ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO Y SIMULACIÓN DE UN SISTEMA DE CLASIFICACIÓN DE OBJETOS EMPLEANDO EL ROBOT INDUSTRIAL BAXTER MEDIANTE EL USO DEL ENTORNO ROS-GAZEBO.

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

CATAGÑA CATAGNIA LUIS PATRICIO

DIRECTOR: ING. PATRICIO JAVIER CRUZ DÁVALOS, PhD.

Quito, septiembre 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Catagña Catagnia Luis Patricio, bajo mi supervisión.

ING. PATRICIO JAVIER CRUZ DÁVALOS, Ph.D.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Catagña Catagnia Luis Patricio, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

Catagña Catagnia Luis Patricio

DEDICATORIA

El presente trabajo de titulación se lo dedico principalmente a mis padres, sin su apoyo y consejo en los momentos más difíciles, no habría culminado este proyecto.

Se lo dedico a mis compañeros y futuros colegas ingenieros por su consejo, paciencia y apoyo dentro y fuera de la universidad.

AGRADECIMIENTO

Con Dios está la sabiduría y el poder; suyo es el consejo y la inteligencia, principalmente agradezco a Dios por darme la sabiduría y entendimiento, gracias a Él pude cumplir una de mis metas más anheladas.

Gracias, a mis padres por sus enseñanzas y comprensión en todo tiempo.

Finalmente a mis amigos y futuros colegas que compartieron apoyo y conocimiento, gracias.

ÍNDICE DE CONTENIDO

RESUMEN

En la actualidad el uso de robots manipuladores industriales ha incrementado, lo que da paso a nuevas técnicas y métodos de programación, como es el caso para el robot Baxter de la empresa Rethink Robotics. En el presente trabajo escrito, inicialmente, se desarrolla un breve estudio de la documentación referente a este manipulador industrial, centrándose en sus características tanto de software como de hardware. Además, se presenta su modelo cinemático directo, mismo que se encuentra accesible al público a través del repositorio oficial GitHub de su fabricante. Luego se desarrolla un estudio del entorno ROS (Robot Operating System), en base al cual es posible controlar al robot Baxter por medio de mensajes y la ejecución de nodos. También se estudia la plataforma de simulación Gazebo, la cual ayuda a simular un entorno virtual con características dinámicas con el objetivo de simular un ambiente real y en este colocar al robot Baxter dentro de una escena industrial especifica. La aplicación desarrollada, hace uso tanto del entorno ROS como del simulador Gazebo, para crear un escenario en el cual el robot Baxter pueda realizar la clasificación de objetos mediante visión artificial empleando la cámara principal con la cuenta este robot manipulador. Los objetos a clasificar son esferas de color verde y cubos rectangulares de color rojo y azul. Para llevar cada uno de estos objetos de una posición a otra se hace uso del control de movimientos de los brazos del robot Baxter. Finalmente se desarrollarán pruebas que demuestran un funcionamiento efectivo en la escena de clasificación creada, estas determinan la efectividad de clasificación tanto de color como de forma, así como tiempos de ejecución de la tarea y el adecuado control de movimiento de los dos brazos con los que cuenta el robot.

PALABRAS CLAVE: Baxter, ROS, Gazebo, Clasificación de Objetos, Visión Artificial, Control de Movimiento.

ABSTRACT

Nowadays, the use of industrial manipulator robots has been increasing, which gives way to new techniques and programming methods, as is the case for the Baxter robot of the company Rethink Robotics. In this paper, initially, a brief study of the documentation concerning this industrial manipulator is developed, focusing on its software and hardware features. In addition, its direct kinematic model is presented, which is accessible to the public through the official GitHub repository of its manufacturer. Then, a study of the ROS (Robot Operating System) environment is developed, based on which it is possible to control the Baxter robot by means of messages and the execution of nodes. The Gazebo simulation platform is also studied, which helps to simulate a virtual environment with dynamic characteristics in order to simulate a real environment and place the Baxter robot in a specific industrial scene. The developed application makes use of both the ROS environment and the Gazebo simulator to create a scenario in which the Baxter robot can classify objects by means of artificial vision using the main camera of this manipulator robot. The objects to be classified are real spheres and rectangular red and blue cubes. To take each of these objects from one position to another, the movement control of the arms of the Baxter robot is used. Finally, tests will be developed to demonstrate effective operation in the created classification scene. These tests determine the effectiveness of classification of both color and shape, as well as execution times of the task and the proper control of movement of the two arms with which the robot has.

KEYWORDS: Baxter, ROS, Gazebo, Object Classification, Artificial Vision, Motion Control.

1. INTRODUCCIÓN

Como punto de partida de este trabajo de titulación se toma a la robótica como una de las herramientas que aportan a los diferentes sistemas un alto nivel de competitividad y productividad, en la actualidad se han desarrollado sistemas automáticos para la ejecución y manipulación de objetos en procesos industriales, uno de ellos es la utilización de sistemas robóticos. El uso más frecuente de este tipo de sistemas se da en países con altos grados de desarrollo tecnológico [39].

Los sistemas robóticos tienen como finalidad complementar y, en ocasiones, sustituir las funciones del ser humano en procesos o tareas en las cuales el grado de peligrosidad es alto e incluso cuando la tarea es demasiada tediosa. Esto provoca que el uso de estos tipos de sistemas sea masivo en varios sectores y varias aplicaciones, ya que posiblemente el mayor beneficio que otorga su empleo es mejorar la calidad de vida del ser humano, por ejemplo, mediante la reducción de horas de trabajo y de riesgos laborales [40].

Las innovaciones tecnológicas avanzan a pasos agigantados y el desarrollo de sistemas robóticos ayudan al control y operación de la producción en ambientes industriales. Un robot manipulador está dotado de extremidades que cumplen diferentes operaciones diversas por lo que están destinados a sustituir la actividad física del hombre en tareas repetitivas, desagradables o peligrosas. El RIA (Robot Institute of America) define a un robot industrial como ‘‘Un manipulador reprogramable que está diseñado para mover materiales, partes, herramientas o dispositivos a través del movimiento de sus extremidades para realizar una variedad de labores’’ [41].

En este sentido, Baxter es una de las soluciones tecnológicas que ha desarrollado la compañía Rethink Robotics con la finalidad de aplicar la robótica en ambientes no solo industriales sino también en áreas de educación e investigación. Este robot posee dos brazos robóticos, (cada extremidad posee 7 grados de libertad), y cuenta con sensores de proximidad al final de cada brazo para evitar colisiones. Además, posee una pantalla y una cámara que permite realizar procesamiento de imágenes [3].

Por otro lado, el mejoramiento continuo de sistemas robóticos ha impulsado a los creadores de software al desarrollo de sistemas con los que la comunicación entre Hombre-Máquina se desarrolla de manera fluida y eficiente. Justamente, el uso del entorno ROS (Robot Operating System) para sistemas robóticos ha facilitado la operación, programación y comunicación entre el ser humano y un robot [42]. ROS, más que un entorno, es una herramienta de código abierto la cual impulsa el futuro de la robótica en el sector industrial gracias a su flexibilidad y facilidad de uso. Esta herramienta ha sido utilizada por grandes

corporaciones como es ABB, iRobot y OTTO MOTORS, las cuales son empresas reconocidas a nivel mundial. El trabajo de esta herramienta está basado en procesar datos mediante nodos, estos nodos son los encargados de recibir, publicar y multiplexar datos de sensores, control, planificación, actuadores y otros mensajes. ROS al ser de código abierto permite y facilita la interconexión de nodos, como, por ejemplo, programados en Python o C++, permitiendo el desarrollo de sistemas robóticos [43].

Una de las plataformas de simulación con las que trabaja ROS es GAZEBO, la cual permite implementar ambientes robóticos en entornos 3D. GAZEBO es un software libre que permite simulaciones realistas, es decir los robots pueden interactuar con el mundo virtual y viceversa, esto da la posibilidad de que al robot desarrollado en este entorno le afecten características físicas como la gravedad y la colisión con objetos [44]. Este simulador adicionalmente da la posibilidad de usar sensores y otros objetos con los que puede interaccionar una plataforma robótica.

En este proyecto se propone realizar la comunicación entre el entorno ROS y el robot Baxter virtual disponible en GAZEBO, con el objetivo de simular el funcionamiento de este robot manipulador para una tarea generalmente requerida industrialmente como es el caso de la clasificación de objetos. Este sistema industrial será visualizado mediante la plataforma de simulación GAZEBO, de esta manera se evidenciará un nuevo entorno de programación que se basa en la ejecución o lanzamiento de nodos.

En este trabajo de titulación se plantea desarrollar un ambiente simulado en donde se cuente con un robot Baxter virtual con el cual se desarrollará la tarea de clasificación de objetos. Dos bandas transportadoras dispensarán los objetos mismos que serán colocados en una caja de clasificación. En particular, el robot Baxter tomará uno de los objetos con una extremidad (la que este más cercana) y lo moverá hasta un punto en el cual pueda realizar la clasificación del objeto por medio de su cámara, la segunda extremidad tomará este objeto y lo clasificará acorde a su color y forma, de esta forma el tiempo de ejecución en la clasificación disminuirá de forma significativa, además, el movimiento del robot Baxter se restringe en el giro de toda la plataforma que lo compone. Por lo que, el presente trabajo de titulación tendrá como producto final demostrable la simulación en el software GAZEBO de un sistema industrial de clasificación de objetos usando un robot manipulador de 2 extremidades con 7 grados de libertad cada una.

En la actualidad Ecuador se encuentra en vías de desarrollo, por lo tanto, al finalizar este trabajo de titulación se pretende incentivar el aprendizaje de la herramienta ROS para sistemas robóticos. Además, este entorno permite contar con una herramienta que

ayudaría al estudio y planteo de controladores para plataformas robóticas complejas a pesar de no contar con el robot físico. La simulación del entorno permitirá determinar si implementar un sistema robótico es factible para cumplir satisfactoriamente la tarea industrial.

1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es:

- Diseño y simulación de un sistema de clasificación de objetos empleando el robot industrial Baxter mediante el uso del entorno ROS-GAZEBO

Los objetivos específicos del Proyecto Técnico son:

- Realizar una revisión bibliográfica de las características, funcionamiento y modelo del robot Baxter, así como el uso del entorno de programación ROS-GAZEBO.

- Desarrollar la escena de la tarea de clasificación de objetos empleando al robot Baxter mediante el software GAZEBO.

- Implementar el algoritmo para la clasificación de los objetos en base a la escena creada.

- Implementar un algoritmo de control de movimiento de los brazos del robot Baxter para la manipulación de los objetos a ser clasificados.

- Verificar el funcionamiento del sistema mediante la realización de pruebas y el análisis de sus resultados.

1.2. ALCANCE

- Se realiza un estudio bibliográfico sobre el entorno ROS (Robot Operating System) y GAZEBO, haciendo referencia a sus conceptos básicos, librerías, ambientes de programación y tipo de ejecución.

- Se realiza un estudio bibliográfico sobre el robot Baxter, además se estudiarán sus características, funcionalidades, sensores y parámetros mecánicos.

- Se estudia y presenta el modelo de cinemática directa del robot Baxter dado por la empresa Rethink Robotics, para conocer cómo se puede realizar el control de posición de sus extremidades.

- Se desarrolla en GAZEBO la escena del sistema industrial donde se colocará al robot Baxter, así como de los objetos a clasificar los cuales son rectangulares de

color rojo y azul con un tamaño de 30 cm de largo, 2 cm de ancho, 2 cm de espesor y otro objeto esférico de color verde el cual tiene como radio 1.5 cm.

- Se obtienen datos de los sensores del robot Baxter virtual y se realiza movimientos de cada uno de sus actuadores (articulaciones), a fin de conocer la forma de interaccionar con el robot Baxter a través de ROS.
- Se realiza la programación necesaria en lenguaje de código abierto para la clasificación de objetos y el respectivo algoritmo para el control de posición de las extremidades del robot Baxter que permita realizar la toma y colocación de objetos.
- Se desarrollan pruebas de efectividad para la clasificación de objetos, así como del posicionamiento de las articulaciones basadas en un set point deseado.
- Se desarrollan pruebas de funcionamiento del sistema en conjunto basados en la toma de un objeto, clasificación del mismo y finalmente su colocación en su respectiva posición final, en base a estas se analizarán los resultados obtenidos enfocados en conocer que tan rápido se realiza la tarea, inicialmente se usará únicamente objetos rectangulares y posteriormente se incorporará a las pruebas el objeto esférico.

1.3. MARCO TEÓRICO

1.3.1. ROBÓTICA INDUSTRIAL

El campo de la robótica es relativamente nuevo, ya que busca cruzar las fronteras de la ingeniería tradicional; su estudio aborda el tema central de la investigación y desarrollo de sistemas mecánicos denominados robots. Estos sistemas poseen un alto grado de complejidad, por lo cual el conocimiento de electrónica, mecánica, ciencias de computación y técnicas de control son fundamentales tanto para el modelamiento como el control del mismo [1].

La robótica industrial es uno de los resultados de la cuarta revolución industrial, en la cual se impulsó el desarrollo tecnológico. La alta demanda de producción fue la causante de la introducción de plataformas robóticas en la industria; en particular, la incorporación de robots manipuladores en una cadena de producción para efectuar tareas repetitivas o peligrosas para el ser humano ayudaron a incrementar la eficiencia y precisión del producto final, además, el riesgo laboral se redujo significativamente.

Según la RIA (Robot Institite of America) se define al robot industrial como *“Un manipulador reprogramable multifuncional diseñado para mover materiales, piezas, herramientas o*

artefactos especiales, mediante movimientos variables programados, para la ejecución de tareas potencialmente muy diversas." [1]

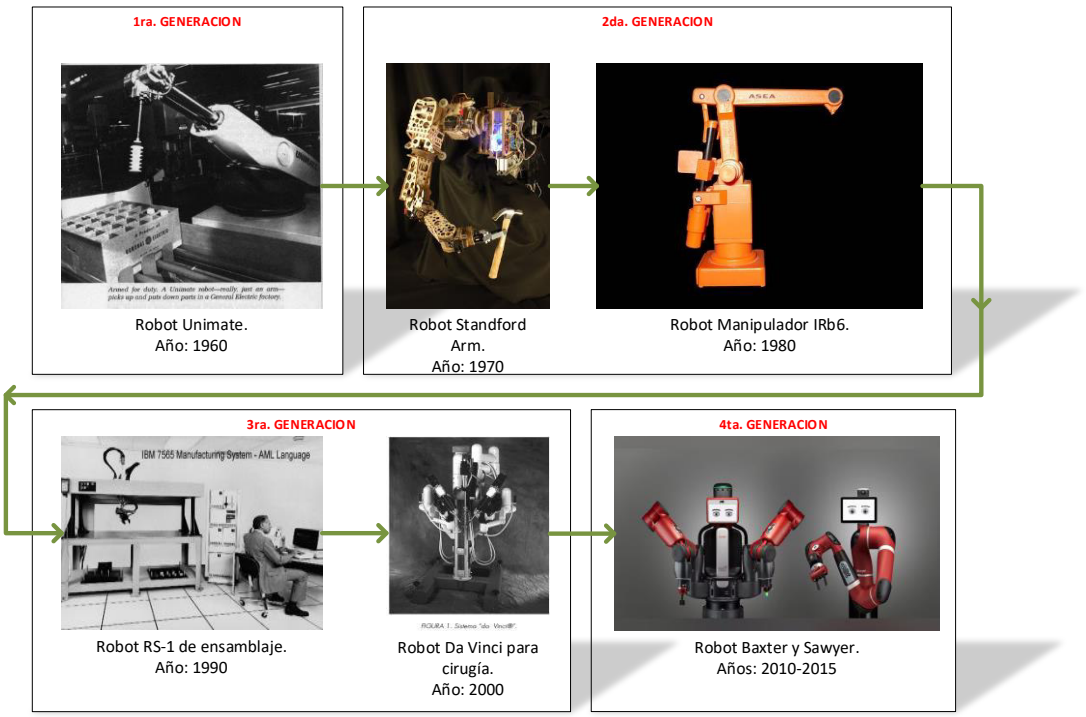


Figura 1.1 Evolución de la Robótica Industrial [3].

Los robots industriales han evolucionado con el pasar del tiempo como se observa en la Figura 1.1, estos sistemas inicialmente fueron controlados con un tipo de programación preestablecido, la cual era almacenada en grandes tambores magnéticos. Además, tenían un tamaño excesivo, eran sumamente pesados y sus actuadores eran hidráulicos. La siguiente generación de robots, a pesar de tener una programación preestablecida, cambio sus tipos de actuadores siendo principalmente electromagnéticos. La tercera generación de robots, utilizó un nuevo ambiente de programación, el cual permite la programación por el usuario dependiendo cual sea la aplicación. Este paso dio como origen al avance en la investigación de los diferentes sistemas de control y algoritmos para la utilización de inteligencia artificial dando paso a la cuarta generación de robot industriales en la que se encuentra al robot Baxter [3].

En la actualidad el avance de la investigación y desarrollo de nuevas tecnologías han dado paso al surgimiento de nuevos sistemas o ambientes de programación. Uno de estos es la programación mediante la interacción directa entre el operador y el robot industrial, este tipo de programación se basa en indicarle que tipo de movimiento debe realizar y el robot deberá memorizar dichos movimientos automáticamente. Esta programación se logra

gracias a la incorporación de sensores y cámaras que interactúan con el usuario u operador [3].

1.3.2. ROBOT MANIPULADOR BAXTER

El robot manipulador Baxter es uno de los robots industriales desarrollados por la compañía Rethink Robotics en el año 2012, año en el cual la compañía lanzo una versión única tanto para la plataforma real como para la virtual. Baxter es un robot humanoide de tipo antropomórfico, es decir, posee dos brazos o articulaciones con siete grados de libertad cada uno [4]. La estructura del Robot Baxter puede ser visualizada en la Figura 1.2a.

Este sistema robótico industrial cuenta con tecnologías de detección de vanguardia que incluyen:

- Detección y control de posición, velocidad y torque en cada articulación
- Cámaras para visión por computadora
- Elementos I/O como botones, perillas y una pantalla montada en la parte superior del Robot.

Además, Baxter cuenta con actuadores de tipo SEAs (Series Elastic Actuators), que son mecanismos que cuenta con resortes deformables que permiten la interacción a nivel humano, como golpes o manipulaciones. Adicionalmente, cada articulación posee sensores de torque, los cuales sirven para activar la seguridad del robot [5]. Justamente, en la Figura 1.2b se observa el mecanismo de resortes y el sensor de torque que conforma el actuador para cada uno de los brazos del robot Baxter.

a) b)

Figura 1.2 a) Robot Manipulador Baxter [4] y b) Actuadores elásticos de la serie Baxter [5].

La programación más sencilla del robot Baxter se realiza en forma de entrenamiento, es decir, se mueven manualmente las articulaciones del modo en que se debe realizar la tarea

de ejecución; al finalizar dichos movimientos, se presiona el botón de navegación que permite memorizar la trayectoria realizada para su posterior ejecución. A este tipo de programación se la ha denominado *"capacitación por demostración",* pero esta no es la única programación que puede ejecutar el robot Baxter, también se la puede realizar por medio de un script escrito ya sea en Python o C++.

1.3.2.1. Características técnicas

El robot Baxter posee una base la cual puede ser fija o móvil, según lo requiera el usuario. Para el caso de una base fija, alcanza una altura de 1.78 metros, mientras que la base móvil le otorga una altura de 1.87 metros [5]. En la Figura 1.3 se visualizan las partes constitutivas del robot Baxter las cuales se describen a continuación.

- Pantalla: sirve para visualizar el estado de ánimo del robot mientras se está ejecutando un evento o acción.
- Brazos Articulados: Estos son iguales entre si y son capaces de ejecutar acciones similares, cada uno de estos tiene seis articulaciones con una pinza de manipulación en cada extremo

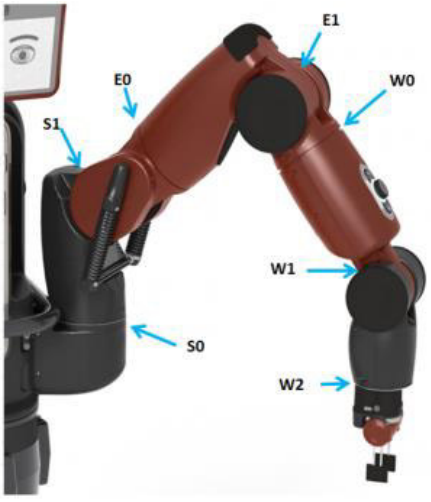


Figura 1.3 Nombre de las Articulaciones del Robot Baxter [5].

Baxter posee un peso alrededor de 80 kilos lo que lo hace sumamente pequeño en comparación a otros robots. Este robot fue diseñado para un trabajo continuo, es decir, puede estar en funcionamiento las 24 horas del día, los 7 días de la semana, sin sufrir mayores daños de hardware. Además, se recomienda una fuente de alimentación ininterrumpida para evitar daños cuando existan cortes de energía [5].

Uno de los puntos más relevantes respecto a este robot es conocer los nombres con los cuales se identifican a cada una de las articulaciones de sus brazos. En la Figura 1.3 se

presenta la nomenclatura que usa el robot Baxter, donde se debe considerar la notación: Muñeca (Wrist) W, Codo (Elbow) E, y Hombro (Shoulder) S.

1.3.2.2. Aplicaciones del Robot Manipulador Baxter

La estructura del robot Baxter está diseñada para trabajar en ambientes industriales ejecutando procesos como Pick and Place, Empaquetado, Clasificación, entre otros. Baxter incorpora cámaras lo que facilita al usuario la realización de tareas complejas, como la clasificación de objetos mediante visión.

Pick and Place.

Este proceso consiste en tomar un objeto, el cual se encuentra en una posición determinada, elevarlo y colocarlo en otra posición [4]. Un ejemplo de esta interacción se muestra en la Figura 1.4.

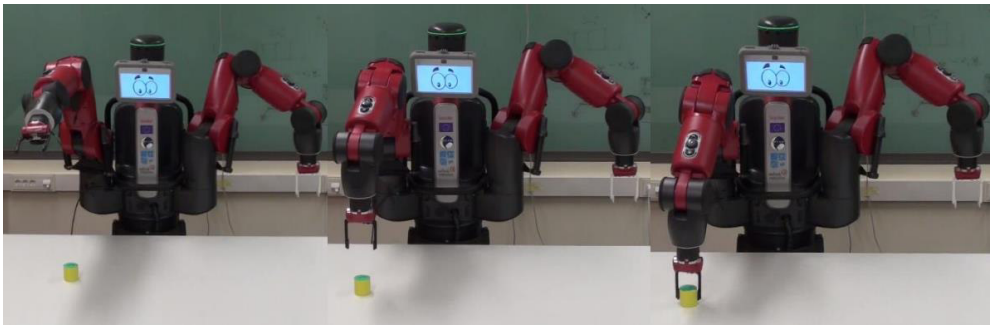


Figura 1.4 Pick and Place con Baxter [4].

Empaquetado

En un proceso de empaque Baxter tiene la capacidad de realizar tareas tanto de recolección como de colocación [6], un ejemplo se observa en la Figura 1.5a.

a) b)

Figura 1.5 Aplicaciones del robot Baxter. a) Empaquetado con Baxter [6]. b) Clasificación con Baxter [7].

Clasificación

Una de las aplicaciones más comunes que puede desarrollar un robot Baxter es la clasificación de objetos, véase Figura 1.5b. Tiene la capacidad de reconocer formas y colores [7].

Programación por Gestos

Una de las aplicaciones más relevantes de la visión artificial que posee el robot manipulador Baxter, es su capacidad de copiar los movimientos del operador con la finalidad de disminuir el proceso de programación del mismo. Este es un claro ejemplo del uso de su cámara ubicada en la parte superior (Cabeza) [19]. En la Figura 1.7 se muestra como el robot Baxter copia el movimiento de las manos del operador por medio de su visión artificial.

Baxter, el robot que aprende a cocinar mirando videos de Youtube

Esta aplicación fue desarrollada por investigadores de la universidad de Maryland y se basa en identificar y reconocer los distintos objetos usados en una cocina. Este sistema tiene la capacidad de interpretar e identificar los diferentes verbos mencionados en el video para la ejecución de las respectivas acciones [20]. En la Figura 1.6 se observa como el robot Baxter mide una cierta cantidad de ingrediente para la receta de cocina.

a) b)

Figura 1.6 Aplicaciones del robot Baxter. a) Uso de visión artificial para programación por Gestos [19]. b) Baxter midiendo la cantidad justa de un ingrediente [20].

Estas son algunas de las innumerables aplicaciones que puede realizar el robot industrial Baxter. En [23] se presenta información sobre las características, el funcionamiento y las posibles aplicaciones que puede ejecutar el robot Baxter a nivel industrial.

1.3.2.3. Modelo Cinemático

El modelo cinemático del Baxter permite el análisis de la posición de cada una de las articulaciones considerando las características geométricas del mismo, para lo cual se

deben tomar inicialmente en cuenta las limitaciones de tamaño de cada articulación. Además, para el cálculo y obtención del modelo cinemático se consideran las longitudes de cada uno de los eslabones. Justamente en la Figura 1.7, se presentan las longitudes de cada eslabón.

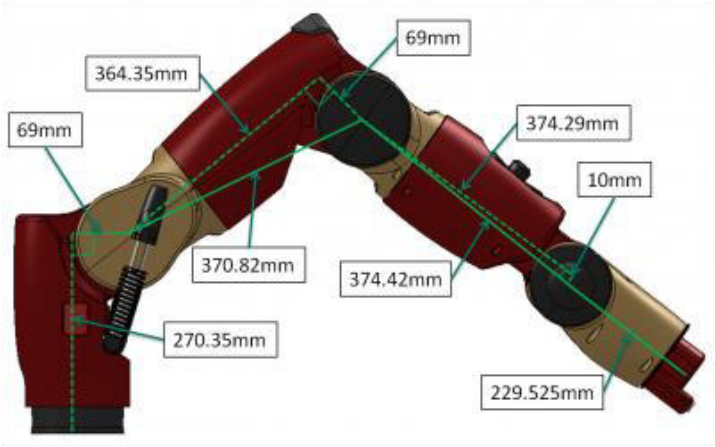


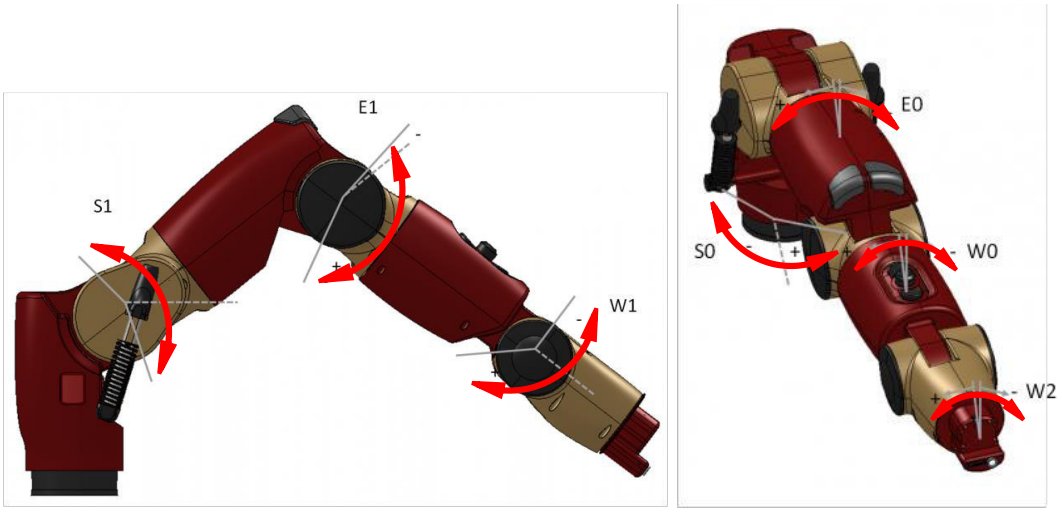
Figura 1.7 Longitud de cada Eslabón para una articulación del Robot Baxter [5].

En la Tabla 1.1 se muestran las medidas de cada eslabón con la respectiva asignación del nombre de la variable.

Tabla 1.1 Longitudes de enlace y desplazamiento del brazo Izquierdo y Derecho.

$\boldsymbol{Variable}$	$\boldsymbol{L_0}$	$\boldsymbol{L_1}$	$\boldsymbol{L_2}$	$\boldsymbol{L_3}$	$\boldsymbol{L_4}$	$\boldsymbol{L_5}$	$\boldsymbol{L_6}$
$\boldsymbol{Valor}\ [\boldsymbol{mm}]$	270.35	69.00	364.35	69.00	374.29	10.00	368.30

Baxter posee dos tipos de articulaciones en cada uno de sus brazos, estas son de tipo bend joints (articulaciones dobladas) o twist joints (articulaciones de torsión). En la Figura 1.8a se presentan las articulaciones dobladas, en este primer caso el giro del actuador es de forma perpendicular al eslabón; mientras que, en la Figura 1.8b se presentan las articulaciones de torsión en donde el giro es paralelo al eslabón [5].



a) b)

Figura 1.8. a) Bend joints b) Twist joints [5].

Una vez conocida la parte estructural de cada uno de los brazos del robot Baxter se debe considerar el campo de movimiento de cada una de las articulaciones. A continuación, se presentan las Tablas 1.2 y 1.3, en donde se detalla estos rangos de movimiento.

Tabla 1.2 Límites y rangos de movimiento de los Bend Joints.

Articulación	Grados		Distancia	Radianes		Distancia
	Limite Min.	Limite Max.		Limite Min.	Limite Max.	
S1	-123	+60	183	-2.147	+1.047	3.194
E1	-2.864	+150	153	-0.05	+2.618	2.67
W1	-90	+120	210	-1.5707	+2.094	3.6647

Tabla 1.3 Límites y rangos de movimiento de los Twist Joints.

Articulación	Grados		Distancia	Radianes		Distancia
	Limite Max.	Limite Max.		Limite Max.	Limite Max.	
S0	-97.494	+97.494	194.998	-1.7016	+1.7016	3.4033
E0	-174.987	+174.987	349.979	-3.0541	+3.0541	6.1083
W0	-175.25	+175.25	350.5	-3.059	+3.059	6.117
W2	-175.25	+175.25	350.5	-3.059	+3.059	6.117

El proceso de obtención del modelo cinemático del robot Baxter conlleva el conocer ciertas ecuaciones básicas que describan la ubicación de una articulación, es decir que definan su posición y orientación. Considerando esto, para representar la orientación de una articulación se hace uso de una matriz rotacional, esta permite representar el cambio de orientación entre dos diferentes sistemas de coordenadas [8]. Además, como las

articulaciones del robot Baxter cuentan con dos tipos de movimientos, es decir, movimientos rotacionales y traslacionales, se define una matriz homogénea la cual se expresa en la Ecuación (1.1).

$$T = \begin{bmatrix} Rotacion_{3x3} & Posicion_{3x1} \\ Perspectiva_{1x3} & Escalon_{1x1} \end{bmatrix} \tag{1.1}$$

Debido al número de grados de libertad que maneja el robot Baxter por cada brazo se hace uso del método denominado Denavit – Hartemberg (D-H) para resolver el problema de la cinemática directa de este robot. El método D-H es aplicado a robots manipuladores que posean una cadena de enlaces conectados entre sí, además, sirve para describir y representar la geometría de un brazo robótico con respecto a un sistema de referencia fijo aplicando matrices de transformación [9]. Este método antes de ser aplicado, se deben considerar ciertas reglas las cuales ayudan a la colocación de los diferentes sistemas de referencia por cada uno de los eslabones involucrados en el sistema. Mayores detalles sobre estas reglas pueden ser revisados en [9].

Basados en dichas reglas, en la Figura 1.9 (Brazo Izquierdo) y 1.10 (Brazo Derecho) se presentan la colocación de los sistemas coordenados acorde al método D-H para cada uno de los brazos del robot Baxter.

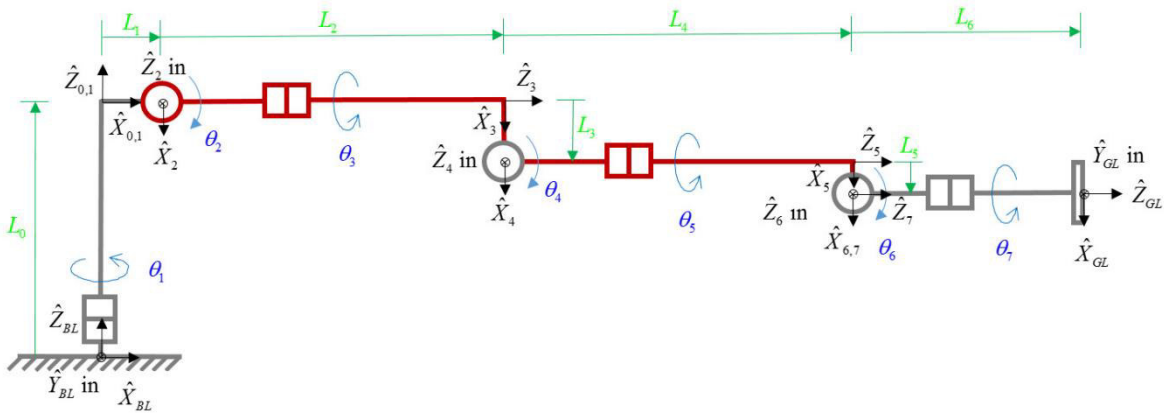


Figura 1.9 Ubicacion de ejes coordenados para el Brazo Izquierdo del Robot Baxter acorde a la convención D-H[10].

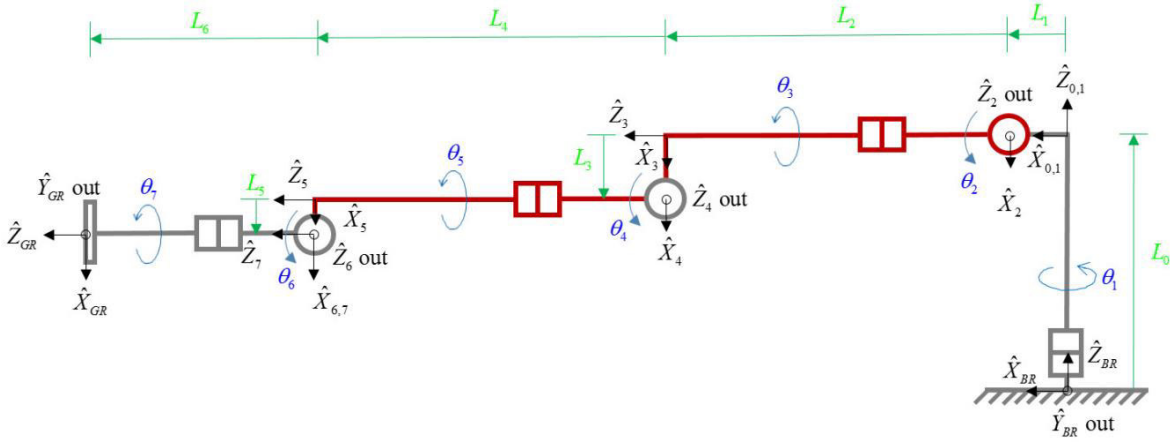


Figura 1.10 Ubicación de ejes coordenados para el Brazo Derecho del Robot Baxter acorde a la convención D-H[10].

Tabla 1.4 Parámetros D-H para el Brazo Izquierdo del Robot Baxter [10].

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90°	L_1	0	$\theta_2 + 90°$
3	90°	0	L_2	θ_3
4	-90°	L_3	0	θ_4
5	90°	0	L_4	θ_5
6	-90°	L_5	0	θ_6
7	90°	0	0	θ_7

Tabla 1.5 Parámetros D-H para el Brazo Derecho del Robot Baxter [10].

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90°	L_1	0	$\theta_2 + 90°$
3	90°	0	L_2	θ_3
4	-90°	L_3	0	θ_4
5	90°	0	L_4	θ_5
6	-90°	L_5	0	θ_6
7	90°	0	0	θ_7

En las Tablas 1.4 y 1.5 se presentan los parámetros modificados para el robot Baxter, estos parámetros son obtenidos tomando como referencia el método descrito en [11], en donde cada marco de coordenadas se localiza justo en cada una de las articulaciones [10]. El robot Baxter es tomado como un caso especial al momento de desarrollar el método D-H debido al alto número de grados de liberta que posee. Como es bien conocido, existe el

método D-H normal o estándar que se describe en [26]. La diferencia principal entre estos dos métodos, estándar y modificado, es la posición en donde se colocan los marcos de coordenadas de cada una de las articulaciones, esto puede ser visualizado en [27] en donde se realiza una comparación de estos métodos y se visualiza ejemplos del proceso aplicativo de cada uno. Generalmente el método convencional para encontrar el modelo cinemático de un manipulador es el estándar, la compañía que diseño al robot Baxter determinó que el método que mejor describe al robot Baxter es el D-H modificado [10].

Como se mencionó anteriormente, el problema de la cinemática directa es resuelto por la multiplicación de las matrices de transformación homogénea. Estas matrices relacionan un sistema de coordenadas denominado (i)-ésimo con otro sistema de coordenadas denominado (i-1)-ésimo. La Ecuación (1.2) define el orden de la multiplicación de estas matrices.

$$T_n^0 = [T_1^0][T_2^1] \dots [T_n^{n-1}] \tag{1.2}$$

En el caso del robot Baxter, el cálculo del modelo cinemático directo se desarrolla por medio de la matriz de transformación homogénea modificada la cual se describe en la Ecuación (1.3). Como se explicó anteriormente, la compañía Rethink Robotics hace uso este método tomando en cuenta que es el que mejor describe los movimientos del robot Baxter [10].

$$ {}^{i-1}_{i}T = \begin{bmatrix} cos(\theta_i) & -sen(\theta_i) & 0 & a_{i-1} \\ sen(\theta_i) * cos(\alpha_{i-1}) & cos(\alpha_{i-1}) * cos(\theta_i) & -sen(\alpha_{i-1}) & -d_i * sen(\alpha_{i-1}) \\ sen(\theta_i) * sen(\alpha_{i-1}) & sen(\alpha_{i-1}) * cos(\theta_i) & \cos(\alpha_{i-1}) & d_i * cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

En base a la matriz dada en la Ecuación (1.3) se procede a reemplazar los valores de cada una de las Tablas 1.4 y 1.5 para los 7 grados de libertad. Las matrices resultantes se presentan en el grupo de Ecuaciones (1.4).

$$ {}^{0}_{1}T = \begin{bmatrix} cos(\theta_1) & -sen(\theta_1) & 0 & 0 \\ sen(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; {}^{1}_{2}T = \begin{bmatrix} -sen(\theta_2) & \cos(\theta_2) & 0 & L_1 \\ 0 & 0 & 1 & 0 \\ -\cos(\theta_2) & -sen(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{2}_{3}T = \begin{bmatrix} cos(\theta_3) & -\text{sen}(\theta_3) & 0 & 0 \\ 0 & 0 & -1 & -L_2 \\ \text{sen}(\theta_3) & cos(\theta_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; {}^{3}_{4}T = \begin{bmatrix} cos(\theta_4) & -\text{sen}(\theta_4) & 0 & L_3 \\ 0 & 0 & 1 & 0 \\ -\text{sen}(\theta_4) & -cos(\theta_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

$$ {}^{4}_{5}T = \begin{bmatrix} cos(\theta_5) & -\text{sen}(\theta_5) & 0 & 0 \\ 0 & 0 & -1 & -L_4 \\ \text{sen}(\theta_5) & cos(\theta_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; {}^{5}_{6}T = \begin{bmatrix} cos(\theta_6) & -\text{sen}(\theta_6) & 0 & L_5 \\ 0 & 0 & 1 & 0 \\ -\text{sen}(\theta_6) & -cos(\theta_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{6}_{7}T = \begin{bmatrix} cos(\theta_7) & -\text{sen}(\theta_7) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \text{sen}(\theta_7) & cos(\theta_7) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

La respuesta del modelo está dada por la Ecuación (1.5).

$$ [{}^{0}_{7}T] = [{}^{0}_{1}T(\theta_1)][{}^{1}_{2}T(\theta_2)][{}^{2}_{3}T(\theta_3)][{}^{3}_{4}T(\theta_4)][{}^{4}_{5}T(\theta_5)]\left[{}^{5}_{6}T(\theta_6)\right][{}^{6}_{7}T(\theta_7)] \quad (1.5) $$

Desarrollando esta ecuación se obtiene:

$$ [{}^{0}_{7}T] = $$

$$ \begin{bmatrix} -(s_1s_3 + c_1s_2c_3)c_4 - c_1c_2s_4 & -c_1c_2c_4 + (s_1s_3 + c_1s_2c_3)s_4 & -s_1c_3 + c_1s_2s_3 & (L_1 + L_2c_2)c_1 - L_3(s_1s_3 + c_1s_2c_3) \\ (c_1s_3 - s_1s_2c_3)c_4 - s_1c_2s_4 & -s_1c_2c_4 - (c_1s_3 - s_1s_2c_3)s_4 & c_1c_3 + s_1s_2s_3 & (L_1 + L_2c_2)s_1 + L_3(c_1s_3 - s_1s_2c_3) \\ s_2s_4 - c_2c_3c_4 & s_2c_4 + c_2c_3s_4 & c_2s_3 & -L_2s_2 - L_3c_2c_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.6) $$

Donde: $s = sen(\theta_i)$ y $c = cos(\theta_i)$, para $i = 1,2,\ldots,7$.

Ahora, asumiendo que la Ecuación (1.7) describe el modelo estándar de la matriz de transformación homogénea.

$$ [{}^{0}_{7}T(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & o_{x_7} \\ r_{21} & r_{22} & r_{23} & o_{y_7} \\ r_{31} & r_{32} & r_{33} & o_{z_7} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.7) $$

Entonces los elementos rotacionales son:

$$ r_{11} = \big((as_4 - c_1c_2c_4)s_6 - (bs_5 + (ac_4 + c_1c_2s_4)c_5)c_6\big)c_7 + \big((ac_4 + c_1c_2s_4)s_5 - bc_5\big)s_7 $$

$$ r_{12} = \big((ac_4 + c_1c_2s_4)s_5 - bc_5\big)c_7 + \big((as_4 - c_1c_2c_4)s_6 - (bs_5 + (ac_4 + c_1c_2s_4)c_5)c_6\big)s_7 $$

$$ r_{13} = -(as_4 - c_1c_2c_4)c_6 - (bs_5 + (ac_4 + c_1c_2s_4)c_5)s_6 \quad (1.8) $$

$$ r_{21} = -\big((ds_4 + s_1c_2c_4)s_6 - (fs_5 + (dc_4 - s_1c_2s_4)c_5)c_6\big)c_7 - \big((dc_4 - s_1c_2s_4)s_5 - fc_5\big)s_7 $$

$$ r_{22} = -\big((dc_4 - s_1c_2s_4)s_5 - fc_5\big)c_7 + \big((ds_4 + s_1c_2c_4)s_6 - (fs_5 + (dc_4 - s_1c_2s_4)c_5)c_6\big)s_7 $$

$$r_{23} = (ds_4 + s_1c_2c_4)c_6 + (fs_5 + (dc_4 - s_1c_2s_4)c_5)s_6$$

$$r_{31} = (hs_6 + (gc_6 + c_2s_3s_5)c_6)c_7 - (gs_5 - c_2s_3c_5)s_7$$

$$r_{32} = -(gs_5 + c_2s_3c_5)c_7 - (hs_6 + (gc_5 + c_2s_3s_5)c_6)s_7$$

$$r_{33} = -hc_6 + (gc_5 + c_2s_3s_5)s_6$$

Donde:

$$a = s_1s_3 + c_1s_2c_3; \quad b = s_1c_3 - c_1s_2s_3; \quad d = c_1s_3 - s_1s_2c_3;$$
$$f = c_1c_3 + s_1s_2s_3; \quad g = s_2s_4 - c_2c_3c_4; \quad h = s_2c_4 + c_2c_3s_4;$$

Mientras que los elementos traslacionales son:

$$^{\circ}x_7 = L_1c_1 + L_2c_1c_2 - L_3a - L_4(as_4 - c_1c_2c_4) - L_5(bs_5 + (ac_4 + c_1c_2s_4)c_5)$$

$$^{\circ}y_7 = L_1s_1 + L_2s_1c_2 - L_3d - L_4(ds_4 + s_1c_2c_4) + L_5(fs_5 + (dc_4 - s_1c_2s_4)c_5) \quad (1.9)$$

$$^{\circ}z_7 = -L_2s_2 - L_3c_2c_3 - L_4h + L_5(gc_5 + c_2s_3s_5)$$

Una vez determinados los valores de la matriz de rotación homogénea, el modelo del movimiento de las articulaciones se presenta en las Ecuaciones (1.10) y (1.11).

- Para el brazo izquierdo

$$[{}^0_7T] = [{}^0_7T(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)] \quad (1.10)$$

- Para el brazo derecho

$$[{}^0_7T] = [{}^0_7T(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)] \quad (1.11)$$

Hasta este punto se ha obtenido la matriz que describe únicamente el movimiento de las articulaciones de cada brazo del robot Baxter sin considerar el aspecto físico del mismo, este aspecto se observa en la Figura 1.2a. Al considerar el aspecto físico del robot Baxter, se concluye que las transformadas anteriormente obtenidas únicamente describen el modelo de cada brazo articulado sin tomar en cuenta tanto las longitudes desde la referencia del sistema de coordenadas hasta el primer eslabón del brazo como la distancia del último eslabón hasta la punta del gripper de agarre de cada brazo. Debido a esto se deben considerar tres matrices de transformación homogénea adicionales para determinar completamente el modelo cinemático directo de cada brazo articulado del robot Baxter, a estas matrices se las denomina matrices faltantes. En total estas matrices faltantes son 6,

debido a que el robot Baxter tiene 2 brazos articulados, la descripción de las mismas se observa en la Tabla 1.6.

Tabla 1.6 Matrices faltantes para completar el modelo cinemático directo del robot Baxter.

Matrices Adicionales o Faltantes		
N. Par	**Variable**	**Descripción**
1er. Par	$[{}^{BL}_{0}T] = [{}^{BR}_{0}T]$	Estas matrices representan el modelo de movimiento desde la referencia de la base de cada brazo robótico hasta la referencia de su primera articulación.
2do. Par	$[{}^{7}_{GL}T] = [{}^{7}_{GR}T]$	Estas matrices representan el modelo de movimiento desde la referencia de la última articulación hasta la referencia de la punta del gripper de cada brazo robótico.
3er. Par	$[{}^{WO}_{BL}T]$, $[{}^{WO}_{BR}T]$	Estas matrices representan al modelo de movimiento desde el sistema de referencias principal hasta la referencia de la base de cada brazo robótico

Una vez analizada la forma física del robot Baxter, el modelo completo de cinemática directa del robot Baxter tomando en cuenta las matrices faltantes se presenta en las Ecuaciones (1.12) y (1.13).

- Para el brazo izquierdo

$$[{}^{W}_{GL}T] = [{}^{WO}_{BL}T][{}^{BL}_{WO}T][{}^{0}_{7}T(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)][{}^{7}_{GL}T] \tag{1.12}$$

- Para el brazo derecho

$$[{}^{W}_{GR}T] = [{}^{WO}_{BR}T][{}^{BR}_{WO}T][{}^{0}_{7}T(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)][{}^{7}_{GR}T] \tag{1.13}$$

En la Tabla 1.6, el primer y segundo par de matrices faltantes representan el sistema desde la base del brazo del robot hasta la primera articulación y desde la última articulación hasta la punta del gripper de cada uno de los brazos del robot Baxter. Estas matrices son definidas utilizando la matriz de rotación y los componentes del vector de posición definidos en las Ecuaciones (1.7) y (1.1) respectivamente.

La Ecuación (1.14) representa el primer y segundo par de matrices faltantes en el modelo del robot Baxter descrito en las Ecuaciones (1.12) y (1.13).

$$[{}^{BL}_{0}T] = [{}^{BR}_{0}T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; [{}^{7}_{GL}T] = [{}^{7}_{GR}T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.14)$$

Donde:

WO: origen de sistema coordenado.

BR: Base derecha del robot.

BL: Base izquierda del robot.

GR: Gripper derecho.

GL: Gripper izquierdo.

El tercer y último par de matrices se obtienen tomando en cuenta los ejes coordenados de la vista superior del robot Baxter, para lo cual se hace uso de la Figura 1.11, en la parte a) de esta figura se observa una vista superior de los ejes coordenados del robot Baxter, mientras que en la b) se indican sus medidas físicas.

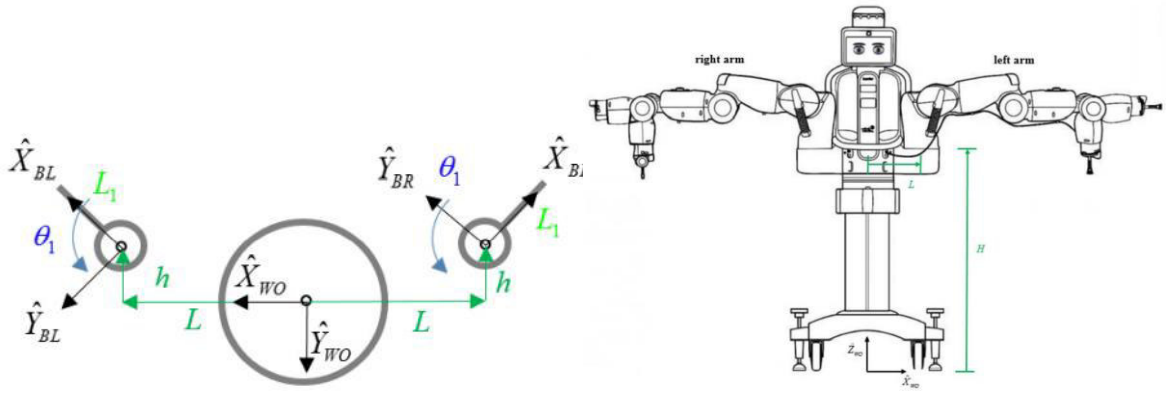


Figura 1.11 Vista del Robot Baxter. a) Ejes coordenados Vista superior. b) Vista frontal [9]

Las matrices de transformación homogénea, que se obtienen a partir de la Figura 1.11, representan el sistema desde la referencia de coordenadas hasta la base de cada uno de los brazos del robot Baxter, como se describe en la Tabla 1.6 y son descritas en la Ecuación (1.15).

$$\left[{}^{WO}_{BL}T\right] = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & L \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & -h \\ 0 & 0 & 1 & H \\ 0 & 0 & 0 & 1 \end{bmatrix}; \left[{}^{WO}_{BR}T\right] = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & -L \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & -h \\ 0 & 0 & 1 & H \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.15)$$

Las variables de la Ecuación (1.15) se definen en la Tabla 1.7. Adicionalmente, las longitudes de la Tabla 1.7 se muestran en la Figura 1.11b.

Tabla 1.7. Longitudes del brazo izquierdo y derecho

$\boldsymbol{Variable}$	$\boldsymbol{Valor}\ [\boldsymbol{mm}]$
$\boldsymbol{L}$	278
$\boldsymbol{h}$	64
$\boldsymbol{H}$	1104

Finalmente, las matrices faltantes de las Ecuaciones (1.14) y (1.15) son remplazadas en las Ecuaciones (1.12) y (1.13) para completar el modelo de cinemática directa del robot Baxter.

Una vez conocido el modelo cinemático del robot Baxter, la compañía Rethink Robotics desarrolló un paquete de ejecución en entorno ROS, el cual contiene los scripts que representan al modelo de cinemática directa, es decir, dicho modelo se encuentra incorporado al sistema por medio de programación, misma que se encuentra desarrollada en varios lenguajes de programación como Python y C++. Para el presente proyecto se hace uso de este paquete sin ninguna modificación debido a que el sistema simulado funciona perfectamente. Adicionalmente, en base a este modelo la compañía Rethink Robotics ha desarrollado los respectivos controladores para cada brazo articulado, que se resumen a continuación.

1.3.2.4. Control para el robot Baxter

Rethink Robotics ha desarrollado dos opciones para controlar la posición de los brazos del robot Baxter con el objetivo de facilitar su uso, ya sea en ambientes reales como en ambientes de simulación. La primera solución se basa en controlar la posición de las articulaciones aplicando un filtro pasa bajos con una frecuencia de corte de 0.2 Hz para suavizar el movimiento de cada articulación y que alcance la posición deseada. La segunda solución es aplicar un control de posición de tipo PID (Proporcional, Integral y Derivativo) con su respectivo lazo de realimentación; además, el paquete cuenta con tres modos adicionales: control de posición en bruto, el control de velocidad y el control de fuerza, estos se basan en un control PID para cada articulación. [35]

Estos modos adicionales de control se diferencian en el algoritmo que aplica cada uno, el fabricante hace énfasis en el uso de estos, ya que, presentan una respuesta arriesgada en el caso de que falle el sistema de alimentación del robot causando daños físicos al usuario. Esto se refiere a que, si se usa uno de estos controladores sobre un robot Baxter real y no se toma en cuenta posibles daños en el sistema de alimentación o tiempos de conexión a la red, el movimiento brusco de los brazos del Baxter puede ocasionar golpes entre las extremidades del robot o causar lesiones al usuario si este se encuentra próximo al robot, esto se debe a que el sistema que usa estos controladores no posee el control de seguridad o modificaciones de seguridad, esto no ocurre con el control de posición, ya que, este cuenta con la seguridad necesaria para evitar estos riesgos mencionados anteriormente y es el control que se usará en el presente proyecto.

Control de posición aplicando un filtro Pasa-Bajos

Un filtro es un elemento que separa señales dependiendo de la frecuencia a la que se encuentra, el filtro pasa bajos cumplen la función de permitir el paso de señales con frecuencias bajas y atenuar las señales de frecuencias altas, los filtros pasa-bajos describen su funcionamiento con la fórmula expresada en la Ecuación 1.13. [37]

$$Tf(s) = (k)\left(\frac{1}{1+\frac{s}{w_c}}\right) \tag{1.13}$$

Donde:

k: es la ganancia del filtro, en el caso del robot Baxter $k = 1$.

w_c: la frecuencia de corte del filtro, esta es la frecuencia a la que la relación de la señal de salida con la entrada es exactamente $\frac{\sqrt{2}}{2}$.

En el caso del robot Baxter, el filtro diseñado por la empresa Rethink Robotics es un filtro de primer orden que posee una frecuencia de corte de 0.2 Hz, el filtro funciona de manera en que la salida del filtro es suavizada cuando existen cambios bruscos en la entrada, como se muestra en la Figura 1.12.

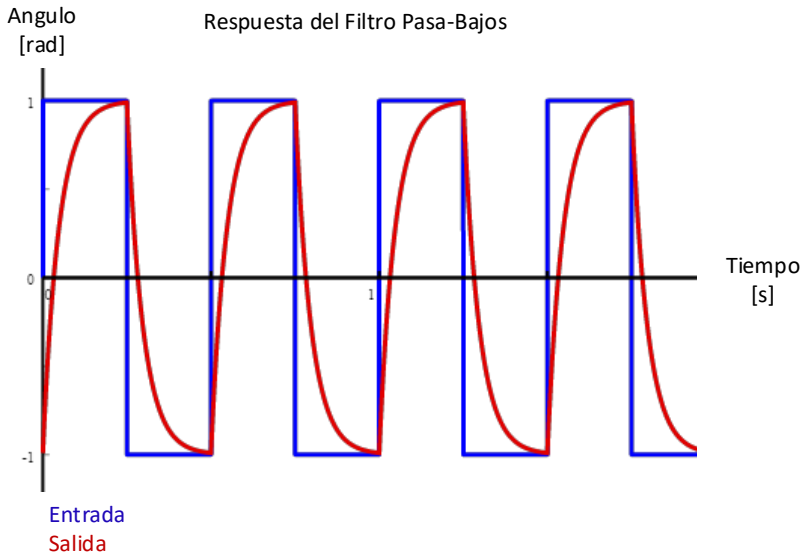


Figura 1.12 Respuesta de un Filtro Pasa-Bajos.

La ecuación (1.13) representa la formula general del filtro pasa bajos en el dominio de Laplace, a esta ecuación, asumiendo k igual a 1, se obtiene la ecuación (1.14).

$$\frac{H(t)}{h(t)} = \left(\frac{w_c}{s + w_c}\right) \tag{1.14}$$

A la ecuación (1.14) se la discretiza por medio del retenedor de orden cero (Zoh) obteniéndose la ecuación (1.15).

$$Y(z) = \frac{H(z)}{h(z)} = \frac{1 - e^{-w_c T}}{z - e^{-w_c T}} \tag{1.15}$$

Donde:

H: es la salida del sistema.

h: es la entrada del sistema

T: es el tiempo.

w_c: es la frecuencia angular de corte, en el caso del robot Baxter $w_c = 1.256$.

Desarrollando la ecuación (1.14) se obtiene:

$$H(z)(z - e^{-w_c T}) = h(z)(1 - e^{-w_c T}) \tag{1.16}$$

A la ecuación (1.16) se le aplica la transformada Z inversa donde T es el tiempo entre muestras, en el caso del robot Baxter $T = 0.008726646$ segundos, se obtiene la ecuación (1.17), este proceso se presenta en [45].

$$H[k+1] - H[k]e^{-w_c T} = h[k](1 - e^{-w_c T}) \quad (1.17)$$

Desarrollando la ecuación (1.17) y despejando para una sola muestra se obtiene:

$$H[k] = \beta H[k-1] + (1-\beta)h[k-1] \quad (1.18)$$

Siendo:

$$\beta = e^{-w_c t} \quad (1.19)$$

Usando la notación $H[k] = H_n \; y \; h[k] = h_n$ se tiene la ecuación (1.19) la cual representa la fórmula básica de un filtro de paso bajo de respuesta de impulso infinito, ecuación con la que trabaja el filtro pasa bajos del robot Baxter.La obtención de esta ecuación puede ser visualizada en [46].

$$H_n = \beta H_{n-1} + (1-\beta)h_n \quad (1.20)$$

En la ecuación (1.21) se observa la fórmula del filtro diseñado por la compañía en función de la posición del ángulo de giro de cada articulación, esta ecuación tiene una estructura igual a la ecuación (1.20) tomando en cuenta que la variable positions es la entrada del sistema (h_n), cmd es la salida del sistema (H_n) y β es igual a 0.98909.

$$cmd[joint] = (0.010900)(positions[joint]) + (0.98909)(cmd[joint]) \quad (1.21)$$

Donde:

$joint$: es la articulación que se desea mover.

$positions$: es el ángulo de giro actual de la articulación.

cmd: es el ángulo de giro al cual se desea llevar a la articulación.

Una vez obtenidas las señales de posición para cada articulación, estas son enviadas por medio de mensajes hacia el robot Baxter virtual o Real.

Controladores PID del robot Baxter

Generalmente un controlador de tipo PID es aquel que controla un sistema en lazo cerrado para alcanzar en su salida un valor deseado. Este tipo de controlador consta de tres elementos denominados acción proporcional, integral y derivativa, un controlador de tipo PID trabaja sobre la señal de error, es decir por medio de su control hace que la señal del error sea tan pequeña como sea posible, la acción proporcional multiplica el error por una

constante kp, la acción derivativa deriva el error y la acción integral integra la señal del error. El controlador PID aplica la ley de control que se expresa en la Ecuación 1.22. [36]

$$c(t) = (kp)e(t) + ki \int e(t)dt + kd\left(\frac{de(t)}{dt}\right) \tag{1.22}$$

Donde:

$c(t)$: señal de control

$e(t)$: señal del error

kp, ki, kd: parámetros del controlador PID

En el caso del robot Baxter, como se mencionó anteriormente, este cuenta con cuatro modos de control que incorporan un controlador de tipo PID cada uno. Estos controladores poseen la arquitectura presentada en la Ecuación (1.22), no obstante, el algoritmo que ejecuta cada uno de estos controles no es el mismo. A continuación, se presentan estos algoritmos representados, adicionalmente, mediante diagramas de flujo.

- **Control de posición**

Es un modo de control básico y fundamental para el movimiento de las extremidades de Baxter denominado control de posición estándar. A este modo ingresan los ángulos de giro de las siete articulaciones con el objetivo de que el robot Baxter alcance dichos ángulos de giro dependiendo de la extremidad o brazo que se desee mover. Este modo de control garantiza la seguridad y comportamiento del robot Baxter [35]. En la Figura 1.13 se presenta el proceso que realiza el algoritmo para garantizar su seguridad y comportamiento.

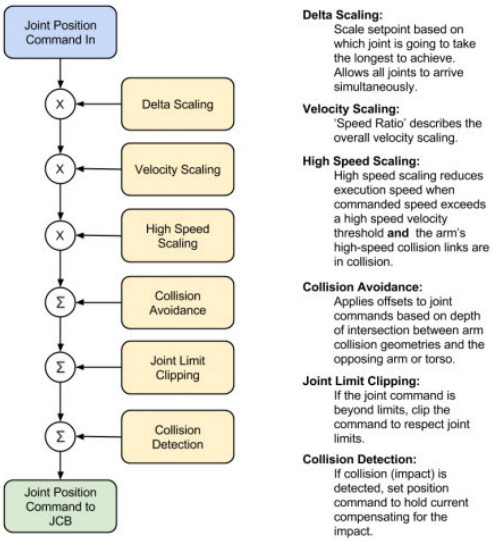


Figura 1.13 Diagrama de flujo para algoritmo del control de posición estándar.[35]

- **Control de posición en bruto**

Es un modo de control de posición más directo debido a que no aplica una compensación para evitar colisiones, es decir, el algoritmo detecta la colisión, pero no la evita. Este modo el movimiento de las articulaciones es más rápido, por esta razón su uso debe ser realizado con cuidado [35]. En la Figura 1.14 se presenta el proceso que realiza el algoritmo para el control de posición en bruto.

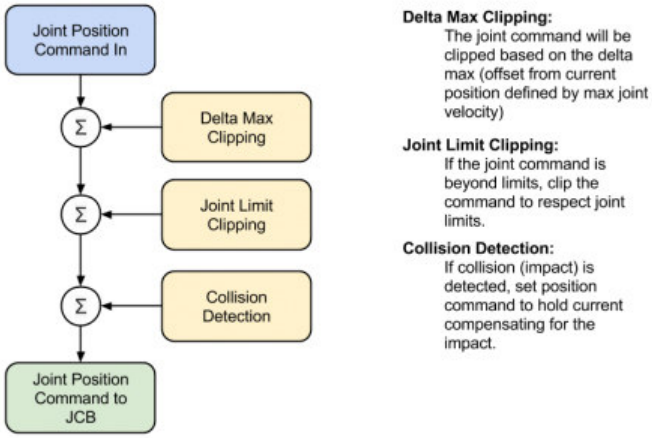


Figura 1.14 Diagrama de flujo para algoritmo de control de posición en bruto.[35]

- **Control de Velocidad**

Este es un modo de control de velocidad en el cual se especifican siete velocidades a las cuales cada articulación debe alcanzar. En este caso se aplican la prevención y detección de colisiones, no obstante, no se garantiza la seguridad y comportamiento del robot Baxter

debido a que si una de las siete velocidades ingresadas llega al límite permitido su control se detendrá mientras que las demás articulaciones continuarán con su respectivo movimiento controlado. Esto puede producir movimientos inesperados y potencialmente peligrosos para el usuario y para el robot mismo [35]. En la Figura 1.15 se presenta el proceso que realiza el algoritmo para el control de velocidad.

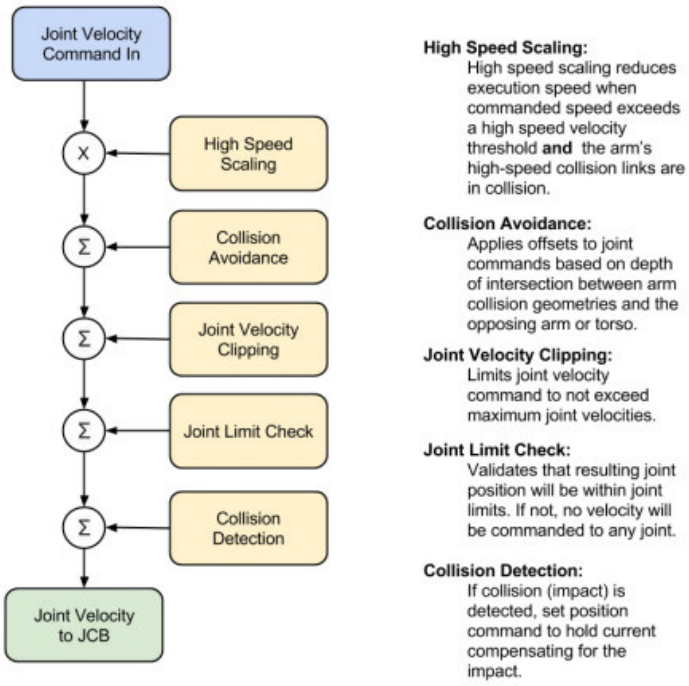


Figura 1.15 Diagrama de flujo para el algoritmo de control de velocidad.

- **Control de Fuerza**

Es un modo de control de fuerza en el cual se especifican siete fuerzas, las cuales cada articulación debe alcanzar. Este modo de control no garantiza la seguridad y comportamiento del robot Baxter debido a la ausencia de la compensación para evitar y detectar colisiones [35]. En la Figura 1.16 se presenta el proceso que realiza el algoritmo para el control de fuerza.

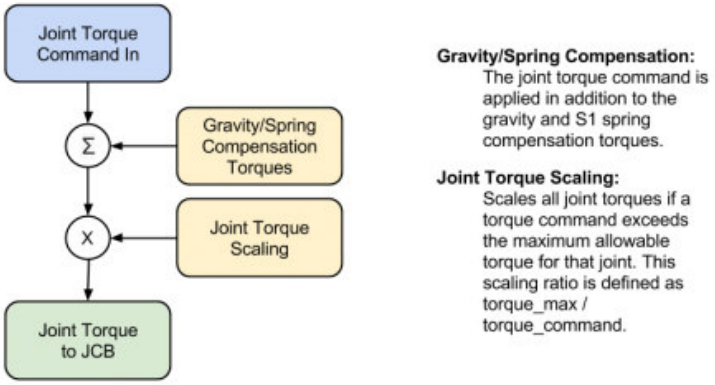


Figura 1.16 Diagrama de flujo para el algoritmo de control de fuerza.[35]

Una vez obtenidas las señales de posición para cada articulación, estas son enviadas por medio de mensajes hacia el robot Baxter virtual o real. Estos modos de control son los que se encuentran en el paquete ROS y funcionan dependiendo del modo que desea implementar el usuario. En el presente proyecto se hace uso del control de posición aplicando un filtro pasa-bajos y el control de posición estándar. El uso del filtro pasa-bajos se debe a que existen posiciones en las cuales los movimientos de las articulaciones se deben realizar lentamente y con suavidad como es el caso de la toma de objetos, además, cuando se usa el filtro pasa bajos el movimiento de las extremidades se realiza independientemente, es decir, no se puede mover las dos extremidades al mismo tiempo debido a que las líneas de programación primero ejecutan el movimiento de una extremidad y después el movimiento de la otra extremidad. Mientras que el uso del control de posición estándar se debe a que otorga la posibilidad de mover las dos extremidades en conjunto. Cabe recalcar que en la aplicación a desarrollar existen posiciones en la cuales es necesario únicamente mover una articulación con suavidad como es el caso de la toma de objetos tanto de la banda transportadora como de la posición luego de realizar la clasificación. También existen posiciones en las cuales es necesario mover las dos extremidades conjuntamente para disminuir el tiempo de ejecución de la aplicación, por ejemplo, las posiciones para el traslado de los objetos. Debido a estos requerimientos de movimiento se usa el filtro pasa bajos y el control de posición. El control de posición incorpora un controlador PID que posee constantes kp, ki y kd, cuyos valores por default se presentan en la Tabla 1.8.

Tabla 1.8 Constantes kp, ki y kd para el control de posición obtenidas por Rethink Robotics.

Valores de las constantes kp, ki y kd			
Articulación	kp	ki	kd
S0	700	0.01	100

S1	10000	100	100
E0	4500	35	1
E1	5500	60	2
W0	1000	30	0.01
W1	900	0.1	0.01
W2	1000	0.1	0.01

En la siguiente sección se detallarán los elementos del entorno ROS y su integración con el robot Baxter.

1.3.3. ENTORNO ROS

ROS (Robot Operating System) es un software de código abierto usado esencialmente para el desarrollo de sistemas robóticos. Este sistema fue introducido a la comunidad por el instituto de investigación Willow Garage en 2007 bajo licencia BSD (Berkeley Software Distribution) y el resto de sus programas son de código abierto (OSS) [12].

El entorno ROS tiene como objetivo la integración y reutilización de programas de uso libre, por lo cual se desarrolló tomando en cuenta plataformas robóticas ya existentes como OpenCV, Player/Stage, Gazebo y otros. Cada una de estas plataformas poseen un paquete de datos para el uso en conjunto con el sistema ROS. Está compuesto por un conjunto extenso de librerías de programación, aplicaciones, drivers y herramientas de simulación las cuales facilitan la creación de aplicaciones tanto para robots simulados como para robots reales [12]. ROS es un marco distribuido de procesos conocido como Nodos, estos nodos permiten el diseño independiente de paquetes ejecutables y el acople libre de estos cuando se requieran, es decir, el uso de Nodos permite ejecutar decisiones independientes sobre el desarrollo y la implementación del sistema robótico.

Como todo sistema operativo ROS posee tanto ventajas como desventajas, variad de las cuales son.

- Ventajas:
    - Permite la resolución de problemas de alto nivel.
    - El aprendizaje para su programación se lo puede realizar observando documentación y códigos de otros programadores.
    - Comunicación con programas de software libre.
- Desventajas

- El proceso de integración de programas de software libre es largo y tedioso.
- Cambios y evoluciones continuas en el software provocan que el sistema se vuelva obsoleto para sus versiones originales.
- Costos altos en cursos para el aprendizaje del entorno ROS.

1.3.3.1. Conceptos básicos

ROS por ser un Meta-sistema operativo posee una gran cantidad de elementos, estos tipos de sistemas se caracterizan por usar recursos divididos e integrarlos en un sistema informático, los recursos son transmitidos de forma ordenada y a altas velocidades. A continuación, se presentarán los elementos más relevantes que conforman el entorno de programación.

Sistema de archivos ROS: Este sistema se encuentra dividido por dos niveles de operación los cuales se denominan como Packages y Stacks.

- Package: el package o paquete es un sistema de archivos de bajo nivel, este sistema contiene en su mayoría ejecutables, modelos, servicios y mensajes herramientas o librerías [21]. En la Figura 1.17 se observa el contenido de un package.

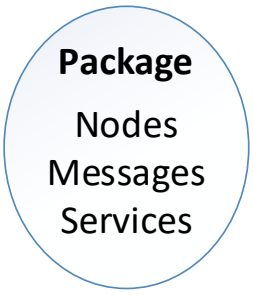


Figura 1.17 Package o Paquete [21].

- Stacks: los stacks o pilas son el conjunto de packages que forman una librería de alto nivel, estos son complementos unos de otros [21]. En la Figura 1.18 se observa la estructura de un stack.

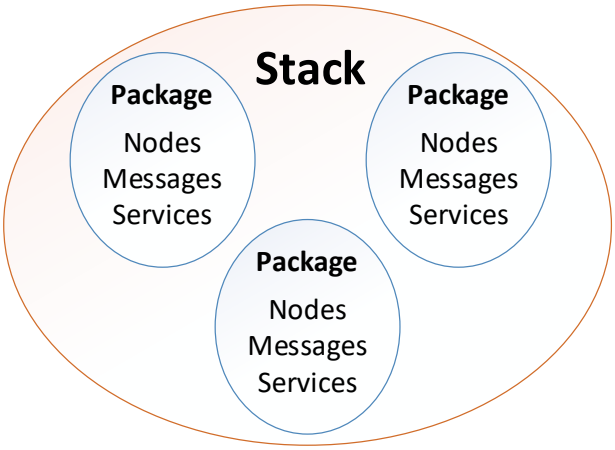


Figura 1.18 Stack o Pila [21].

Nodos: En el entorno ROS al nodo se lo conoce como un proceso individual que realiza un cómputo o cálculo determinado, este cómputo cumple la función de enviar, recibir u ofrecer servicios entre dos o más nodos. La característica fundamental que poseen los nodos es su nombre, ya que, cada uno de estos deben poseer un nombre único para un funcionamiento óptimo. El nombre único es aquel que lo identifica y lo distingue de un grupo de nodos que se ejecutan en conjunto [21, 22]. Pueden ser programados en lenguajes como C++ y Phyton, para hacer uso de estos tipos de programación en la cabecera del programa se debe añadir la librería *roscpp* o *rospy*, respectivamente. La comunicación entre nodos es independiente y el uso de estos dos lenguajes de programación en un mismo proyecto no altera la ejecución del mismo, la comunicación funcionara normalmente siempre y cuando los mensajes que se envíen y reciban sean del mismo tipo.

Topics: Los topics son los sistemas de transporte por donde se enrutan los mensajes que un nodo envía o recibe, adicionalmente, el topic o tema es usado dependiendo del tipo de contenido de mensaje que envía o recibe [21]. La semántica que manejan los topics es de tipo publish/subscribe, es decir, para recibir o enviar información los nodos deben informar al Master que se desea publicar o suscribirse a él para el envío y recepción de datos [22]. Una de las características principales de los topics es que estos son unidireccionales, por lo cual, un nodo que se encuentra enviando información por un topic nunca recibirá información por el mismo topic de que la información enviada está siendo leída. Para recibir una respuesta de lectura es necesario realizar otro tipo de comunicación entre nodos [12]. El uso de topics ayuda al sistema general debido a que desacoplan la producción del consumo de información en un proceso definido. Esto se debe a que varios nodos pueden estar publicando o suscribiéndose por un mismo topic, pero con la diferencia de que un nodo desconoce la existencia de otro nodo cuando los dos se encuentra conectados por el mismo topic.

Services: Es un tipo de sistema de transporte en el cual la comunicación entre nodos se realiza de manera request/response. Este tipo de comunicación se debe a que en ciertos sistemas es necesario recibir una respuesta del nodo con el que se está comunicando [23]. En el entorno ROS un servicio se define como la forma en la que se envía y recibe respuesta por medios de mensajes.

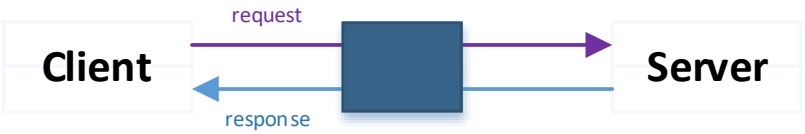


Figura 1.19 Service [21].

En la Figura 1.19 se observa que un nodo puede ofrecer un servicio (server) y un nodo cualquiera tiene la capacidad de acceder a dicho servicio (client). Este procedimiento es efectuado por medio de mensajes tanto para el envío como para la recepción de información.

Messages: Los mensajes en el entorno ROS son definidos como estructuras de datos, que poseen campos escritos. Estos mensajes son usados como medio de comunicación entre nodos vía Topics [21]. La estructura de datos de los mensajes es similar a la estructura de C++, el entorno ROS tiene un paquete de mensajes definido como std_msgs, estas estructuras de datos son descritos en la Tabla 1.9.

Tabla 1.9 Tipos de datos (Equivalencia).

msg	$C++$
Bool	Unit8_t
int8	int8_t
unit8	unit8_t
int16	int16_t
unit16	unit16_t
int32	int32_t
unit32	unit32_t
int64	int64_t
unit64	unit64_t
float32	float
float64	doublé
String	std::string
Time	ros::Time
duration	ros::Duration

Master: Es un nodo del núcleo del sistema conocido como *roscore*, al ser activado proporciona un registro de nombres, servicios y parámetros los cuales hacen posible la comunicación entre nodos lanzados individualmente dentro del sistema; sin el Master los nodos no podrían encontrarse. En la Figura 1.20 se observa que el *roscore* está intermedio en la conexión entre nodos.

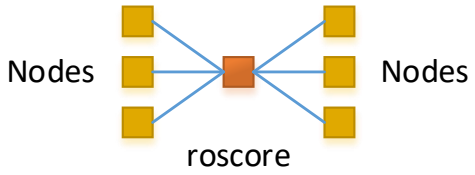


Figura 1.20. Master [21].

Launch: La ejecución de nodos, el llamar a servicios, leer y publicar en tópicos se puede realizar por medio de una terminal; sin embargo, el entorno ROS proporciona una herramienta que facilita la ejecución de varios nodos, llamar varios servicios y ejecutar varias herramientas a la vez, esta herramienta se la conoce como *roslaunch.* Esta es ejecutada sobre un archivo *.launch,* este archivo es descrito en XML y contiene todos los nodos, servicios y herramientas a ejecutar. Para la ejecución de estos tipos de archivos se debe considerar abrir una Terminal y escribir la siguiente línea de comando, adicionalmente es necesario ubicarse en el directorio donde se encuentra el proyecto a ejecutar.

$ roslaunch nombre_paquete nombre_archivo.launch

1.3.3.2. Comunicación

Como se indicó en la sección anterior, la comunicación del entorno ROS se efectúa bajo mensajes, básicamente son el medio de comunicación que usan los nodos para enviar y/o recibir datos [12]. En el entorno ROS la comunicación entre dos o más nodos se divide en tres tipos:

- Topic
- Service
- Action

Los dos primeros fueron estudiados como conceptos en la sección anterior, ahora se desarrollará un estudio más a fondo para determinar las diferencias que existen entre cada uno de estos tipos de comunicación.

Topic (Tema).

Este tipo de comunicación se define como mensajes de tema, esta comunicación se denomina Publisher/Subscribe en donde el Publisher (Editor) envían cierta información y el subscribe (Subscriptor) reciben dicha información. Este tipo de comunicación es unidireccional debido a que el nodo editor no recibe información de que el nodo subscriptor haya recibido la información correctamente. Como se mencionó anteriormente, para que los nodos tengan una comunicación efectiva es necesario subscribirse al nodo Master [12]. Por ejemplo, la comunicación con características de unidireccionalidad y asincrónica es adecuada cuando existen datos de sensores en el sistema, ya que, un sensor envía datos continuamente a través de una conexión.

Service (Servicio).

Este tipo de comunicación se define como mensajes de servicio la cual tiene una estructura de Client/Server en donde el cliente solicita un servicio y el servidor es el responsable de la respuesta del servicio solicitado. Estos mensajes de servicio tienen como característica principal la bidireccionabilidad síncrona, este tipo de comunicación sirve cuando se requiera tanto la solicitud como la respuesta realizando un intercambio de mensajes sincrónicos [12]. Este método de comunicación de ROS, a diferencia de los Topic, son comunicaciones de mensajes de una sola vez, es decir, la comunicación se efectúa únicamente hasta cuando se complete la solicitud y la respuesta del servicio, posteriormente los nodos en comunicación se desconectarán. Por esto los mensajes de servicio son muy útiles al momento de generar comandos para la realización de un proceso que involucre robots, además la carga de red es relativamente baja.

Action (Acción)

Este tipo de comunicación es definido como mensajes de acción en donde el cliente define la acción y el servidor ejecuta el proceso para efectuar la acción requerida, una vez terminado el proceso el servidor envía al cliente comentarios y resultados del proceso ejecutado, como se describe la estructura que posee esta comunicación es de tipo Client/Server [12]. En los mensajes de acción la transmisión de datos es bidireccional como en los mensajes de servicio y asincrónica como en la comunicación de mensajes de tema.

1.3.3.3. Manipulador Baxter en el entorno ROS

La compañía Rethink Robotics ha desarrollado una versión de Baxter la cual es usada específicamente por organizaciones académicas e investigación. Esta versión posee un diferente software a la versión original.

La versión del robot de investigación Baxter cuenta con una configuración SDK la cual se ejecuta desde una estación de trabajo. El SDK proporciona al sistema una interfaz de programación de aplicación (API) la cual es accesible con el entorno de código abierto ROS. [13]. La API sirve para:

- La ejecución directa de comandos y ROS scripts en la operación del robot de investigación Baxter.
- Permite ejecutar al Baxter como ROS Master y así permitir la comunicación con cualquier dispositivo que se encuentre en la red de Baxter.
- Permite la lanzar y ejecutar nodos ROS con la finalidad de comunicarse con el robot.
- Permite el control de sensores y junturas de cada extremidad del robot.

Rethink Robotics creó un paquete de ejecución para el robot Baxter el cual se encuentra accesible al público en el repositorio en GitHub del siguiente enlace: https://github.com/RethinkRobotics/baxter_simulator

Los requerimientos para ejecutar este paquete de datos son los siguientes:

- Ubuntu 16.04 LTS.
- ROS Indigo o Ros Kinetic.
- Python 2 (Obligatorio).

En el siguiente capítulo se realizará una breve explicación del proceso de instalación de este paquete, haciendo referencia los puntos más relevantes en dicho proceso.

1.3.4. ENTORNO DE SIMULACIÓN GAZEBO.

El entorno de simulación Gazebo es un software libre que puede ser reconfigurado, ampliado y modificado con el objetivo de mejorar su sistema. Este programa es compatible con el entorno ROS, es decir, puede ser ejecutado a través de comando ROS y utilizar las APIs del entorno para el control del sistema robótico enviando y recibiendo datos entre estos dos softwares.

Gazebo es un programa usado como simulador para ambientes de tipo 3D, cinemáticos, dinámicos y multi-robots. Una de las características principales de este simulador es que permite crear entornos reales y tridimensionales con un alto grado de complejidad. Esto hace que el sistema o mundo desarrollado sea lo más realista posible, dando paso a la capacidad de generar ambientes con características físicas como la gravedad. Una de las características más importantes es la interacción entre los robots situados en el sistema y

el mundo generado ya que Gazebo permite al robot ejecutar movimientos como tomar o empujar objetos y rodar o deslizarse por el suelo.

1.3.4.1. Biblioteca de modelos

Gazebo posee una biblioteca de modelos básicos los cuales se presenta únicamente cuando el programa es abierto en el sistema operativo. Esta biblioteca hace uso de una conexión a internet para ser visualizada en el software, la misma contiene objetos para crear el mundo en el cual se genera el entorno de simulación en donde interactuará un robot específico. Estos objetos pueden ser bloques, estacionamientos, mesas, sillas e incluso hay una sección dedicada al robot Turtlebot 3. Como es una biblioteca básica está limitada en modelos industriales como bandas transportadoras o motores [28].

Al ser una biblioteca básica el modelo del robot Baxter no se encuentra disponible, en las subsecciones siguientes se detalla cómo importar modelos externos como es el caso del robot Baxter.

1.3.4.2. Diseño de modelos compatibles

Como se mencionó anteriormente Gazebo posee una biblioteca básica de modelos, pero no está delimitado solamente al uso de los mismos. En esta sección se desarrollará un estudio en donde el principal objetivo es diseñar modelos que acepte este entorno de simulación. La necesidad de diseñar modelos que acepte el software se debe a que, en la biblioteca básica, no existen modelos industriales como es el caso de las bandas transportadoras, las cajas y los objetos de clasificación que se necesitan para el presente proyecto.

Existen 2 métodos para el diseño de modelos compatibles para la plataforma Gazebo.

- El primer método consiste en generar un archivo con cierta programación denominada XML, el cual especifica los parámetros físicos del modelo diseñado. Este tipo de programación también se conoce como SDFormat el cual es un formato estable, robusto y extensible capaz de describir aspectos estáticos y dinámicos de un sistema robótico o ambiental [14]. La extensión para este archivo de programación es .urdf, .world, .sdf o .launch, varios ejemplos de programación pueden ser revisados en [14].
- El segundo método se desarrolla tomando en cuenta programas secundarios como Solidworks, Blender, entre otros. Una vez diseñado el modelo en cualquiera de estas plataformas se procede a exportar el archivo a una extensión determina. Por ejemplo, Solidworks posee una herramienta que facilita la construcción del archivo

de extensión .launch .urdf y .stl del modelo diseñado, mientras que Blender permite exportarlo como extensión .stl y .dae.

Las extensiones antes mencionadas son compatibles con Gazebo lo que facilita la ejecución dentro del entorno de desarrollo.

1.3.4.3. Importación de modelos a Gazebo

Como se mencionó anteriormente Gazebo no cuenta con el modelo del robot Baxter, por lo que la importación de este modelo es fundamental. En este caso la empresa Rethink Robotics ha diseñado un paquete de simulación el cual será estudiado en la subsección siguiente, por el momento la explicación se centra en la importación de archivos a Gazebo.

En el apartado anterior se habló de archivos con extensiones .urdf, .stl y .dae archivos con los cuales trabaja el paquete de simulación del robot Baxter, para cada uno de estos tipos de archivos existe una forma de importarlos hacia el entorno Gazebo. En todos los casos, una vez ejecutado el entorno de simulación Gazebo, la importación de modelos ya sean de objetos como de robots se pueden desarrollar de dos maneras distintas.

- Desde un archivo de extensión .launch.
- Desde un programa escrito en C++ o Python.

Para el caso del proyecto se usará el archivo de extensión .launch para lanzar el entorno de simulación incluido el robot Baxter y el archivo de extensión .py para cargar los modelos adicionales de la escena.

Importar un archivo de extensión URDF

Esta extensión posee las características físicas y dinámicas del modelo del robot u objeto creado. La forma más fácil de obtener este archivo es construyendo el modelo en Solidworks y con la herramienta de exportación obtener el archivo. Este archivo puede ser ejecutado por medio de un archivo de extensión .py con las siguientes líneas de código [29].

```
1  block_xml = ''
2  with open (model_path + "block/model.urdf", "r") as block_file:
3      block_xml=block_file.read().replace('\n', '')
```

La primera línea de código garantiza que la variable *block_xml* no contenga datos, en la segunda línea de código la variable *model_path* contiene el directorio en donde se encuentran los modelos, adicionalmente se apunta a la carpeta que contenga el archivo de

extensión .urdf y lo añade a la variable *block_file*. Finalmente, en la tercera línea de código, el contenido de la variable *block_file* es cargado en la variable *block_xml*, la función *read()* sirve para mostrar la figura en el momento que se ejecute el código [25].

Importar un archivo de extensión STL y DAE

Estas extensiones son obtenidas a través de la herramienta de exportación de SolidWorks o Blender (los modelos que componen al robot Baxter fueron desarrollados en Blender), para su respectiva importación es necesario navegar sobre el archivo urdf.xacro (tipo de extensión para programación XML) y tomar en cuenta los siguientes cambios.

Cambio 1: En la sección modelo/visual/mesh se carga el directorio donde se encuentra el archivo de extensión .dae [24].

Cambio 2: En la sección modelo/collision/mesh se carga el directorio del archivo de extensión .stl [24].

Esta importación hace uso de dos archivos de extencion .urdf.xacro, el primero para añadir los archivos de extensión .dae y .stl, mientras que el segundo es un archivo global que contiene más archivos similares al primero. Una vez realizado los cambios se procede a usar el archivo de extensión .launch para cargar las figuras en el entorno de simulación [25].

Existen varios procesos de importación de modelos, los procedimientos estudiados en esta sección hacen referencia únicamente para el paquete de simulación del robot Baxter debido a que todos los archivos de ejecución se encuentran dentro del paquete de simulación del mismo. En el Capítulo 2 se describirá el proceso de importación tanto para los objetos de clasificación como para las bandas transportadoras, cajas y mesas usadas en el proyecto.

1.3.4.4. Manipulador Baxter en el simulador GAZEBO

La compañía Rethink Robotics de igual forma como creo el paquete ROS para el robot Baxter creo el paquete de simulación respectivo en código abierto. Este paquete de simulación contiene cada uno de los modelos que componen el robot industrial Baxter, estos archivos son de extensión .urdf, .stl y .dae. Para obtener acceso a los archivos se debe acceder al repositorio en GitHub en [25].

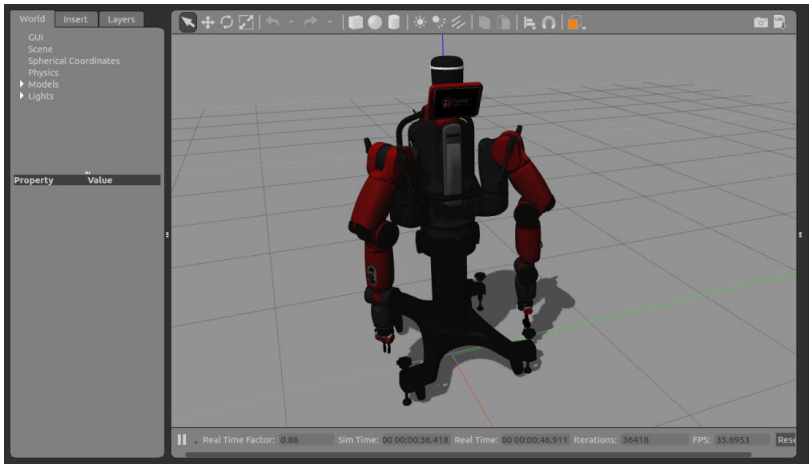


Figura 1.21 Baxter ejecutado en la plataforma Gazebo.

En la Figura 1.21 se observa el robot Baxter ubicado en la plataforma de simulación Gazebo. Este robot al contar con todas sus características reales es necesario habilitarlo para realizar algún movimiento determinado [13]. Este proceso será evidenciado en el capítulo siguiente, en el cual se detalla cómo se carga el paquete de simulación del robot Baxter al entorno Gazebo.

1.3.5. CLASIFICACIÓN DE OBJETOS POR VISIÓN

En esta sección se realizará un estudio de una de las aplicaciones comúnmente usadas por el robot Baxter, esta aplicación se denomina clasificación de objetos por visión haciendo uso de la cámara principal que posee el robot Baxter en la parte superior.

La visión artificial juega un papel importante en el desarrollo de sistemas de manipulación robóticos. Los sistemas de visión artificial permiten que un sistema robótico adquiera características de flexibilidad, adaptación y capacidad de reorganización. Su uso ayuda a la ejecución de tareas como:

- Reconocimiento de piezas.
- Posicionamiento y orientación de objetos en base a un punto de referencia.
- Clasificación de objetos.
- Ensamblaje de piezas.

La visión artificial es denominada como un subcapítulo de la inteligencia artificial y también se la conoce con el nombre de visión por computadora. Estos sistemas son usados cuando la simple detección de presencia no es una fuente suficiente de información para el desarrollo de un proceso, en este caso un proceso de clasificación de objetos [15,16].

1.3.5.1. Métodos de Clasificación

Por el Color

Uno de los métodos más comunes de clasificación usado en sistemas industriales es la clasificación por color, ya que en ciertas circunstancias del proceso existen piezas u objetos similares, pero de un color distinto. Este método hace uso de un barrido de imagen para detectar el color del objeto para realizar su respectiva clasificación. Este barrido se logra gracias al uso de procesamiento de imagen básico, característica propia del sistema de visión artificial [17]. En la Figura 1.22 se observa el diagrama de flujo básico que sigue el método de clasificación de objetos por color.

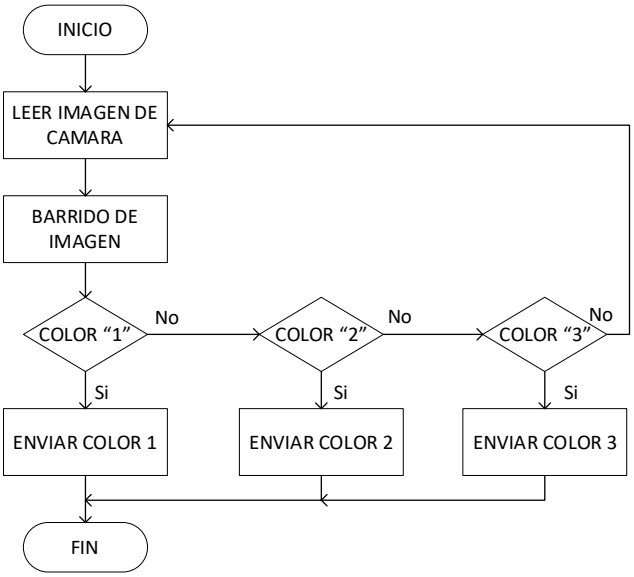


Figura 1.22 Diagrama de flujo básico para la clasificación por color [17].

Por la Forma

El método de clasificación por forma hace uso de un algoritmo más complejo que la clasificación por color, en este caso el procesamiento de imagen conlleva lo siguiente: la conversión de la imagen a un sistema binario, la eliminación de pixeles con el objetivo de eliminar pixeles no pertenecientes al borde y calcular la nueva región blanca creada en el proceso. Finalmente se hace uso de un algoritmo de detección de bordes, denominado CANNY [17], para calcular el número de bordes del objeto y en base a este número determinar la forma del objeto. Este método se usa principalmente cuando el objeto posea una forma cuadrada, circular o triangular.

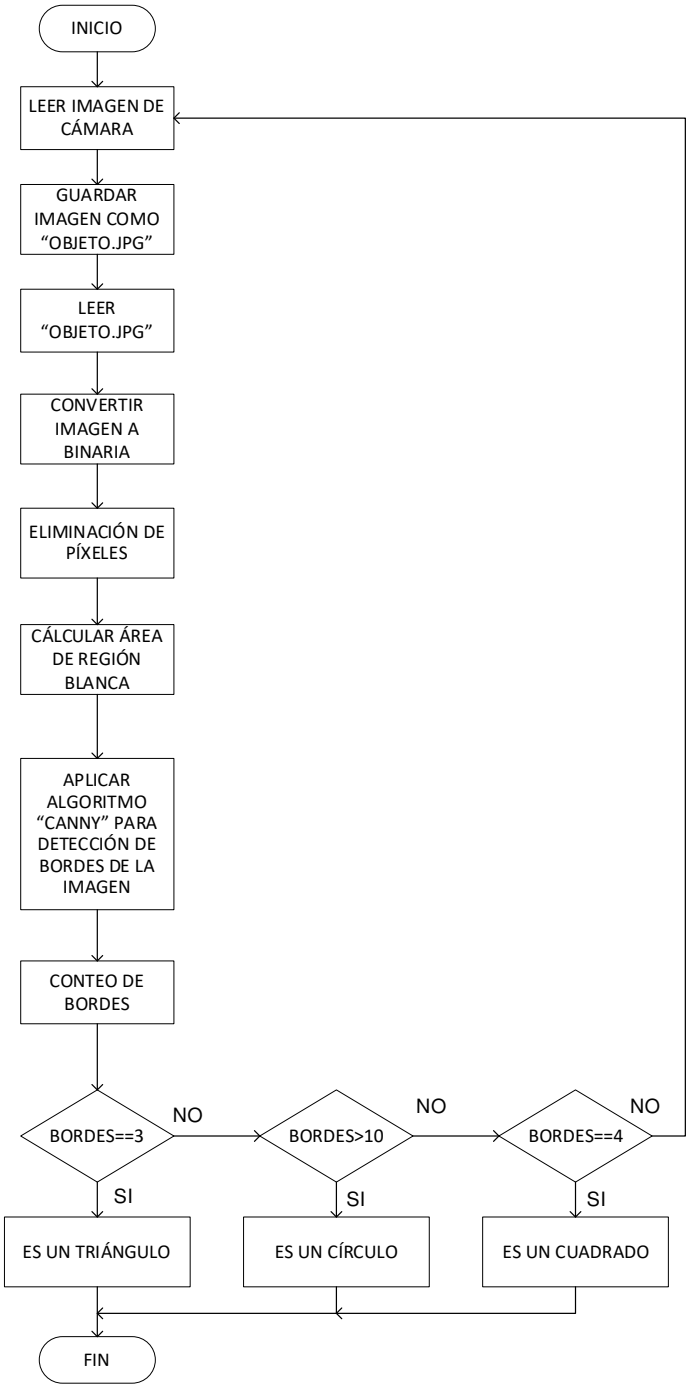


Figura 1.23 Diagrama de flujo para la clasificación por forma usando el método de CANNY [17].

En la Figura 1.23 se observa el diagrama correspondiente a la clasificación por formas usando el método de CANNY. Se puede observar todo el procedimiento incluido los números de bordes que son usados para determinar la forma del objeto.

Cabe indicar que, en la clasificación de objetos por la forma, el algoritmo de CANNY no es el único usado para la detección de contornos, existe una gran variedad de algoritmos que realizan esta tarea como los siguientes:

- *Detector de contorno Shen – Castan (ISEF, infinite symmetric exponential filter):* este detector se basa en el detector de CANNY con la diferencia que hace uso de un filtro óptimo con la finalidad de obtener un detector de contorno más preciso [30].
- *Detector de contorno por Segmentación de Secuencias:* Esta técnica hace uso de una matriz que representa a la imagen, consiste en analizar cada elemento de esta matriz buscando un punto de segmentación [31].
- *Fuzzy Wavelet Framework (FWF):* Técnica híbrida usada para entornos ruidosos con 5 etapas, en donde la lógica difusa es fundamental para el algoritmo [31].

Cada uno de estos usa un algoritmo complejo para la detección de contornos, pero esto no quiere decir que uno sea mejor que otro. Dado que en el presente proyecto los objetos a clasificar poseen formas básicas, es suficiente emplear el algoritmo de CANNY para desarrollar la tarea de clasificar objetos.

1.3.6. RECONOCIMIENTO DE OBJETOS

En el entorno de visión artificial el reconocimiento de objetos se define como la tarea de encontrar e identificar un objeto dentro de una imagen o video, es decir, esta tarea cumple la función de determinar la posición de un objeto en una imagen determinada de forma continua y fiable. El objetivo principal del reconocimiento de objetos es tomar la información no relevante para el ojo humano y compararla de alguna forma con un objeto preestablecido en el sistema [18]. Un sistema de reconocimiento de objetos se encuentra conformado de:

- Un sensor basado en visión.
- Un algoritmo de extracción y conversión a sistema binario.
- Un sistema clasificatorio basado en características predefinidas.

Los métodos de reconocimiento de objetos se encuentran divididos en tres grandes grupos:

- *Aproximación basada en modelos de objetos:* este método de reconocimiento hace uso de técnicas de detección, como por ejemplo la detección de bordes.
- *Métodos basados en apariencia:* Este método hace uso de imágenes precargadas en el sistema, es decir, una especie de base de datos con las cuales la comparación

se realiza con facilidad. Este método compara bordes, escala de grises o emparejamiento del gradiente entre otros.

- *Métodos basados en características:* este método usa el emparejamiento de las características tanto del objeto como de la imagen, estas características pueden ser parches de superficie, bordes lineales y esquinas.

En este capítulo se desarrolló el estudio teórico acerca del entorno ROS, el simulador Gazebo, el modelo de cinemática directa del robot Baxter, las características y capacidades de un robot Baxter y finalmente se explicaron brevemente los métodos usados en la clasificación de objetos por visión artificial. En el próximo capítulo se desarrollará la metodología con la que trabaja el sistema de clasificación de objetos.

2. METODOLOGÍA

Este capítulo se enfoca en detallar la metodología con la cual el proyecto de titulación se desarrolló, para lo cual se abordará el desarrollo tanto de la fase de diseño como de la fase de simulación. Respecto a la fase de diseño, se describirá específicamente cada uno de los algoritmos y programas diseñados en las diferentes plataformas, con el objetivo de presentar la metodología con la cual se trabajará en cada una de las secciones que representan al sistema clasificador de objetos, al finalizar este apartado se desarrollará un archivo de ejecución con el objetivo de facilitar la ejecución de varios archivos a la vez. Mientras que respecto a la fase de simulación se presentará tanto el sistema de control como el entorno de simulación con el objetivo de obtener resultados acordes al proyecto.

2.1. DESARROLLO DEL ENTORNO DE SIMULACIÓN PARA LA CLASIFICACIÓN DE OBJETOS

Como se ha indicado en el capítulo anterior, el entorno de simulación para la escena de clasificación planteada será Gazebo, este software permite tanto el diseño como la importación de modelos. Los modelos de este proyecto fueron desarrollados de diversas formas, por ejemplo, el modelo del robot Baxter fue desarrollado en Blender mientras que el modelo de las bandas transportadoras se desarrolló en Solidworks, cada uno de estos modelos se enlistan a continuación.

- Robot Baxter.
- Bandas Transportadoras.
- Caja.

- Mesa.
- Objetos a clasificar: Cubo Rectangular y Esfera.

2.1.1. ROBOT BAXTER

Como se menciona en la Sección 1.3.4.4, el paquete de simulación del robot Baxter es descargado del repositorio oficial de la empresa que lo fábrica Rethink Robotics. Este repositorio contiene los archivos de extensión .dae y .stl necesarios para su respectiva simulación [25].

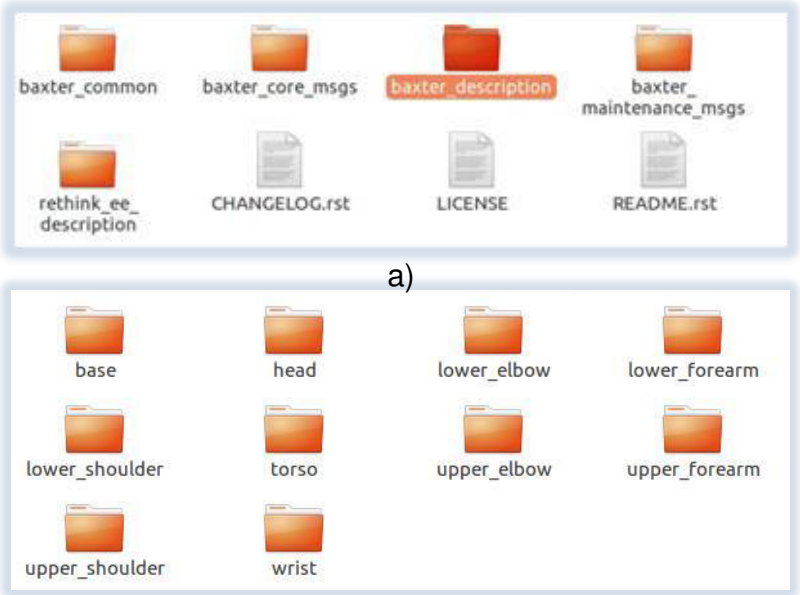
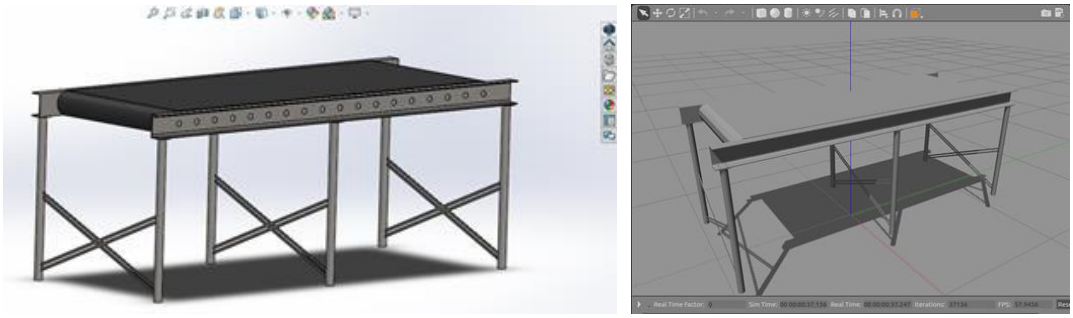


Figura 2.1 Paquete de simulación robot Baxter. a) Repositorio descargado. b) Carpetas de modelos que componen al robot Baxter.

En la Figura 2.1a se observa el repositorio descargado, mientras que en la Figura 2.1b se observa las carpetas que contienen los objetos que componen al robot Baxter. Estas carpetas se encuentran en la carpeta baxter_description de la Figura 2.1a.

2.1.2. BANDAS TRANSPORTADORAS

Como se mencionó en la Sección 1.3.4.1, a pesar de que Gazebo posee una librería de modelos, está no presenta un modelo de banda transportadora, por lo cual, se lo diseñó en SolidWorks. En la Figura 2.2a se puede observar el diseño implementado en este software.



a) b)

Figura 2.2 Banda transportadora. a) Modelo en SolidWorks. b) Modelo en Gazebo.

La banda transportadora posee una altura de 80 cm, una longitud de 1.5 metros y 80 cm de ancho. Estas medidas se acoplan al entorno de simulación en donde no interfieren en el rango de movimiento de las articulaciones del robot Baxter.

SolidWorks posee una herramienta de exportación denominada “Export as URDF”, herramienta que al ser ejecutada crea una carpeta en donde no solamente existe el archivo .urdf sino que adicionalmente crea archivos .stl y .launch. Para el caso de importar el modelo a Gazebo se hace uso únicamente del archivo .stl. En la Figura 2.2b se presenta el modelo importado a Gazebo. Adicionalmente, a cada banda transportadora se le incorpora un sensor de proximidad el cual posee un rango de detección de 0.05 y 0.2 m, finalmente la banda transportadora se visualiza en la Figura 2.3.

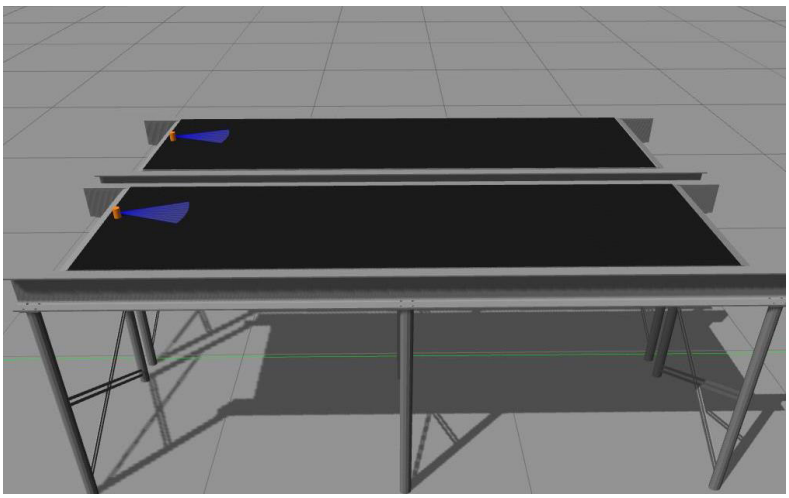
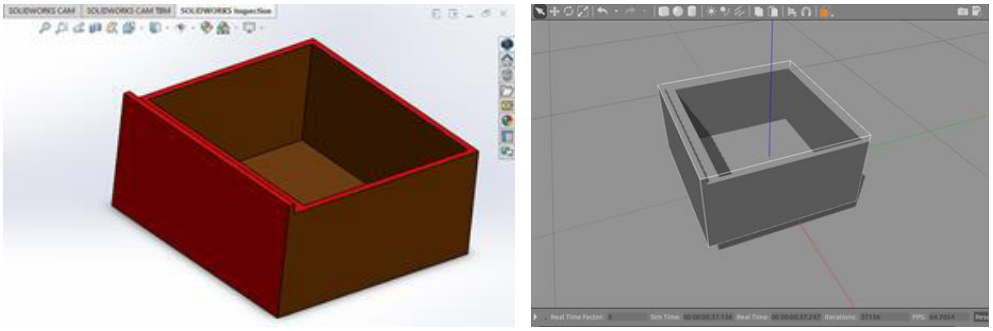


Figura 2.3 Banda Transportadora con sensor de presencia.

2.1.3. CAJA

El modelo de la caja es diseñado en SolidWorks y, de forma similar al anterior, su archivo de extensión .stl es obtenido mediante la herramienta denominada “Export as URDF”. Este

diseño se puede observar en la Figura 2.4a y en la Figura 2.4b se presenta el diseño importado en Gazebo. El proceso y contenido de exportación es similar al de la banda transportadora.



a) b)

Figura 2.4 Caja. a) Modelo en SolidWorks. b) Modelo en Gazebo.

La caja se diseña con una altura de 32 cm, ancho 69 cm y largo 69 cm, además cada cara posee 2 cm de espesor.

2.1.4. MESA

La mesa es un modelo propio de Gazebo por lo que no es necesario un diseño externo. En la Figura 2.5 se presenta la que es empleada y, como se indica, este modelo es tomado directamente de la librería de Gazebo.

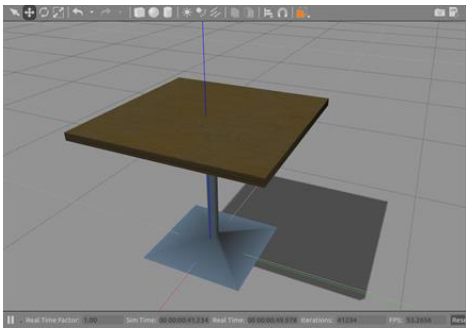


Figura 2.5 Mesa (librería de modelos Gazebo).

2.1.5. OBJETOS A CLASIFICAR

Los objetos a usar en el presente proyecto son geométricamente básicos, por lo cual su diseño se limita a líneas de programación en formato de un archivo .urdf. A continuación, se detalla la programación para cada objeto.

2.1.5.1. OBJETO 1: CUBO RECTANGULAR

Como se mencionó anteriormente el archivo de extensión .urdf contiene las características físicas y dinámicas del modelo diseñado. Dentro de estas características se encuentran las

dimensiones del objeto, su peso y los respectivos tensores de inercia. Respecto a este último, en base a las dimensiones del objeto es posible determinar los tensores de inercia. El termino tensor de inercia para un sólido rígido es un tensor de segundo orden que expresa el contenido de la inercia rotacional del cuerpo [32].

El modelo del cubo rectangular es diseñado con una altura de 0.02 m, un ancho de 0.02 m y una profundidad de 0.3 m. Además, se asume que el objeto tiene una masa de 0.01 kg.

Tabla 2.1 Parámetros físicos para el cubo rectangular.

Altura	Ancho	Profundidad	Masa
$h = 0.02\ [m]$	$w = 0.02\ [m]$	$d = 0.3\ [m]$	$m = 0.01\ [kg]$

Respecto al tensor de inercia se emplea el de un cubo rectangular, mismo que viene dado por la Ecuación (2.1) [32].

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} = \begin{bmatrix} \frac{1}{12}m(h^2+d^2) & 0 & 0 \\ 0 & \frac{1}{12}m(w^2+h^2) & 0 \\ 0 & 0 & \frac{1}{12}m(w^2+d^2) \end{bmatrix} \tag{2.1}$$

Estos valores deben ser obligatoriamente especificados en el código fuente que genera el sólido para evitar desplazamientos erróneos en la simulación. En la Figura 2.6 se muestra la simulación en Gazebo de los modelos de cubo rectangular, para el proyecto es necesario dos de estos objetos con las mismas características físicas y dinámicas, pero de diferente color, el cambio de este parámetro se efectúa en base a líneas de programación.

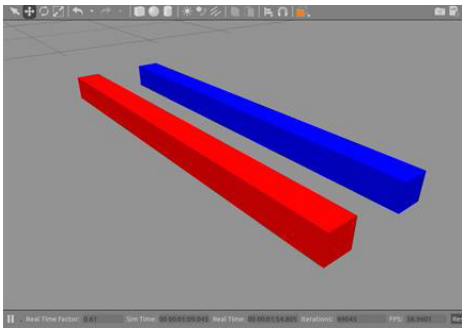


Figura 2.6 Cubos rectangulares en Gazebo.

Las líneas para la modificación de color es la siguiente:

<material>Gazebo/Red</material>

Esta línea de programación se encuentra situada al final del código, haciendo referencia al color del objeto, como se observa la modificación se desarrolla cambiando Red por colores básicos como son Yellow, Blue, Green o Purple, entre otros.

2.1.5.2. OBJETO 2: ESFERA

El diseño de la esfera es similar al cubo rectangular, en este caso se cambia las características físicas y dinámicas del objeto tomando en cuenta el radio de la esfera y el tipo de figura que se desea diseñar.

La Ecuación (2.2) sirve para calcular los tensores de inercia para una esfera [32] y para su cálculo se toma como referencia los datos de la Tabla 2.2.

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} = \begin{bmatrix} \frac{2}{5}mr^2 & 0 & 0 \\ 0 & \frac{2}{5}mr^2 & 0 \\ 0 & 0 & \frac{2}{5}mr^2 \end{bmatrix} \tag{2.2}$$

Tabla 2.2 Parámetros físicos para una esfera.

Radio	Masa
$r = 0.015\ [m]$	$m = 0.05\ [kg]$

En la Figura 2.7 se observa el modelo de la esfera implementado en Gazebo. Cabe resaltar que, para este proyecto, el color de la esfera únicamente será verde con el propósito de analizar los resultados de la clasificación cuando se presente un objeto con color y forma diferente.

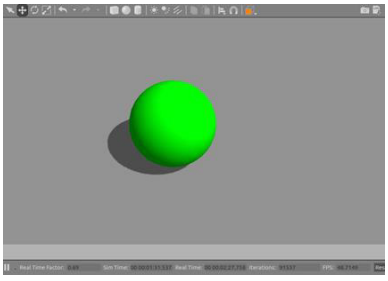


Figura 2.7 Esfera en Gazebo.

2.1.6. CONSTRUCCIÓN DE LA ESCENA

En esta sección se incorporarán todos los archivos diseñados, incluyendo el paquete de simulación del robot Baxter. Para este fin se debe tomar en cuenta la creación de 4 archivos de ejecución, 2 de los cuales corresponden al paquete de simulación del robot Baxter y estos son:

- baxter.world: Carga las características físicas y gravitacionales simulando un entorno real.
- baxter_world.launch: Es el archivo encargado de ejecutar el baxter.world y además ejecuta el paquete de simulación del robot Baxter.

Los archivos restantes son:

- mundo.py: Este archivo es el encargado de contener la programación en phyton de los modelos extras a cargar en la escena de simulación.
- mundo.launch: Este es el archivo principal, el cual desarrolla la ejecución del baxter_world.launch y mundo.py en conjunto, una vez ejecutado, en el entorno Gazebo se abre automáticamente y los modelos son cargados como se muestra en la Figura 2.8, donde se observa cada uno de los modelos importados para generar el escenario en el cual se desarrollará la clasificación de objetos con el robot Baxter.

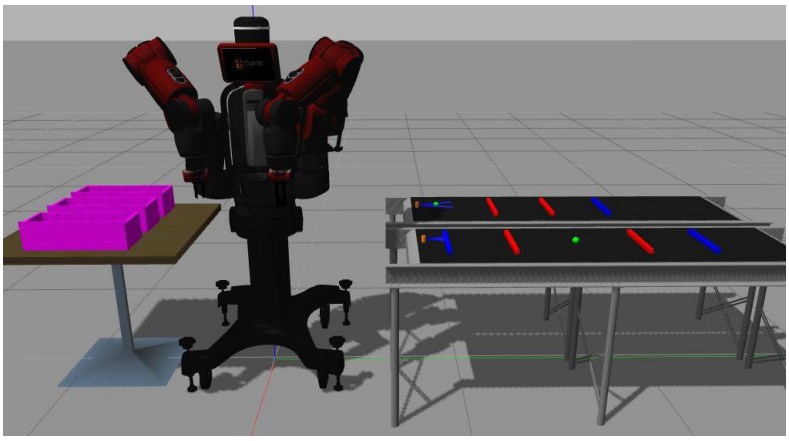


Figura 2.8 Modelos importados para la escena de simulación en Gazebo.

2.2. DESARROLLO DE LA INTERACCIÓN CON EL ROBOT BAXTER

Como se explicó anteriormente, el robot Baxter posee varias formas de programación, es decir, sus movimientos pueden ser comandados por medio de scripts, por medio del teclado e incluso por medio de una palanca de X-BOX [13]. El método seleccionado para el desarrollo del clasificador de objetos es la programación por script de Python, esto debido a la complejidad del algoritmo a construir. No obstante, el uso de la cinemática directa del robot Baxter requiere que se conozcan el ángulo de giro de cada bend joint y twist joint que se desea alcanzar en cada una de las posiciones de los brazos del Baxter, es decir, para las posiciones de toma, colocación y clasificación. Para conocer cada uno de los ángulos de giro de las articulaciones que debe alcanzar cada brazo en una respectiva posición se puede interactuar con el robot Baxter mediante un dispositivo externo, que en este caso

será el teclado de la computadora. Esta interacción consiste en llevar cada brazo del robot Baxter a la posición deseada y enlistar cada ángulo de giro de las articulaciones, ángulos con los cuales se construirá el script final para el proceso planteado. Este procedimiento se lleva a cabo tomando en cuenta los ejemplos suministrados en el paquete ROS del robot Baxter [38].

2.2.1. MOVIMIENTOS POR TECLADO

Como se mencionó anteriormente, es necesario conocer el ángulo de giro de cada articulación en cada una de las posiciones de toma, colocación y clasificación, mismos que serán obtenidos luego de mover cada articulación por medio del teclado de la computadora llevando cada brazo del Baxter a las posiciones mencionadas. Los movimientos por teclado ayudan en el incremento o decremento del ángulo de giro de cada una de las articulaciones, por lo cual es necesario construir un script con el cual al ser presionada una tecla se provoca el movimiento de una de las articulaciones de los brazos del robot Baxter.

Una de las facilidades que presenta el paquete ROS del robot Baxter, es la capacidad de contener un script de ejemplo para la interacción de movimientos por teclado. Este ejemplo se encuentra en el paquete denominado baxter_examples con el nombre joint_position_keyboard. Este archivo está escrito en lenguaje Python el cual en su interior posee líneas de programación ROS, con las cuales la inicialización de nodos, el envío y recepción de mensajes se puede desarrollar exitosamente.

2.2.1.1. Descripción del script de movimiento

El script de movimiento denominado *"joint_position_keyboard"* tiene como objetivo principal la interacción entre el ambiente de simulación del robot Baxter y el teclado de la computadora. Esto sirve para colocar a cada una de las articulaciones en las posiciones de toma, colocación y clasificación. Una vez obtenidas las posiciones antes mencionadas se procede a generar el script de movimiento para el proceso planteado.

Al ser presionada cierta tecla una de las articulaciones asignadas produce un movimiento. La asignación de teclas para cada articulación se presenta en la Tabla 2.3 y cada uno de los movimientos posee un paso de 0.1 radianes. Este paso puede ser cambiado en su respectiva línea de código, en el presente proyecto se realizó con un paso de 0.01 radianes debido a que la exactitud es fundamental para cada posición de los brazos del Baxter.

Tabla 2.3 Tabla de asignación de ángulos para cada tecla.

Brazo Izquierdo			**Brazo Derecho**		
Tecla	**Articulación**	**Acción**	**Tecla**	**Articulación**	**Acción**
9	S0	Incremento	4	S0	Incremento
6	S0	Decremento	1	S0	Decremento
8	S1	Incremento	3	S1	Incremento
7	S1	Decremento	2	S1	Decremento
o	E0	Incremento	r	E0	Incremento
y	E0	Decremento	q	E0	Decremento
i	E1	Incremento	e	E1	Incremento
u	E1	Decremento	w	E1	Decremento
l	W0	Incremento	f	W0	Incremento
h	W0	Decremento	a	W0	Decremento
k	W1	Incremento	d	W1	Incremento
j	W1	Decremento	s	W1	Decremento
.	W2	Incremento	v	W2	Incremento
n	W2	Decremento	z	W2	Decremento
,	Gripper Close	Open	c	Gripper Close	Open
m	Griper Open	Close	x	Griper Open	Close
/	Gripper Calibrate	Calibrate	b	Gripper Calibrate	Calibrate

En la Figura 2.9 se observa el diagrama de flujo que representa el algoritmo de movimiento por teclado.

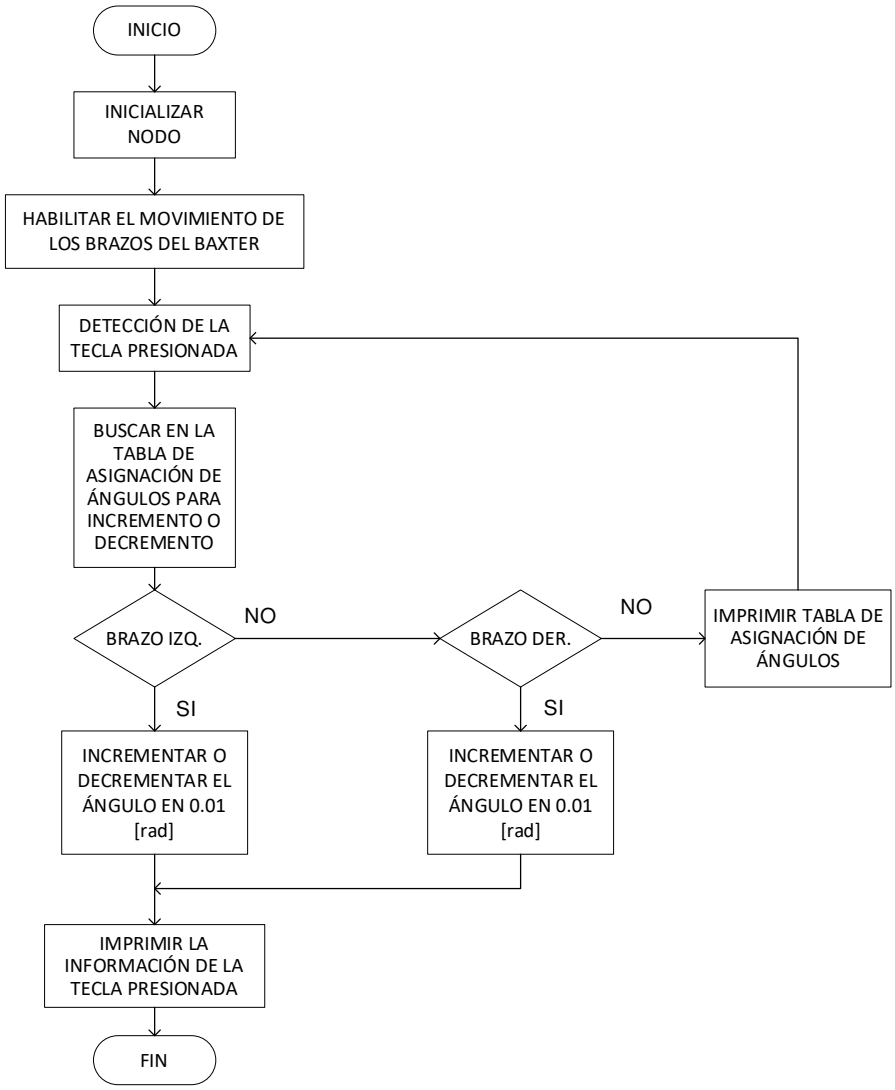


Figura 2.9 Diagrama de flujo para el algoritmo de movimientos por teclado.

El algoritmo consiste en:

- Inicializar o abrir un nodo para la comunicación entre el teclado y el robot Baxter.
- Habilita las extremidades del robot Baxter para su libre movimiento.
- Detecta la tecla que se presiona.
- Ejecuta el movimiento de la articulación con un paso de 0.1 radianes.
- Espera para la siguiente interacción con el teclado.

La finalización del nodo o cierre del programa se ejecuta mediante la tecla ESC o presionando Crtl+C, adicionalmente se deshabilitan las extremidades del robot Baxter.

2.2.1.2. Ejecución del Script de Movimiento

La ejecución de este script se realiza mediante la ejecución del siguiente comando en la ventana Terminal.

rosrun baxter_examples joint_position_keyboard.py

El comando *rosrun* permite la ejecución de un archivo ejecutable sin la necesidad de encontrarse en el directorio del archivo, es decir, sin ubicarse en la carpeta scripts del paquete baxter_examples. En la Figura 2.10 se presenta el mensaje de ejecución del script de movimiento, en esta ventana se observa la inicialización del nodo y la habilitación de movimientos para el robot Baxter.

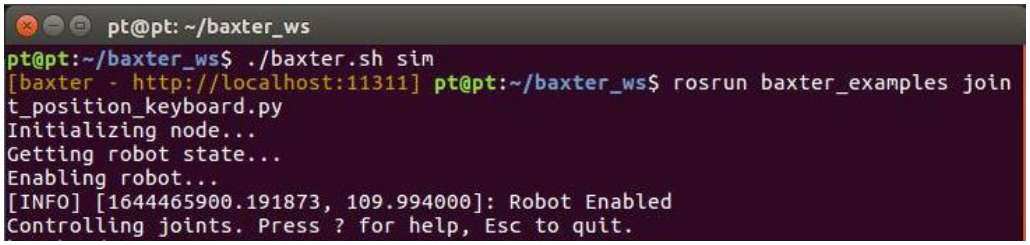
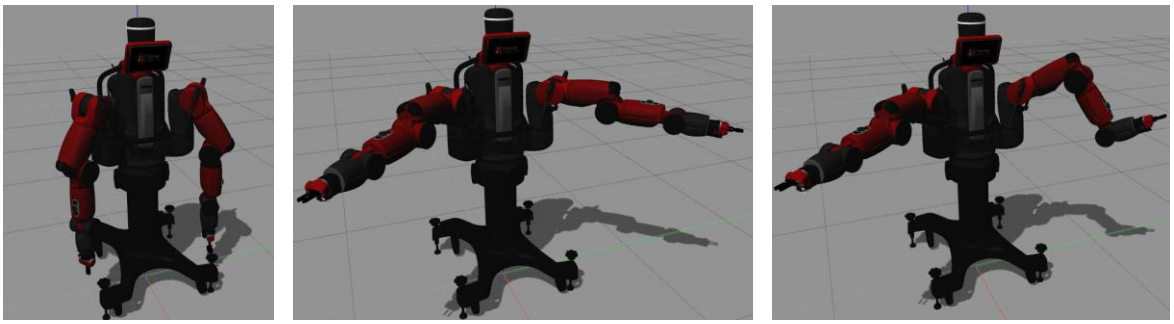


Figura 2.10 Mensaje en la ventana Terminal de la ejecución del script de movimientos por teclado.

En la Figura 2.11a se presenta el estado inicial del robot Baxter, mientras que en la Figura 2.11b y 2.11c se presenta el movimiento de las extremidades al ser presionadas diversas teclas descritas en la Tabla 2.3.



a) b) c)

Figura 2.11 Estados del robot Baxter. a) Posición por default b) Posición con ángulos seteados a cero c) Posición accionando solo tres articulaciones.

Para la Figura 2.11b se presenta la Tabla 2.4 en la cual se observa el número de veces que se presionó las respectivas teclas para colocar en cero a cada ángulo de giro de las articulaciones.

Tabla 2.4 Número de interacciones por tecla para cambiar la posición de los brazos del Baxter, desde la posición inicial hasta la posición de ángulos cero.

Izquierda				Derecha			
Int	**Art**	**Ang. Inicial**	**Ang. Final**	**Int**	**Art**	**Ang. Inicial**	**Ang. Final**
0	E0	0.0	0.0	0	E0	0.0	0.0
50	E1	0.5	0.0	50	E1	0.5	0.0
20	S0	0.2	0.0	30	S0	-0.3	0.0
104	S1	1.04	0.0	105	S1	1.05	0.0
50	W0	-0.5	0.0	40	W0	-0.4	0.0
4	W1	-0.04	0.0	4	W1	-0.04	0.0
5	W2	-0.05	0.0	3	W2	-0.03	0.0

En la Figura 2.11c se observa el cambio únicamente para el brazo izquierdo del robot Baxter, en este caso se realiza el cambio de las articulaciones S1, W1 y E1, el cambio de estos ángulos se presenta en la Tabla 2.5.

Tabla 2.5 Interacciones para el cambio de ángulo de giro en tres articulaciones.

Int	**Art**	**Ang. Inicial**	**Ang. Final**
50	S1	0.0	-0.5
150	E1	0.0	1.5
150	W1	0.0	-1.5

2.3. DESARROLLO DEL ALGORITMO DE CONTROL PARA LA TOMA Y COLOCACIÓN DE OBJETOS

Como se menciona en la sección anterior, el algoritmo de control para la toma y colocación de objetos se desarrolla en base a los ángulos de giro de cada una de las articulaciones de cada brazo, esto se debe a que se está usando la cinemática directa del robot Baxter.

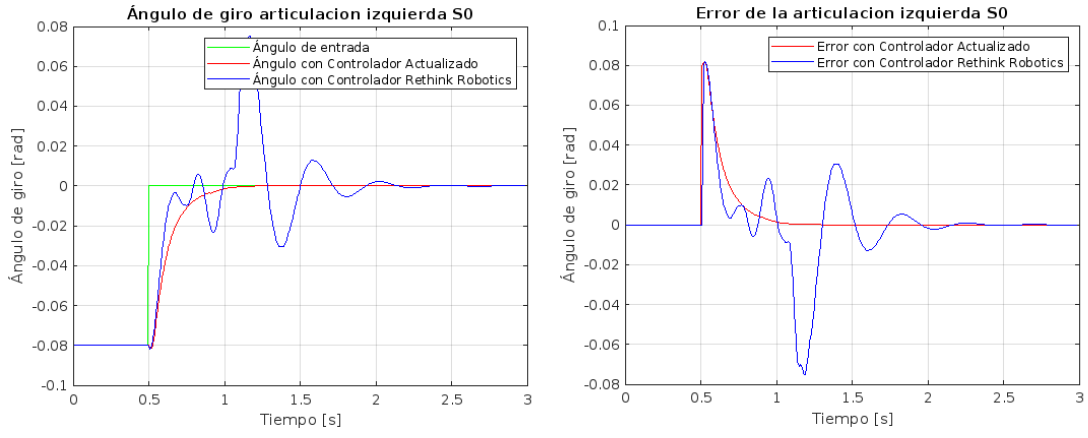
Este algoritmo de control se basa en el uso del filtro pasa bajos y el controlador de tipo PID mencionado en la Sección 1.3.2.4, estos se encuentran incorporados en el paquete ROS desarrollado por la empresa Rethink Robotics. A continuación, en la Tabla 2.6 se presenta los valores de las constantes del controlador PID para las cuales el control de posición de las articulaciones trabaja de manera adecuada sin producir movimientos bruscos cuando se mueven las extremidades para alcanzar una posición determinada. Cabe recalcar que los valores que se presentan en la Tabla 2.6 son tanto para el brazo derecho como para el

izquierdo debido a la similitud de los mismos y son obtenidos de forma empírica, es decir, se sintonizó el controlador de cada articulación por medio del método de prueba y error.

Tabla 2.6 Valores de las constantes para el controlador PID de los brazos del robot Baxter (Derecha/Izquierda)

Valores de las constantes kp, ki y kd			
Articulación	kp	ki	kd
S0	7000	0.01	700
S1	10000	0.01	700
E0	4500	0.35	100
E1	5500	0.60	200
W0	1000	0.30	100
W1	9000	0.1	100
W2	1000	0.1	100

Los valores de la Tabla 2.6 son valores sintonizados debido a que los valores del controlador original, descritos en la Tabla 1.8 de la Sección 1.3.2.4, producen oscilaciones en la respuesta, por lo que estos valores ajustados son con los que se trabaja en el proyecto. En la Figura 2.12a se puede observar la respuesta de la articulación S0 con los valores descritos en la Tabla 1.8 (Rethink Robotics) y la Tabla 2.6 (Actualizado), mientras que en la Figura 2.12b se observa el error de la articulación S0 empleando los dos controladores.



a) b)

Figura 2.12 Respuesta de la articulación S0 del brazo izquierdo del robot Baxter. a) Ángulo de giro. b) Error en el sistema.

La Figura 2.12 contiene graficas obtenidas por el tópico "*set_point, process_value y errro*", con ayuda del comando "*rostopic echo"* se obtuvieron los valores que se envían cuando el

control de posición se está ejecutando, valores visualizados con ayuda de MatLab, en la Figura 2.12a se observa que ambas respuestas pasan de una posición de -0,08 [rads] a una posición de 0 [rads] aunque la respuesta empleando el controlador de Rethink Robotics posee mayor oscilación antes de estabilizarse. En la Figura 2.12b se observa la señal del error que se genera cuando se aplica el lazo de realimentación, de igual forma se observa oscilaciones en el error provocado por el controlador de Rethink Robotics.

Para determinar si el controlador sintonizado es mejor que el controlador desarrollado por la compañía Rethink Robotics es necesario calcular cifras de desempeño, en este caso se calcula el índice de rendimiento denominado ISE, este índice es igual a la integral de la señal del error al cuadrado como se muestra en la ecuación (2.3), entre menos valor posea dicho índice mejor será el controlador implementado.

$$ISE = \int e^2(t)\, dt \tag{2.3}$$

En la Tabla 2.7 se muestra el resultado después de aplicar la ecuación 2.7 a la señal del error de cada uno de los controladores.

Tabla 2.7 Índice de desempeño ISE

Indice	Controlador Actualizado	Controlador Rethink Robotics
ISE	0.0005804	0.001201

Con los datos presentados en la Tabla 2.7 se determina que un mejor desempeño posee el controlador actualizado cuyo ISE es de 0.0005804, este controlador será el que se usará durante el presente proyecto.

En esta sección se construye un algoritmo el cual ejecuta una serie de posiciones y mediante el controlador PID cada una de estas posiciones es alcanzada con rapidez y efectividad. Además, se hace uso del filtro pasa bajos para las posiciones que necesitan movimientos suaves como es el caso de la toma de objetos. El algoritmo de control envía un ángulo de giro al robot Baxter virtual y por medio del controlador de tipo PID el valor del ángulo de giro seteado es alcanzado. La salida de este lazo de control puede ser observada por medio del comando "*rqt_plot*" ingresado en una ventana terminal, además, visualmente en la plataforma de Gazebo se observa el movimiento de la articulación a la cual se le está aplicando dicho control. En la Figura 2.13a se puede observar el diagrama de flujo del algoritmo de movimiento para cada una de las posiciones.

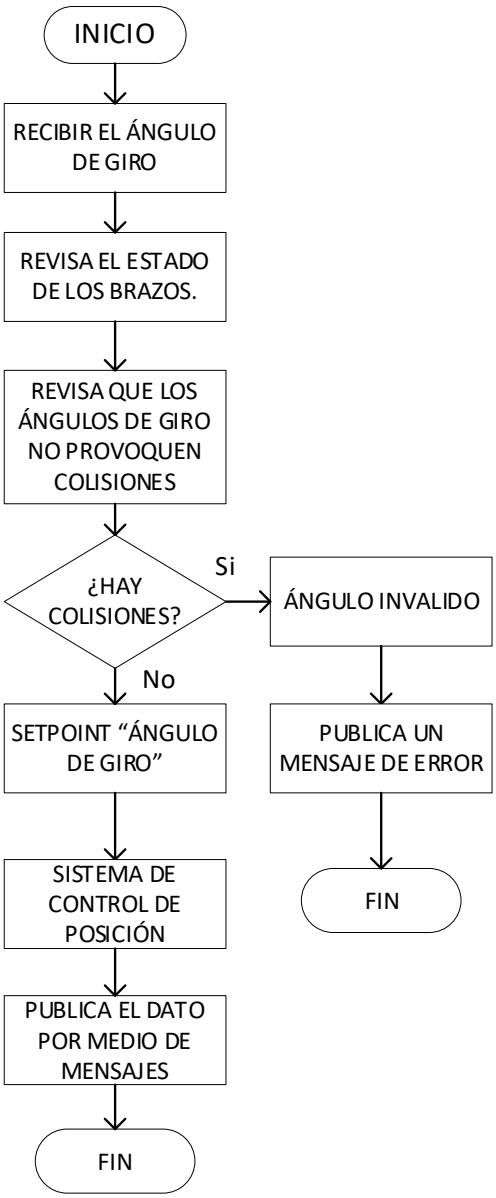


Figura 2.13 Diagrama de flujo del algoritmo de control.

Como se puede observar el algoritmo de control no solamente ejecuta el control de posición cada articulación, sino que también revisa el estado del robot y ejecuta un algoritmo de detección de colisiones para que los brazos articulados del Baxter no se superpongan entre si y tampoco se superpongan con los demás modelos presentes en la simulación.

La ejecución de la secuencia de posiciones se desarrolla en base a los ángulos de giro de las articulaciones obtenidos luego de ejecutar el script de movimiento por teclado. Para obtener cada una de las posiciones, el script de movimiento es ejecutado sobre la escena desarrollada en la Sección 2.1, es decir, las posiciones de toma, colocación y clasificación

son obtenidas con la escena real del proceso. Una vez obtenidas dichas posiciones se crea un script, el cual ejecuta esa secuencia de ángulos de giro, en cada ejecución de posición se emplea el algoritmo de control, con el cual los movimientos se generan automáticamente alcanzando los ángulos ingresados en cada posición.

El algoritmo debe tomar en cuenta la posición inicial de los brazos del robot Baxter, a esta posición se la conoce como posición de destrabe, inicialmente al ser colocado el robot Baxter en el ambiente Gazebo este se presenta como una posición de default que puede ser observada en la Figura 2.11a. La posición de destrabe consiste en asignarle ángulos a cada articulación con el objetivo de colocarlos en una determina posición antes de comenzar con el proceso, estos ángulos pueden ser cambiados en base a programación en el script “tuck_arm”, ubicado en la carpeta scripts. En el presente proyecto se consideran los ángulos dados por el paquete ROS para la posición de destrabe, en la Figura 2.14 se observa esta posición.

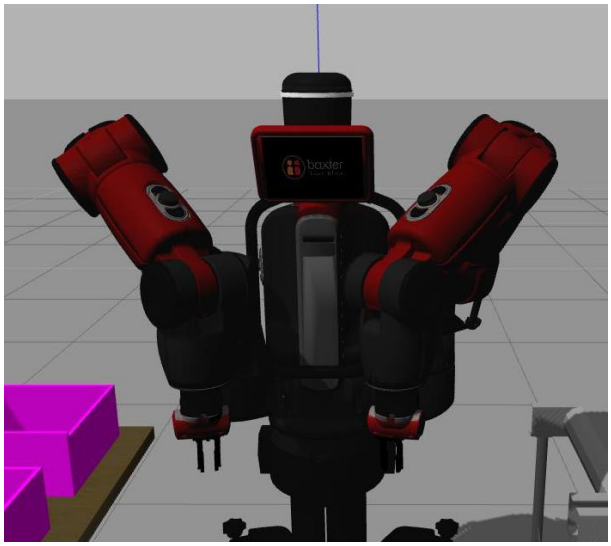


Figura 2.14 Posición de destrabe de los brazos del robot Baxter.

2.3.1. TOMA DE OBJETOS

El proceso comienza tomando en cuenta que el objeto posee una posición específica en la banda transportadora, es decir, se encuentra estático sobre la banda transportadora. Para que el brazo izquierdo del robot Baxter llegue a esta posición para tomar el objeto, los ángulos de giro previamente obtenidos se cargan dentro de un script que al ser ejecutado el código el brazo izquierdo del Baxter pasa de la posición de destrabe a la posición de toma de objetos de forma automática. Estas posiciones se componen de ángulos de giro, los cuales al ser seteado ingresan al control de posición de cada uno de ellos y como resultado la articulación se mueve hasta alcanzar el ángulo seteado.

Adicionalmente, el valor de cada ángulo de giro para cada una de las posiciones de toma, colocación y clasificación pueden ser obtenido de dos maneras.

2.3.1.1. Primera Forma

En el software Gazebo existe una lista de los modelos usados en la escena, se accede a World/Modelos/Baxter/Articulación/ángulo, en esta sección se muestra el valor del ángulo de giro de la articulación seleccionada. En la Figura 2.15 se presenta el ángulo para la articulación S1 del brazo izquierdo.

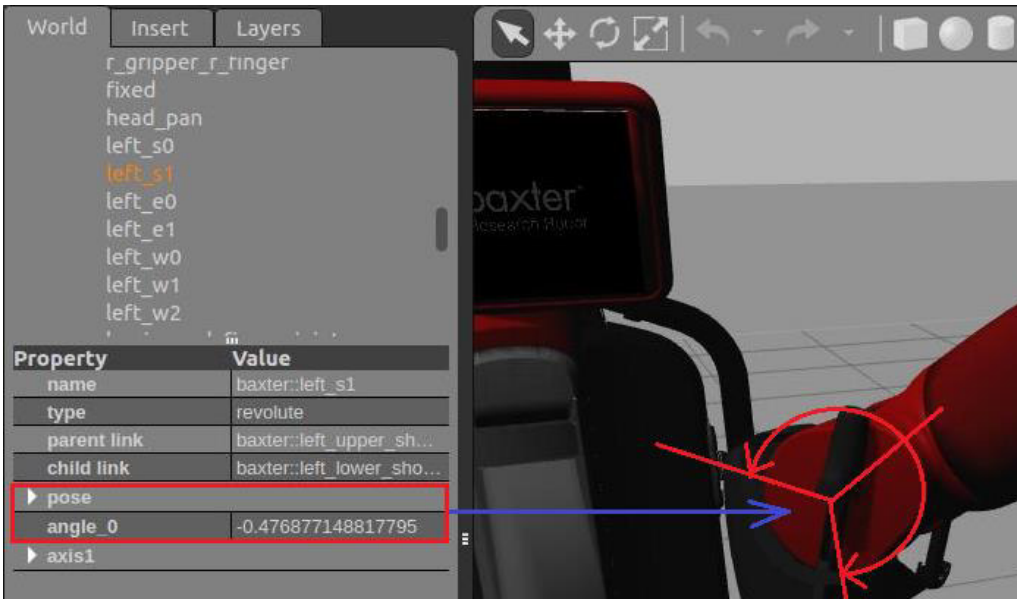


Figura 2.15 Ángulo de una articulación por Gazebo

2.3.1.2. Segunda Forma

Como se lo ha mencionado anteriormente, el paquete de Baxter posee varios ejemplos con los cuales la interacción con Baxter es más sencilla, existe un ejemplo denominado *joint_states* el cual se puede ejecutar mediante el siguiente comando en la ventana Terminal.

rostopic echo/robot/joint_states

Este comando muestra los estados de las articulaciones de cada brazo del robot Baxter, en la Figura 2.16 se puede observar la ejecución de este comando y su resultado.

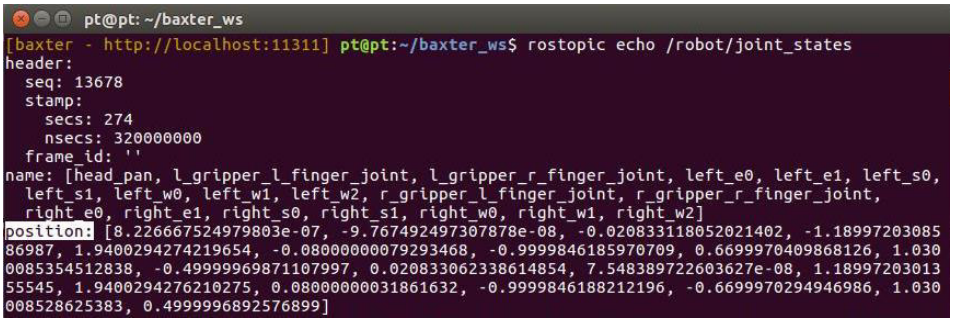


Figura 2.16 Información de los ángulos actuales de las articulaciones del robot Baxter.

Para esta sección se hará uso de la Segunda Forma para conocer los estados de las articulaciones por mayor facilidad, mientras que, se considerarán las dos formas en el capítulo de resultados.

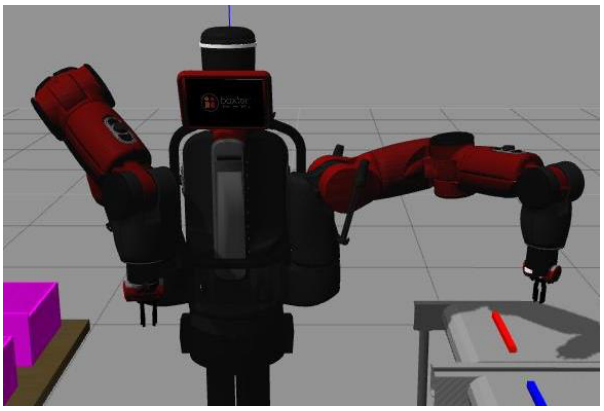


Figura 2.17 Posición para la toma de los objetos de la banda.

En la Figura 2.17 se observa la posición del brazo izquierdo del robot Baxter para la toma de objetos, mientras que en la Tabla 2.8 se observan los ángulos de giro para la posición toma de objetos.

Tabla 2.8 Ángulos de giro para la posición Toma de Objetos.

Articulación	E0	E1	S0	S1	W0	W1	W2
Angulo de giro	-1.6	1.2	1.15	-0.1	1.5	1.7	-1.8

Finalmente, estos ángulos obtenidos son cargados en el script de movimientos para su ejecución automática.

2.3.2. POSICIÓN DE AGARRE

La posición de agarre de objetos se desarrolla mediante el uso del filtro pasa bajos, con el objetivo de obtener movimientos lentos y suaves para que el gripper de agarre tome el objeto sin un contacto previo para evitar que el objeto se mueva de su posición. En la Figura 2.18 se observa varias posiciones en donde se agarra un objeto dentro de la aplicación.

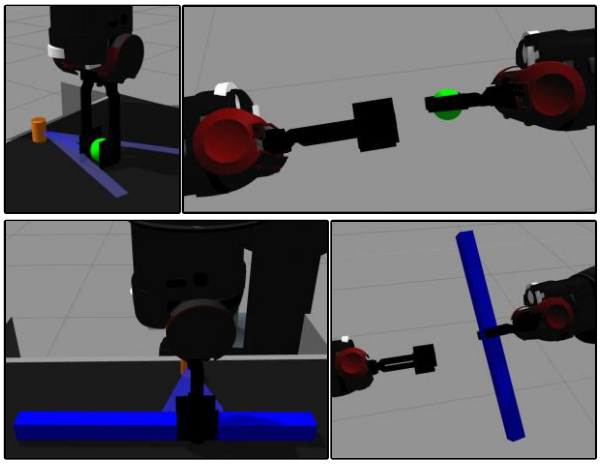


Figura 2.18 Posiciones de agarre para objetos dentro de la aplicación.

En la Figura 2.19a y 2.19b se presenta la vista superior y lateral del tamaño del gripper y el tamaño del objeto rectangular cúbico a recoger, para la validación de que el robot Baxter virtual pueda recoger el objeto se toma en cuenta el centro de masa del mismo y 6 líneas de agarre sobre el objeto como se muestra en la Figura 2.19 c).

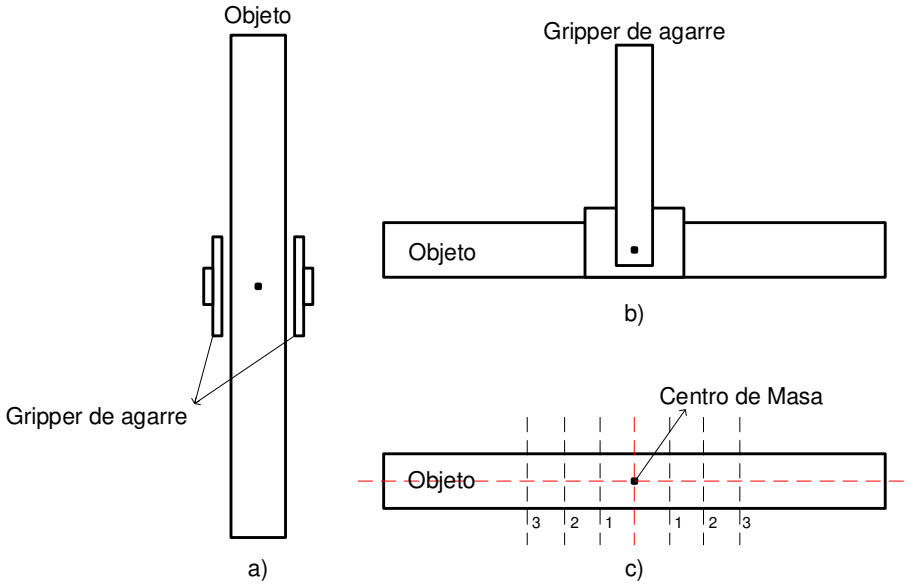


Figura 2.19 Gripper de agarre – Objeto cúbico rectangular. a) Vista Superior. b) Vista lateral. c) Centro de agarre y líneas de agarre del objeto.

En la Figura 2.19 c) se presentan las líneas de agarre para un objeto de forma cúbica rectangular, el espacio entre líneas es de 2 cm al igual que el espacio entre el centro de masa y la línea 1, esto ayudará en la Sección 3.2 a determinar el desfase para el agarre de un objeto.

2.3.3. COLOCACIÓN DE OBJETOS

La colocación de objetos se desarrolla de igual manera que en la toma de objetos, en la Figura 2.20 se observa el punto en el cual el brazo derecho del robot Baxter coloca el objeto en la caja seleccionada.

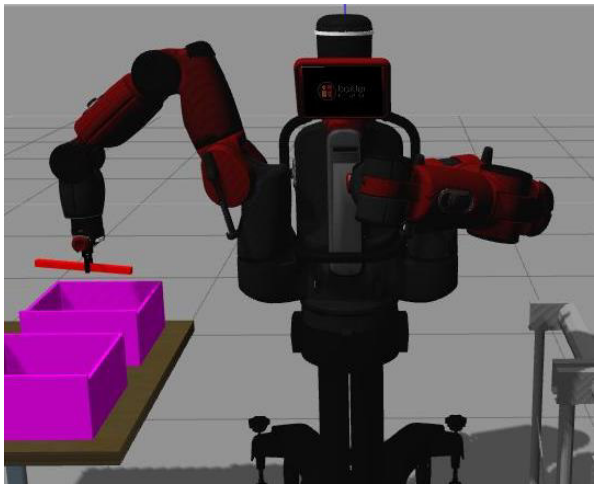


Figura 2.20 Posición para colocar los objetos de la caja correspondiente.

En la Tabla 2.9 se presentan los ángulos de giro para la posición colocación de objetos, estos ángulos son cargados en el script de control de movimiento.

Tabla 2.9 Ángulos de giro para la posición Colocación de Objetos

Articulación	E0	E1	S0	S1	W0	W1	W2
Angulo de giro	1.69	1.72	-1.22	-1.25	-0.4	1.45	1.97

Una vez obtenidos todos los ángulos de giro de cada una de las articulaciones del robot Baxter, se procede a cargar cada dato en el script de movimientos, el cual al ser ejecutado procede a mover cada brazo automáticamente a cada una de las posiciones aplicando el control de posición de cada articulación y el filtro pasa bajos con la finalidad de que realice el proceso planteado en el proyecto

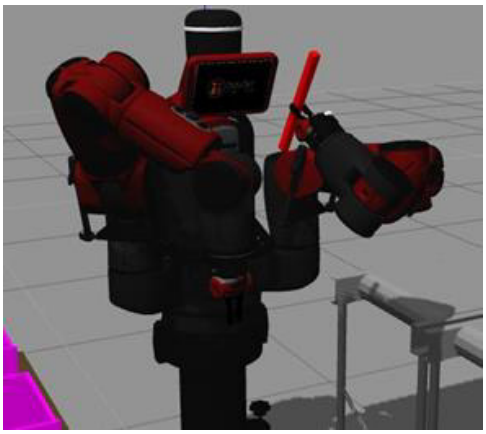
2.4. DESARROLLO DEL ALGORITMO DE CLASIFICACIÓN DE OBJETOS

El algoritmo de clasificación de objetos tiene como objetivo principal identificar el color y la forma del objeto. En el caso de los colores, se desea detectar el color rojo, azul y verde, para el caso de la forma es necesario identificar entre un cubo rectangular y una esfera.

En el desarrollo del algoritmo de clasificación se hace uso de la cámara principal del robot Baxter denominada como *head_camera*. Esta cumple su función como sensor de visión principal como se menciona en la Sección 1.3.6, además es necesario abrir un nodo con

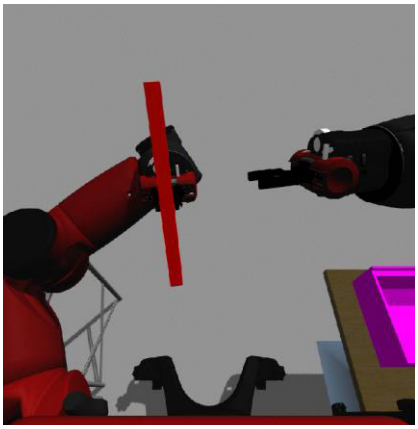
el cual se logra la recepción de la imagen que registra la cámara. Una vez obtenida la imagen se aplican los diferentes algoritmos explicados en las Figuras 1.17 y 1.18.

El uso de la cámara principal del robot Baxter limita al usuario a generar un punto en donde el robot Baxter pueda analizar el objeto tomado por el gripper correspondiente, para este caso se genera un punto en donde el brazo izquierdo del Baxter presenta el objeto a la cámara. Este punto es denominado como posición de clasificación, posición que se presenta en la Figura 2.21, adicionalmente la cámara se encuentra ubicada en la parte superior del Baxter representada en el rectángulo de color rojo como se muestra en la figura denominada “Vista desde el objeto”.



Vista General

Vista desde el Objeto



Vista desde la Cámara principal

Figura 2.21 Posición de clasificación del robot Baxter.

Una de las ventajas de trabajar con tipos de programación de alto nivel es la existencia de librerías que facilitan el uso de programación compleja como es el caso de la librería de OpenCV para Python y para ROS. Esta es una de las librerías comúnmente usada en aplicaciones de visión artificial, en el proyecto el uso de esta librería es fundamental para el procesamiento de imágenes adquiridas desde Gazebo.

2.4.1. ADQUISICIÓN DE LA IMAGEN A PROCESAR

La adquisición de imagen se centra en recibir los datos que producen la cámara principal del robot Baxter, para acceder a estos datos es necesario usar la comunicación publisher/suscriber. En el caso del presente proyecto, la suscripción implementada se la conoce como nodo secuencia, este método se basa en conectarse al tópico y consecuentemente capturar la imagen por medio de un nodo. El tópico al cual se suscribe el script de Python se denomina */cameras/head_camera/image* y el nodo por el cual se realiza la comunicación se denomina *initial_state_from_cameras.* Cabe recalcar que el mensaje tanto para el publicador como para el suscriptor debe ser de tipo *Image,* de no ser el caso la recepción de la imagen no se efectúa.

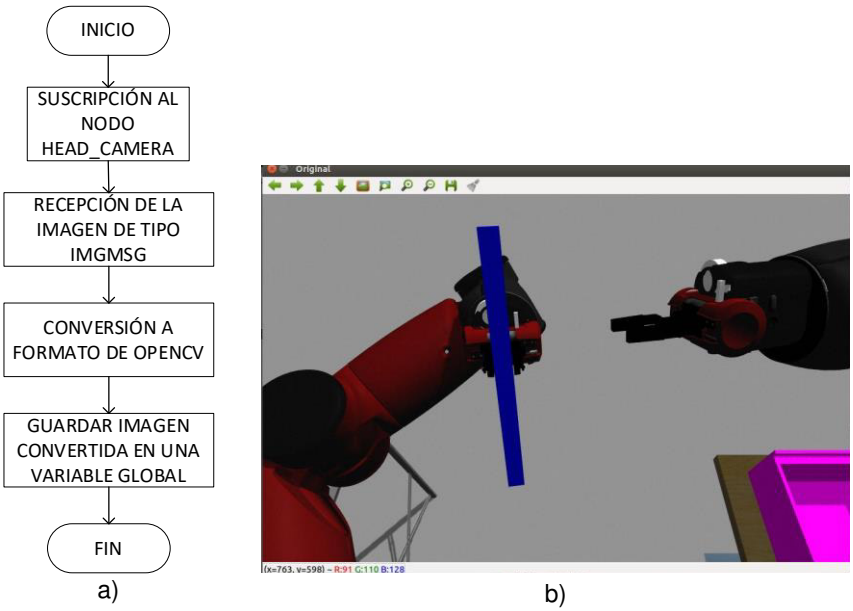


Figura 2.21 Adquisición de imagen. a) Diagrama de flujo. b) Imagen recibida.

En la Figura 2.22a se observa el diagrama de flujo que se implementa para el proceso de adquisición de imagen, una particularidad de esta adquisición de datos es que posee una taza de adquisición de 10 Hz con el objetivo de minimizar el flujo de datos y no sobrecargar al sistema. En la Figura 2.22b se puede observar la imagen recibida desde la *head_cámara* del robot Baxter situado en el entorno Gazebo.

2.4.2. DETECCIÓN DE COLOR

Una vez adquirida la imagen de la cámara del robot Baxter se procede a realizar un procesamiento de imágenes con el objetivo de determinar el color del objeto ubicado en el gripper del robot Baxter. Para realizar esta sección se deben considerar lo colores tanto de los objetos como del robot Baxter debido a que parte de sus extremidades poseen una tonalidad de color rojo. Una de las partes principales de este algoritmo es definir los rangos

en los cuales se encuentran los colores a detectar, OpenCV hace uso de código HSV (Hue, Saturation, Value), el cual es una transformación lineal del código RGB (Rojo, Verde, Azul); por lo que es necesario determinar cada uno de estos rangos con este tipo de código.

En la Tabla 2.10 se muestran los límites tanto inferiores como superiores para cada color en código HSV, estos rangos son los usados en el presente proyecto.

Tabla 2.10 Limites HSV para colores Rojo, Verde y Azul.

Color	Rojo (HSV)	Azul (HSV)	Verde (HSV)
Límite superior	(0, 255, 255)	(130, 255, 255)	(70, 255, 255)
Límite inferior	(0, 52, 115)	(110, 50, 50)	(50, 100, 50)

En esta sección se considera un desafío el poder diferenciar entre la tonalidad de color rojo del robot Baxter y la tonalidad del color rojo del objeto. Para este caso se toma en cuenta la construcción de una imagen secundaria en la escala de grises la cual se superpone sobre la imagen original con la finalidad de resaltar aún más la forma del objeto, en la Figura 2.23 se observa el procesamiento de la imagen recibida desde su imagen original hasta la imagen resultante luego de aplicar este filtro sobre la imagen HSV.

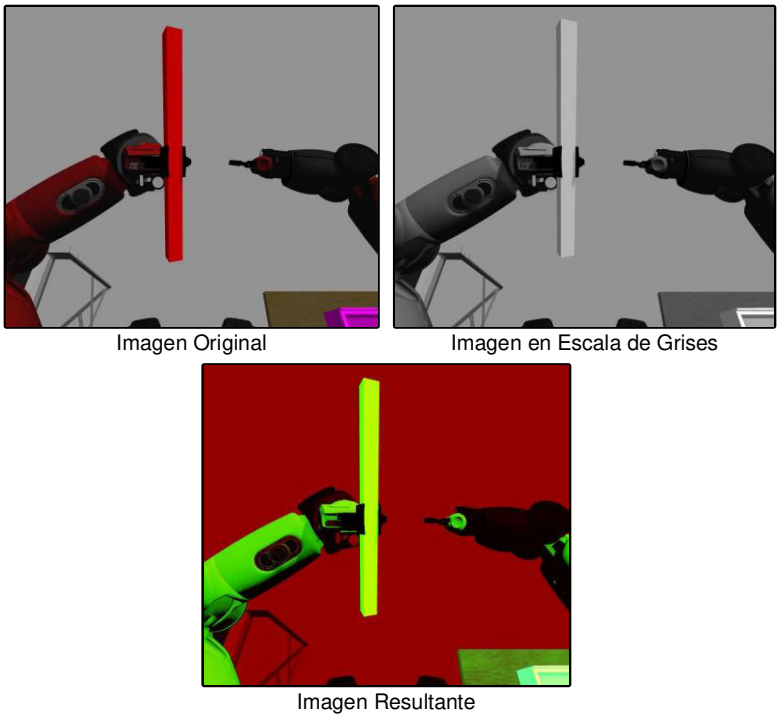


Figura 2.23 Filtro para el color rojo aplicado en la imagen HSV.

Una vez obtenida la imagen filtrada de la Figura 2.23 se construye una imagen binaria, en donde el color a detectar genera zonas blancas.

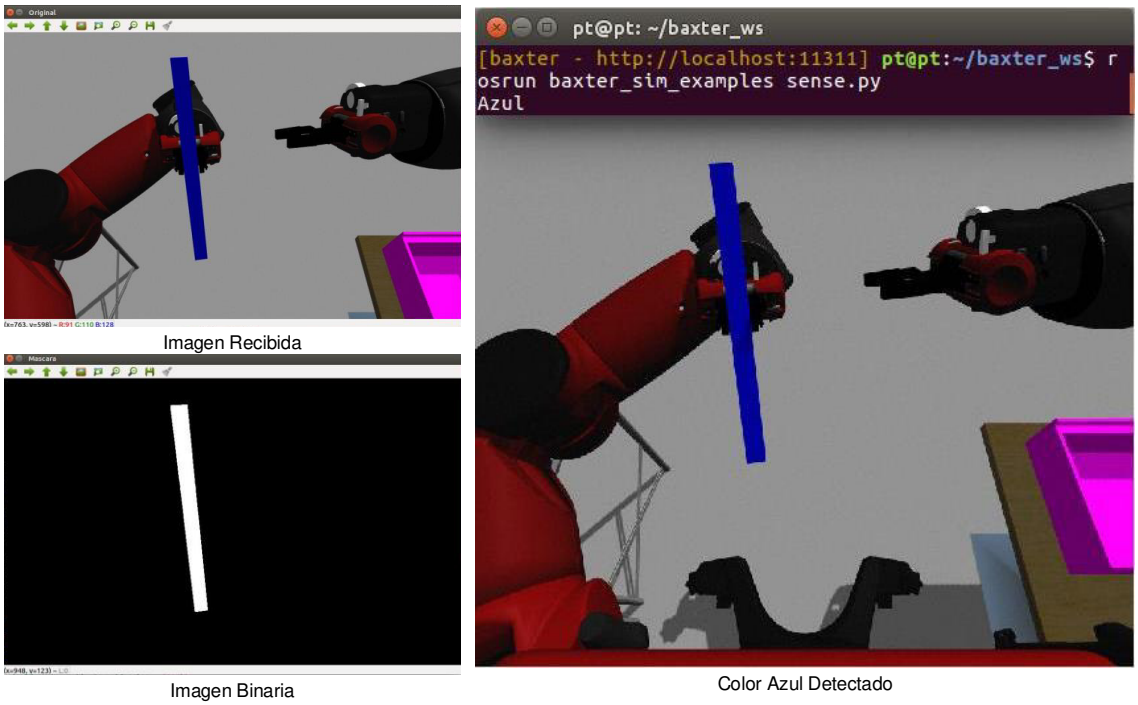


Figura 2.22 Imagen obtenida en el algoritmo de detección de color.

En la Figura 2.24 se presentan el procesamiento de la imagen desde su detección hasta la determinación del color del objeto. Finalmente, en la Figura 2.25 se presenta el diagrama de flujo final con el cual la detección se efectúa adecuadamente.

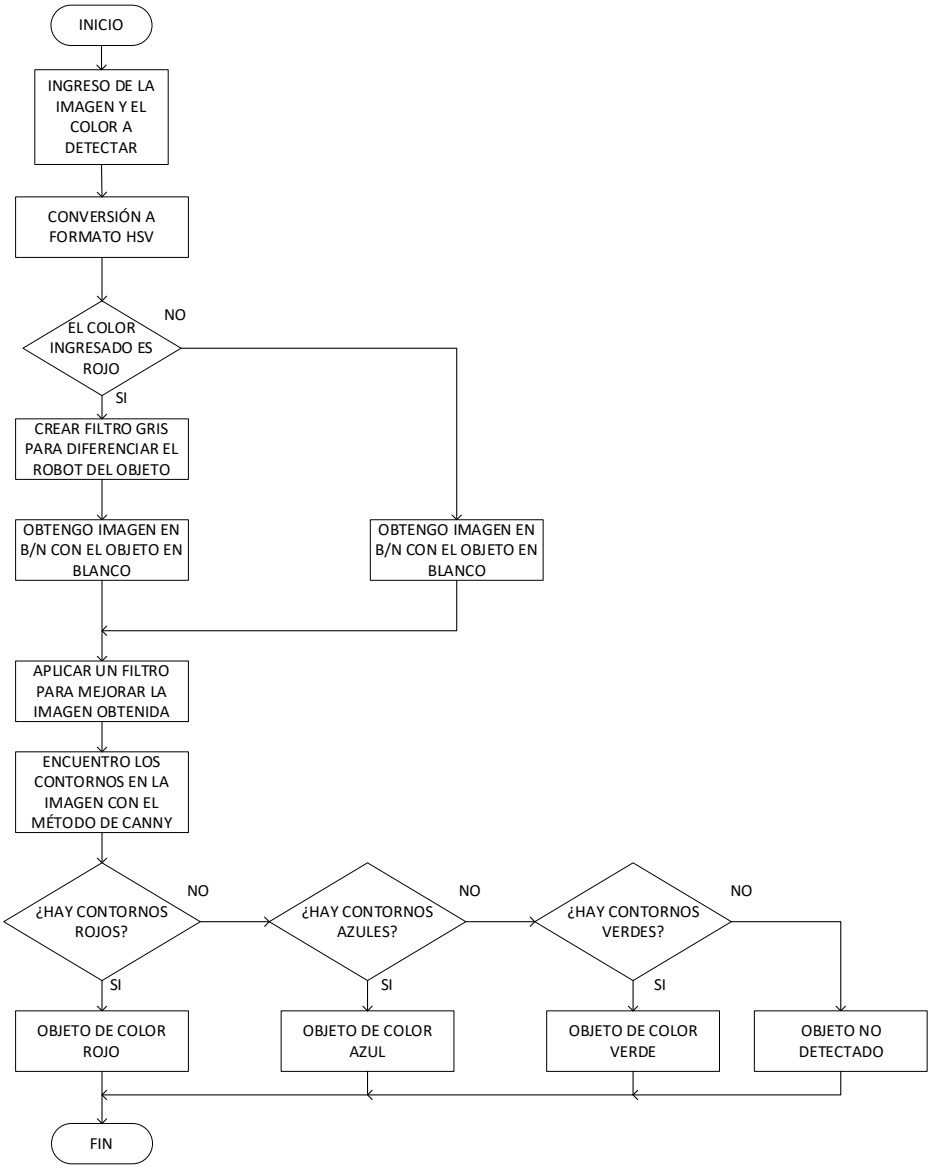


Figura 2.23 Diagrama de flujo del algoritmo implementado para detectar el color de un objeto por medio de la cámara del robot Baxter.

2.4.3. DETECCIÓN DE FORMA

El algoritmo detección de forma para objetos sólidos se presentó en la Sección 1.3.5.1, en esta se detalla el procedimiento que se debe seguir para determinar la forma de un objeto determinado. En este proyecto los objetos tienen formas geométricas básicas, por lo cual, este algoritmo es la continuación del algoritmo de detección de color, luego de determinar el color del objeto se aplica la técnica de Canny de detección de bordes. Una vez obtenido el borde de la figura geométrica se realiza una secuencia que cuente el número de bordes de la figura, así con el resultado se determina qué tipo de forma posee la figura, en el proyecto estas formas pueden ser circulares haciendo referencia a la esfera y rectangulares haciendo referencia al cubo rectangular.

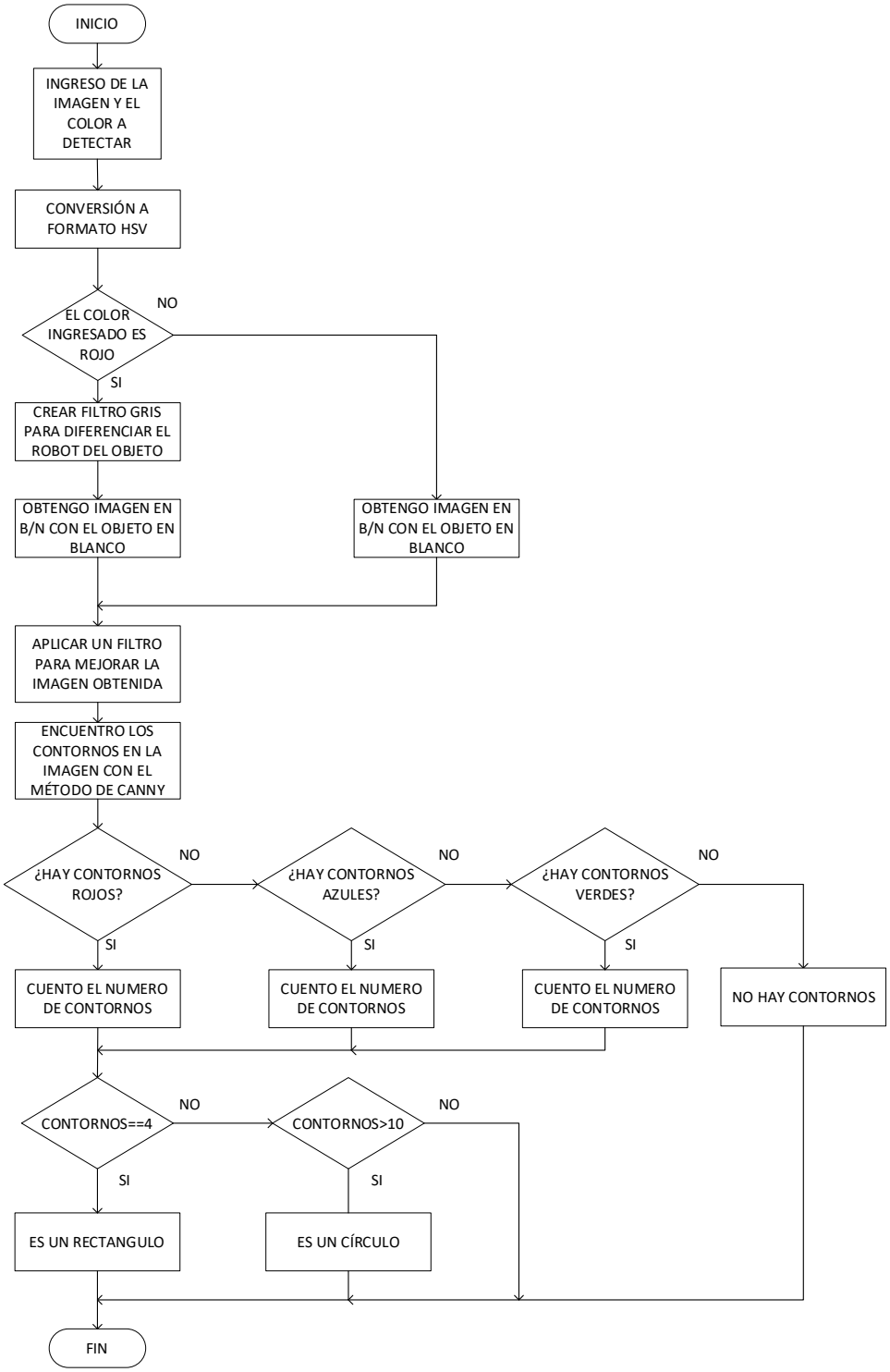


Figura 2.24 Diagrama de flujo del algoritmo implementado para detectar la forma de un objeto por medio de la cámara del robot Baxter.

En la Figura 2.26 se puede observar el diagrama de flujo que ejecuta el detector de forma, mientras que en la Figura 2.27 se observa el resultado en la terminal luego de aplicar el algoritmo de detección de color y detección de forma.

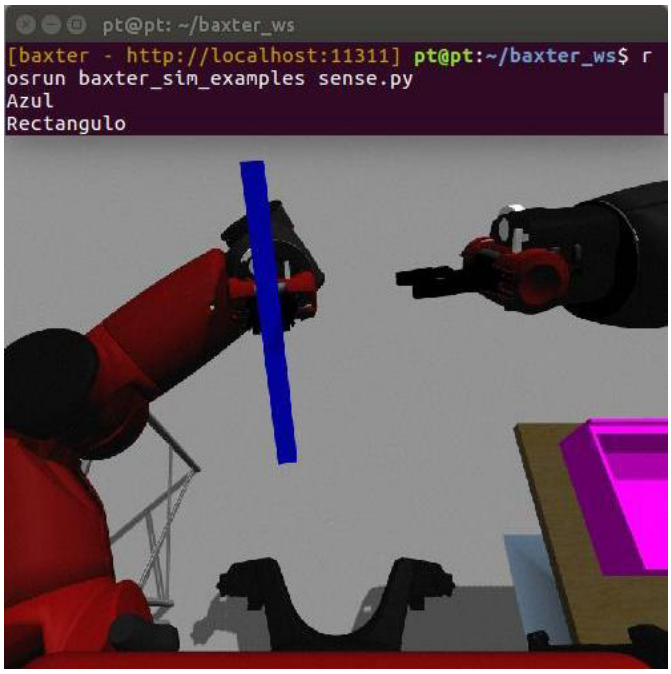


Figura 2.25 Ejecución del script para detección de color y forma.

En la siguiente sección se desarrolla un paquete de ejecución que incorporara todos los scripts desarrollados y modelos utilizados para la escena de clasificación con el objetivo de facilitar una ejecución conjunta.

2.5. DESARROLLO DEL PAQUETE DE EJECUCIÓN PARA EL SISTEMA COMPLETO

Un paquete de ejecución de ROS es desarrollado para facilitar la ejecución de varios programas que conforman un sistema. En este caso el paquete ayudará a ejecutar los diferentes archivos usados para colocar al robot Baxter en Gazebo, colocar su escena de simulación correspondiente, ejecutar los algoritmos tanto para el movimiento automático de cada brazo como para la ejecución del algoritmo de clasificación mediante visión artificial. Existe un procedimiento general el cual ayuda a su respectiva creación y se explica en la página oficial de ROS [33], este procedimiento se desarrolla por medio de una ventana Terminal.

CREACIÓN UN PAQUETE DE EJECUCIÓN

La creación del paquete de ejecución se desarrolla sobre el mismo paquete ROS del robot Baxter, en este caso es necesario dirigirse al directorio *baxter_ws/src/baxter_simulator*, una vez situado en esta carpeta se ejecuta el siguiente comando.

catkin_create_pkg “my_pkg” “library”

En donde *"my_pkg"* hace referencia al paquete que se desea crear, en el proyecto este paquete se denomina *baxter_sim_examples* y *"library"* hace referencia a las librerías con las que se trabaja en el paquete, en el proyecto las librerías usadas son: *rospy, std_msgs, rospack, baxter_core_msgs, baxter_gazebo* entre otras.

Una vez creado el paquete en la carpeta se presentan dos archivos denominados *CMakeLists.txt* y *package.xml*, en estos archivos se encuentran las diferentes librerías con las cuales se está trabajando, librerías que pueden ser añadidas o retiradas directamente desde el código fuente, más información sobre estos tipos de archivos y su creación pueden ser consultados en [33].

AÑADIR CONTENIDO AL PAQUETE

Una vez desarrollados lo algoritmos correspondientes para la tarea de clasificación de objetos, el paso siguiente es desarrollar el archivo de ejecución, el cual sirve para colocar al robot Baxter en el entorno Gazebo y los respectivos modelos de la escena diseñada, como se mencionó en la sección 1.3.4.2 en el formato SDformat existe un archivo de extensión *.launch*, el cual al ser ejecutado por el comando *roslaunch* desde la terminal desplegara los nodos correspondientes al robot Baxter y a los nodos del script que contiene los modelos de la escena desarrollada.

El paquete de ejecución tiene por nombre *baxter_sim_examples* como se mencionó en la sección anterior, este paquete posee las siguientes carpetas:

- Config: Carpeta de archivos de configuración de parámetros.
- Launch: Carpeta de archivos de ejecución.
- Models: Carpeta de modelos desarrollados en diferentes plataformas
- Scripts: Carpeta de programas para distintas aplicaciones.

Una de las cosas importantes para que cada uno de los archivos puedan ser tomados como parte del paquete es que todos sean archivos ejecutables, caso contrario no serán accesibles por el comando *rosrun* o *roslaunch*, para convertir un archivo a ejecutable se debe realizar el siguiente procedimiento.

- Acceder al directorio del archivo por medio de la terminal.
- Ejecutar el comando *chmod +x* sobre el archivo correspondiente.

Una vez realizado el procedimiento anterior, sobre la carpeta correspondiente se debe ejecutar el comando *ls*, cuando esto suceda todos los archivos convertidos a ejecutables deben estar con letras de color verde como se muestra en la Figura 2.28.

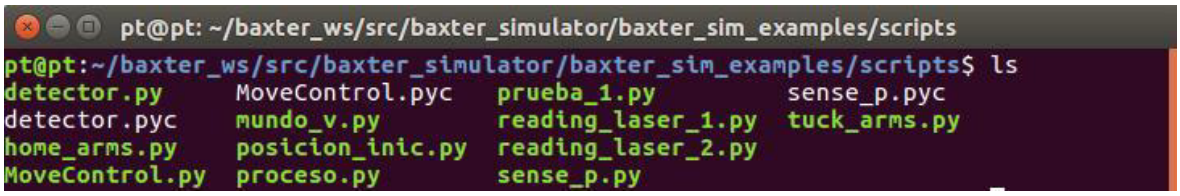


Figura 2.26 Archivos ejecutables para el proceso de clasificación mediante visión artificial.

Para completar el ingreso de un paquete a la carpeta de ROS se debe ejecutar el comando *catkin_make* en el directorio raíz, en este proyecto la carpeta directorio raíz se denomina *baxter_ws*, una vez aplicado el comando automáticamente serán añadidas las respectivas líneas de código al paquete principal que ejecuta el nodo master como se menciona en [34].

En el siguiente capítulo se presentará tanto los resultados más relevantes como el análisis de cada uno de ello, este análisis tiene como objetivo principal validar la aplicación desarrollada en el proyecto con la metodología descrita anteriormente.

3. RESULTADOS Y DISCUSIÓN

El capítulo de resultados y discusión se encuentra conformado tanto por la fase de simulación y la fase de análisis de resultados. La primera fase consta de realizar una simulación del sistema integrado considerando un diferente orden en los objetos a clasificar y describir las diferentes pruebas que se realizarán y la fase de análisis se enfoca en el tratamiento de los resultados que se obtienen en el transcurso de la simulación, resultados relevantes que ayuden a validar la metodología. Estas pruebas son descritas a continuación:

- Pruebas de posicionamiento de cada una de las articulaciones.
- Pruebas de cambios de objetos para el análisis de efectividad del algoritmo de clasificación.
- Pruebas de la efectividad de que la extremidad llegue al punto para tomar el objeto lo pase a la otra articulación y finalmente llegue a la posición final.
- Prueba de rapidez del sistema midiendo el tiempo que se demora el robot Baxter virtual en ejecutar la clasificación de objetos.

3.1. PRUEBAS DE POSICIONAMIENTO

El desarrollo de esta prueba se realiza ingresando un ángulo de giro al algoritmo de control de posición y observar el ángulo que alcanza cada una de las articulaciones. La misma prueba se desarrollará 5 veces con el fin de obtener más de una medición para cada articulación. Como se mencionó en la Sección 2.3.1 existen dos formas de conocer el ángulo de giro de cada una de las articulaciones, formas con las cuales se trabajará en esta sección, es decir, en cada prueba se obtiene dos mediciones, la primera medición se denomina Valor Gazebo y la segunda medición se denomina Valor ROS, por lo tanto, el número de errores obtenidos en las pruebas son 10 por cada brazo del robot Baxter, los resultados del error de posicionamiento se presentan en la Tabla C.1 y C.2 del ANEXO C.

Tabla 3.1 Error Promedio en la Prueba de Posicionamiento.

Error Promedio				
Art.	**Brazo Izquierdo**		**Brazo Derecho**	
	Medición Gazebo	**Medición ROS**	**Medición Gazebo**	**Medición ROS**
S0	0,073240141	0,080836481	0,03327	0,0329105
S1	0,105345727	0,109828541	0,4583244	0,459088
E0	0,008815646	0,006800251	0,166032269	0,16611479
E1	0,003632921	0,001788376	0,100275258	0,09880505
W0	0,364868555	0,317962196	0,121006567	0,12274149
W1	0,022386587	0,027400472	0,011479612	0,0132699
W2	0,14141842	0,141194418	0,3856884	0,31642

En la Tabla 3.1 se presenta el error promedio de posición del ángulo de giro de cada articulación.

Con los datos registrados en la Tabla 3.1 se procede a calcular el error en mediciones para cada articulación, este error es equivalente a la raíz cuadrada de la suma de cada error obtenido al cuadrado, esto se expresa en la Ecuación (3.1).

$$Err = \sqrt{Error_G^2 + Error_R^2} \tag{3.1}$$

Donde:

Err: Es el error en la medición.

$Error_G$: Es el error promedio obtenido de la medición en el simulador Gazebo.

$Error_R$: Es el error promedio obtenido de la medición en el entorno ROS.

Tabla 3.2 Error en la medición para cada brazo del Baxter.

Brazo Izquierdo (Left)		Brazo Derecho (Right)	
Art.	**Error Final [%]**	**Art.**	**Error Final [%]**
S0	0,073995668	**S0**	0,076027629
S1	0,661164862	**S1**	0,664288283
E0	0,23444911	**E0**	0,235108594
E1	0,139731562	**E1**	0,140656559
W0	0,516542455	**W0**	0,523550497
W1	0,047757083	**W1**	0,039491806
W2	0,572159371	**W2**	0,464616748

A partir de los resultados mostrados en la Tabla 3.2 se concluye que el error del ángulo de giro de cada articulación oscila aproximadamente entre el 0.04% y el 0.7%, dando como resultado que el posicionamiento de cada brazo es adecuado para el proceso.

3.2. PRUEBAS DE EFECTIVIDAD DE LA EXTREMIDAD

Esta prueba se divide en dos partes, la primera consiste en comprobar que cada uno de los brazos del robot Baxter ejecuten la secuencia correspondiente, esta secuencia consiste en que el brazo izquierdo tome a los objetos de cada una de las bandas transportadoras, los coloque en una posición en la cual el brazo derecho toma al objeto del brazo izquierdo y lo pasa a las cajas determinadas para su clasificación, la segunda parte consiste en determinar el agarre y transporte del objeto para lo cual se hace uso de la Figura 2.19 de la sección 2.3.2, figura en donde se determina los límites de agarre del objeto..

Tabla 3.3 Prueba de Transporte de un objeto de un punto a otro empleando los dos brazos del Baxter.

Parte 1: Prueba de Transporte de la banda a las cajas					
Objeto	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**
Cubo rectangular	SI	SI	SI	SI	SI
Esfera	NO	SI	SI	SI	SI

En la Tabla 3.3 se presentan los resultados de cinco pruebas desarrolladas para el transporte de un objeto empleando los dos brazos del robot Baxter. A partir de estos registros se concluye que la primera esfera simulada no siempre es transportada, esto debido a un mal agarre por parte del gripper del brazo empleado, no obstante, las siguientes pruebas muestran un transporte adecuado tanto de esferas como de cubos rectangulares, cabe mencionar que si una esfera es tomada correctamente por un brazo del Baxter, en la interacción con el otro brazo, este tomará de forma adecuada a la esfera completando exitosamente el transporte de la misma.

Tabla 3.4 Prueba de agarre y transporte de un objeto empleando un solo brazo del Baxter.

Parte 2: Prueba de Agarre y Transporte (objeto cúbico rectangular)			
Prueba 1			
Límite 1	**Límite 2**	**Agarre**	**Transporte**
Centro de Masa	Línea 1	SI	SI
Línea 1	Línea 2	SI	SI
Línea 2	Línea 3	SI	NO
Línea 3	Extremo del objeto	SI	NO
Prueba 2			
Límite 1	**Límite 2**	**Agarre**	**Transporte**
Centro de Masa	Línea 1	SI	SI
Línea 1	Línea 2	SI	SI
Línea 2	Línea 3	SI	NO
Línea 3	Extremo del objeto	SI	NO
Prueba 3			
Límite 1	**Límite 2**	**Agarre**	**Transporte**
Centro de Masa	Línea 1	SI	SI
Línea 1	Línea 2	SI	SI
Línea 2	Línea 3	SI	NO
Línea 3	Extremo del objeto	SI	NO

Los registros de la Tabla 3.4 muestran los resultados de la prueba de agarre y transporte en la cual únicamente se hizo uso de un objeto con forma cúbica rectangular, esto debido a que el agarre y transporte de la esfera es un caso especial ya que para un agarre y transporte exitoso el centro de masa de la esfera debe coincidir con el centro del gripper.

En la Tabla 3.4, el límite 1 y límite 2 representan la zona en donde se coloca el gripper para agarrar el objeto, por lo que se concluye que para un objeto cúbico rectangular todas las veces que el gripper de agarre es colocado en las líneas 1 y 2 la toma y transportación del objeto se ejecuta con normalidad, mientras que en las líneas 3 logra hacer un correcto agarre, pero no transportaba el objeto hasta el punto requerido, esto se debe a que, el centro de masa del objeto se encuentra por fuera del rango requerido para un correcto agarre y transporte.

A partir de estos resultados y la explicación de la sección 2.3.2 se concluye que el desfase aceptable entre el efector final del Baxter virtual y el centro de masa del objeto es de 4 cm.

3.3. PRUEBAS DE CLASIFICACIÓN PARA CAMBIOS DE OBJETOS

Esta prueba se desarrolla en base a la existencia de esferas de color verde en cada una de las bandas transportadoras, objetos con los cuales la clasificación no solamente es por color sino también por forma, es decir, estos objetos son mezclados entre objetos de color rojo, color azul y de forma cúbica rectangular.

El objetivo de esta prueba es determinar que el algoritmo de clasificación mediante visión artificial se esté ejecutando continuamente y determinar si el objeto fue o no colocado en su caja respectiva, para esta prueba se desarrolla en 5 escenarios diferentes con 5 objetos cada uno. Dos de estos escenarios únicamente contendrán objetos de forma cúbica rectangular, dos de los escenarios faltantes contienen una esfera cada uno y en el escenario final se tiene en su mayoría esferas de color verde, cabe indicar que los objetos son proporcionados mediante las dos bandas transportadoras. En la Tabla 3.5 se puede observar los datos registrados en esta prueba.

Tabla 3.5 Pruebas clasificación para cambios de objetos.

Escenario 1				
Objeto	**Forma**	**Color**	**Clasificación V.A**	**Colocación en caja**
1	Cubo rectangular	Rojo	SI	SI
2	Cubo rectangular	Rojo	SI	SI
3	Cubo rectangular	Rojo	SI	SI
4	Cubo rectangular	Azul	SI	SI
5	Cubo rectangular	Rojo	SI	SI
Escenario 2				
Objeto	**Forma**	**Color**	**Clasificación V.A**	**Colocación en caja**
1	Cubo rectangular	Rojo	SI	SI
2	Cubo rectangular	Azul	SI	SI
3	Cubo rectangular	Azul	SI	SI
4	Cubo rectangular	Azul	SI	SI
5	Esfera	Verde	SI	NO
Escenario 3				
Objeto	**Forma**	**Color**	**Clasificación V.A**	**Colocación en caja**
1	Cubo rectangular	Azul	SI	SI
2	Cubo rectangular	Rojo	SI	SI
3	Cubo rectangular	Rojo	SI	SI
4	Esfera	Verde	SI	SI
5	Cubo rectangular	Rojo	SI	SI
Escenario 4				

Objeto	Forma	Color	Clasificación V.A	Colocación en caja
1	Cubo rectangular	Azul	SI	SI
2	Cubo rectangular	Rojo	SI	SI
3	Cubo rectangular	Azul	SI	SI
4	Cubo rectangular	Rojo	SI	SI
5	Cubo rectangular	Azul	SI	SI
Escenario 5				
Objeto	**Forma**	**Color**	**Clasificación V.A**	**Colocación en caja**
1	Esfera	Verde	SI	SI
2	Cubo rectangular	Azul	SI	SI
3	Esfera	Verde	SI	SI
4	Esfera	Verde	SI	SI
5	Esfera	Verde	SI	SI

Se debe tomar en cuenta que las pruebas registradas en la Tabla 3.5 fueron desarrolladas continuamente, es decir, la simulación del sistema fue continuo. A partir de los resultados presentados en la Tabla 3.3 se observa que el robot Baxter al encontrarse con la primera esfera en el sistema, el robot Baxter la clasifica, pero no la coloca en la caja correspondiente, esto se debe a que en la simulación la esfera fue agarrada en una posición no adecuada y cuando es tomada por el brazo derecho del robot Baxter este la suelta debido a que no se encuentra en posición para un agarre efectivo. Una de las causas de un mal agarre, es debe a que al ser una esfera esta puede moverse del punto de agarre debido a la acción de la gravedad que se encuentra aplicada sobre el objeto, lo que no ocurre con un objeto cúbico rectangular ya que estos objetos fueron clasificados y colocados en sus cajas exitosamente.

A partir de los resultados mostrados en la Tabla 3.3 se concluye que el algoritmo de clasificación que usa de sensor la cámara principal del robot Baxter virtual es efectivo cuando se incorporan esferas de color verde al sistema.

3.4. PRUEBAS DE RAPIDEZ DEL SISTEMA

Esta prueba consiste en medir el tiempo que se demora el robot Baxter virtual en ejecutar todo el proceso de clasificación, este tiempo será medido mediante el parámetro “sim time” de la plataforma Gazebo. Se debe tomar en cuenta que el tiempo que demora el robot Baxter virtual en ejecutar un movimiento depende de la tasa de envió de datos por mensaje. El paquete ROS del robot Baxter sugiere una tasa de 20Hz para el envió de datos, tasa con la cual se ejecuta este proyecto. Se hace uso de este valor debido a que, si el código es ejecutado en un robot Baxter real, este no tenga problemas de comunicación en el envío

de datos. Por esto, esta tasa es con la cual se desarrollan las pruebas respectivas, los resultados de estas pruebas se presentan en la Tabla 3.6.

Tabla 3.6 Registro de datos para la prueba de rapidez

Tiempo de clasificación [s]							
Objeto	**Forma**	**Color**	**P. 1**	**P. 2**	**P. 3**	**P. 4**	**P. 5**
1	Esfera	Verde	5,8635	6,5358	5,9415	5,8399	6,2120
2	Cubo rectangular	Azul	8,0175	7,8372	8,0213	7,9989	7,9145
3	Cubo rectangular	Rojo	8,0175	8,5982	8,0012	8,3417	8,3990
4	Cubo rectangular	Rojo	8,1300	9,2790	8,5240	8,0990	9,1870
5	Cubo rectangular	Rojo	8,4450	9,4248	8,3450	9,0544	9,2870
6	Esfera	Verde	7,2825	8,5014	8,0178	8,0308	8,4550
7	Cubo rectangular	Azul	7,9830	8,7948	8,5698	8,0132	8,4760
8	Cubo rectangular	Rojo	7,0770	8,8020	8,7340	9,1287	8,0588
9	Cubo rectangular	Azul	8,3175	10,2564	9,5633	8,7522	9,0133
Tiempo total de clasificación [s]			69,133	78,0296	73,717	73,258	75,002
Tiempo total de clasificación [min]			1,1522	1,3005	1,2286	1,2210	1,2500
Tiempo total de simulación [s]			86,280	101,40	98,32	88,450	92,30
Tiempo total de simulación [min]			1,4380	1,6900	1,6387	1,4742	1,5383

A partir de los datos mostrados en la Tabla 3.6 se concluye que el tiempo promedio para clasificar un objeto esférico de color verde es de 7,068 segundos, mientras que, para clasificar un objeto cúbico rectangular sea de color rojo o color azul el tiempo promedio es de 8,5274 segundos, Finalmente el tiempo promedio que se demora en realizar el proceso de toma, clasificación y colocación de 9 objetos de diferente forma y color es de 93,35 segundos (1,555 minutos). Adicionalmente en los datos obtenidos se observa una variación en cada valor registrado, esto se debe, a que los tiempos de envío de flujo de datos para cada mensaje depende en gran parte del procesador de la máquina. En un ambiente real estos tiempos no se ven afectados, ya que, el robot Baxter posee un procesador independiente el cual ayuda a incrementar la velocidad del sistema.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

- Se realizó un estudio detallado del ambiente de programación ROS-GAZEBO haciendo referencia a sus características y funcionalidades, tomando como punto de partida la aplicación de este sistema sobre el robot Baxter virtual. Además, se desarrolló un estudio del modelo cinemático directo del robot Baxter, modelo proporcionado por la compañía Rethink Robotics.

- En el desarrollo de la escena de clasificación industrial se observó y estudió diferentes formas de diseñar los modelos, para lo cual se deben tomar en cuenta las características físicas y dinámicas del objeto. Por lo que el cálculo de inercias y dimensiones del objeto es de suma importancia ya que Gazebo toma en cuenta estas características para simular el objeto, de no obtener un buen diseño la simulación puede no ejecutarse correctamente.

- En el algoritmo de clasificación de objetos mediante visión artificial se observa que primero se debe realizar la clasificación por color y después la clasificación por forma, esto se debe a que, en el proceso de clasificación, de la imagen recibida por la cámara del Baxter se obtiene una imagen binaria (Blanco y Negro) en la cual el objeto de color rojo, azul o verde se presenta en color blanco, en este punto ya no es posible la clasificación por color. Luego se cuenta los contornos de la imagen obtenida y se determina su forma, con este proceso el algoritmo se ejecuta una sola vez y no dos veces que es el caso de que primero se determine la forma y luego el color.

- En el algoritmo de control de movimientos se toma en cuenta un controlador de tipo PID para cada una de las articulaciones de los brazos del robot Baxter, también se utilizó un filtro pasa bajos cuando el movimiento de la extremidad sea suave, como es el caso de la toma de objetos, el controlador de tipo PID fue sintonizado por el método de prueba y error observando la curva de respuesta de la articulación a modificar. La validación de los valores de las constantes del controlador PID se realizó mediante el índice de desempeño denominado ISE cuyo valor es 0.0005804.

- En la prueba de posicionamiento se determinó que el error de posición en cada una de las articulaciones varía entre 0,04% y 0,7%, porcentajes con los cuales el error de posición en cada articulación es aceptable. En el caso de la prueba de efectividad si un objeto no es tomado cerca de su centro de masa por el efector final el objeto es agarrado, pero no transportado, por lo cual se determina que el desfase entre el centro de la paleta del efector final y el centro de masa del objeto debe ser de 4 cm para que agarre y transporte el objeto. En el caso de una esfera este rango se limita a que estos centros deben coincidir con gran precisión debido al tamaño de la esfera, de no ser así, la esfera es soltada en el transporte.

- En la prueba de clasificación para cambios de objetos se observa que, al ser incorporadas esferas en las bandas transportadoras, el algoritmo de clasificación entrega resultados correctos, es decir, la esfera es clasificada; sin embargo, existen

casos en donde la esfera es clasificada pero no transportada, esto se debe a que la esfera se ve afectada por la gravedad, por lo que esta se pudo haber movido de su posición en reposo provocando que el efector final no la agarre adecuadamente.

- Finalmente se realizó la prueba de rapidez del sistema de clasificación, esta prueba es medida por el tiempo de simulación, parámetro visible en la plataforma de simulación Gazebo, dando como resultado un tiempo promedio de simulación de 1.555 minutos, tiempo en el cual se desarrolla la toma, clasificación y colocación de 9 objetos en las cajas correspondientes. En conclusión, el tiempo que se demora un robot Baxter real en desarrollar la clasificación mencionada es de 1.555 minutos, validando de esta forma el uso de la aplicación desarrollada en este proyecto de titulación.

4.2. RECOMENDACIONES

- El uso del modelo de cinemática directa limita mucho a los movimientos del robot Baxter, es decir, una mayor parte de las líneas de código contiene las posiciones a las que se desea llevar cada una de las articulaciones. Se propone para un trabajo futuro el uso de la cinemática inversa del robot Baxter, la cual será de gran ayuda para ya no mover a una posición determinada a sus extremidades sino moverlas a una posición especifica en el plano. Además, el uso de la cinemática inversa da paso a incorporar algoritmos de agarre para objetos en movimiento, cálculos de centro de masa de objetos entre otros.

- Se recomienda desarrollar los movimientos de los objetos por medio de scripts para la simulación de bandas transportadoras en el entorno Gazebo, debido a que, este programa no acepta la simulación de motores que giren continuamente y muevan la banda transportadora. Para un trabajo futuro se puede desarrollar un objeto que mediante un script se incorpore como joint a la banda transportadora y que esta articulación sea de tipo traslacional para así simular el movimiento del objeto sobre la banda transportadora.

- Para abrir un elemento o modelo en el entorno de simulación Gazebo se debe tomar en cuenta que tipo de elemento es, en este caso si es un modelo de robot, banda o mesa, es recomendable hacerlo mediante un script XML, mientras que si es un modelo URDF es recomendable hacerlo mediante un script de Python.

- En esta aplicación se desarrollaron objetos de formas básicas como lo es un cubo rectangular y una esfera, para un trabajo futuro se recomienda desarrollar modelos

más reales, es decir, modelos con texturas y materiales diferentes que existan en la industria, con el objetivo de desarrollar un algoritmo de clasificación mediante visión artificial, el cual pueda ser entrenado de forma en que detecte una mayor cantidad de objetos.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] I. Olier Caparroso, O. Avilés, y J. Hernández Bello, "Una introducción a la robótica industrial", Ciencia e IngenieríaNeogranadina, vol. 8, pp. 53-67, jun. 1999.

[2] C. Chávez, "Robotica Industrial (introduccion, conceptos y momenclatura)", www.youtube.com, Sep. 08, 2021. https://www.youtube.com/watch?v=x_PvOL_Cvps&ab_channel=C%C3%A9sarCh %C C3%A1vez-Ingenier%C3%ADa- (accessed Dec. 06, 2021).

[3] P. J. Guaraca Medina and J. L. Ochoa Ochoa, “Estudio de la programación y operación de los robots industriales KUKA KR16-2 y KR5-2 ARC HW,” dspace.ups.edu.ec, Feb. 2015, Accessed: Dec. 06, 2021. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/7744.

[4] R. Chans Veres, “Simulación de un proceso industrial para la incorporación de robots,” ruc.udc.es, Jun. 2018, Accessed: Dec. 06, 2021. [Online]. Available: http://hdl.handle.net/2183/21893.

[5] S. Muñoz Montoya and S. Valencia Sánchez, “Sistema de pick and place para objetos en movimiento sobre el plano, con el Cobot Baxter,” repository.eia.edu.co, 2020, Accessed: Dec. 06, 2021. [Online]. Available: https://repository.eia.edu.co/handle/11190/2726.

[6] Baxter-making Rethink Robotics closes its doors. New Atlas. 2018. Retrieved from: https://newatlas.com/rethink-robotics-closing-down/56664/

[7] R. Amadeo, “Hands-on with Baxter, the factory robot of the future,” Ars Technica, Jun. 15, 2014. https://arstechnica.com/gadgets/2014/06/hands-on-with-baxter-the-factory-robot-of-the-future/ (accessed Dec. 06, 2021).

[8] G. A. Díaz Guevara and S. Sarmiento Navarro, “Diseño e implementación de funciones de interacción emocional y sensorial para el robot Baxter en el contexto del proyecto Human-Robot Scaffolding,” repository.javeriana.edu.co, Apr. 2018, Accessed: Dec. 09, 2021. [Online]. Available: http://hdl.handle.net/10554/38799.

[9] V. H. Gómez Tejada, “Operación de remachado mediante robot manipulador industrial basado en ROS,” idus.us.es, 2015, Accessed: Dec. 09, 2021. [Online]. Available: http://hdl.handle.net/11441/36944.

[10] R.L. Williams II, “Baxter Humanoid Robot Kinematics”, Internet Publication, https://www.ohio.edu/mechanical-faculty/williams/html/pdf/BaxterKinematics.pdf, April 2017.

[11] J.J. Craig, 2005, Introduction to Robotics: Mechanics and Control, Third Edition, Pearson Prentice Hall, Upper Saddle River, NJ.

[12] Yoonseok Pyo, Hancheol Cho, Leon Jung, Darby Lim, “ROS Robot Programming (English),” Robotsource, Dec. 01, 2017. http://community.robotsource.org/t/download-the-ros-robot-programming-book-for-free/51 (accessed Jan. 04, 2022).

[13] C. Fairchild and T. L. Harman, ROS robotics by example : learning to control wheeled, limbed, and flying robots using ROS Kinetic Kame, 2nd ed. Birmingham, Uk: Packt Publishing, 2017.

[14] Open Source Robotics, “SDFormat Specification,” sdformat.org, 2020. http://sdformat.org/spec (accessed Jan. 06, 2022).

[15] Iván García S., Víctor Caranqui S, “LA VISIÓN ARTIFICIAL Y LOS CAMPOS DE APLICACIÓN,” Tierra Infinita, vol. Vol. 1 Núm. 1 (2015), Jul. 2015, doi: 10.32645/26028131.76.

[16] E. Á. Sobrado Malpartida, “Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot,” Pontificia Universidad Católica del Perú, May 2011, [Online]. Available: http://hdl.handle.net/20.500.12404/68.

[17] J. Porras and M. De La Cruz, “CLASIFICATION SYSTEM BASED ON COMPUTER VISION.” Accessed: Jan. 19, 2022. [Online]. Available: https://www.urp.edu.pe/pdf/id/2881/n/clasification-system-based-on-computer-vision.

[18] L. G. Tena, H. Sossa, A. Alvarado, O. O. Vergara, V. M. H. Zubía, and F. J. L. Benavides, “Reconocimiento de objetos en una plataforma robótica móvil,” Cultura Científica y Tecnológica, no. 55, Dec. 2015, Accessed: Jan. 08, 2022. [Online]. Available: https://erevistas.uacj.mx/ojs/index.php/culcyt/article/view/747.

[19] N. T. Fitter and K. J. Kuchenbecker, “Teaching a Robot Bimanual Hand-Clapping Games via Wrist-Worn IMUs,” Frontiers in Robotics and AI, vol. 5, Jul. 2018, doi: 10.3389/frobt.2018.00085.

[20] J. Otero, “Baxter, el robot que aprende a cocinar mirando vídeos de YouTube,” El blog de Javier Otero, Feb. 01, 2015. http://www.javierotero.info/2015/02/baxter-el-robot-que-aprende-cocinar.html.

[21] J. Velasco Seguido-Villegas and M. Ferre Pérez, “Análisis y comparación de las principales plataformas de simulación robótica y su integración con ROS,” oa.upm.es, Sep. 01, 2019. https://oa.upm.es/56724/.

[22] G. del Olmo Hernández, M. R. Aragüés Muñoz, G. López Nicolás, and Universidad de Zaragoza, Simulación en Gazebo de robots móviles para tareas de transporte y manipulación. Zaragoza: Universidad de Zaragoza, 2019.

[23] “Cómo Funciona El Robot Baxter | Tecnología,” Portal Multimedia Científica Y Popular. https://es.wordssidekick.com/how-baxter-robot-works-24445 (accessed Jan. 01, 2022).

[24] Programación Extrema, “Robótica: de modelo Blender a Gazebo,” www.youtube.com, 2019. https://www.youtube.com/watch?v=uKwaWlHUG-8 (accessed Jan. 17, 2022).

[25] Rethink Robotics, “baxter_common,” GitHub, Dec. 30, 2015. https://github.com/RethinkRobotics/baxter_common (accessed Jan. 17, 2022).

[26] R. P. Paul, Robot manipulators : Mathematics, programming and control., 1st Edition. Cambridge, MA, USA.: The MIT Press, 1981.

[27] Dr. José Antonio Garrido Natarén, José Antonio Hernandez Reyes,MC. José Luis Fernando Palomeque Loyo, Carlos Armando Gutiérrez González, “ESTUDIO DE UNA CADENA CINEMÁTICA UTILIZANDO LOS MÉTODOS DENAVIT-HARTENBERG Y CRAIG MODIFICADO - PDF Free Download,” docplayer.es. https://docplayer.es/95237332-Estudio-de-una-cadena-cinematica-utilizando-los-metodos-denavit-hartenberg-y-craig-modificado.html (accessed Jan. 19, 2022).

[28] Open Source Robotics Foundation, “Gazebo : Tutorials,” gazebosim.org, 2014. https://gazebosim.org/tutorials (accessed Jan. 19, 2022).

[29] Open Source Robotics Foundation, “Gazebo : Tutorial : URDF in Gazebo,” gazebosim.org, 2014. http://gazebosim.org/tutorials/?tut=ros_urdf (accessed Jan. 19, 2022).

[30] Diego A. Macrini and Guillermo Baruh, “Sistema de Reconocimiento Automático de Formas,” www.cs.toronto.edu. https://www.cs.toronto.edu/~dmac/images/ProjectFiles/sraf/srafdoc/sraf.html (accessed Jan. 19, 2022).

[31] A. Cardona, L. Garelli, and J. Gimenez, “COMPARISSON OF METHODS FOR BORDER DETECTION IN SAR SATELLITE IMAGES,” Mecánica Computacional, vol. I Vol XXXVII, pp. 2067–2075, Nov. 2019, Accessed: Jan. 20, 2022. [Online]. Available: http://www.amcaonline.org.ar.

[32] “Tensor de inercia,” Wikipedia, Oct. 02, 2021. https://es.wikipedia.org/wiki/Tensor_de_inercia#:~:text=El%20tensor%20de%20inercia%20es (accessed Jan. 22, 2022).

[33] J. E. Riva, “es/ROS/Tutoriales/CreandoPaquetes - ROS Wiki,” wiki.ros.org, Aug. 05, 2021. http://wiki.ros.org/es/ROS/Tutoriales/CreandoPaquetes (accessed Oct. 08, 2021).

[34] W. Woodall, “catkin/Tutorials/create_a_workspace - ROS Wiki,” wiki.ros.org, May 11, 2017. http://wiki.ros.org/catkin/Tutorials/create_a_workspace (accessed Oct. 08, 2021).

[35] Rethink Robotics, “Arm Control Modes - sdk-wiki,” sdk.rethinkrobotics.com. https://sdk.rethinkrobotics.com/wiki/Arm_Control_Modes (accessed Apr. 28, 2022).

[36] K. Ogata, Sebastián Dormido, Raquel Dormido Canto, A. Mariani, and E. Al, Ingeniería de control moderna. Madrid: Prentice Hall, 2010.

[37] T. L. Floyd and Rodolfo Navarro Salas, Dispositivos electrónicos. México: Pearson Educación, 2008.

[38] RethinkRobotics, “baxter_examples/scripts at master · RethinkRobotics/baxter_examples,” GitHub, Dec. 30, 2015. https://github.com/RethinkRobotics/baxter_examples/tree/master/scripts (accessed May 02, 2022).

[39] Astigarraga, E. (2015), ‘‘Impacto de la Tecnología en las organizaciones Los retos del futuro: tecnología y personas’’, Noviembre 2015. [En línea]. Disponible en: http://www.consejo.org.ar/congresos/material/12congresoadmi/Trabajo2.5.pdf [Accedido: 17-agosto-2021].

[40] VALVERDE-CASTRO, Byron Iván. La importancia de la Robótica como eje en el desarrollo de la sociedad. Polo del Conocimiento, [S.l.], v. 5, n. 8, p. 1368-1377, ago. 2020. ISSN 2550-682X. Disponible en: < https://polodelconocimiento.com/ojs/index.php/es/article/view/1668>. Fecha de acceso: 02 sep. 2021 doi:http://dx.doi.org/10.23857/pc.v5i8.1668.

[41] De Jesús García, Marisol, and Carlos Alberto García Delgadillo. “Diseño de Una Interfaz Para Un Manipulador Tipo Scara Que Pueda Usarse Para Ensamblado

Electrónico." Tesis.ipn.mx, 18 June 2008, tesis.ipn.mx/handle/123456789/48. Accessed 3 Sept. 2021.

[42] Jason M. O'Kane, ''A Gentle Introduction to ROS'', Independently published, oct 2013, 978-1492143239, Available at url: http://www.cse.sc.edu/~jokane/agitr/ [Acceded: 20-ago-2021].

[43] Fankhauser, Péter & Jud, Dominic & Wermelinger, Martin & Hutter, Marco. (2017). Programming for Robotics - Introduction to ROS. 10.13140/RG.2.2.14140.44161.

[44] Fairchild, Carol, and Thomas L Harman. ROS Robotics by Example: Learning to Control Wheeled, Limbed, and Flying Robots Using ROS Kinetic Kame. Birmingham, Uk, Packt Publishing, 2017.

[45] A. S. Chipule Pérez, G. Águila Rodríguez, and R. Posada Gómez, "DISEÑO DE UN FILTRO DIGITAL PASA BAJAS DE PRIMER Y SEGUNDO ORDEN A PARTIR DE CIRCUITO RC," core.ac.uk, Nov. 2016, Accessed: Jun. 23, 2022. [Online]. Available: https://core.ac.uk/display/229036096?utm_source=pdf&utm_medium=banner&utm_campaign=pdf-decoration-v1

[46] w3.css, "First-Order Low-Pass Filter Discretization," controlsystemsacademy.com, Apr. 30, 2019. http://controlsystemsacademy.com/0020/0020.html (accessed Aug. 25, 2022).

ANEXOS

En la sección de anexos se incorporan un manual de usuario, en el cual se detalla los pasos que se debe seguir para poner en marcha la aplicación de clasificación de objetos mediante visión artificial haciendo uso de del robot Baxter. Además, se incorpora un anexo en el cual se describen las características de hardware y software del robot Baxter, esta documentación es proporcionada por la compañía Rethink Robotics. Finalmente se incorpora un anexo en el cual se presenta una tabla cuyo contenido son los errores de las pruebas de posicionamiento del robot Baxter.

A continuación, se presentan los anexos mencionados:

ANEXO A. Manual de usuario.

ANEXO B. Introducción al Robot Baxter.

ANEXO C. Resultados de los errores obtenidos para la prueba de posicionamiento.

ESCUELA POLITÉCNICA NACIONAL

Campus Politécnico "J. Rubén Orellana R."

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
DECANATO

ORDEN DE EMPASTADO

De conformidad con el Artículo 101 del **REGLAMENTO DE RÉGIMEN ACADÉMICO DE LA ESCUELA POLITÉCNICA NACIONAL** aprobado por Consejo Politécnico en Octubre de 2017 y el **INSTRUCTIVO PARA EL PROCEDIMIENTO DE TRABAJOS DE TITULACIÓN DE LAS CARRERAS DE PRE-GRADO Y PROGRAMAS DE POSGRADO DE LA FIEE,** aprobado por Consejo de Facultad el 24 de enero de 2019 que establece que "*El Decano de la FIEE, remitirá los informes de calificación al Director del Trabajo de Titulación y le solicitará un informe de conformidad en relación a las observaciones planteadas, este informe será presentado al Decano en un plazo no mayor a diez días, quien a su vez emitirá la autorización de impresión y encuadernación final del Trabajo de Titulación o Tesis de Grado.",* una vez verificado el cumplimiento del formato de presentación establecido, autorizo la impresión y encuadernación final del Trabajo de Titulación presentado por el señor:

LUIS PATRICIO CATAGÑA CATAGNIA

Fecha de autorización: 6 de septiembre de 2022

Firmado electrónicamente por:
MARTHA CECILIA
PAREDES PAREDES

Dra. Martha Cecilia Paredes
Decana

wg.