%
Rotación de velocidad	60rpm ±10%
Motor torque.	Torque 1.5N.m
Estabilidad de la estructura	-
Forma de las piezas	Principios inventivos basados en TIPS.
Material resistente	Resistencia a la tracción (50MPa (4,700psi))
Confiabilidad	Tiempo sin fallas (24 meses)
Complejidad de las piezas	Principios inventivos basados en TIPS.
Procesos de manufactura	Número de procesos de manufactura < 5
Adaptabilidad y funcionalidad	Número de entornos de trabajo < 3
Configuración robótica compleja	Principios inventivos basados en TIPS.

Referencia: (Riaño, Peña, & Sánchez, 2018).

4.1.2 Selección de tecnologías

A partir de la selección de tecnologías se podrá determinar los diferentes elementos a emplear y el alcance de cada uno de ellos en el desarrollo del proyecto de robótica a partir de lo dispuesto en el método propuesto.

4.1.3 Sensórica Inmersa en el robot

Los sensores inmersos en el robot son aquellos que se encuentran fijos en la estructura del robot y permiten navegar de manera autónoma en un entorno. Dentro de estos sensores se encuentran los sensores activos y pasivos. Los sensores activos tienen la capacidad de transmitir algún tipo de energía al entorno. Estos sensores proporcionan de manera directa medidas de distancia al entorno. Entre los más conocidos se encuentran

la luz infrarroja, luz laser, ultrasonidos, ondas de radio. Los sensores pasivos los cuales generan directamente una señal eléctrica en respuesta a un estímulo externo, sin la necesidad de una fuente de energía externa, tomando la energía del estímulo. Por otra parte, la sensorica Propioceptiva es aquella que mide las variables internas del robot y se encuentra integrada al sistema de control. Finalmente, la sensorica exteroceptiva mide las variables externas del robot que se generan en su entorno.

4.1.4 Sensórica incorporada al entorno

Los sensores incorporados al entorno son aquellos que proporcionan al robot la capacidad de contar con un comportamiento flexible en su entorno, alejándolo de realizar tareas repetitivas, preprogramadas y acercándolo a un funcionamiento inteligente en relación con el área de trabajo. Estos sensores están compuestos principalmente de sensores activos y pasivos los cuales están dispuestos en lugares específicos del entorno. Es así como un robot apoyado con sensores incorporados al entorno es un dispositivo adaptable a un sinnúmero de tareas. En la tabla 4-1 se detalla las ventajas y desventajas de cada uno de los tipos de sensores a emplear en el método propuesto.

Tabla 4-1: Ventajas y desventajas de los diferentes tipos de sensores para robótica móvil

Tipo de Sensor	Ventajas	Desventajas	Referentes
Sensores Activos	-Transmiten sus propios pulsos de energía, los cuales no dependen de la luz ni de la luz diurna	-Pueden llegar a ser más costosos, y requiere mayor capacidad de procesamiento. -la energía radiada puede ser influenciada o inferida por otras fuentes.	(SungBu, I.O.Lee, D.I.Cho, & JangMyung, 2006)
Sensores Pasivos	-Los datos son más fáciles de interpretar, El reflejo de las distintas bandas del espectro se pueden utilizar para clasificar tipos de cobertura	-Requieren de luz solar para generar imágenes.	(Zenatti, Fontanelli, Palopoli, Macii, & Nazemzadeh, 2016)
Sensores Propioceptivos	-Son sensores diseñados especialmente para la recepción de señales internas del sistema robótico como posición del robot y nivel de batería	-Su utilización es condicionada debido a su especificidad con la que fueron concebidos y desarrollados	(Salter, Michaud, Létourneau, Lee, & Werry, 2007)

Tabla 4-1: Ventajas y desventajas de los diferentes tipos de sensores para robótica móvil (Continuación)

Sensores Exteroceptivos	Tienen la capacidad de medir las señales externas del robot, aunque se encuentren inmersos en la plataforma robótica.	-Similares a las de los sensores activos y pasivos.	(A General Framework for Temporal Calibration of Multiple Proprioceptive and Exteroceptive Sensors, 2014)
Sensores Incorporados al Entorno	Brindan un estado global sobre el entorno en el cual se está desarrollando el sistema robótico	-Capacidad de procesamiento adicional. -Requieren desarrollo de sistemas de anclaje y soporte en el entorno.	(Sanders, 2008)

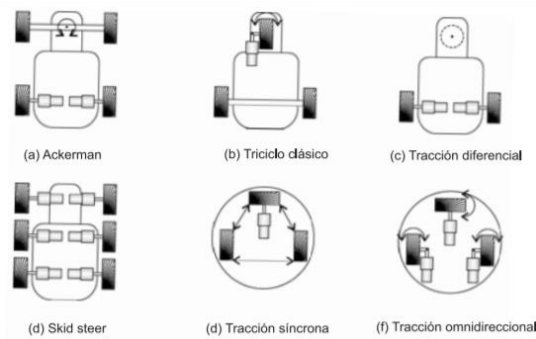
4.1.5 Locomoción

La locomoción hace referencia al movimiento que realiza un agente robótico para moverse de un lugar a otro y desplazarse en función del espacio. La locomoción varía en términos de forma, estructura, velocidad y otros elementos. Los robots móviles se clasifican por el tipo de locomoción los cuales son en general ruedas, patas y orugas. Se toma como una componente de la metodología propuesta la locomoción por ruedas.

En los diferentes tipos de locomoción se cuenta, con la locomoción diferencial. Esta no cuenta con ruedas directrices, por tanto, modifica la velocidad relativa de las ruedas izquierda y derecha. La locomoción omnidireccional posee la capacidad de desplazarse en cualquier dirección sin la necesidad de alcanzar una orientación específica previa, ya que debe contar como mínimo con 3 ruedas activas. En los robots en configuración *Ackerman*, las dos ruedas delanteras giran para controlar la orientación y las traseras se mantienen paralelas. En el triciclo clásico la rueda delantera sirve tanto para la tracción como para el direccionamiento, en el eje trasero sus ruedas son pasivas y se mueven libremente. En la configuración llamada *Skid Steer* se tiene una configuración similar a la diferencial, pero esta al contar como mínimo con cuatro ruedas con tracción y direccionamiento tiene la capacidad de realizar desplazamientos transversales. Finalmente está la configuración síncrona, en donde los motores están separados para la traslación y rotación, lo cual

garantiza el control en línea recta. En la Figura 4-4 se representa de manera gráfica los diferentes tipos de locomoción a base de ruedas en robots móviles.

Figura 4-4: Configuraciones de locomoción.



De igual manera en la Tabla 4-2 se detalla la tabla con las ventajas y desventajas de los diferentes tipos de locomoción.

Tabla 4-2: Ventajas y desventajas de los diferentes tipos de locomoción para robótica móvil

Tipos de Locomoción	Ventajas	Desventajas	Referentes
Diferencial	-Económico -Fácil de implementar -Diseño Simple	-Difícil de controlar -Requiere control de precisión para trayectorias rectas	(Rubio, Valero, & Llopis, 2019)
Omnidireccional	-Permiten movimientos complejos	-Implementación compleja -El movimiento en línea recta no está garantizado por restricciones mecánicas, es necesario implementar técnicas de control.	(Terakawa, Komori, & Matsuda, 2019)
Ackerman	-Fácil de implementar -Un sistema simple de 4 barras controla la dirección	-El robot puede moverse instantáneamente adelante o atrás pero no lateralmente por el deslizamiento de las ruedas	(Hrbáček, Ripel, & Krejsa, 2010)
Triciclo Clásico	-Por su arquitectura no presenta deslizamiento	-Requiere guiado no holónimo.	(A. Batlle & Barjau, 2009)
Skid Steer	-Su arquitectura permite desplazamientos laterales	-El costo de implementación puede ser más alto que otras arquitecturas.	(Madow, Martínez, Morales, & Blanco, 2007)
Síncrona	-Motores separados para traslación y rotación simplifican el control -El control en línea recta está garantizado mecánicamente.	-Diseño complejo. -Difícil Implementación	(Mohd Anuar & Sapiee, 2018)

4.1.6 Manipulación

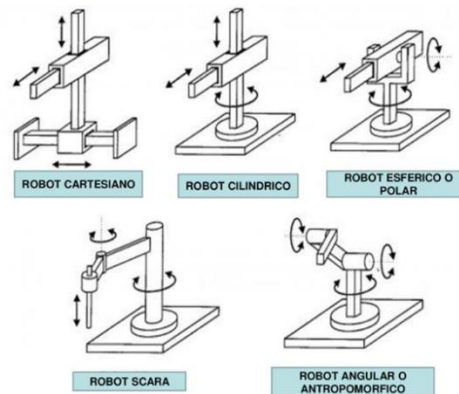
Un robot industrial es un manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas.

Un robot manipulador está constituido principalmente por cinco elementos. Estructura mecánica, transmisiones, sistemas de accionamiento, sistema sensorial, elementos o efectores finales. Una vez definido esto es importante describir de manera general cada uno de los tipos de robots manipuladores. El robot cartesiano emplea 3 dispositivos deslizantes perpendiculares entre sí, para generar movimientos de acuerdo a los tres ejes cartesianos X, Y, Z, con aplicaciones en máquinas de fresado, dibujo o plotter.

El robot cilíndrico se basa en una columna vertical que gira sobre la base. También cuenta con dos dispositivos deslizantes que pueden generar movimientos sobre los ejes Z e Y, con aplicaciones en soldadura, traslado de materiales y montaje. El robot esférico o polar utiliza un brazo telescópico que puede bascular en torno a un eje horizontal. Este eje telescópico está montado sobre una base giratoria. Las articulaciones proporcionan al robot la capacidad de desplazar el brazo en una zona esférica, las aplicaciones más comunes están en soldadura por punto, fundición a presión, soldadura por gas y por arco.

El robot de brazo articulado o *scara* se compone de una columna que gira sobre la base. El brazo contiene una articulación, pero solo puede realizar movimientos en un plano. En el extremo del brazo contiene un eje deslizante que se desplaza en Z, sus aplicaciones más comunes están en operaciones de ensamblaje, pintura y fundición a presión. Finalmente, el robot antropomórfico está constituido por dos componentes rectos que simulan el brazo y antebrazo humano, sobre una columna giratoria. Los antebrazos están conectados mediante articulaciones que se asemejan al hombro y al codo. En la figura 4-5, se representa de manera gráfica los diferentes tipos robots manipuladores, y en la tabla 4-3 se denotan las ventajas y desventajas de cada uno de los manipuladores previamente mencionados.

Figura 4-5: Tipos de robots manipuladores



Referencia: (Barrientos, Peñin, Balaguer, & Aracil, 1997)

Tabla 4-3: Ventajas y desventajas de los diferentes brazos manipuladores

Tipos de brazos manipuladores	Ventajas	Desventajas	Referentes
Scara	<ul style="list-style-type: none"> -Volumen de trabajo mayor que volumen del robot. -Apto para tareas de montaje. -Velocidad y Fuerza. 	<ul style="list-style-type: none"> -Modelo cinemático complejo -Solo permite tareas superficiales. -Sensible a la suciedad del ambiente. 	(B.S.K.K & Ahmed, 2014)
Cartesiano	<ul style="list-style-type: none"> -Movimiento lineal en 3 dimensiones -Modelo cinemático sencillo -Estructura rígida -Puede cubrir grandes volúmenes- 	<ul style="list-style-type: none"> -Volumen de trabajo menor que el del volumen del robot -Solo permite tareas superficiales -Sensible a la suciedad del ambiente. 	(McTaggart, y otros, 2017)
Cilíndrico	<ul style="list-style-type: none"> -Modelo cinemático simple -Volumen de trabajo mayor que el del robot -Permite accionadores hidráulicos 	<ul style="list-style-type: none"> -Región de acceso limitada. -Sensible a la suciedad del ambiente. 	(Andreev & Peregudova, 2020)
Esférico o polar	<ul style="list-style-type: none"> -Volumen de trabajo mucho mayor que el del robot 	<ul style="list-style-type: none"> -Modelo cinemático complejo -Sensibilidad a la suciedad del ambiente -Región de acceso limitada 	(Dikra, Badreddine, Larbi, & Jalal, 2019)
Angular o antropomórfico	<ul style="list-style-type: none"> -Máxima flexibilidad -Volumen del trabajo mayor que el volumen del robot -permite accionadores eléctricos. 	<ul style="list-style-type: none"> -Modelo cinemático complejo (movimientos lineales.) 	(Kabanov & Balabanov, 2018)

4.1.7 Desarrollo de la aplicación

Para el desarrollo de la aplicación fue importante tener claro un conjunto de pasos y alternativas a escoger como lo es el desarrollo local o *cloud*, así como el entorno de desarrollo y las herramientas de control de versiones.

Los proveedores de Paas (Plataforma como servicio) en la nube, disponen de un entorno de desarrollo e implementación completo, con recursos que permiten entregar desde aplicaciones sencillas hasta aplicaciones específicas habilitadas para la nube. en función del consumo que se tenga de los servicios en función del tiempo, se debe realizar un pago de los mismos.

Al igual que el IaaS (Infraestructura como servicio), el Paas incluye además infraestructura (servidores, almacenamiento y redes), pero también incluye *middleware* y herramientas de desarrollo. En el caso de la robótica, actualmente las compañías proporcionan Paas con *middleware* de robótica previamente instalado, así como también simuladores e IDE's de desarrollo.

Por otra parte, un servidor local requiere de una selección de características técnicas que brinden el desempeño adecuado y emplear el tiempo suficiente para realizar la instalación de cada una de las aplicaciones y validar la compatibilidad de la misma, de igual manera resolver todos los errores que puedan presentarse.

En la tabla 4-4 se detalla una comparación el uso de Paas en la nube, y una plataforma a nivel local.

Tabla 4-4: Comparación entre Servidor nube y servidor local

Servidor en la Nube de tipo Paas	Servidor Local
Precios más bajos y escalables	Altos costos de los equipos
Actualizaciones automáticas	Costos de actualización y renovación
Sin costo de infraestructura ni soporte.	Necesidad de espacio físico.
No es necesario realizar Backup	Respaldo manual
Información disponible 24/7	Costos por acceso remoto
Altos estándares de seguridad	La seguridad depende de la organización
Pago por servicios Paas	Costo del servidor, proceso de configuración y mantenimiento
Escalabilidad de recursos infinita	Limitado a la capacidad del servidor.

Finalmente, las herramientas de control de versiones, nos permiten mantener los desarrollos de manera organizada por medio de un repositorio que normalmente se expone en la nube para ser consumido e intervenido por otros desarrolladores, estas permiten resolver conflictos a la hora de que el equipo deba realizar cambios sobre un mismo programa, revertir y deshacer cambios en el código fuente. En la tabla 4.5 detallamos el software de control de versiones más utilizadas.

Tabla 4-5: Características del software de control de versiones

Software de control de versiones	Características
Git	-Su sistema de trabajo está basado en ramas -ramas pueden tener una línea de progreso diferente de la rama principal -Es libre
CVS	-Emplear el modelo cliente-servidor -Mantiene actualizada la copia de trabajo del archivo.
Apache Subversión	-Emplear el modelo cliente-servidor -Directorios están versionados junto con las operaciones de copia, eliminación, movimiento y cambio de nombre.
Mercurial	-Alto rendimiento. -Escalabilidad con capacidades avanzadas de ramificación y fusión y un desarrollo colaborativo totalmente distribuido. -Interfaz web integrada.
Monotore	-Brinda apoyo para la internacionalización y localización. -utiliza un protocolo personalizado eficiente y robusto llamado Netsync.

4.1.8 Simulación

La simulación robótica contempla las técnicas de programación conocidas como programación fuera de línea (*offline programming OLP*), esta permite estudiar múltiples escenarios de un robot antes de colocarlo en producción, los diferentes errores que puedan presentarse pueden predecirse a tiempo. Esta técnica de programación es la mejor manera de maximizar el retorno de inversión. Es importante a la hora de realizar la simulación de un agente robótico tener en cuenta las siguientes consideraciones:

Realizar el desarrollo de modelos, realizar pruebas y optimizaciones, lo que se quiere con esto es analizar el comportamiento de sistemas muy complejos y difíciles de evaluar por otros procedimientos.

Conocer cuál será el comportamiento de los sistemas antes de construirlos, sin perder de vista que los valores finales de los simuladores serán aproximaciones de los valores reales.

Reprogramar el proceso fuera de una línea de fabricación que se encuentre ya en producción, si por alguna razón, cambian las necesidades; así se reduce el tiempo de parada de las máquinas y robots.

Anticipar el funcionamiento y puesta en servicio de las líneas de producción, ya que es un sistema independiente y puede realizarse en paralelo con el montaje de células robóticas.

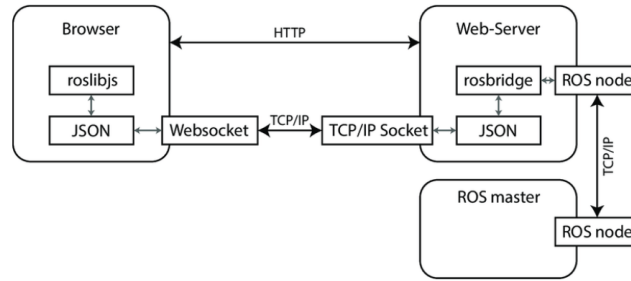
Diseñar correctamente las trayectorias de los agentes robóticos. Durante el modelado los programas simuladores informan de cualquier colisión o pérdida de proximidad entre los elementos del modelo y del entorno.

4.1.9 Desarrollo web

Las aplicaciones robóticas en el marco de la industria 4.0 requieren de aplicativos web que permitan interactuar con los agentes robóticos que se encuentren en campo con el fin de integrarlos al ciclo productivo de la compañía, por lo cual es importante definir diferentes componentes que son importantes a la hora de la implementación en un proyecto de robótica.

- **Arquitectura de comunicación entre bajo y alto nivel.** La topología normalmente es bajo el modelo cliente servidor. Esta arquitectura consiste en un cliente que realiza peticiones a un servidor el cual le da respuesta. Este último posee un sistema operativo multiusuario distribuido a través de una red de computadores que a su vez recibe y transmite la información de los diferentes robots en campo. En la figura 4-6 se representa una arquitectura bajo el modelo cliente servidor propuesta para la interacción de robots por medio de una GUI.

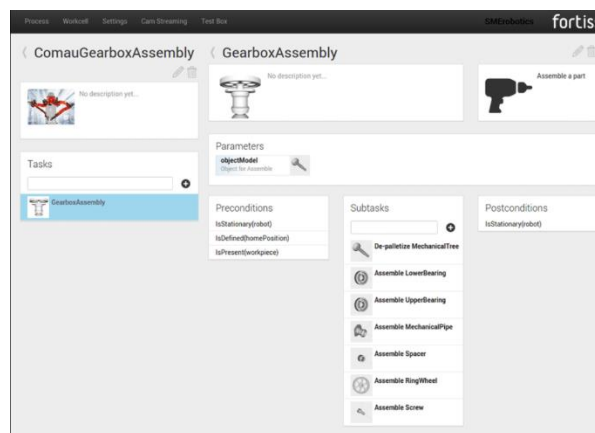
Figura 4-6: Arquitectura cliente servidor aplicado a robótica móvil en el uso de tareas jerárquicas por medio de una GUI



Referencia: (profander, 2014).

- Desarrollo de interfaz gráfica.** Es el entorno visual de imágenes y objetos por medio del cual una máquina y un usuario final interactúan. Lo que se logra con esto es simplificar la comunicación entre una máquina y un usuario. En las aplicaciones robóticas es fundamental contar con interfaces graficas sencillas de comprender y usar, que sea fácil recortar su funcionamiento, contar con datos ricos en el momento de la operación, que la información esté ordenada mediante menús, iconos, barras etc. Esto a raíz de que las aplicaciones robóticas son complejas y una interfaz gráfica sencilla permite optimizar los procesos productivos y una curva de aprendizaje acelerada para cualquier persona indiferente de su grado de conocimiento. En la figura 4-7 se expone una interfaz gráfica web, la cual está diseñada para colocar de manera consecutiva tareas en ensamble para un robot manipulador.

Figura 4-7: Interfaz gráfica para ejecución de tareas de una celda de ensamble por medio de un robot manipulador

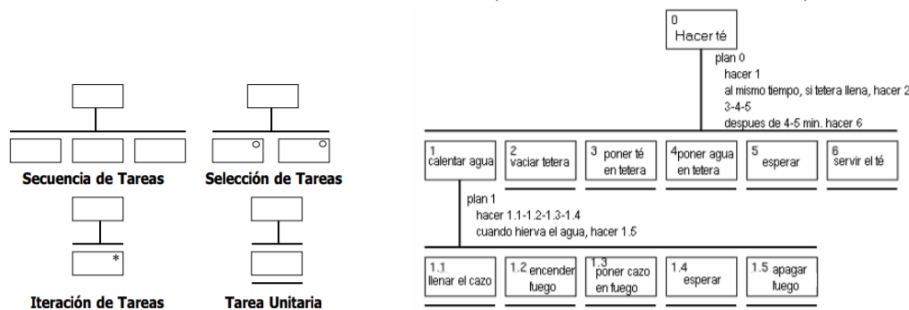


Referencia: (profander, 2014)

4.1.10 Análisis jerárquico de tareas

Es una técnica de análisis de tareas. Este análisis se realiza a partir de una descripción de tareas en términos de operaciones y planes. Las operaciones (descomposición en subtareas) son actividades que realizan las personas para alcanzar un objetivo, y los planes son una descripción de las condiciones que se tienen que dar cuando se realiza cada una de las actividades (Gea & Gutierrez, 2002). El formato grafico se parece a un árbol con ramas y subramas en función de las necesidades. A la hora de la descomposición de una tarea en subtareas, es posible representar cuatro tipos de descomposiciones. Secuencia, es una descomposición en un conjunto ordenado temporalmente de una secuencia de tareas. Selección, es un conjunto de tareas de las que se tendrá que elegir una de ellas. Iteración, es la repetición de un subconjunto de tareas. Finalmente, la Tarea unitaria es una actividad indivisible según el nivel de detalle dado. Por último, el análisis de tareas implica tres etapas enlazadas: recogida de información, diagramación y análisis.

Figura 4-8: Representación de un análisis jerárquico de tareas



Referencia: (Gea & Gutierrez, 2002)

Como se representa en la figura 4-9, Algunas tareas se pueden desglosar con mayor detalle en secuencias. Un plan describe el conjunto de operaciones necesarias para llevar a cabo una actividad, o bien, muestra las circunstancias por las que una operación es realizada antes que otra. Estos planes se añaden a la tabla jerárquica. La descripción de la información se realiza en forma de tabla o en forma de diagrama de árbol que describa las relaciones entre tareas y subtareas (Gea & Gutierrez, 2002).

4.1.11 Fuentes de Datos

En software las fuentes de datos son una colección organizada de información estructurada o datos, típicamente almacenados electrónicamente en un computador para su posterior recuperación, análisis, y/o transmisión. En la robótica definir las fuentes de datos toma importancia a la hora de integrar los sistemas robóticos en las cadenas productivas, ya que por medio estas fuentes de datos se asignarán los trabajos a los agentes robóticos en función de la cola de pedidos que se tengan.

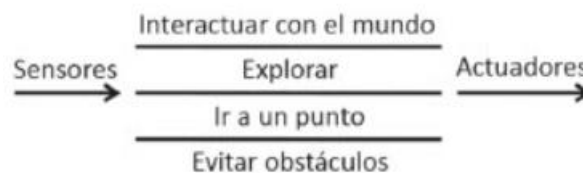
- **Motor de base datos.** Los motores de bases de datos son programas específicos, que tienen como propósito ser el puente entre la base de datos y las aplicaciones que las emplean. Cada uno de estos cumple una tarea específica, que van desde crear la base de datos hasta administrar el uso y acceso a esta. Un motor de base de datos debe cumplir con los siguientes aspectos. Abstracción de la información, independencia, consistencia, seguridad, integridad, respaldo, recuperación y tiempo corto de respuesta. En la actualidad encontramos motores de bases de datos como DB2, SQL server y Oracle considerados los más potentes en el mercado actual.
- **Definir esquemas involucrados en la aplicación.** Definir los esquemas involucrados en la aplicación con el fin de identificar cuáles de estos esquemas realizarán consultas a las bases de datos y establecer un CRUD (Create, Read, Update, Delete). A partir de lo anterior, definir una interfaz que permita representar de manera clara y ordenadas todos los datos que provengan de la fuente de datos con las funciones propias de un CRUD y la actualización de datos en función de las acciones del robot.
- **Despliegue de la interfaz relacionada con la fuente de datos.** A partir de los esquemas definidos es importante realizar una lógica que garantice la integridad de los datos y proporcionar una interfaz funcional para la correcta manipulación de los datos provenientes y que se añaden a las fuentes de datos. Lo anterior requiere realizar el despliegue de estas interfaces con el fin de validar todos los escenarios posibles.

4.1.12 Selección de Control y navegación

Para el uso de esta metodología es indispensable el uso de técnicas de control y navegación tanto para aplicaciones de robótica móvil como de robótica manipuladora. Por lo anterior a continuación se detallarán los diferentes tipos de navegación y control.

- Navegación reactiva.** La navegación reactiva es una de las estrategias de navegación de un robot móvil. La cual consiste en responder de manera inmediata a los estímulos del entorno. Las arquitecturas de control llevan a cabo las funciones de percepción, representación, cognición y actuación, necesarias para proveer una navegación autónoma y segura. En tal sentido integran elementos de hardware y software necesarios para su implementación, organización e interconexión (Acosta, Gallardo, & Pérez, 2016). La arquitectura basada en comportamientos busca descomponer tareas complejas de navegación en un conjunto de comportamientos simples bien delimitados, con esto se logra la flexibilidad para añadir nuevos comportamientos, respuesta en tiempo real a obstáculos no modelados y bajo costo computacional. En la figura 4-9 se representan los paradigmas de navegación reactiva

Figura 4-9: Paradigma de arquitectura basada en comportamientos



Fuente: (Acosta, Gallardo, & Perez, 2016)

Otra técnica reactiva de navegación es la denominada campos potenciales, esta técnica considera al robot como una partícula, que está bajo la influencia de un campo potencial artificial, cuyas variaciones modelan el espacio libre (Navegación en Robots, 2011). La función potencial U en un punto p del espacio euclídeo, se define sobre el espacio libre y consiste en la composición de un potencial atractivo $U_a(p)$, que atrae al robot hacia la posición destino, y otro repulsivo $U_r(p)$ que lo hace alejarse de los obstáculos, como se representa en la ecuación 4.1.

$$U(p) = U_a(p) + U_r(p) \quad (4.1)$$

La fuerza artificial $F(p)$ a la que afecta el vehículo en la posición p , por el potencial artificial $U(p)$ resulta la ecuación 4.2.

$$F(p) = -\nabla U(p) \quad (4.2)$$

Al igual que la función potencial, la fuerza artificial es el resultado de la suma de una fuerza de atracción $F_a(p)$, proveniente de la posición destino, y otra fuerza de repulsión $F_r(p)$ debidas a los obstáculos del entorno de trabajo, como se representa en la ecuación 4.3.

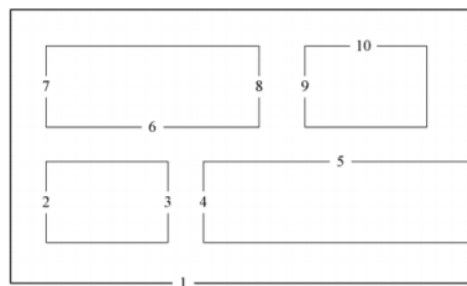
$$F(p) = F_a(p) + F_r(p) \quad (4.3)$$

Finalmente. La navegación basada en campos potenciales se basa en llevar a cabo las siguientes secuencias de acciones: Calcular el potencial $U(p)$ que actúa sobre el vehículo en la posición actual p según la información suministrada por los sensores. Luego Determinar el vector fuerza artificial $F(p)$ según la ecuación (4.2). Finalmente, en virtud del vector calculado construir las consignas adecuadas para los actuadores del vehículo que hagan que éste se mueva según el sentido, dirección y aceleración especificadas por $F(p)$.

- **Navegación con Modelo.** La navegación de robots móviles a partir de modelos se crea a partir del problema de la localización y posicionamiento de robots móviles en un ambiente desconocido y cambiante. Dado lo anterior se han propuestos modelos como geométricos, topológicos y semánticos y métodos de planificación de rutas partir de los modelos o mapas previamente construidos.

Un modelo o mapa geométrico representa los objetos de acuerdo a sus relaciones geométricas absolutas. Este puede ser una representación en forma de grilla, o más abstracto aun, un mapa de polígonos o líneas como se detalla en la figura 4-10.

Figura 4-10: Representación de mapas Geométricos

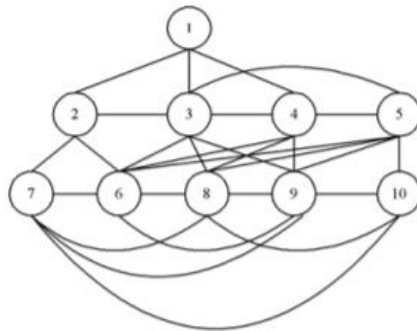


Fuente: (Bambino, 2008)

Por otra parte, los mapas topológicos se basan en la relación geométrica entre las características observadas más que en su posición absoluta. El resultado se presenta bajo la forma de un grafo donde los nodos representan las características observadas y los arcos representan la relación entre las características.

A diferencia de los mapas geométricos, los mapas topológicos pueden ser construidos y actualizados sin considerar la estima de la posición del robot. Esto significa que cualquier error en esta representación será independiente de cualquier error en las estimaciones de la posición del robot. Esto permite generar mapas de grandes dimensiones sin preocuparse por los errores de odometría del robot, dado que, todas las conexiones entre nodos son relativas. En la figura 4-11 se representa un mapa de tipo topológico.

Figura 4-11: Representación de mapas Topológicos

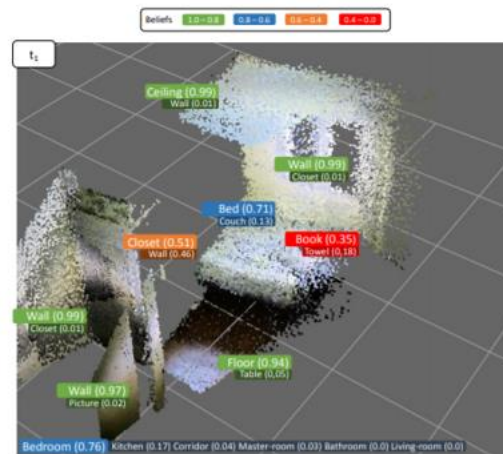


Fuente: (Bambino, 2008)

Por otro lado, los mapas semánticos son representaciones del mundo que permiten a un robot entender no sólo los aspectos espaciales de su lugar de trabajo, sino también el significado de sus elementos (objetos, habitaciones, etc.) y como los humanos interactúan con ellos (e.g. funcionalidades, eventos y relaciones). Para conseguirlo, un mapa semántico añade a las representaciones puramente espaciales, tales como mapas geométricos o topológicos, meta-información sobre los tipos de elementos y relaciones que pueden encontrarse en el entorno de trabajo como se ilustra en la figura 4-12. Esta

meta-información, denominada conocimiento semántico o de sentido común, se codifica típicamente en bases de conocimiento.

Figura 4-12: Representación de mapas semánticos



Fuente: (Ruiz, 2016)

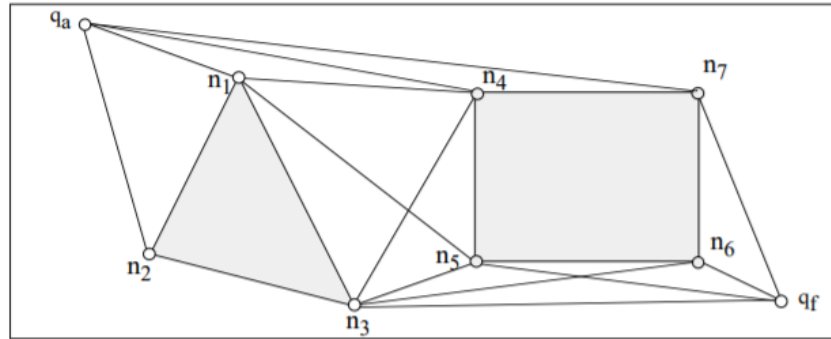
- Métodos de planificación de rutas.** Los métodos de planificación de rutas están encaminados a encontrar una ruta segura capaz de llevar al vehículo desde la posición actual a la posición de destino. El término de ruta segura implica el cálculo de un camino al menos continuo en posición, que sea libre de obstáculos. En vista a lo anterior es necesario emplear métodos de planificación de rutas, los cuales tengan la capacidad de generar las referencias que se le entregará al control de movimiento del robot móvil. Es por ello que a continuación se detallarán los diferentes métodos de planificación de rutas.

Se toma como punto de partida la planificación basada en grafos de visibilidad la cual, por medio de un enfoque geométrico, supone un entorno bidimensional en el cual los obstáculos están modelados mediante polígonos. Para la generación del grafo este método induce el método de visibilidad, según el cual dos puntos del entorno visible si y solo si se pueden unir mediante un segmento rectilíneo que no intercepte ningún obstáculo. Si dicho segmento resulta tangencial a algún obstáculo se consideran los puntos afectados como visibles. (Navegación en Robots, 2011)

De esta manera es como se definen los nodos del grafo de visibilidad la posición inicial, la final y todos los vértices de los obstáculos del entorno, el grafo resulta de la unión mediante arcos de todos aquellos nodos que sean visibles.

En la figura 4-13 se detalla el grafo de visibilidad construido a partir de los obstáculos poligonales existentes en el entorno y las configuraciones inicial q_a y final q_f .

Figura 4-13: Grafo de visibilidad en un entorno de dos obstáculos



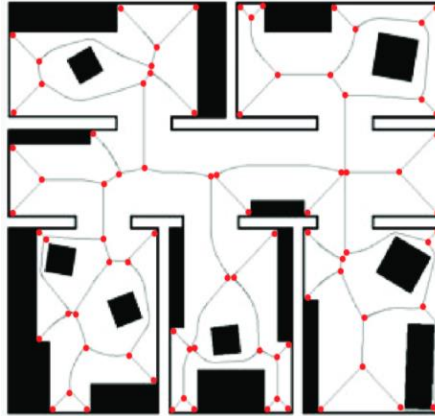
Fuente: (Navegación en Robots, 2011)

Finalmente, mediante un algoritmo de búsqueda en grafos se elige la ruta que una la configuración inicial con la final minimizando alguna función de coste. La ruta que cumple el objetivo de la navegación queda definida como una sucesión de segmentos que siguen los requisitos especificados.

Otro método de planificación de rutas es el que está basado en diagramas de Voronoi. Este método sitúa la ruta lo más alejada posible de los obstáculos. Con ello elimina el problema presentado por los grafos de visibilidad de construir rutas semilibres de obstáculos. Dado un espacio en el que se encuentran situados varios puntos de interés. Los diagramas de Voronoi dividen dicho espacio en secciones que se sitúan bajo la influencia de cada punto. Matemáticamente se define como la unión de los puntos del plano más cercanos a un punto dado.

Dado un mapa objetivo primero se aplica un ensanchamiento de seguridad a los contornos de los objetos para simplificar el trazado y evitar aristas. Después se unen los vértices y otros puntos de interés y se trazan las mediatrices de los segmentos. El camino que asegura que el robot se encuentre lo más alejado de los obstáculos en todo momento será el que marquen dichas mediatrices. En la figura 4-14 se representa la aplicación de diagramas de Voronoi en un entorno cerrado con diferentes obstáculos y paredes.

Figura 4-14: Mapa de una casa aplicando diagramas de Voronoi.



Fuente: (Rojas, Lorente, & de Quirós, 2002)

Este método es considerado simple e intuitivo ya que los algoritmos de navegación deben calcular el camino óptimo para el desplazamiento del robot no son complejos y por lo tanto no necesitaran de grandes recursos hardware. Solo se necesita el plano del recorrido. Y este puede obtenerse y digitalizarse de muchas maneras.

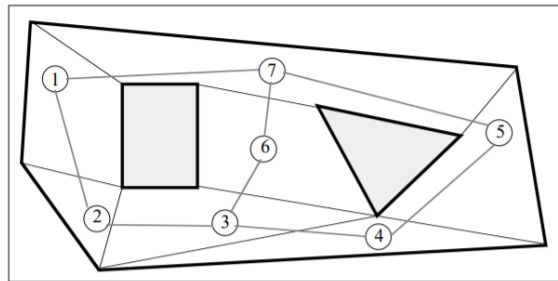
En la planificación basada en modelado del espacio libre se aplica a arquetipos de entornos con obstáculos poligonales, y la planificación en este caso se realiza mediante el modelado del espacio libre. Esto es posible por medio de cilindros rectilíneos generalizados (CRG). Al igual que los diagramas de Voronoi con el uso de los CRG se pretende que el vehículo navegue lo más alejado de los obstáculos. De forma que la ruta que lleve al robot desde una configuración inicial hasta otra final estará compuesta por una serie de CRG interconectados, de tal modo que la configuración de partida se encuentre en el primer cilindro de la sucesión y la final en el último.

El último método de planificación a presentar es la planificación basada en la descomposición de celdas. Este se fundamenta en una descomposición de celdas de espacio libre. Así, la búsqueda de una ruta desde una postura inicial q_a hasta otra final q_f , consiste en encontrar una sucesión de celdas que no presente discontinuidades, tal que la primera de ellas contenga a q_a y la última a q_f . Al contrario que los métodos expuestos anteriormente, no encuentra una serie de segmentos que modele la ruta, sino una sucesión de celdas; por ello, se hace

necesario un segundo paso de construcción de un grafo de conectividad, encargado de definir la ruta. (Navegación en Robots, 2011)

Para la planificación según el método de descomposición en celdas, se precisa la resolución de dos problemas: la descomposición del espacio libre en celdas y la construcción de un grafo de conectividad. El primero de ellos implica construir unas celdas con determinada forma geométrica tal que resulte fácil de calcular un camino entre dos configuraciones distintas pertenecientes a la celda como se ilustra en la figura 4-15, y la comprobación para averiguar si dos celdas son adyacentes debe contar con la mayor simpleza posible. Aparte de estas características, la descomposición global del espacio libre implica que no deben existir solapamientos entre celdas y que la unión de todas ellas corresponde exactamente al espacio libre.

Figura 4-15: Representación gráfica de método de descomposición de celdas



Fuente: (Navegación en Robots, 2011)

4.1.13 Análisis cinemático y dinámico para Sistema de Manipulación

Los robots manipuladores en general se caracterizan por tener limitaciones de diseño relacionado con la estabilidad y la distribución de equilibrio y peso. Por lo anterior a la hora de diseñar un sistema de control robótico es importante realizar un análisis cinemático, esto implica calcular las posiciones relativas entre el sistema de coordenada, lo cual causa un aumento de incertidumbre y un error acumulado en las transformaciones.

Es importante tener en cuenta que generalmente el análisis de la cinemática se aborda de forma directa para así calcular la posición del punto final del robot como función de valores

articulares o ángulos, y de forma inversa para calcular el valor de las coordenadas articulares como función de la posición final (Ramirez Arias, Fonseca, & Astrid, 2012). En la tabla 4-6 se hace una descripción entre la cinemática inversa y directa, teniendo claro que a partir del análisis de la cinemática directo es posible realizar un análisis inverso.

Tabla 4-6: Descripción de Cinemática directa e inversa

Cinemática Directa	Cinemática Inversa
Se conoce: Ángulos articulares y geometría de los eslabones	Se conoce: Posición y orientación del elemento terminal referido a la base
Debe determinar: Posición y orientación del elemento referido a la base.	Debe determinar: Ángulos articulares y geometría de los eslabones para alcanzar orientación y posición de la herramienta

Todo lo anterior implica un análisis dinámico del robot manipulador. Este análisis relaciona el movimiento del robot y las fuerzas implicadas en el mismo. El modelo dinámico establece relaciones matemáticas entre las coordenadas articulares, sus derivadas (velocidad y aceleración), las fuerzas y torque aplicadas a las articulaciones o en el efector final y los parámetros del robot como masas de los eslabones, inercias, etc.

A raíz de lo anterior, para el análisis dinámico de un robot, se proponen diferentes métodos como el de Euler–Lagrange y Newton-Euler, los cuales se exponen en la tabla 4-7 con sus respectivas características.

Tabla 4-7: Características de los métodos para el análisis dinámico


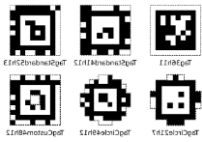


Método	Características
Euler–Lagrange	Este modelo conduce a unas ecuaciones finales bien estructuradas donde aparecen de manera clara los diversos pares y fuerzas que intervienen en el movimiento (inercia, centrífuga, Coriolis, gravedad)
Newton-Euler	Se parte de la ley de conservación de par y fuerza. Se aplica habitualmente en robots de 6 grados de libertad, ya que el coste computacional es mucho menor en comparación con el método Euler–Lagrange.

Una vez se tienen definidos los modelos del manipulador, para su puesta en práctica es importante conocer la trayectoria que seguirá dentro de su espacio de trabajo, un análisis de singularidades con respecto a las posturas del robot en sí y el manipulador que puede generar comportamientos indeseados (Cardoso, Fernández, Marrero-Osorio, & Guardado, 2017).

4.1.14 Marcadores fiduciales visuales y sistemas de decodificación

Los marcadores fiduciales visuales son herramientas que ayudan a la localización de un robot en un entorno industrial. Los sistemas de marcadores fiduciales tienen diseños únicos y varias formas geométricas, los cuales podrían detectarse fácilmente por medio de cámaras tradicionales. En años recientes se han propuesto diferentes tipos de marcadores fiduciales los cuales poseen características como técnicas de codificación, resistencia ante la oclusión y precisión de medida. En la tabla 4-8 se denotarán las características de cada uno de ellos y su representación.

Tabla 4-8: Características de los marcadores fiduciales visuales

Tipo de Marcador	Características	Representación	Algoritmos de Decodificación
ARTag	<ul style="list-style-type: none"> - Patrones planos bitonales con un número de identificación único. - Codificado por medio de técnicas de sumas de verificación y corrección de errores (FEC). 		-Ar-Tags Alvar
AprilTag	<ul style="list-style-type: none"> - Sistema de codificación lexicográfico. - Son de tipo bidimensional. - Codificación de datos entre 4 y 12 bits. 		- vpDetectorAprilTag
CALTag	<ul style="list-style-type: none"> - Localización de los puntos de calibración mediante subpíxeles. - Alta densidad de área. - Robustez bajo oclusión, iluminación irregular. - Minimización de falsos positivos y falsos negativos. - Procesamiento automático sin ajuste de parámetros. 		- ChromaTag
Rune-Tag	<ul style="list-style-type: none"> - Patrón de puntos derivado de códigos de corrección de errores. - Compuestas por un conjunto de elementos circulares de alto contraste dispuestos espacialmente en capas concéntricas. - Interior del marcador se deja libre para cualquier carga útil adicional del usuario 		-LidarTag

Finalmente, para estos marcadores fiduciales visuales se han propuesto diferentes herramientas de decodificación las cuales en su gran mayoría son *open source* y están disponibles para su uso según la aplicación, así como rendimiento y están ligadas a los autores que han propuesto este tipo de marcadores visuales.

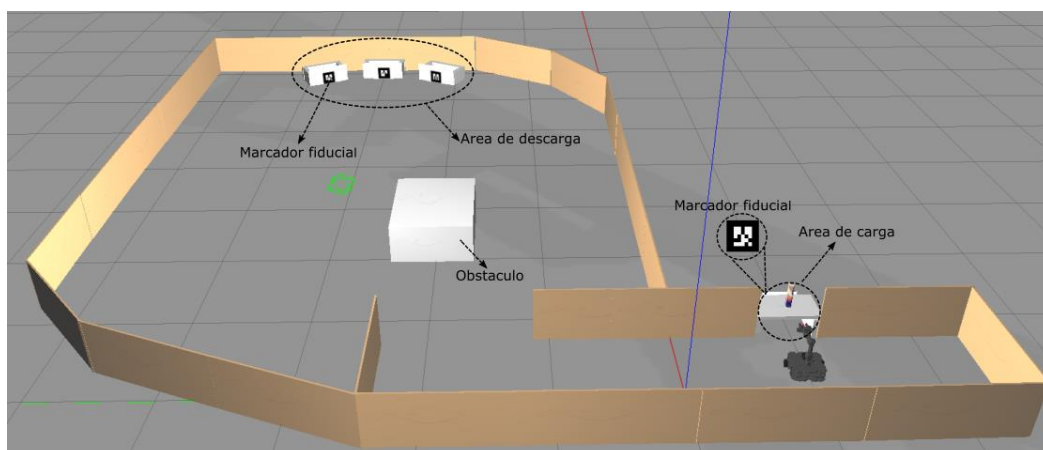
5. Validación experimental y resultados

En esta sección se conducirá un experimento de validación que consiste en el desarrollo de una aplicación robótica logística en la que un robot móvil manipulador se desplaza en un entorno interior. La misión de navegación consiste en tomar un objeto el cual es transportado desde una posición de origen a una de destino a partir de marcadores fiduciales visuales tipo AR-Tags. Para la interacción con el robot se desarrolla un sitio web el cual permite realizar la creación, edición, asignación y borrado de pedidos los cuales son enviados al robot móvil manipulador para que ejecute la tarea encomendada.

5.1 Descripción del experimento

La metodología se probó en un entorno interior que simula una instalación industrial y cuyo plano se observa en la figura 5-1. El espacio consta de un área en la cual se encuentra el robot, inicialmente en las proximidades del objeto que debe tomar para llevar a otro punto todo esto guiado a partir de marcadores fiduciales de tipo AR-Tag, los cuales se encargan de alinear el robot con el área de carga, así como también de tres AR-Tags que están dispuestos en el área de descarga, los cuales sirven de referencia al robot con respecto al pedido encomendado. Finalmente se cuenta con obstáculos fijos como paredes y obstáculos dinámicos como cajas, que aparecen de manera aleatoria en la ruta de navegación del robot.

Figura 5-1: Entorno de diseño experimental en el simulador Gazebo



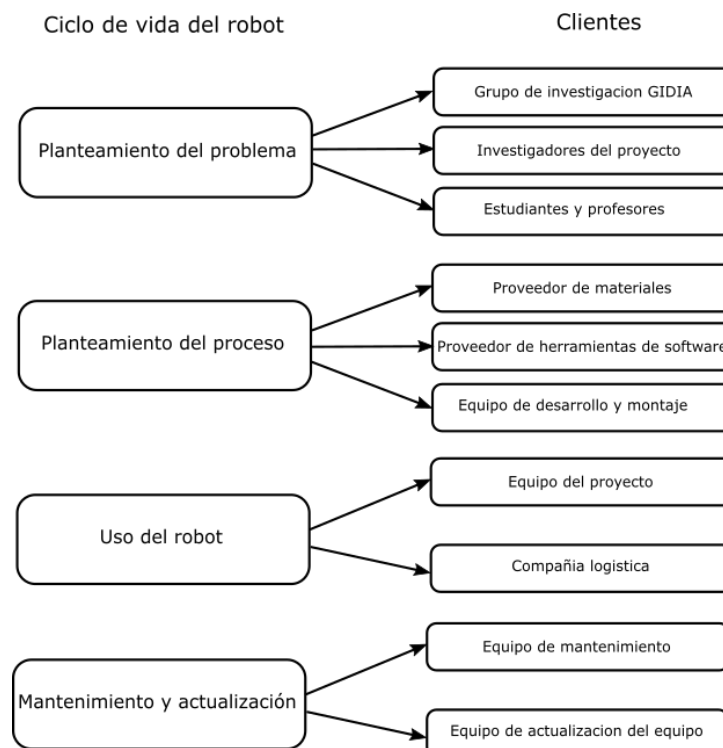
La aplicación se desarrolla en un entorno que simula instalaciones industriales en el cual un robot móvil manipulador debe llevar un objeto de un punto a otro, La metodología aplicada se planteó en la figura 4-1 y a continuación se describe el desarrollo de cada una de las etapas de la misma.

5.2 Establecimiento de requisitos

La metodología contempla una etapa de levantamiento de requerimientos en la cual se entrevista al usuario de la aplicación. En este caso los requerimientos están relacionados con un proceso logístico en el cual un robot manipulador debe llevar objetos localizados en un punto dentro de unas instalaciones industriales hasta un punto de destino y entregar el objeto que transporta, en este caso se realizará por medio de un simulador el cual permitirá integrar todos los recursos para alcanzar el objetivo de esta aplicación.

En la figura 5-2 se muestra las etapas del ciclo de vida de vida consideradas para el proyecto, y los clientes asociados en cada una de ellas. La identificación de los diferentes clientes en cada una de las etapas es vital para un correcto levantamiento de las necesidades en un proyecto de robótica.

Figura 5-2: Relación de clientes con el ciclo de vida del robot



Con la identificación de los usuarios en cada una de las etapas del ciclo de vida, se procede a describir las necesidades de cada uno de los actores. Las necesidades identificadas para el desarrollo de un robot manipulador para aplicaciones logísticas se presentan en la tabla 5-1.

Tabla 5-1: Necesidades asociadas al robot manipulador logístico.

Necesidades asociadas al robot manipulador para aplicaciones logísticas	
Ciclo de vida	Necesidades de los usuarios
Planeamiento del proyecto	Espacio de trabajo adecuado para logística interior
	Capacidad de proporcionar el mismo resultado en cada ciclo de operación
	Estructura que proporcione estabilidad y escalabilidad.
	Proporcionar un prototipo virtual para análisis y simulaciones
Planeamiento del proceso	Control correctamente sintonizado y tolerante a estímulos
	Software de tipo <i>open source</i> escalable en el tiempo
	Entorno de desarrollo en la nube de tipo plataforma como servicio con los elementos requeridos para el desarrollo.
	Herramienta de control de versiones
	Robot móvil visualmente agradable y diseño relacionado con el entorno de trabajo logístico.
Uso del robot	Capacidad de carga de por lo menos 20 kg.
	Autonomía de operación de hasta 2 horas.
	Robot manipulador con capacidad de integrarse con el robot móvil el cual pueda tomar y entregar objetos a la cadena de producción de forma segura y sin comprometer la dinámica del robot móvil.
	El robot manipulador debe contar con los suficientes grados de libertad para tomar objetos a alturas hasta de 15 cm con respecto al piso.
	Capacidad de carga del robot manipulador debe ser de 400g con pinzas como efectores finales para el transporte de objetos cilíndricos.
	Adaptable para ejecutar diferentes tareas con puntos de destino dinámicos con referencias visuales.
	Capacidad de evadir obstáculos fijos como dinámicos en el entorno logístico.
	Tecnológicamente compatible para abstraer nuevas tecnologías.
	Interfaz hombre maquina en la web que permita la creación, modificación, actualización y eliminación de pedidos para asignarlos al robot en campo y almacenarlos en fuentes de datos.
	Interfaz hombre máquina que permita el manejo manual del robot para labores de mantenimiento y verificación.
Mantenimiento y actualización	Componentes fáciles de desmontar e intercambiar
	Componentes estándar y comercialmente accesibles

5.3 Selección de tecnologías

En este caso las tecnologías seleccionadas en función de la metodología propuesta son las siguientes.

5.3.1 Sensórica

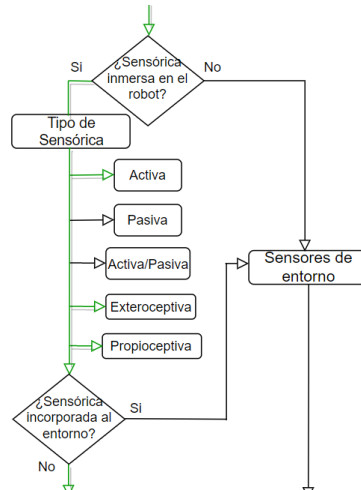
De acuerdo a como sugieren los requisitos en la tabla 5-1, la navegación requiere una detección de obstáculos para lo cual se cuenta con alternativas como sensor por ultrasonido, sensor laser y cámaras, al comparar las características de cada uno de estos se decide emplear un sensor laser de tipo activo, ya que presenta el mejor rango de lectura y un bajo consumo computacional.

Los destinos dinámicos a partir de marcadores visuales requieren de sistemas de visualización, por tanto, se hace uso de una cámara integrada al robot de tipo exteroceptiva, la cual presenta una fácil integración con una plataforma robótica y una resolución adecuada para la identificación de marcadores a distancias de hasta 7 metros.

Para identificar la posición y orientación del robot en la navegación autónoma se hace necesario emplear un sensor de tipo encoder y un giroscopio, los cuales son sensores propioceptivos y hacen parte de la plataforma robótica. Estos sensores cuentan con cualidades suficientes según los requisitos establecidos en el proyecto.




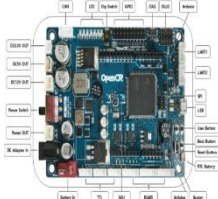
en la figura 5-3, se ilustran los tipos de sensores seleccionados a partir de la metodología propuesta, denotando el flujo con color verde.

Figura 5-3: Apartado de metodología correspondiente a la sensórica



Una vez identificados cada uno de los tipos de sensores empleados, en la Tabla 5-2 se dará una breve descripción de los sensores empleados los cuales están inmersos en la plataforma robótica.

Tabla 5-2: sensores empleados en la validación experimental

Sensor	Características	Ilustración
Raspberry Pi Camera Module v2.1	-Peso 3g -Dimensiones Alrededor de 25 x 24 x 9 mm -Resolución 8MP -Foco Fijo 1m-inf -Sensor Sony IMX219	
Encoder Dinamixel XM430-W210	-Modos de control. Velocidad, Posición. -Tasa de Baudios. 9600 bps - 4,5 Mbps -Torque 2,7 Nm (a 11,1 V, 2,1 A) -Velocidad sin carga hasta 95rpm (@ 14.8V)	
Sensor laser de distancia LDS-01	-Operación de voltaje 5VDC. 400mA -Distancia de detección 120mm ~ 3,500mm. -Tasa de muestreo 1,8 kHz Peso. 125g	
Giroscopio de 3 ejes, Acelerómetro de 3 ejes. (ICM-20648)	-Se encuentra embebido en la placa controladora OpenCR (IMU). 3-Axis Gyroscope with Programmable FSR of $\pm 250\text{dps}$, $\pm 500\text{dps}$, $\pm 1000\text{dps}$ and $\pm 2000\text{dps}$ - 3-Axis Accelerometer with Programmable FSR of $\pm 2\text{g}$, $\pm 4\text{g}$, $\pm 8\text{g}$ and $\pm 16\text{g}$ • Onboard Digital Motion Processor (DMP) - VDD operating range of 1.71V to 3.6V	

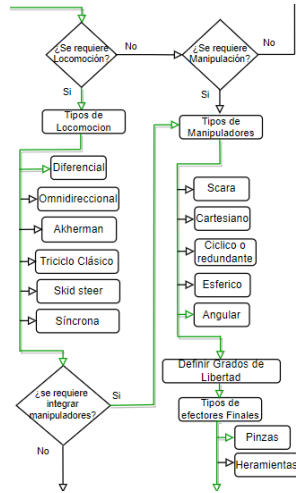
5.3.2 Locomoción y manipulación

Para la locomoción se decide emplear un robot de tipo diferencial debido a que los entornos logísticos tienden a ser complejos y este tipo de configuración permite al robot interactuar en espacios reducidos ya que pueden girar sobre su propio eje. Este robot es el Turtlebot 3, en su configuración *waffle* desarrollado por la compañía Robotics®. La plataforma robótica se integra a un brazo manipulador denominado OpenManipulator de tipo angular, el cual cuenta con cuatro grados de libertad y un efector final que para este caso es una pinza para el transporte de objetos cilíndricos, características requeridas en la fase de levantamiento de requisitos. Esta integración es desarrollada por la empresa Robotics® y permite un uso transparente sin afectar la física de la plataforma robótica. En la figura 5-4

Método para el desarrollo de aplicaciones en robótica de servicios basados en industria 4.0 y computación en la nube

se detalla el flujo tomado en este caso, el cual involucra una locomoción de tipo diferencial y un brazo manipulador de tipo angular con un efector final del tipo pinzas.

Figura 5-4: Apartado de metodología correspondiente a la locomoción y manipulación



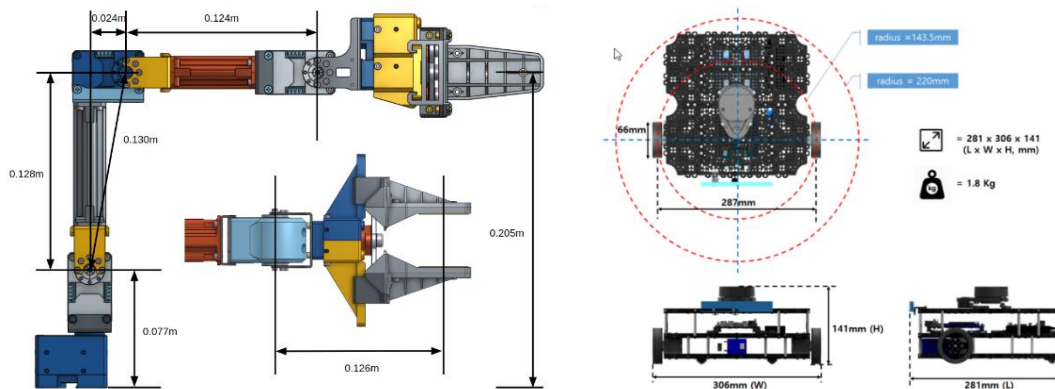
Teniendo la plataforma robótica y el brazo manipulador definido, en la tabla 5-3 y en la tabla 5-4 se detallan las características técnicas de la plataforma robótica y el brazo manipulador. Finalmente, en la figura 5-5 se ilustran las medidas de la plataforma robótica, así como del brazo manipulador.

Tabla 5-3: Características técnicas turtlebot 3

Ítems	Especificación
Velocidad máxima de translación	0.26 m/s
Velocidad máxima de rotación	1.82 rad/s
Capacidad de carga	30 kg
Dimensiones	281mm x 306mm x 141mm
Peso total	1.8 kg
Tiempo de funcionamiento	2h
Tiempo de carga	2h 30min
MCU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
Actuadores	DYNAMIXEL XM430-W210
LDS (Laser Distance Sensor)	360 Laser Distance Sensor LDS-01
Cámara	Raspberry Pi Camera Module v2.1
IMU	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 3 Axis
Power connectors	3.3V / 800mA 5V / 4A 12V / 1A
Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
Conexión con PC	USB
Firmware upgrade	via USB / via JTAG

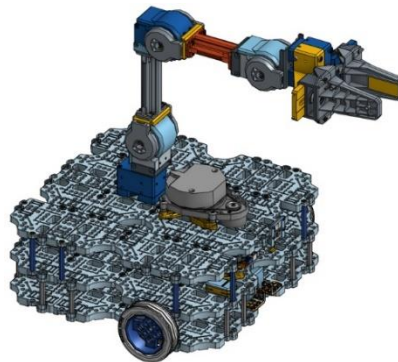
Tabla 5-4: Características técnicas del OpenManipulator-X

Ítems	Especificación
Actuadores	DYNAMIXEL XM430-W350-T
Voltaje	12V
Grados de libertad (DOF)	5 (4 g + 1 DOF Gripper)
Capacidad de carga	500g
Velocidad de las articulaciones	46 RPM
Peso	0.70 kg
Extensión	380 mm
Rango de movimiento de la pinza	20~75 mm
Comunicación	TTL Level Multidrop BUS
Software	ROS, DYNAMIXEL SDK, Arduino, Processing
Controlador principal	PC, OpenCR

Figura 5-5: Dimensiones de plataforma robótica y brazo manipulador

Fuente: <https://emmanual.robotis.com/docs/en/platform/turtlebot3/manipulation/>

En la figura 5-6 se representa el robot manipulador integrado empleado en la validación experimental para el transporte de mercancías en el entorno logístico.

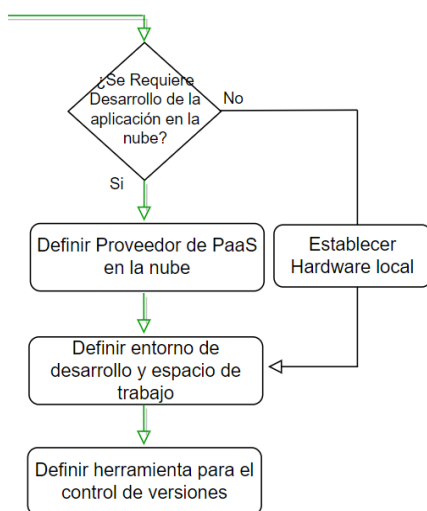
Figura 5-6: Plataforma Turtlebot 3 integrada con OpenManipulator

Fuente: <https://emmanual.robotis.com/docs/en/platform/turtlebot3/manipulation/>

5.3.3 Entorno de desarrollo

Según los requisitos establecidos se decide tomar el flujo de la metodología que involucra una PaaS (plataforma como servicio), el espacio de trabajo es ROS (*Robotic Operating System*) en su versión *Melodic* y un sistema operativo base, Linux Ubuntu 18.04 LTS. En la figura 5-7 se indica el flujo a seguir en esta etapa de la metodología.

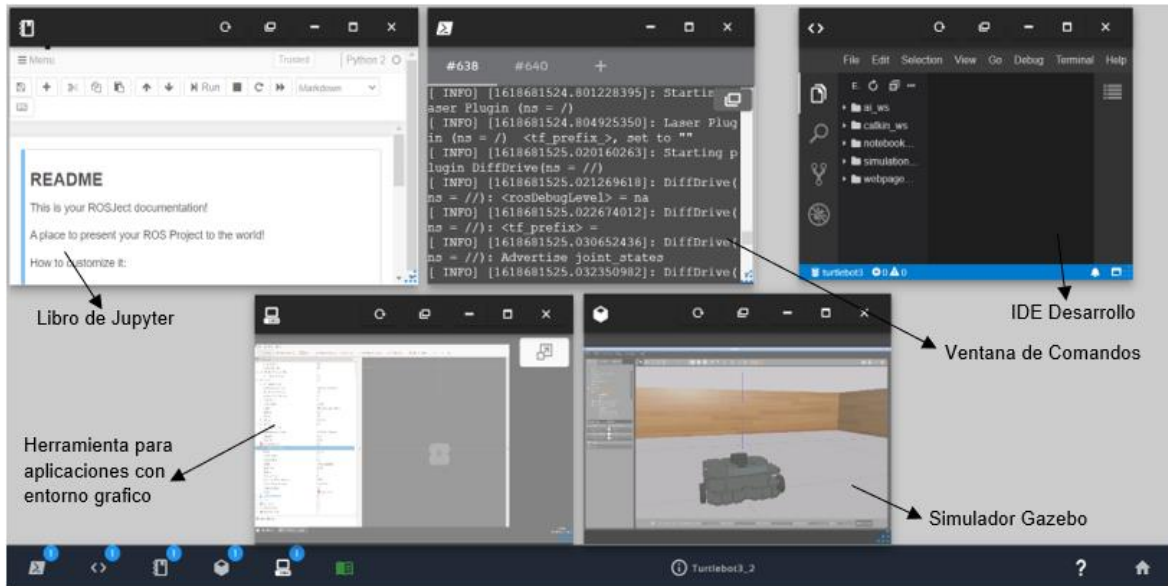
Figura 5-7: Apartado de metodología correspondiente al entorno de desarrollo



La PaaS empleada se denomina ROS Development Studio de la empresa The Construct®. La cual ofrece un entorno de desarrollo en la nube, está provisionada con ROS en su versión Melodic, cuenta con un sistema operativo Linux Ubuntu 18.04 LTS, un IDE de desarrollo basado en Visual Studio Code, un libro de Júpiter, herramienta para el despliegue de aplicaciones gráficas, así como el simulador Gazebo y una ventana de comandos. Esta plataforma como servicio nos ofrece hasta 8 horas continuas de desarrollo por día de manera gratuita. Desde esta plataforma es posible desplegar aplicaciones web propias para realizar diferentes pruebas y desplegar los desarrollos en un laboratorio de robots reales que se tiene dispuesto por la compañía The Construct®.

En la figura 5-8 se representa la PaaS empleada en la validación experimental, indicando cada uno de sus componentes.

Figura 5-8: Entorno de desarrollo ROS development Studio (RDS).

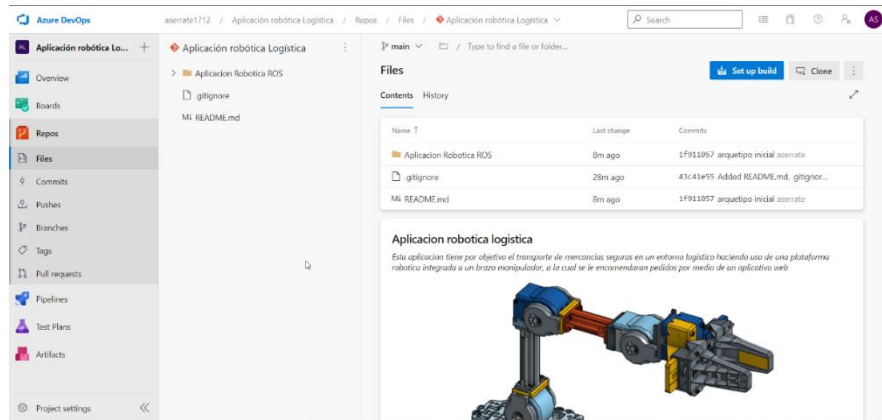


El espacio de trabajo en este caso es el Robot Operating System (ROS), el cual es un *middleware* robótico con la capacidad de proveer servicios standard como la abstracción de hardware, el control de dispositivos de bajo nivel, el paso entre mensajes entre procesos y el mantenimiento de paquetes. Está basado en una arquitectura de grafos donde el procesamiento toma lugar en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros. El middleware está orientado para ejecutarse en sistemas UNIX (Ubuntu -Linux).

La herramienta de control de versiones empleada para el almacenamiento de la aplicación del robot y el aplicativo web es Git. Se establece un repositorio de Git estará contenido en la herramienta de Azure DevOps propia de Microsoft®, con el fin de a futuro integrarse con metodologías de desarrollo de software como lo es DevOps e incorporando etapas como lo es la integración continua y el despliegue continuo de las aplicaciones. Todo lo anterior por medio de los pipelines dispuestos para esto en la herramienta de Azure DevOps.

En la figura 5-9, se representa la interfaz gráfica de Azure DevOps, la cual contiene, herramientas para el marco de trabajo en equipos ágiles, así como la integración de repositorios Git y herramientas para el despliegue continuo de software.

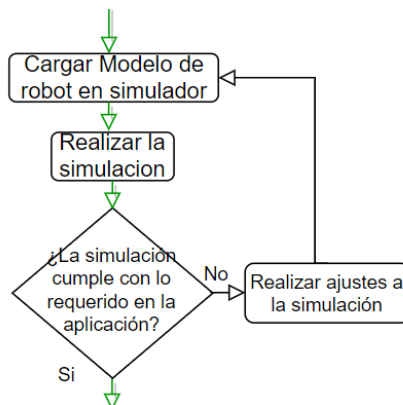
Figura 5-9: Interfaz gráfica de Azure DevOps



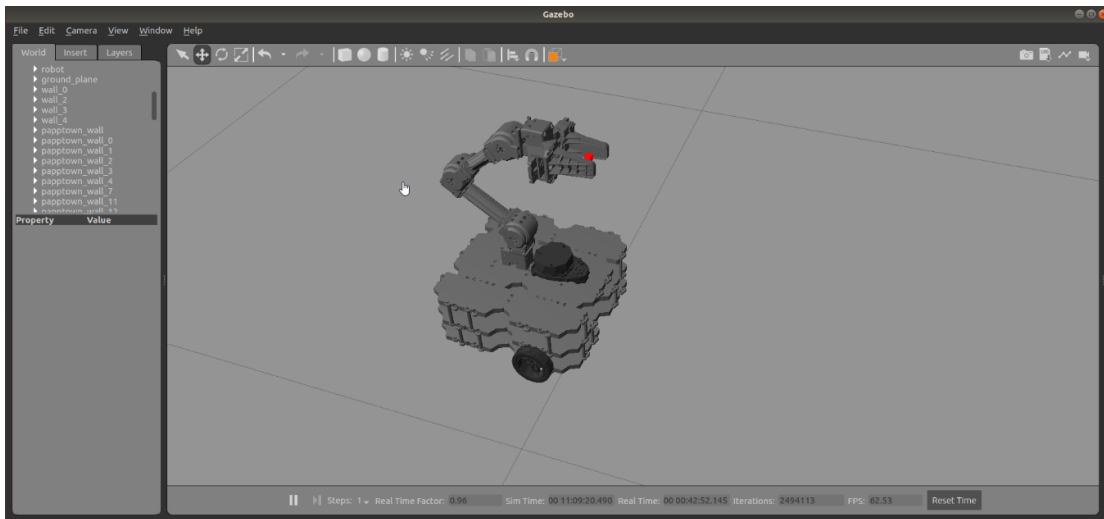
5.3.4 Simulación

Para la simulación se empleó el software Gazebo el cual se encuentra integrado con ROS, se cargó el modelo 3D del Turtlebot 3 manipulador provisto por la empresa Robotis®, el cual viene modelado con todas las fuerzas mecánicas y motrices con el fin de obtener una simulación realista, como es solicitado en los requisitos del proyecto. En la figura 5-10 se detalla en el apartado de la metodología en donde la simulación cumple de entrada con lo requerido en la aplicación según lo que se requiere.

Figura 5-10: Apartado de metodología correspondiente a la simulación

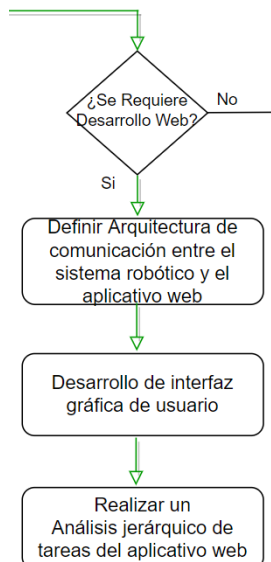


En la figura 5-11, se ilustra el robot manipulador cargado en el simulador Gazebo.

Figura 5-11: Simulador Gazebo con robot *turtlebot 3 open manipulator*

5.3.5 Desarrollo web

A partir de los requisitos solicitados se debe definir una arquitectura de comunicación entre el sistema robótico y el aplicativo web, así como una fuente de datos, por tanto, se decide tomar el flujo de la metodología que integra el desarrollo web con sus diferentes etapas. En la figura 5-12 se indica el flujo a seguir a partir de la metodología propuesta.

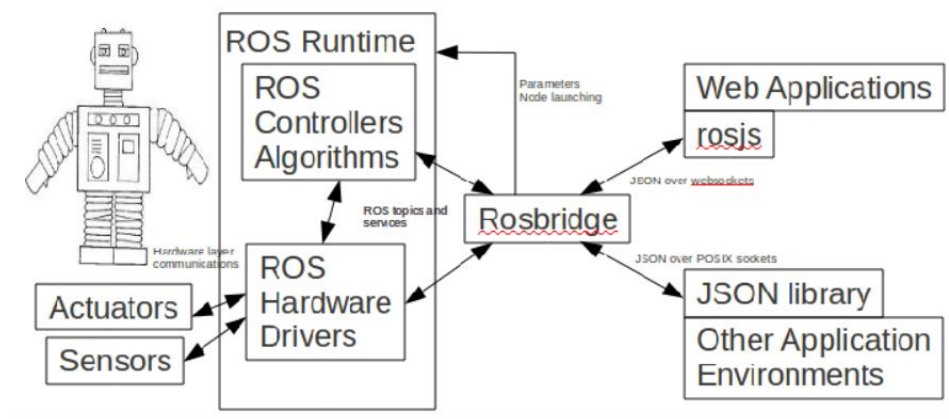
Figura 5-12: Apartado de metodología correspondiente al desarrollo web

Para este caso se ha empleado la API Rosbridge. Esta es una capa de abstracción del *middleware* de ROS con un marco de desarrollo de aplicaciones minimalistas y estándar accesible para programadores de aplicaciones que no son de tipo robóticas. Rosbridge proporciona un acceso programático basado en *sockets* que apuntan a las interfaces del robot y algoritmos proporcionados por ROS.

Rosbride facilita el uso de tecnologías web como JavaScript por medio de librerías como *rosjs*, con el fin de ampliar el uso de las aplicaciones robóticas integrando conceptos como lo son las interfaces hombre-máquina (Crick, Jay, Osentoski, Pitzer, & Jenkins, 2017).

En la figura 5-13 se ilustra la arquitectura empleada en la validación experimental, en donde se tiene el ROS runtime el cual están ejecutando las aplicaciones para el control del robot en un entorno logístico, así como la lectura de sensores y escritura a actuadores. Por otra parte, está *rosbridge* dispuesto en un servidor apuntando a ROS enviando parámetros para la ejecución de nodos y realizando la lectura de mensajes de ROS runtime desde y hacia fuentes externas. Desde la aplicación web por medio de la librería *rosjs* empleando JavaScript es posible enviar y recibir datos en tiempo real en las diferentes interfaces webs, por medio de *rosbride* ya que este actúa como un puente entre las aplicaciones web y el runtime de ROS

Figura 5-13: Arquitectura de *rosbridge* integrada con ROS

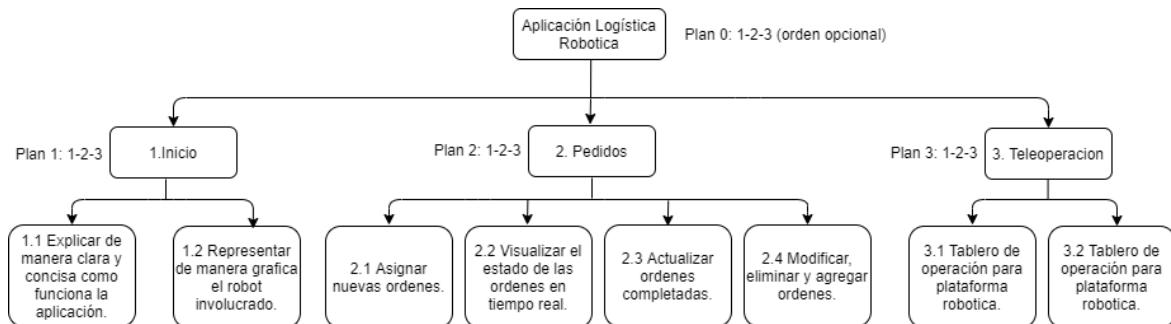


Fuente: (Crick, Jay, Osentoski, Pitzer, & Jenkins, 2017).

Una vez definida la arquitectura de comunicación se da paso a realizar un análisis jerárquico de tareas, el cual permitirá describir las funcionalidades que tendrán el aplicativo

web y la jerarquía de las mismas. En la figura 5-14, se presenta el análisis jerárquico de las tareas de la aplicación web.

Figura 5-14: Análisis jerárquico de la aplicación web logística robótica



Completado el análisis, se detallan las interfaces hombre-máquina realizadas en la aplicación logística la cual se denomina ROS-Orchestador, estas interfaces fueron desarrolladas con tecnologías como HTML, PHP y CSS. En la figura 5-15 se presenta la interfaz de inicio en la cual se da una introducción al objetivo de la aplicación y sus funcionalidades. En la figura 5-16 se presenta la interfaz de pedidos en donde es posible visualizar los pedidos en una tabla con los campos id único, código *tag* o marcador de destino, fecha de creación, comentarios, así como también editar y eliminar pedidos, y dos botones en la parte superior, uno para cargar los pedidos en estado pendientes al robot manipulador y el otro para crear nuevos pedidos.

En la figura 5-17 se presenta una ventana para crear un nuevo pedido en la cual se solicitan datos como el marcador deseado, fecha de creación y un comentario que permita identificar el pedido. En la figura 5-18 se presenta la interfaz de ejecución de pedidos la cual permite conectarse con ROS por medio de *rosbridge* y una ventana con los mensajes de *logs* que envía *rosbride*. De igual manera ubicar el pedido a ejecutar y un botón de ejecutar pedido para colocar el robot en operación con los datos del pedido. Finalmente, en la figura 5-19 se presenta la interfaz de teleoperación la cual permite conectarse con *rosbride* y por medio de los botones enviar los comandos de velocidad respectivos a la plataforma robótica. Esta interfaz es opcional y fue creada para labores de mantenimiento y posicionamiento de la plataforma robótica en caso de fallos.

Método para el desarrollo de aplicaciones en robótica de servicios basados en industria 4.0 y computación en la nube

Figura 5-15: Interfaz de inicio



Figura 5-16: Interfaz de pedidos

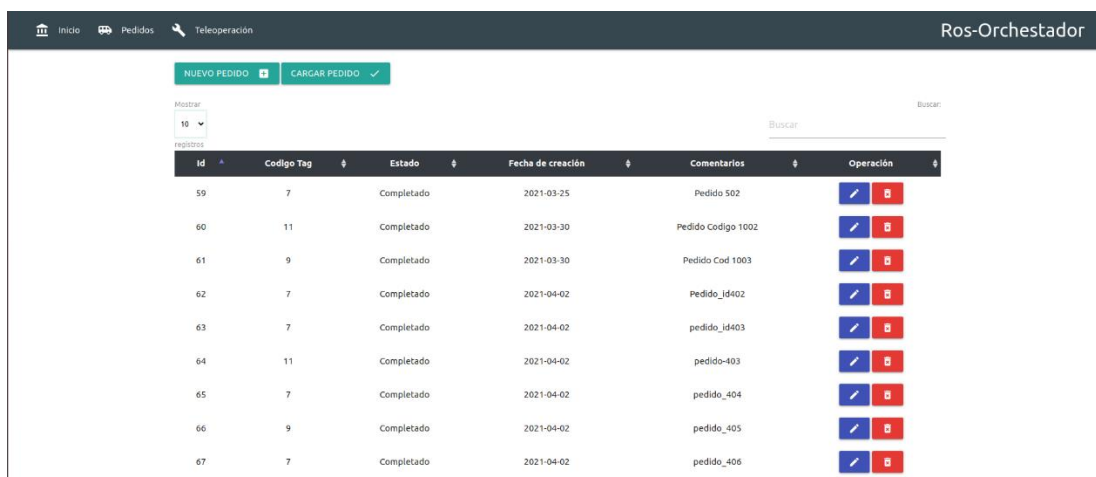


Figura 5-17: Interfaz para ingresar datos

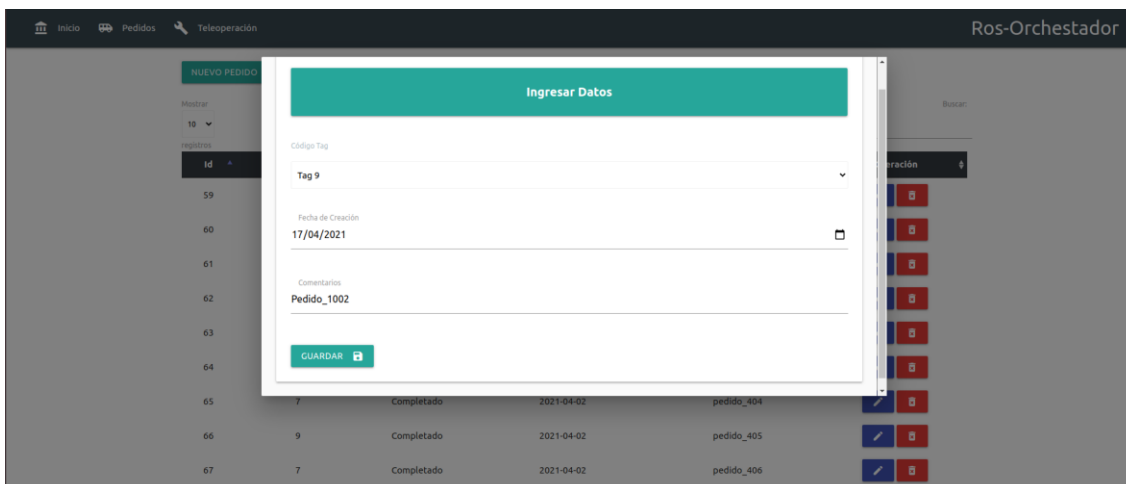
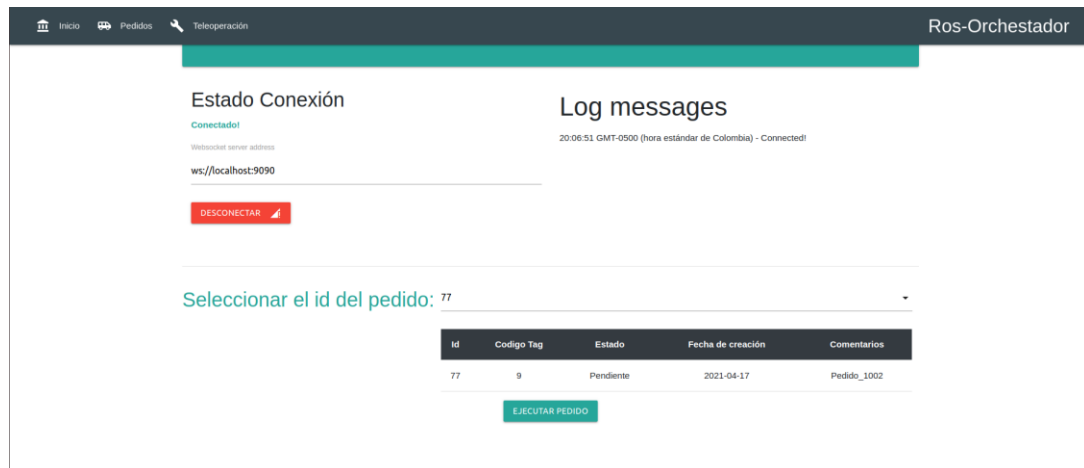
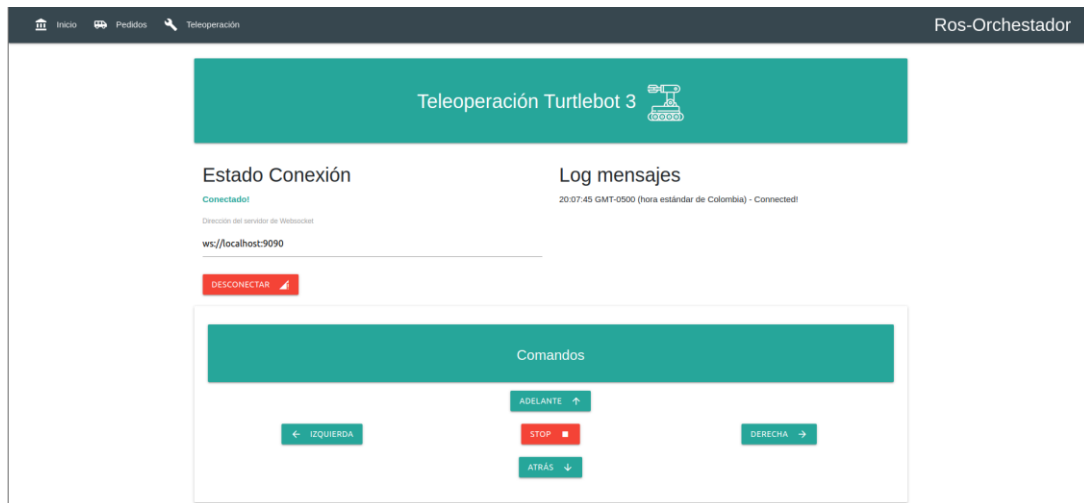


Figura 5-18: Interfaz de ejecución de pedidos**Figura 5-19:** Interfaz de teleoperación

5.3.6 Fuente de datos

Según los requisitos solicitados, se decide adoptar el flujo de la metodología que involucra una fuente de datos con el fin de administrar los pedidos en sus diferentes estados. En la figura 5-20, se ilustra el apartado de la metodología correspondiente a las fuentes de datos.

El motor de base de datos empleado es MySQL®. Este es un sistema de gestión de bases de datos relacionales de código abierto con un modelo cliente-servidor. En el modelo cliente servidor, MySQL crea una base de datos para almacenar información en donde los clientes pueden realizar solicitudes escribiendo instrucciones en lenguaje SQL específicas

en la herramienta de consultas MySQL, la aplicación del servidor responderá con la información solicitada y ésta se expondrá a los clientes.

se empleó el servidor XAMPP® para la administración de las fuentes de datos y el sitio web. XAMPP® es una distribución de apache que incluye diferentes softwares libres, como Linux, Apache, MySQL, PHP y Perl, lo que lo hace muy adecuado según los requisitos de la aplicación al ser código abierto y por su capacidad de integración de herramientas. En la tabla 5-5, se detallan cada una de las características de las herramientas que componen XAMPP®.

Figura 5-20: Apartado de metodología correspondiente a las fuentes de datos

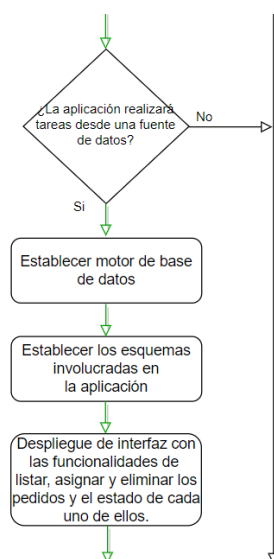
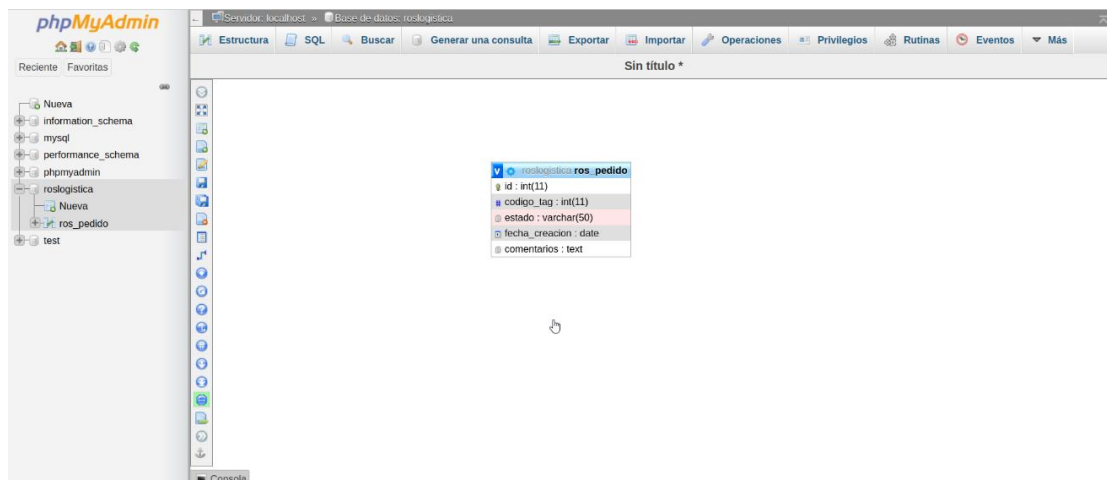


Tabla 5-5: Características de las herramientas de XAMPP®.

Herramienta	Características
Linux	Es el sistema operativo en donde estará instalada la aplicación, este es de distribución libre y posee alto rendimiento.
Apache	Es un servidor web de código abierto para exponer contenidos web
MySQL	Se emplea para el almacenamiento de datos para servicios web.
PHP	Es un lenguaje de programación del lado del servidor que permite crear páginas web, es independiente de plataformas y soporta varios sistemas de bases de datos.
Perl	Este lenguaje de programación se usa en la administración del sistema, el en desarrollo web y en la programación de red.

En la figura 5-21 se ilustra la interfaz gráfica de MySQL, en el cual se creó una base de datos denominada “roslogistica” y se maneja una única tabla llamada “ros_pedido”, ésta tabla contiene los campos id como llave primaria de tipo entero, código_tag de tipo entero de hasta 11 dígitos, estado de tipo *varchar* de hasta 50 caracteres, fecha_creacion de tipo fecha y comentario de tipo texto.

Figura 5-21: Interfaz gráfica MySQL inmersa en el servidor XAMPP®



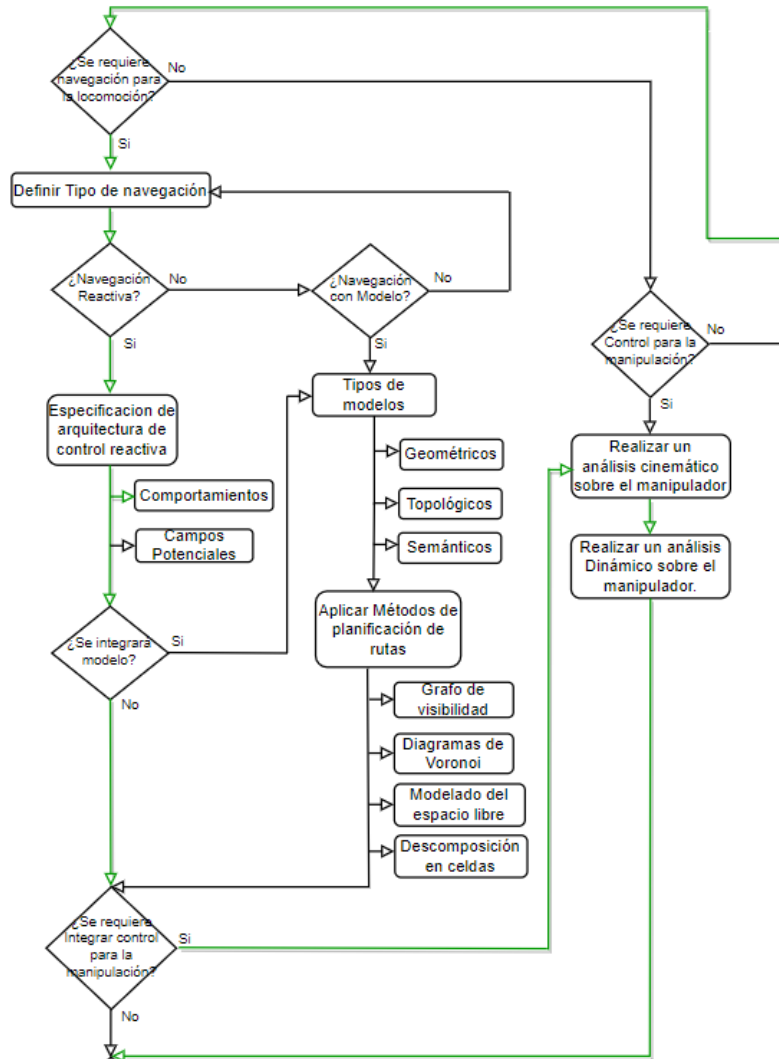
5.3.7 Control de navegación y manipulación

A partir de los requisitos de la aplicación que involucra un entorno logístico el cual implica mapas de entornos cambiantes y obstáculos dinámicos, se decide emplear una navegación reactiva bajo la arquitectura de control por comportamientos apoyada por marcadores fiduciales visuales, la cual se integra por el control de manipulación por medio del software Movelt®, para el transporte seguro de mercancías en el entorno logístico. En la figura 5-22 se detalla el flujo a seguir en el apartado de la metodología relacionado con el control de navegación y manipulación.

Haciendo uso del software ROS se propone implementar una arquitectura reactiva por comportamientos que permita la navegación autónoma de la plataforma robótica turtlebot 3 en una misión de entrega de mercancías a partir del marcador fiducial visual asignado. En esta misión de navegación la plataforma robótica debe enfrentarse a diferentes obstáculos del entorno. Todo esto por medio de nodos que permitan proporcionar datos en unidades de medida conocidas como lo son, la distancia del sensor láser (m), así como también los datos proporcionados por el *enconder* (m) y el Gyrosensor (°), que

proporcionan la posición actual del robot y su orientación en el plano (x,y) y a su vez publicando mensajes a la plataforma robótica Turtlebot 3 de tipo velocidad lineal (m/s) y angular(rad/s).

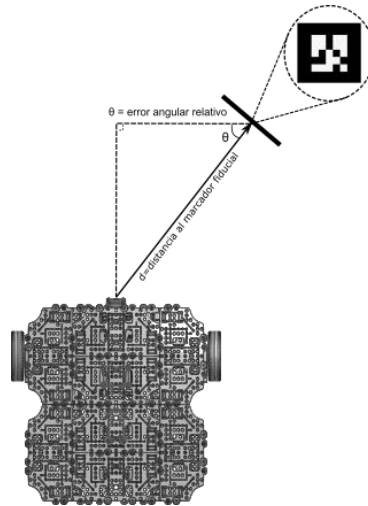
Figura 5-22: Apartado de metodología correspondiente al control de navegación y manipulación



A continuación, se describe cada uno de los comportamientos que hacen parte de la arquitectura de control para la plataforma robótica, como lo son la aproximación para la toma y descarga de objetos, avanzar al destino, orientación al destino y la evasión de obstáculos.

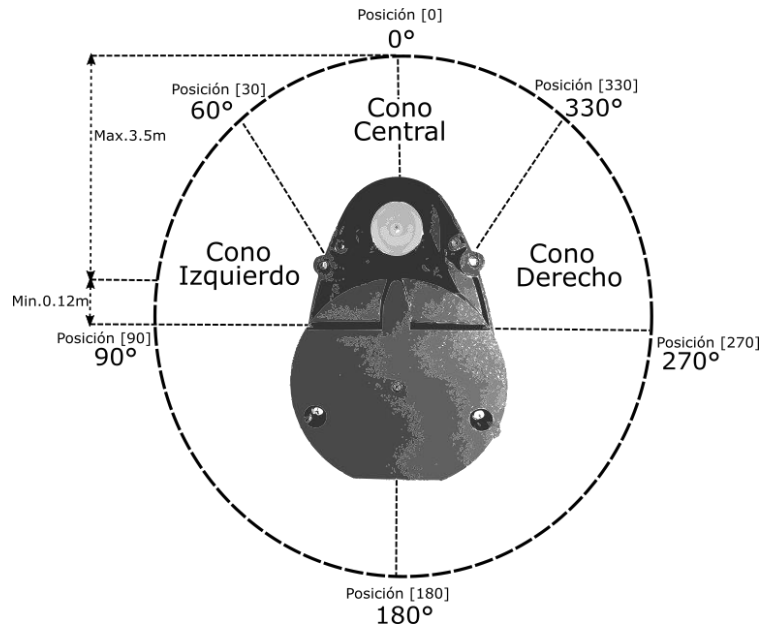
En el primer comportamiento de aproximación para la toma y descarga de objetos, el robot el cual ya tiene por línea de vista al marcador fiducial le es posible alinearse con el marcador por medio del error angular relativo y por medio de un controlador *fuzzy* que incorpora seis reglas de inferencia, el cual controla la velocidad lineal y angular. Si el error angular relativo con respecto al marcador fiducial es mayor a 0.5° el robot se detiene y se alinea nuevamente con el marcador fiducial, y continua con el desplazamiento lineal hasta llegar a una distancia mínima en carga de 0.5m y en descarga de 0.1m. En la figura 5-23, se ilustra la forma en la que se obtiene el error angular y la distancia relativa al marcador fiducial.

Figura 5-23: Representación de distancia y error angular con respecto al Marcador fiducial



El segundo comportamiento es el de evasión de obstáculos. Este comportamiento se tiene como parámetros de entrada la distancia actual que provee sensor láser el cual tiene un rango de lectura de 360° con una resolución de 1° por lectura. Para este comportamiento se emplean tres conos de detección cada uno de 60° . Para el conjunto de valores de distancia obtenidos en cada cono, se realiza un promedio y el resultante es enviado al controlador *fuzzy* el cual incorpora 27 reglas de inferencia, para enviar al robot las velocidades lineal y angular en todo momento. La activación de este comportamiento se da cuando en uno de sus tres conos se detecta un valor de distancia menor a 1.8 m. En la figura 5-23 se presenta la ubicación de cada uno de los conos de detección, los ángulos y la distancia de lectura.

Figura 5-24: Conos de detección en sensor laser



El tercer comportamiento es el de orientación al destino en el cual se plantea un control *fuzzy* con una entrada que es el error angular el cual viene dado por la ecuación 5.1

$$\theta_{error} = \theta_{hipotenusa} - \theta_{robot} \quad (5.1)$$

Donde el $\theta_{hipotenusa}$ es el ángulo que se calcula en todo momento, y es producto de la relación de la posición actual del robot con respecto a la meta y está dado por la ecuación 5.2.

$$\theta_{hipotenusa} = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) \quad (5.2)$$

Y θ_{robot} es el ángulo de orientación del robot entregado a través del Gyrosensor. Se considera como la salida de este controlador la velocidad angular del Turtlebot 3, para este comportamiento se debe tener en cuenta los dos tipos de giros que puede tener, el giro en sentido horario son valores de velocidad angular negativos y en el sentido antihorario serán valores de velocidad angular positivos.

El cuarto comportamiento es avanzar al destino. Para este comportamiento se propone un controlador *fuzzy*, cuya entrada será la distancia actual a la que se encuentra el robot con respecto a la meta, para este comportamiento se deben cumplir reglas que le permita al

TurtleBot 3 disminuir la velocidad lineal a medida que la diferencia con el punto de llegada es menor.

En la ecuación (5.3) se describe el cálculo empleado para conocer la distancia actual (m):

$$d_{actual} = \sqrt{(\Delta x)^2 + (\Delta y)^2} \quad (5.3)$$

Dónde:

d_{actual} , es la distancia del robot con respecto a la meta.

Δx , es la diferencia de la posición final menos la posición actual del robot en el eje x.

Δy , es la diferencia de la posición final menos la posición actual del robot en el eje y.

Una vez definido cada uno de los comportamientos, se aborda la manera en que estos interactúan en la arquitectura de control propuesta para la plataforma robótica. La conmutación de los comportamientos se da de la siguiente manera.

Inicialmente el robot adopta el comportamiento aproximación para la toma de objetos, lo cual implica orientarse con respecto al marcador fiducial único y avanzar hasta el objeto por medio de la conmutación de comportamientos, tomarlo y luego alejarse del área de carga, hasta 0.7m tomando como referencia el marcador fiducial. Posteriormente y según el área de trabajo se habilita el comportamiento de orientación al destino, una vez el error angular es menor que a grados, este se deshabilita e inmediatamente se habilita el comportamiento de avanzar al destino. Si durante este comportamiento el error angular calculado para el primer comportamiento supera los dos grados, inmediatamente se deshabilita el comportamiento de avanzar al destino y regresa al comportamiento de orientación al destino, corrigiendo el error angular generado por el desplazamiento lineal. Esto ocurre hasta que la distancia actual con respecto a la meta o punto de interés sea menor a 0.01 m.

La conmutación entre los dos controladores se hace con la finalidad de evitar que la plataforma robótica se desoriente completamente en medio de su recorrido para ir a la meta.

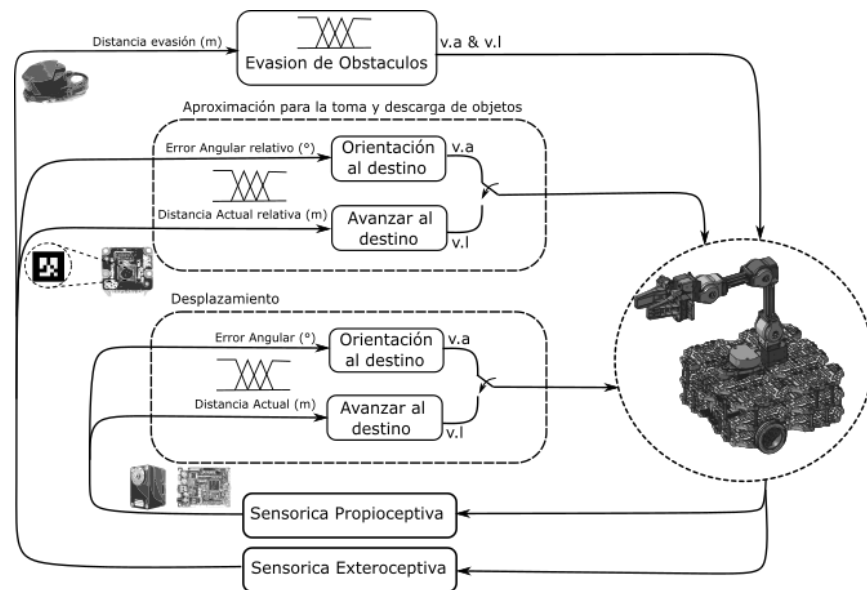
Una vez establecida la interacción entre los tres primeros comportamientos se continúa con la adición de un cuarto comportamiento, que será la evasión de obstáculos, la cual

tendrá como condición de activación la distancia mínima detectada por el promedio de los tres conos del sensor laser.

Si la distancia mínima es menor a 1.8 m se activa dicho comportamiento e inhabilita los comportamientos de orientación y avanzar al destino. Las condiciones que aseguran que la plataforma robótica supere un obstáculo es la ausencia de objetos que se encuentren a una distancia mínima de 1.8 m en cualquiera de sus conos.

Una vez se ha llegado al punto de interés, el robot se orienta a 0° con el fin de visualizar los marcadores fiduciales para la descarga del objeto que lleva consigo. Una vez identificado el marcador adecuado según su misión logística, se activa el comportamiento de aproximación para la descarga de objetos, en el cual la plataforma robótica se orienta con respecto al marcador fiducial y avanza hacia él. Hasta que la distancia con respecto al marcador fiducial sea 0.01m. En la figura 5-24 se ilustra la arquitectura de control para una misión de navegación, donde se hace uso de la sensorica propioceptiva y exteroceptiva la cual entrega valores de distancia y orientación a cada comportamiento según corresponda.

Figura 5-25: Arquitectura de control reactiva



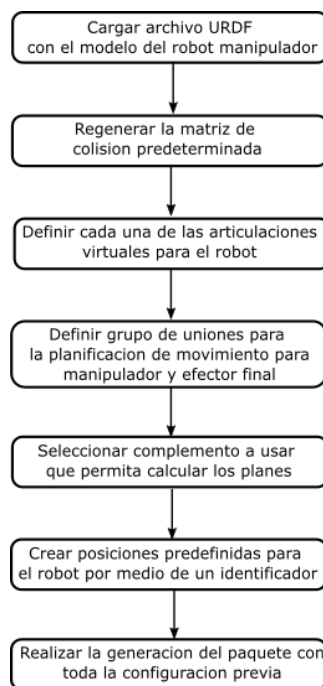
Una vez definido el control para la locomoción, se hace un análisis cinemático y dinámico para el control del manipulador. En este caso para el Open Manipulator X, se emplea el *framework* Moveit! (Motion Planning Framework). Este *framework* proporciona las

trayectorias necesarias para que el brazo manipulador y efector final se ubiquen en un lugar determinado.

Realizar el movimiento de un brazo manipulador no es una labor trivial, ya que se deben producir una secuencia de movimientos que deben seguir cada articulación del brazo en coordinación con otras articulaciones, a esto se le llama planificación de movimiento. El resultado de una planificación de movimiento es la secuencia de movimientos que deben realizar todas las articulaciones del brazo para pasar de la ubicación actual a la deseada. Moveit! proporciona el plan que deben seguir las articulaciones para mover un brazo de una posición a otra.

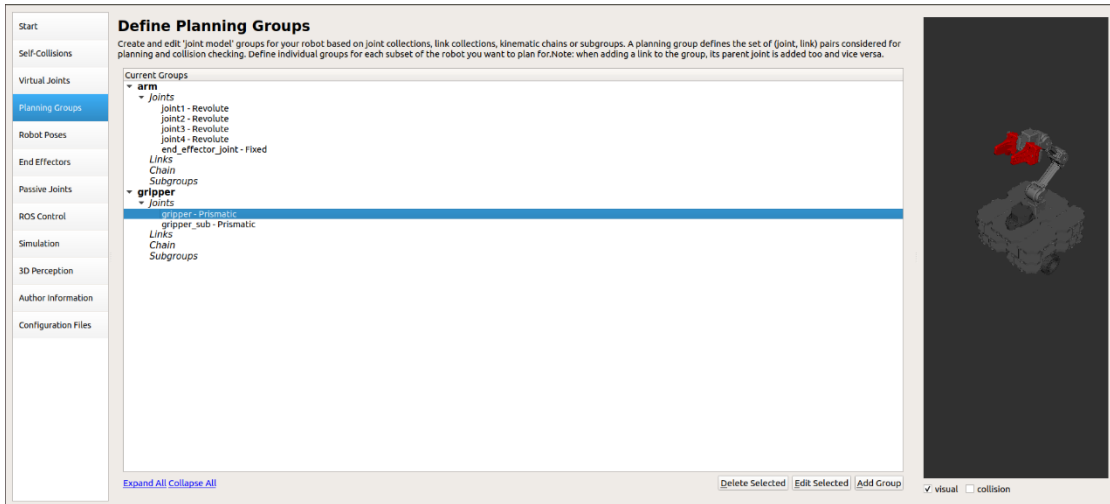
Para iniciar la configuración con Moveit en ROS, se debe ejecutar el asistente de configuración, el cual a partir de un archivo de configuración URDF (Universal Robot Description Format) el cual es una colección de archivos que describen la física del robot como la longitud de las diferentes partes del brazo, qué articulaciones contiene, entre otras cosas. En la figura 5-25, se detalla cada uno de los pasos a la hora de configurar el robot con el asistente de Moveit.

Figura 5-26: Etapas de configuración en el asistente de Moveit!



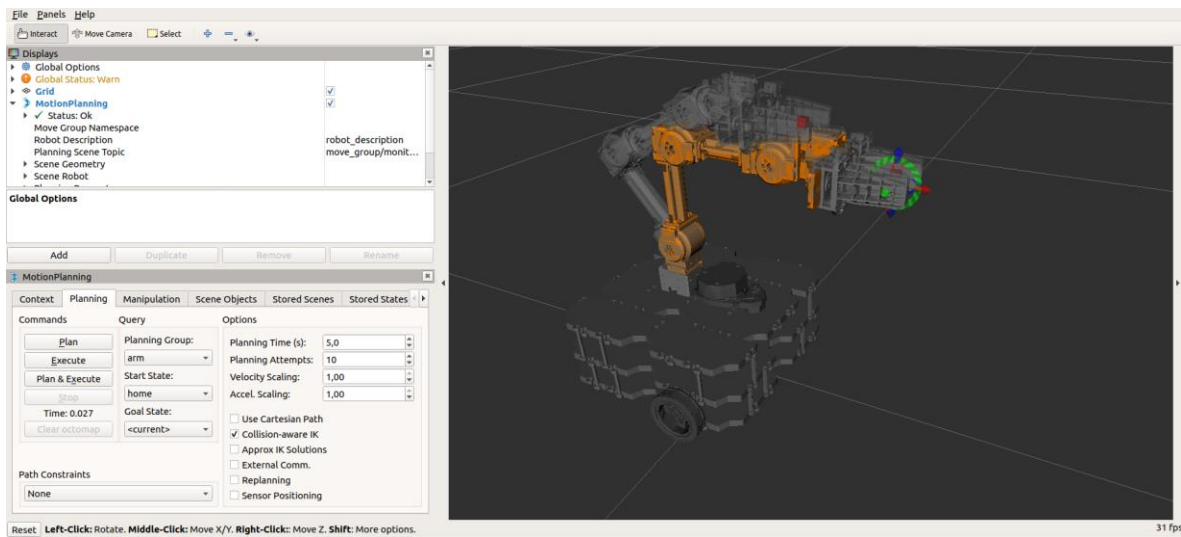
En la figura 5-26, se ilustra la interfaz gráfica del asistente de MoveIt! para el Open Manipulator X.

Figura 5-27: Interfaz gráfica de MoveIt!



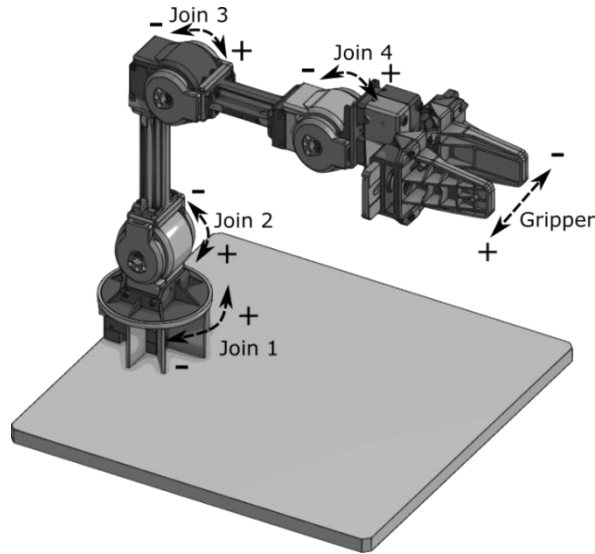
Una vez configurado el brazo manipulador con Moveit, se procede a realizar la planeación de cada una de las posiciones que tomará el robot manipulador, por medio del aplicativo Rviz. El cual representa gráficamente cual será el movimiento del robot en cada una de las posiciones que se configuren. En la figura 5-27, se ilustra la interfaz gráfica de Rviz la cual está integrada con Moveit para la planeación de trayectorias.

Figura 5-28: Interfaz gráfica de Rviz integrado con Moveit!



Para esta aplicación se planearon tres posiciones del robot, a saber: Toma del objeto, Posición de reposo y Entrega del objeto. En la figura 5-28, se detalla el nombre cada una de las uniones con sus nombres y el efector final del robot manipulador.

Figura 5-29: Uniones del brazo manipulador y efector final



En la tabla 5-13 se detallan cada una de las posiciones de las uniones expresadas en radianes, para cada una de las posiciones finales programadas.

Tabla 5-6: Posiciones del brazo manipulador para la aplicación logística

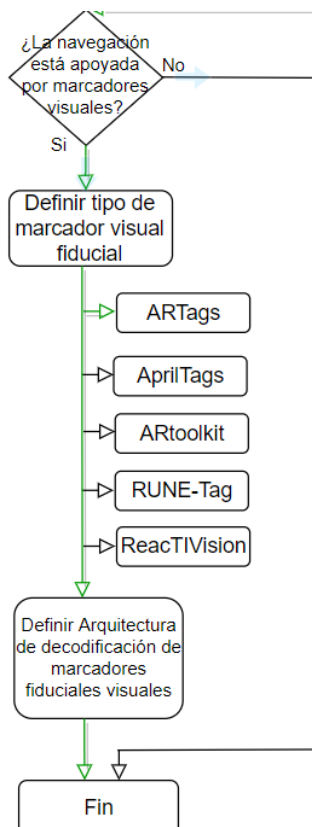
Nombre	Joint1 (rad)	Joint2 (rad)	Joint3 (rad)	Joint4 (rad)	Gripper (rad)
Toma de objeto	0.0	0.1	-0.15	0.0	0.19
Posición de Reposo	0.0	-1.0	0.30	0.70	-0.005
Entrega de objeto	0.0	-1.0	-0.30	0.0	0.19

Finalmente, para la integración de los movimientos del brazo manipulador se emplea la librería de Python denominada `moveit_comander`, la cual envía las posiciones deseadas al grupo del manipulador y al del efector final.

5.3.8 Marcadores fiduciales

A partir de los requisitos de la aplicación que involucra puntos de destino y origen cambiantes se hace necesario el uso de una navegación apoyada por marcadores fiduciales visuales, por tanto, se selecciona el flujo que corresponde al uso de marcadores visuales seleccionando el marcador fiducial tipo ARTag como se ilustra en la figura 5-30, siendo este una versión optimizada de los tradicionales ARtoolkit la cual permite una fácil integración con ROS. Los ARTags son un sistema de marcadores que utiliza la teoría de codificación digital para obtener una tasa de confusión entre marcadores y falsos positivos muy baja con un tamaño de marcador pequeño requerido, empleando un método de enlace de bordes para proporcionar una inmunidad sólida a las variaciones de iluminación. Los marcadores ARTag son patrones planos bi-tonales que contienen un número de identificación único codificado con técnicas digitales robustas de sumas de verificación y corrección de errores hacia adelante (FEC). ARTag cuenta con las tasas de error más bajas y cuantificables numéricamente, no requiere un umbral de escala de grises como lo hacen otros sistemas de marcadores y puede codificar hasta 2002 ID únicos diferentes sin necesidad de almacenar patrones. (Fiala, 2005).

Figura 5-30: Apartado de metodología correspondiente a los marcadores fiduciales

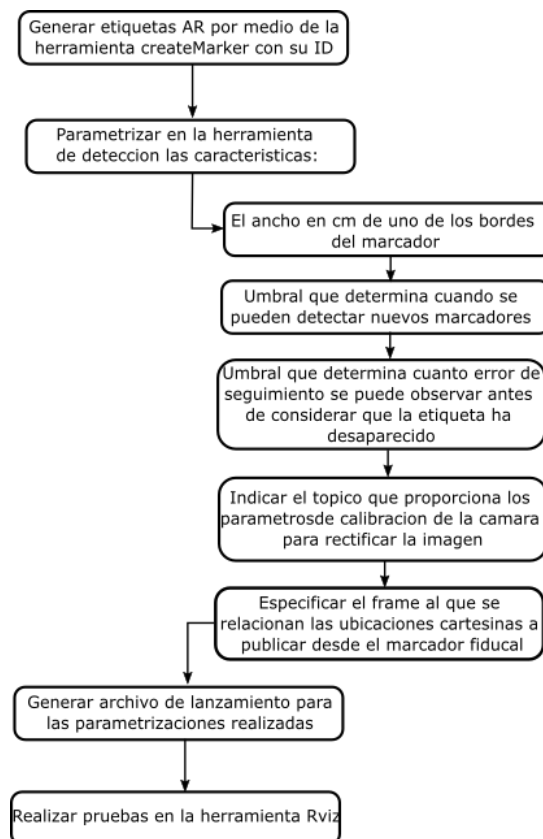


Para la decodificación de los ARTags se empleó la herramienta de decodificación denominada ar_track_alvar la cual se encuentra integrada con ROS y ofrece las siguientes características.

- Generación de etiquetas AR de tamaño, resolución y codificación de datos / ID variables.
- Identificación y seguimiento de la pose de múltiples etiquetas. Esto permite estimaciones de pose más estables, robustez a las oclusiones y seguimiento de objetos de varios lados.

ar_track_alvar presenta umbrales adaptables para manejar una variedad de condiciones de iluminación, seguimiento basado en flujo óptico para una estimación de pose más estable y un método de identificación de etiquetas mejorado que no se ralentiza significativamente a medida que aumenta el número de etiquetas. En la figura 5-31, se detalla cada uno de los pasos a la hora de configurar la lectura de los marcadores fiduciales ar-tags.

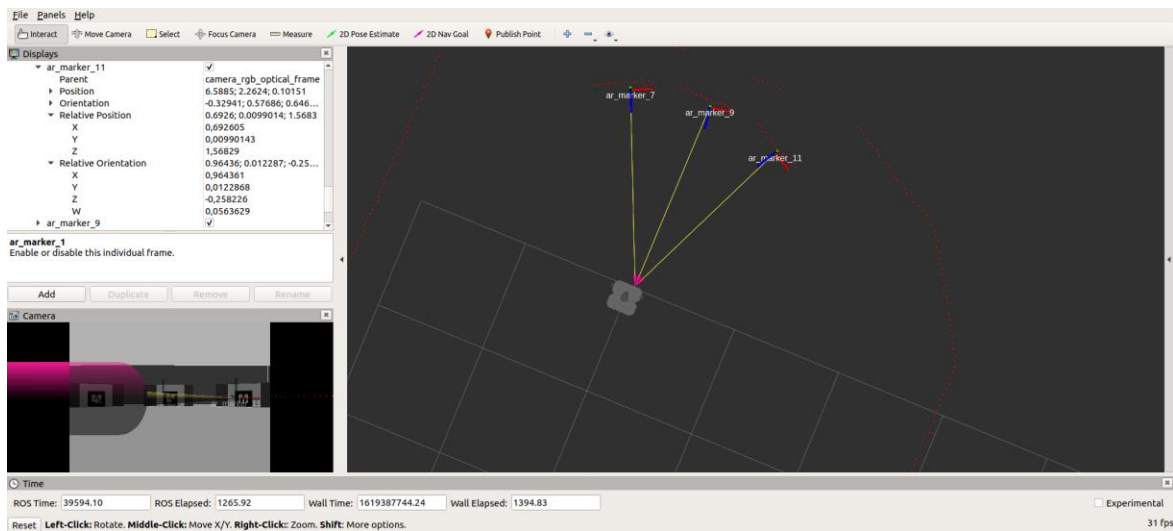
Figura 5-31: Procedimiento para el uso de la herramienta ar_track_alvar



Para el experimento de validación se seleccionaron 4 AR-Tags, 1 para el área de carga y 3 para el área de descarga con ID's 1,7,9,11 respectivamente.

Por medio de la herramienta Rviz, se realizaron diferentes pruebas para su correcta identificación, como se ilustra en la figura 5-32, en donde se hace la identificación de los marcadores fiduciales del área de descarga.

Figura 5-32: Identificación de los marcadores fiduciales en la herramienta Rviz



5.4 Resultados y análisis

Se procede a realizar pruebas que describen cada uno de los tres comportamientos que se llevan a cabo para la navegación autónoma y a través de gráficos se ilustra el desempeño de cada comportamiento en relación con la velocidad lineal y angular a controlar y el comportamiento del brazo manipulador haciendo un análisis de las velocidades y el torque empleado.

El experimento de validación tiene como origen el sistema de coordenadas y con un ángulo de orientación de 0° , con coordenadas de interés de $x=5\text{ m}$, $y=4\text{ m}$. Para luego realizar la aproximación con respecto al AR tag 7.

En la figura 5-33 se ilustra los datos obtenidos de las posiciones X y Y de la plataforma robótica.

Figura 5-33: Recorrido de robot móvil en una misión logística.



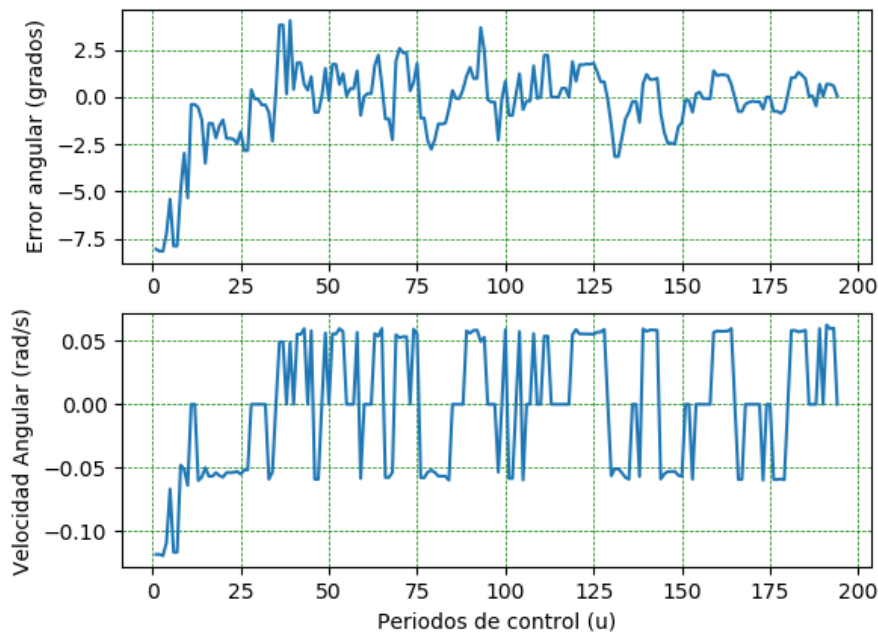
A partir de la misión de navegación previamente realizada se procede a realizar la validación de cada uno de los comportamientos por separado.

5.4.1 Validación de comportamiento de aproximación para la toma y descarga de objetos

Para la prueba de comprobación del funcionamiento del controlador *fuzzy* se adquieren datos de la velocidad angular y velocidad lineal con respecto al error angular relativo y la distancia relativa entregadas por el algoritmo de decodificación para el *tag* destinado para la carga de los objetos.

En la figura 5-34 se ilustra como varía el error angular en contraste con la velocidad angular en los diferentes periodos de control y en la figura 5-35 la variación de la velocidad lineal en función de la distancia.

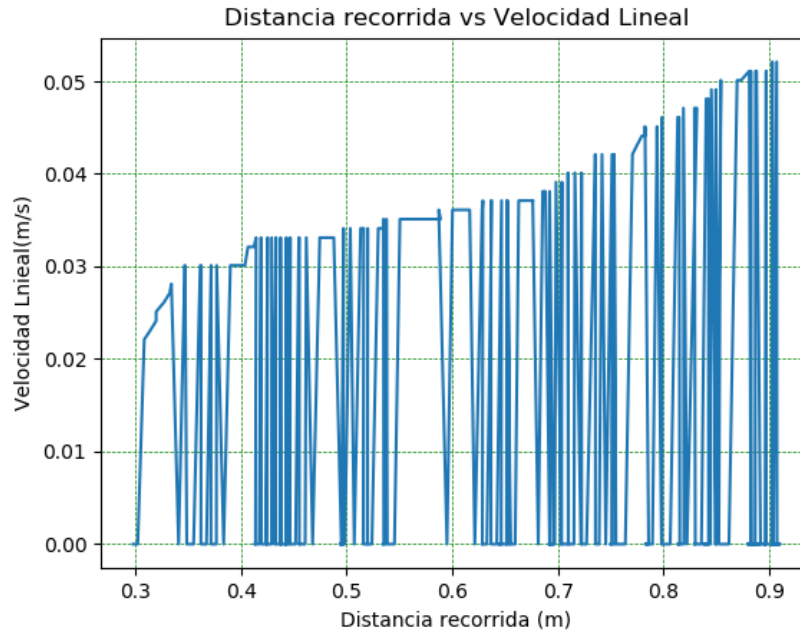
Figura 5-34: Error angular y velocidad angular en función de los periodos de control



Al analizar la Figura 5-34, se encuentra que a medida que el error angular relativo disminuye, la velocidad angular disminuye hasta que se cumpla en umbral de activación de 0.5° . Al superar este umbral se evidencia una velocidad angular de 0° para darle paso al controlador de avance al destino. A medida que el robot se encuentra más cerca del marcador fiducial las medidas de error angular relativo se encuentran más estables y con menos ruido lo cual permite una orientación más precisa a la hora de tomar el objeto en el área de carga.

Con respecto a la Figura 5-35, se evidencia que a menor distancia la velocidad lineal es menor, y se presentan velocidades de 0 m/s continuamente y esto debido a la corrección constante del control de orientación al destino las cuales en el punto medio y final se muestran con menor frecuencia y esto debido a la cercanía con el marcador fiducial.

Figura 5-35: Velocidad lineal en función de la distancia recorrida



5.4.2 Validación de comportamiento evasión de obstáculos

Este comportamiento se lleva a cabo cuando el sensor laser en cualquiera de sus tres conos de detección obtiene una medida menor a 1.8 m, esta medida será la condición que le permite hacer la variación entre las diferentes funciones de pertenencia y proveer en simultaneo valores de velocidad angular y lineal al momento de realizar una evasión de obstáculos. En la figura 5-33, se evidencia la ruta que tomó el robot tanto para evadir las paredes una vez realizada la carga de los objetos en un área ajustada, como para evadir el obstáculo que se encuentra en su trayectoria hacia el punto de destino de manera imprevista. Lo anterior gracias al conjunto de reglas implementadas para el comportamiento de evasión, se evidencia que la plataforma robótica al enfrentarse a las paredes describe una trayectoria amplia con respecto a obstáculo debido a que éste es fijo, mientras que al encontrar el obstáculo de manera imprevista, el robot describe un cambio de trayectoria más pronunciado.

Para las figuras 5-36, 5-37 y 5-38 se toma una muestra de los periodos de control donde se evalúa el comportamiento de cada una de las distancias de los tres conos de detección, con respecto a las velocidades angulares y lineales que presentan variaciones en función de los cambios de distancia.

Figura 5-36: Prueba de validación de comportamiento de evasión de obstáculos para la distancia del cono derecho

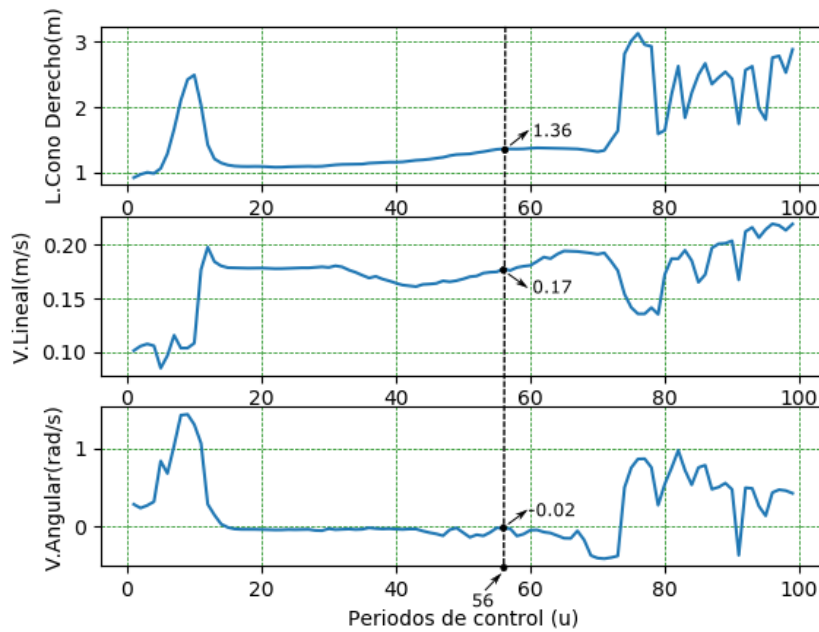


Figura 5-37: Prueba de validación de comportamiento de evasión de obstáculos para la distancia del cono central

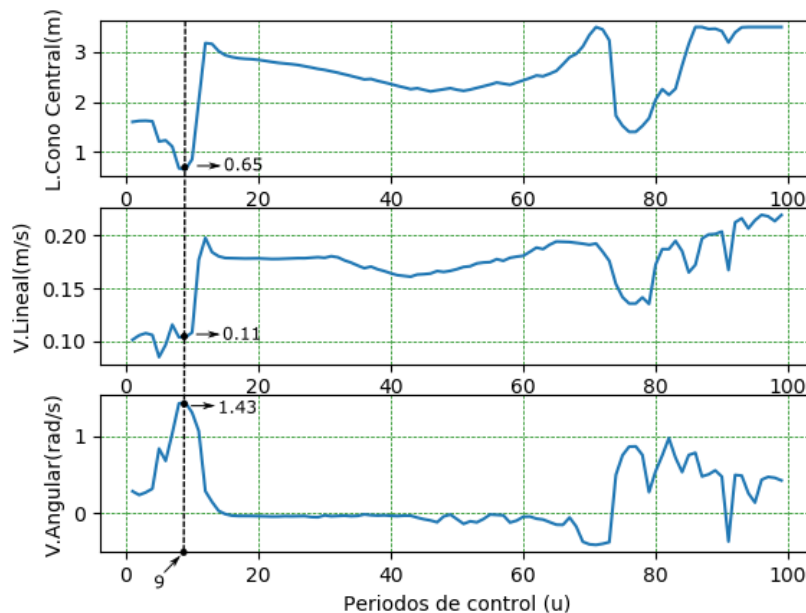
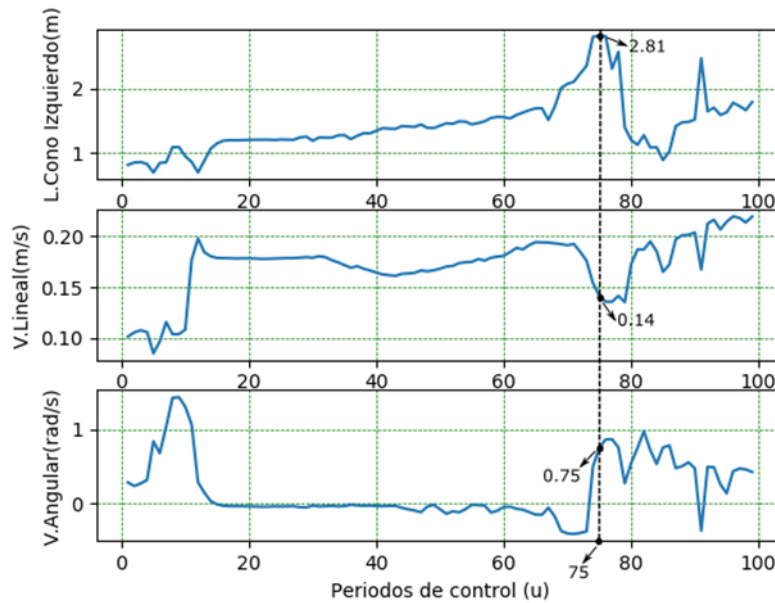


Figura 5-38: Prueba de validación de comportamiento de evasión de obstáculos para la distancia del cono izquierdo

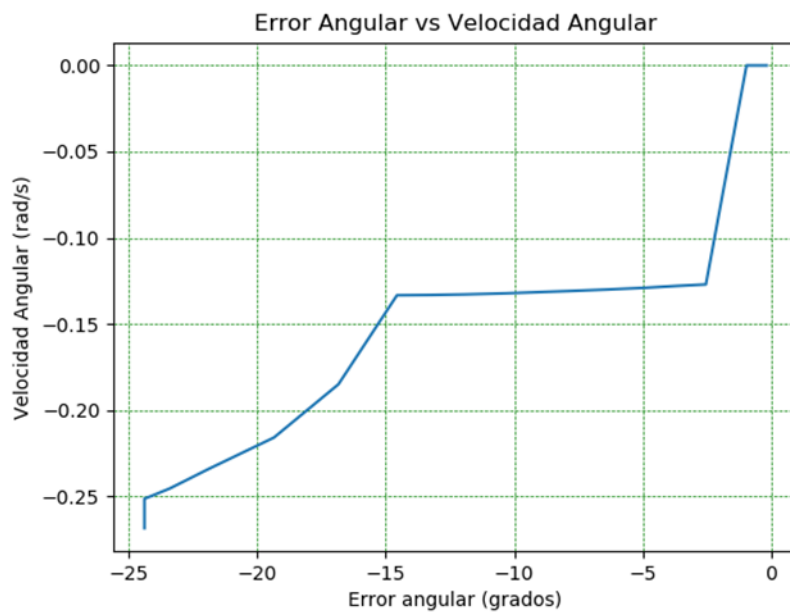


En la Figura 5-35 para la prueba de validación del cono derecho en el periodo de control 56, se evidencia una distancia de 1.36 m, una velocidad lineal de 0.17 m/s y una velocidad angular de -0.02 rad/s. Esto obedece a que en ese periodo de control para los demás conos de detección hay distancias considerables para que el robot realice un desplazamiento lineal con una velocidad angular muy baja. Por otro lado, en la figura 5-36 para la prueba de validación del cono central, en el periodo de control 9, se evidencia una distancia de 0.65 m, una velocidad lineal de 0.11 m/s y una velocidad angular de 1.43 rad/s y esto ocurre debido a que se encuentra un obstáculo frontal muy cerca de la plataforma robótica y el cono de detección con más distancia disponible es el derecho. Lo cual implica una velocidad lineal muy baja y un giro a la derecha con una velocidad angular positiva muy alta. Finalmente, en la figura 5-37 para la prueba de validación del cono izquierdo en el periodo de control 75, se evidencia una distancia de 2.81m, una velocidad lineal de 0.14 m/s y una velocidad angular de 0.75 rad/s, lo que en consecuencia se da a raíz de una distancia en el cono derecho de más de 3 metros aun teniendo una distancia similar en el cono izquierdo pero inferior al del cono derecho, en este caso el controlador prefiere apuntar al cono con la distancia más segura.

5.4.3 Validación de comportamiento orientación al destino

Para la prueba de comprobación del funcionamiento del controlador *fuzzy* para la orientación al destino, se adquieren datos del error angular y la velocidad angular durante el giro de la plataforma robótica para posicionarse en un ángulo específico. Para esta prueba se estableció como punto final las coordenadas (5,4) en unidades de metros y por medio de la ecuación 5.1 se calcula el error angular en todo momento para que la plataforma robótica se mantenga orientada con respecto al punto de interés. En la figura 5-38 se ilustra como varía la velocidad angular en función del error angular.

Figura 5-39: Prueba de validación de comportamiento de orientación al destino



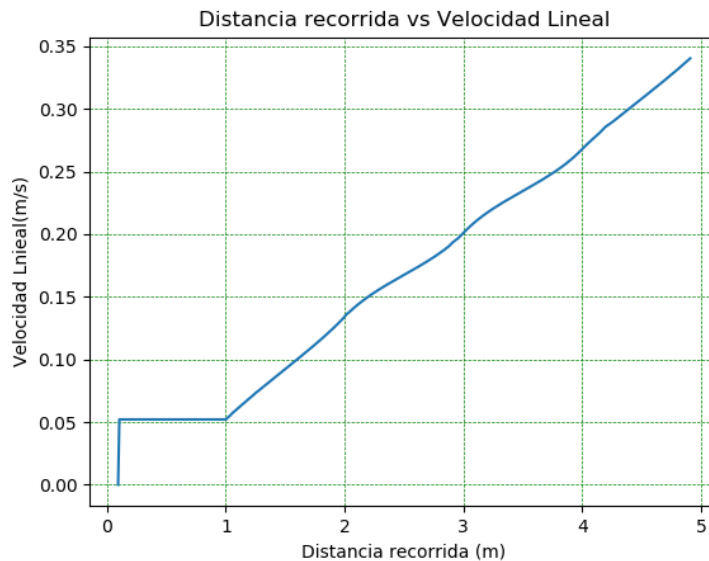
Al analizar la figura 5-39 se encuentra que, a medida que el error angular disminuye la velocidad angular disminuye de manera proporcional gracias a las funciones de pertenencia del controlador *fuzzy* de orientación al destino. la disminución de velocidad se lleva a cabo hasta que el error angular es menor a 2° la cual indicará el fin a este comportamiento como se ilustra en la figura 5-38.

5.4.4 Validación de comportamiento avanzar al destino

La prueba de validación de comportamiento avanzar al destino se adquieren datos de la distancia actual a la que se encuentra el robot con respecto al destino y de la velocidad lineal, ambas se ejecutan por el cálculo de la ecuación 5.3, donde se actualiza la distancia actual con respecto al destino.

Como se ilustra en la figura 5-39, a medida que disminuye la distancia con respecto al destino, la velocidad lineal tiende a disminuir gradualmente, gracias a las diferentes funciones de pertenencia previstas en dicho controlador *fuzzy*.

Figura 5-40: Prueba de validación de comportamiento avanzar al destino



5.4.5 Prueba de validación para brazo manipulador en la toma de objeto

Para la prueba de validación, se tomaron los datos de velocidad y torque de cada una de las articulaciones del robot en los periodos de control en lo que se tenía la tarea programada de tomar el objeto.

A partir de la figura 5-41, se encuentra que la articulación tres fue la que alcanzó mayor velocidad, aproximadamente 1 m/s, seguida por la articulación dos que alcanzó velocidades similares, la articulación cuatro alcanzó aproximadamente 0.6 m/s y la articulación uno se mantuvo en 0 m/s debido a que no se realizaron movimientos de

translación del brazo. Por otra parte, según la figura 5-42, se evidencia un mayor esfuerzo en la articulación 2 de 0.14Nm para mover todo el brazo el cual se vuelve negativo con el fin de mantener la posición del brazo en equilibrio estático, de igual manera hay un esfuerzo negativo en la articulación tres el cual se mantiene constante y en las articulaciones uno y cuatro el esfuerzo fue mínimo en todo momento.

Figura 5-41: Velocidades de articulaciones en brazo manipulador al tomar el objeto.

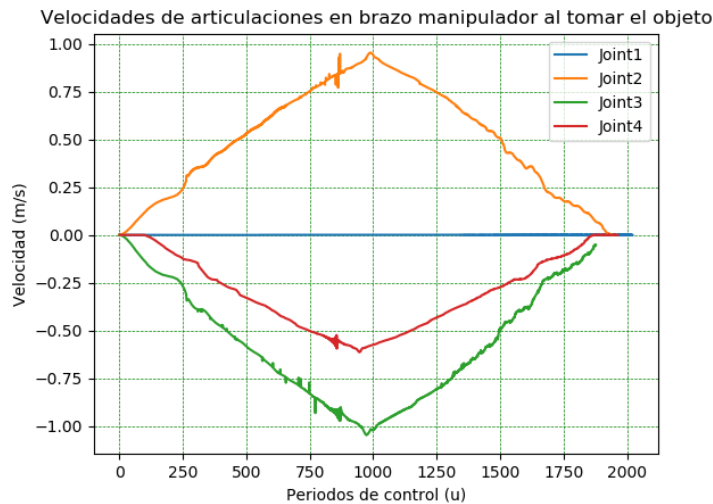
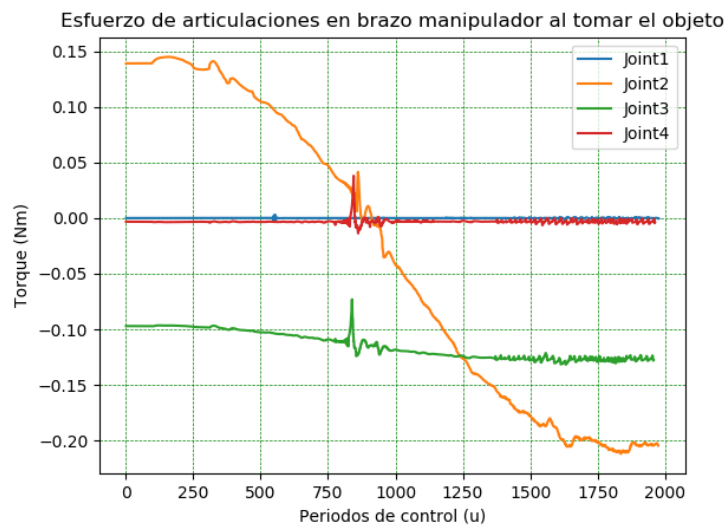


Figura 5-42: Esfuerzo de articulaciones en brazo manipulador al tomar el objeto.



5.4.6 Prueba de validación para brazo manipulador en la entrega de objeto

Para esta prueba de validación, se tomaron los datos de velocidad y torque de cada una de las articulaciones del robot, en los periodos de control en los que se tenía la tarea programada de entregar el objeto y teniendo en cuenta que el brazo manipulador se encuentra cargado.

A partir de la figura 5-43 se encuentra que la articulación dos fue la que alcanzó mayor velocidad aproximadamente 0.9 m/s, seguido por la articulación cuatro que alcanzó velocidades de 0.6m/s, la articulación tres alcanzó 0.5 y la articulación uno se mantuvo en 0 m/s debido a que no se realizaron movimientos de translación del brazo. Con respecto a la toma del objeto se evidencian velocidades diferentes y esto a raíz de que se busca posiciones diferentes con un mayor grado de inclinación de la articulación cuatro. Por otra parte, según la figura 5-44, se evidencia un mayor esfuerzo en la articulación dos por encima de 0.15 Nm para mover todo el brazo, articulación la cual toma valores negativos con el fin de mantener la posición del brazo en equilibrio estático. De igual manera hay un esfuerzo negativo en la articulación tres el cual se mantiene constante, en la articulación cuatro el esfuerzo fue mayor que en la toma del objeto y la articulación uno se mantuvo igual. Según lo evidenciado se encuentra un mayor esfuerzo por parte del brazo manipulador a la hora de completar la entrega del objeto, lo cual obedece a dos razones principales como lo es un ángulo mayor para para las articulaciones del brazo manipulador y el peso del objeto.

Figura 5-43: Velocidades de articulaciones en brazo manipulador al entregar el objeto

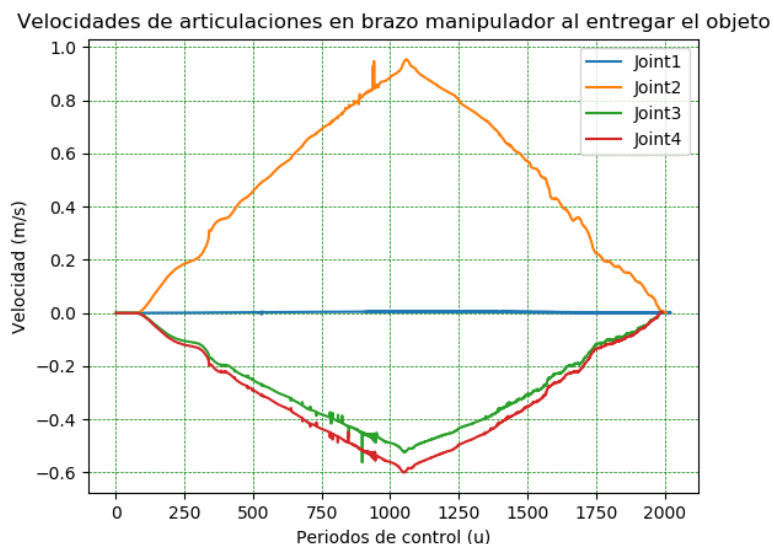
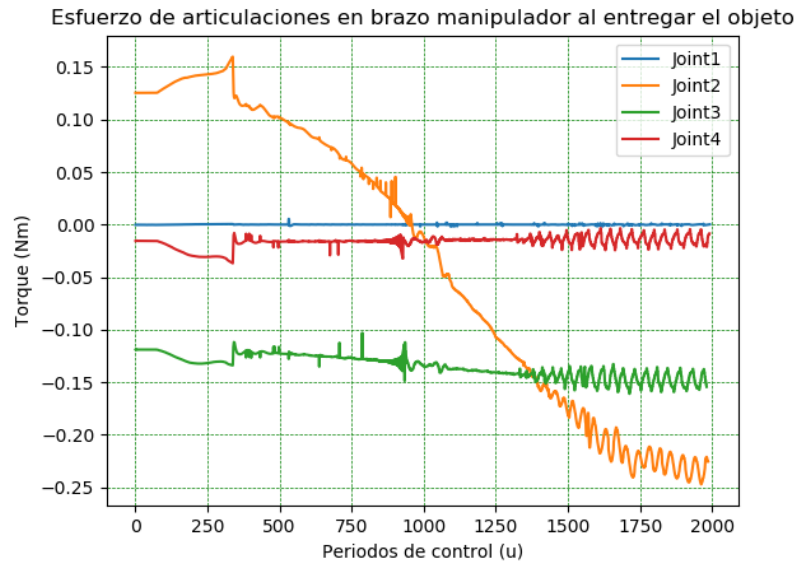


Figura 5-44: Esfuerzo de articulaciones en brazo manipulador al entregar el objeto



5.4.7 Prueba de validación de conmutación de los comportamientos

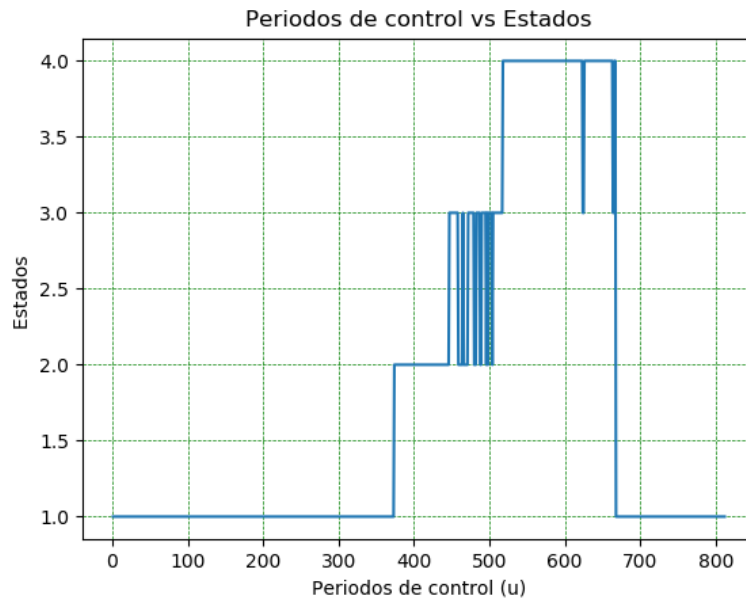
Una vez se realizadas las pruebas de validación de cada uno de los comportamientos, se evalúa en que periodos de control ocurre la conmutación de comportamientos. Por tanto, estos comportamientos fueron etiquetados con un número de la siguiente forma.

- 1-Aproximacion para la toma y descarga de objeto.
- 2-Evasión de obstáculos.
- 3-Orientación al destino.
- 4- Avanzar al destino.

A partir de lo ilustrado en la figura 5-45 se identifica que entre el periodo de control 0 y 390 se mantiene el comportamiento de aproximación para la toma del objeto. Luego entre el periodo de control 390 hasta 500 se da el comportamiento de evasión de obstáculos con una conmutación recurrente con el comportamiento de orientación al destino. Esto ocurre ya que a medida que el robot evade el obstáculo hasta obtener el umbral de distancia de conmutación en los conos de detección, esta conmuta al comportamiento orientarse al

destino, pero cuando lo hace identifica que no se ha superado el obstáculo y vuelva a realizar la evasión hasta superarlo completamente sin perder su destino objetivo. Entre el periodo de control que inicia en 510 y va hasta 660 la plataforma robótica avanza hacia el destino corrigiendo el error de orientación en dos ocasiones. Finalmente, entre el periodo 660 y 805 regresa al comportamiento de aproximación, en este caso para la entrega del objeto.

Figura 5-45: Validación de conmutación de los comportamientos



5.4.8 Prueba de validación de tiempos de respuesta

Una forma de validar globalmente la metodología es a partir de los tiempos de ejecución de la tarea completa desde que se asigna el pedido al robot a través de la interfaz de usuario hasta que el robot llega al final del recorrido entrega la botella y envía la notificación de que el pedido fue completado.

Por tanto, para cada uno de los destinos dispuestos a través de marcadores fiduciales se realizaron 10 pedidos, se tomaron los tiempos de ejecución y se obtuvieron valores como tiempo, promedio y desviación estándar. En la tabla 5-7, 5-8 y 5-9 se evidencian los valores obtenidos para cada uno de los destinos.

Tabla 5-7: Tiempos de ejecución para destino con marcador fiducial AR tag 7

Prueba	Tiempo (minutos)	Promedio (minutos)	Desviación estándar
1	6.00	5.925	0.08720984
2	5.95		
3	6.01		
4	5.85		
5	5.89		
6	6.05		
7	5.79		
8	5.91		
9	5.82		
10	5.98		

Tabla 5-8: Tiempos de ejecución para destino con marcador fiducial AR tag 9

Prueba	Tiempo (minutos)	Promedio (minutos)	Desviación estándar
1	5.00	4.99	0.06349978
2	5.04		
3	5.07		
4	4.98		
5	4.95		
6	4.90		
7	5.10		
8	5.02		
9	4.92		
10	5.01		

Tabla 5-9: Tiempos de ejecución para destino con marcador fiducial AR tag 11

Prueba	Tiempo (minutos)	Promedio (minutos)	Desviación estándar
1	5.93	5.884	0.06850791
2	5.89		
3	5.95		
4	5.98		
5	5.85		
6	5.80		
7	5.78		
8	5.82		
9	5.94		
10	5.90		

Con respecto a la tabla 5-7 y 5-9 que corresponde al AR tag 7 y 11, se encuentra un tiempo promedio de 5.925 minutos y 5.884 minutos, así como una desviación estándar de 0.08720984 y 0.06850791, lo cual indica que los datos se encuentran poco

dispersos, así como también hay una buena repetibilidad en tiempos de operación para los destinos en cuestión. Por otro lado, en la tabla 5-8 que corresponde al AR tag 9 se tiene un tiempo promedio de 4.99 minutos, aproximadamente 1 minuto menos con respecto a los AR tags 7 y 11. Esto obedece a que como está dispuesto el AR tag 9 en el entorno, le permite al robot alinearse de manera más fácil con este y por tanto invertir menos tiempo y esfuerzo, ya que a mayor distancia del marcador fiducial, los valores de distancia y orientación entregados por los marcadores fiduciales presentan variaciones considerables. Con respecto a la desviación estándar en el AR tag 9 fue de 0.06850791 lo cual indica de igual manera una buena repetibilidad, lo que permite tener tiempos estimados de operación y sobre esto realizar proyecciones en función de la capacidad de operación de los robots en campo.

5.4.9 Prueba de validación de entregas exitosas

Se realiza un número significativo de pruebas de validación del método teniendo en cuenta el criterio de cuantas entregas exitosas se hicieron y cuantas no. De un total de 20 pruebas de asignación de pedidos a diferentes destinos, 2 de ellas presentaron problemas al momento de la recolección o la entrega del objeto a transportar lo que obedece a un 90% de entregas exitosas y a un 10% de entregas fallidas.

Durante el desplazamiento del robot en entornos estrechos el robot manipulador en ocasiones colisionaba con el objeto a transportar lo cual genera problemas a la hora de tomar el objeto, esto se debe a la incertidumbre de la medida proporcionado por el marcador fiducial, lo que es usual en los entornos reales y a que solo se tiene un marcador fiducial como referencia para ubicar el robot en el área de carga y tomar el objeto a transportar.

Si bien el porcentaje de pruebas de validación exitosas es alto conviene identificar alternativas de mejora para aumentar el porcentaje de éxito, como lo sería el uso de sistemas de apoyo visuales adicionales en el área de carga para la correcta ubicación del brazo manipulador tanto para la toma como entrega del objeto.

6. Conclusiones y recomendaciones

6.1 Conclusiones

- El middleware ROS en su versión *Melodic* permitió el desarrollo del método propuesto desde todas sus perspectivas de la integración de la sensorica, el desarrollo de la arquitectura de control, la integración con aplicativos webs, la integración con simuladores y con *frameworks* para la integración de robótica manipuladora.
- El *framework* Moveit! permitió realizar la integración y manipulación de un brazo robótico OpenManipulator X de manera fácil y rápida por medio de sus métodos de planificación de posiciones para cada una de sus articulaciones, los cual resolvió muchos problemas a la hora de integrar un robot manipulador en el experimento de validación.
- La API de *rosbridge* por medio de su librería *roslibjs*, permitió realizar una conexión con el *middleware* ROS, para la lectura de tópicos involucrados en la aplicación web, por medio del lenguaje JavaScript. Logrando con esto una integración mayor en aplicaciones logísticas industriales.
- Ros Development Studio, es la herramienta de plataforma como servicio en la nube que permitió ahorrar tiempo en configurar el *middleware* ROS y las demás

herramientas involucradas, así como un ahorro en los recursos de hardware para el desarrollo de la aplicación logística.

- La herramienta de Azure DevOps permitió realizar el control de versiones respectivo durante el desarrollo de la aplicación robótica y deja el panorama abierto para la integración y despliegue continuo de software en los ambientes pre-productivos y productivos de manera automática.
- Por medio de los experimentos de validación realizado fue posible evidenciar que la metodología propuesta para el desarrollo de aplicaciones robóticas logísticas es adecuada para el desarrollo de diferentes aplicaciones de esta naturaleza, así como también brindar un panorama amplio a la hora de desarrollar futuras aplicaciones en robótica de servicio.
- Los modelos de computación en la nube para aplicaciones robóticas abren un panorama mayor en las aplicaciones robóticas y su caracterización permitió identificar el modelo más adecuado para el desarrollo de la metodología propuesta.
- La caracterización de los elementos de la industria 4.0 aplicables a la robótica de servicio permitió identificar de manera clara los diferentes tipos de sensores y herramientas de desarrollo para el desarrollo de la metodología propuesta.
- Con el diseño e implementación de la arquitectura de control reactiva por comportamientos conformada por la aproximación para la toma y descarga de objetos, avanzar al destino, orientación al destino y evasión de obstáculos, se logró la navegación autónoma de la plataforma robótica a partir de la información suministrada por la sensorica propioceptiva y exteroceptiva.
- El proceso de diseño e implementación de controladores *fuzzy* para las velocidades de los motores de la plataforma robótica permitió que este pudiera realizar los cuatro comportamientos, los cuales cumplieron satisfactoriamente el requerimiento de navegación autónoma, para la arquitectura reactiva propuesta en este trabajo.

- Las pruebas de validación permitieron corroborar los tres comportamientos, para hacer mejoras considerables en cada uno de los controladores *fuzzy* de orientación, desplazamiento lineal y evasión.
- El controlador *fuzzy* más robusto para el control de la velocidad lineal y angular de los motores de la plataforma robótica, es el de evasión de obstáculos el cual, al ser diseñado bajo 29 reglas, pudo contener la totalidad del universo del discurso, aplicando su respuesta en pruebas de navegación autónoma.
- Los marcadores fiduciales son elementos de gran utilidad a la hora de realizar aplicaciones con robots de servicios ya que sus algoritmos de decodificación son precisos y confiables de igual manera permite flexibilizar las aplicaciones robóticas en los casos que se requieran cambios en el entorno gracias a la gran cantidad de identificadores que se pueden emplear si variar la arquitectura de control.
- El simulador Gazebo permitió realizar las diferentes simulaciones teniendo en cuenta las diferentes físicas de la plataforma robótica, así como del brazo manipulador lo que permitió realizar un acercamiento cercano con el mundo real y que la hora de su implementación lo cambios sean mínimos.

6.2 Recomendaciones

- En los experimentos de validación Integrar una cámara en el entorno que permita proporcionar al robot las coordenadas del elemento a tomar haciendo uso de marcadores fiduciales.
- Incorporar en la metodología el manejo de múltiples robots tanto para el control de navegación como para la coordinación web en asignación de tareas logísticas.

-
- En los experimentos de validación incorporar una cámara que cubra el entorno de trabajo y permita incorporar modelos de planificación de rutas fusionados con una arquitectura reactiva.
 - En los experimentos de validación emplear otro tipo de marcadores fiduciales para la orientación del robot en el entorno.
 - Emplear otro tipo de cámaras incorporadas en el robot que permitan obtener un mayor alcance y reducir el error inducido a la hora de decodificar los marcadores fiduciales.
 - Implementar técnicas de inteligencia artificial que permitan generar más suavidad a la hora de realizar el cambio de comportamientos.
 - Emplear diferentes plataformas como servicio en la nube que permitan el desarrollo de aplicaciones robóticas que permitan un despliegue a entornos reales e integración continua en el ciclo de vida del software haciendo uso de metodologías ágiles como DevOps.

Bibliografía

- Baalbaki, I., & Xie, X. (2009). A decision framework for operation management of reconfigurable mobile service robots in hospitals. *IFAC Proceedings Volumes*, 151-156.
- Deuseab, J. R. (2014). Methodology for the Planning and Implementation of Service Robotics in Industrial WorN Processes. *Conference on Assembly Technologies and System*, 41-46.
- Dourado, C., da Silvaa, S., da Nóbrega, R., Barros, A., Sangaiah, A., Rebouças, P., . . . Victor. (2019). A new approach for mobile robot localization based on an online IoT system. *Future Generation Computer Systems*, 859-881.
- Dröder, K., lBobka, P., Germann, T., Gabriel, F., & Dietrich, F. (2018). A Machine Learning-Enhanced Digital Twin Approach for Human-Robot-Collaboration. *Procedia CIRP*, 187-192.
- Erboz, G. (2017). How To Define Industry 4.0: Main Pillars Of Industry 4.0. *7th International Conference on Management* (pág. 9). Nitra: ICoM.
- Farinelli, A., Boscolo, N., Zanutto, E., & Pagello, E. (2017). Advanced approaches for multi-robot coordination in logistic scenarios. *Robotics and Autonomous Systems*, 34-44.
- Giusti Bravo, F. (13 de Septiembre de 2016). *beetrack*. Recuperado el 20 de Diciembre de 2019, de Robótica Industrial: una mirada actual y lo que viene: <https://www.beetrack.com/es/blog/robotica-industrial-una-mirada-actual-y-lo-que-viene>
- Hehua, Y., Qingsong, H., Yingying, W., Wenguo, W., & Muhammad, I. (2017). Cloud robotics in Smart Manufacturing Environments: Challenges and countermeasures. *Computers & Electrical Engineering*, 56-62.
- Jin, W., IIKwag, S., & DaeKo, Y. (2020). Optimal capacity and operation design of a robot logistics system for the hotel industry. *Tourism Management*, 103-971.

- Kousi, N., Koukas, S., Michalos, G., Makris, S., & Chryssolouris, G. (2016). Service Oriented Architecture for Dynamic Scheduling of Mobile Robots for Material Supply. *Procedia CIRP*, 18-22.
- Kuhner, D., Fiederer, L., & Aldingerad, J. (2019). A service assistant combining autonomous robotics, flexible goal formulation, and deep-learning-based brain-computer interfacing. *Robotics and Autonomous Systems*, 98-113.
- Li, H., & Savkin, A. (2018). An algorithm for safe navigation of mobile robots by a sensor network in dynamic cluttered industrial environments. *Robotics and Computer-Integrated Manufacturing*, 65-82.
- Madrid, R. (19 de Octubre de 2019). *Redaccion Medica*. Obtenido de Robots: los nuevos ayudantes hospitalarios: <https://www.redaccionmedica.com/noticia/los-robots-los-nuevos-ayudantes-hospitalarios-88603>
- Ozaki, K., Kagaya, H., & SatoshiHirano. (2013). Preliminary Trial of Postural Strategy Training Using a Personal Transport Assistance Robot for Patients With Central Nervous System Disorder. *Archives of Physical Medicine and Rehabilitation*, 59-66.
- Rivera, Z., Simone, M., & Guida, D. (2019). Unmanned Ground Vehicle Modelling in Gazebo/ROS-Based Environments. *Machines*, 21.
- Schuessler, Z., Schuessler, H., & Strohaber, J. (2018). Robotic-assisted hysterectomy in a community hospital after seven years of experience. *Laparoscopic, Endoscopic and Robotic Surgery*, 42-45.
- Velagic, J., & Balta, H. (2010). Design and Development of Control and Communication Systems for Complex Mobile Robot and Manipulator Structure. *IFAC Proceedings Volumes*, 307-312.
- Xiaa, C., Zhanga, Y., Wang, L., Coleman, S., & Liua, Y. (2018). Microservice-based cloud robotics system for intelligent space. *Robotics and Autonomous Systems*, 139-150.
- Yan, H., Hua, Q., Wang, Y., Wei, W., & Imrand, M. (2017). Cloud robotics in Smart Manufacturing Environments: Challenges and countermeasures. *Computers & Electrical Engineering*, 56-65.
- Yavasa, V., Deniz, Y., & Ozenb, Y. (2020). Logistics centers in the new industrial era: A proposed framework for logistics center 4.0. *Transportation Research Part E: Logistics and Transportation Review*, 101-864.

- Zhu, D. (2018). IOT and big data based cooperative logistical delivery scheduling method and cloud robot system. *Future Generation Computer Systems*, 709-715.
- Bhavsar, P., Patel, S., & Sobh, T. (2019). Hybrid Robot-as-a-Service (RaaS) Platform (Using MQTT and CoAP). *International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (pág. 16). Atlanta: IEEE.
- D. Bogataj, D. B. (2019). The ageing workforce challenge: investments in collaborative robots or contribution to pension schemes, from the multi-echelon perspective. *International Journal of Production Economics*, 106.
- Gonzales, p., Calvo, I., Etxberria, I., & Zulueta, E. (2015). Hacia un framework basado en ROS para la implementación de Sistemas. *Jornadas de Automática*, (págs. 1050-1057). Bilbao.
- J. Guiochet, M. M. (2017). Safety-critical advanced robots: a survey. *Rob Auton Syst*, 52.
- OMRON. (2019). *OMRON Automatizacion Industrial*. Obtenido de industrial omron: <https://industrial.omron.es/>
- Pyo, Y., Nakashima, K., & Kuwahata, S. (2015). Service robot system with an informationally structured environment. *Robotics and autonomous System*, 148-165.
- Sadek, L., & Ayad, S. (2015). TOWARDS A NEW CLOUD ROBOTICS APPROACH. *Proceedings of the 8th International Symposium on Mechatronics and its Applications (ISMA12)* (pág. 5). Sharjah: UAE.
- Sakamoto, J., Kiyoyama, K., Matsumoto, K., & Pyo, Y. (2018). Development of ROS-TMS 5.0 for informationally structured environment. *ROBOMETCH JOURNAL*.
- Schumann, H. (25 de Junio de 2018). *ZIVID*. Obtenido de <https://blog.zivid.com>
- Toris, R., Kent, D., & Chernova, S. (2014). Toris, R., Kent, D., & Chernova, S. (2014). The robot management system: a framework for conducting human-robot interaction studies through crowdsourcing. HRI 2014. *HRI*, (pág. 16).
- Ullah, I. (2019). Modeling and simulation of complex manufacturing phenomena using sensor signals from the perspective of Industry 4.0. En C. Chen, & T. Hartmann, *Advanced Engineering Informatics* (pág. 358). ScienceDirect.
- Hu, G., Peng, W., & Wen, Y. (2012). Cloud Robotics:Architecture, Challenges and Applications. *IEEE Network* , 21-28.

- Koubaa, A. (2014). A Service-Oriented Architecture for Virtualizing. *International Conference on Architecture of Computing Systems*, 196-208.
- Mbunge, E., Fashoto, S., & Dlamini, S. (2020). Modelling a Gossip-based Protocol for Enhanced Dynamic Routing. *asian journal of information technology*, 203-206.
- Yoon, J., & Park, H. (2015). A Cloud-based Integrated Development Environment for robot software Development. *Journal of Institute of Control, Robotics and Systems*, 173-178.
- A General Framework for Temporal Calibration of Multiple Proprioceptive and Exteroceptive Sensors. (2014). En J. Kelly, & G. S. Sukhatme, *Experimental Robotics* (págs. 195-209). Springer.
- A. Battle, J., & Barjau, A. (2009). Holonomy in mobile robots. *Robotics and Autonomous Systems* .
- Acosta, G., Gallardo, j., & Perez, R. (2016). Reactive control architecture for autonomous mobile robot navigation. *Revista Chilena de Ingeniería*.
- Almasri, M., Elleithy, K., & Alajlan, A. (2016). Sensor Fusion Based Model for Collision Free Mobile Robot Navigation. *Sensors*, 16-24.
- Andreev, A., & Peregudova, O. (2020). On global trajectory tracking control of robot manipulators in cylindrical phase space. *International Journal of Control*, 3003-3015.
- B.S.K.K, I., & Ahmed, M. (2014). Modelling and Control of SCARA Manipulator. *Procedia Computer Science*, 106-113.
- Bambino, i. (2008). *Una Introducción a los Robots Móviles* .
- Barrientos Sotelo, V. R., García Sánchez, J. R., & Silva Ortigoza, R. (2007). Robots Móviles: Evolución y Estado del Arte. *Polibits*, 12-17.
- Barrientos, A., Peñin, L. F., Balaguer, C., & Aracil, R. (1997). *Fundamentos de Robotica*. Madrid: McGraw-Hill.
- Cardoso, E., Fernandez, A., Marrero-Osorio, S. A., & Guardado, P. (2017). Modelos cinemático y dinámico de un robot de cuatro grados de libertad. *Ingeniería Electrónica, Automática y Comunicaciones*, 56-75.

- Crick, C., Jay, G., Osentoski, S., Pitzer, B., & Jenkins, O. C. (2017). Rosbridge: ROS for Non-ROS Users. En H. I. Christensen, & O. Khatib, *Robotics Research* (págs. 493-504). Switzerland: Springer International Publishing.
- Dikra, E., Badreddine, A., Larbi, E., & Jalal, E. (2019). Optimal Trajectory Planning for Spherical Robot Using Evolutionary Algorithms. *Procedia Manufacturing*, 960-968.
- Fiala, M. (2005). ARTag, a fiducial marker system using digital techniques. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (págs. 590-596). IEEE.
- Gea, M., & Gutierrez, I. (2002). *La interacción persona-ordenador*. Granada: Universidad de Granada.
- Goergen, R., Maboni, M., & Gioppo de Souza, M. (2018). Design of an Experimental Workbench for Force Control Tests with Pneumatic Actuators. *Proposal of an Autonomous System for Inspection of Structures* (págs. 52-57). São Paulo: ABIMAQ.
- Hrbáček, J., Ripel, T., & Krejsa, J. (2010). Ackermann mobile robot chassis with independent rear wheel drives. *Power Electronics and Motion Control Conference (EPE/PEMC), 2010 14th International*.
- Kabanov, A., & Balabanov, A. (2018). The modeling of an anthropomorphic robot arm. *International Conference on Modern Trends in Manufacturing Technologies and Equipment (ICMTMTE 2018)*.
- Madow, A., Martínez, J. L., Morales, J., & Blanco, J. L. (2007). Experimental kinematics for wheeled skid-steer mobile robots. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International*, (págs. 1222-1227).
- McTaggart, M. & -B., Razjigaev, A., Rowntree, T., Vijay, K., Wade-McCue, S., & Tow, A. (2017). Mechanical Design of a Cartesian Manipulator for Warehouse Pick and Place. *arxiv*.
- Mohd Anuar, K., & Sapiee, M. (2018). Synchronous Mobile Robots Formation Control. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 16.
- Navegación en Robots. (2011). En Y. Torres, & A. Ines, *Planificación de trayectorias en robots móviles* (págs. 21-52).
- Nielsen, I., Dang, Q.-V., Bocewicz, G., & Banaszak, Z. (2017). A methodology for Implementation of Mobile Robot in Adaptive Manufacturing Environments. *Journal of Intelligent Manufacturing*, (págs. 1171 - 1188).

- profander, S. (2014). *Implementation and Evaluation of multimodal input/output channels for task-based industrial robot programming*.
- Ramirez Arias, J. L., Fonseca, R., & Astrid. (2012). Mathematical modeling of the direct. *Revista Ingeniería Solidaria*, vol. 8, 46-52.
- Riaño, C., Peña, C., & Sánchez, H. (2018). Aplicación de técnicas de desenvolvimiento de producto para el desarrollo de un robot antropomórfico. *Revista UIS Ingenierías*, 21-34.
- Rojas, D. B., Lorente, L. E., & de Quirós, C. B. (2002). *Planificación local basada en sensores para un manipulador móvil en tareas de colaboración con humanos*. Madrid: Universidad Carlos III de Madrid.
- Rossmann, J., Ruf, H., & Schlette, C. (2009). Model-Based Programming “by Demonstration”– Fast Setup of Robot Systems (ProDemo). En T. Kroger, & F. Wahl, *Advances in Robotics Research: Theory, Implementation, Application* (págs. 159-168). Springer.
- Rubio, f., Valero, F., & Llopis, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advance Robotic Systems*.
- Ruiz, J. (2016). *Probabilistic Techniques in Semantic*. malaga: Universidad de malaga.
- Salter, T., Michaud, F., Létourneau, D., Lee, D., & Werry, I. (2007). Using proprioceptive sensors for categorizing human-robot interactions. *2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Arlington: IEEE.
- Sanders, D. (2008). Environmental sensors and networks of sensors. *Sensor Review*.
- SungBu, K., I.O.Lee, D.I.Cho, & JangMyung, L. (2006). DYNAMIC LOCALIZATION OF A MOBILE ROBOT WITH ACTIVE BEACON SENSORS. *IFAC Proceedings Volumes*, 921-925.
- Terakawa, T., Komori, M., & Matsuda, K. (2019). Motion Analysis of an Omnidirectional Mobile Robot with Wheels Connected by Passive Sliding Joints. *Advances in Mechanism and Machine Science*, 2279-2288.
- Vinh, Q., Ewa, I., Bøgh, S., & Bocewicz, G. (2013). Modelling and Scheduling Autonomous Mobile Robot for a Real-World Industrial Application. *IFAC Conference on Manufacturing Modelling, Management and Control*. Russia.

Zenatti, F., Fontanelli, D., Palopoli, L., Macii, D., & Nazemzadeh, P. (2016). Optimal placement of passive sensors for robot localization. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.