UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TÍTULO

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE INTERACCIÓN COLABORATIVO ENTRE EL ROBOT HUMANOIDE NAO Y EL ROBOT OMNIDIRECCIONAL ROBOTINO PARA LA EJECUCIÓN Y COORDINACIÓN DE TAREAS DE LOCOMOCIÓN EN ENTORNOS CONTROLADOS

AUTOR

Barcia Ayala, Orlando Giovanni

TRABAJO DE TITULACIÓN

Previo a la obtención del grado académico en
MAGÍSTER EN ELECTRÓNICA Y AUTOMATIZACIÓN

TUTOR

Figueroa Olmedo, Junior Rafael

Santa Elena, Ecuador

Año 2025

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TRIBUNAL DE SUSTENTACIÓN

Firmado electrónicamente por:
ALICIA GERMANIA ANDRADE VERA

Ing. Alicia Andrade Vera, Mgtr.
COORDINADORA DEL PROGRAMA

Firmado electrónicamente por:
JUNIOR RAFAEL FIGUEROA OLMEDO

Ing. Junior Figueroa Olmedo, Mgtr.
TUTOR

Firmado electrónicamente por:
PATRICIO JAVIER CRUZ DAVALOS

Ing. Patricio Cruz Dávalos, Ph.D.
DOCENTE ESPECIALISTA

EFREN VINICIO HERRERA MUENTES
Digitally signed by EFREN VINICIO HERRERA MUENTES
Date: 2025.02.07 11:13:35 -05'00'

Ing. Efrén Herrera Muentes, Ph.D.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
MARIA MARGARITA RIVERA GONZALEZ

Abg. María Rivera González, Mgtr.
SECRETARIA GENERAL
UPSE

UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por BARCIA AYALA ORLANDO GIOVANNI, como requerimiento para la obtención del título de Magíster en Electrónica y Automatización.

TUTOR

Firmado electrónicamente por:
JUNIOR RAFAEL
FIGUEROA OLMEDO

Ing. Junior Figueroa Olmedo, Mgtr.

Santa Elena, 13 de diciembre de 2024

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

DECLARACIÓN DE RESPONSABILIDAD

Yo, **BARCIA AYALA ORLANDO GIOVANNI**

DECLARO QUE:

El trabajo de Titulación, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE INTERACCIÓN COLABORATIVO ENTRE EL ROBOT HUMANOIDE NAO Y EL ROBOT OMNIDIRECCIONAL ROBOTINO PARA LA EJECUCIÓN Y COORDINACIÓN DE TAREAS DE LOCOMOCIÓN EN ENTORNOS CONTROLADOS previo a la obtención del título en Magíster en Electrónica y Automatización, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Santa Elena, 13 de diciembre de 2024

EL AUTOR

Firmado electrónicamente por:
ORLANDO GIOVANNI
BARCIA AYALA

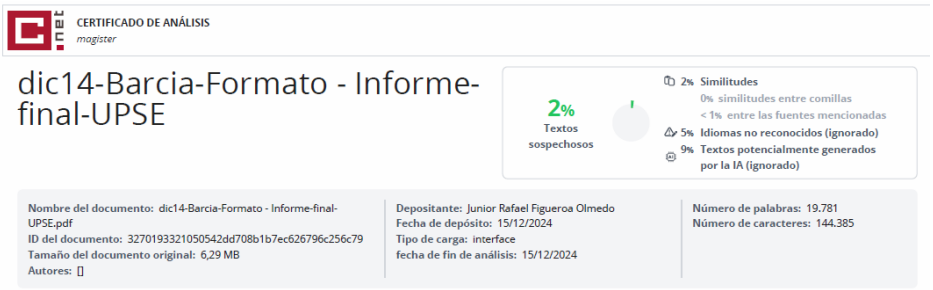
Barcia Ayala Orlando Giovanni

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE CIENCIAS DE LA INGENIERÍA
INSTITUTO DE POSTGRADO

CERTIFICACIÓN DE ANTIPLAGIO

Certifico que después de revisar el documento final del trabajo de titulación denominado DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE INTERACCIÓN COLABORATIVO ENTRE EL ROBOT HUMANOIDE NAO Y EL ROBOT OMNIDIRECCIONAL ROBOTINO PARA LA EJECUCIÓN Y COORDINACIÓN DE TAREAS DE LOCOMOCIÓN EN ENTORNOS CONTROLADOS, presentado por el estudiante, BARCIA AYALA ORLANDO GIOVANNI fue enviado al Sistema Antiplagio COMPILATIO, presentando un porcentaje de similitud correspondiente al 2%, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.


CERTIFICADO DE ANÁLISIS
magister

dic14-Barcia-Formato - Informe-final-UPSE

2% Textos sospechosos

2% Similitudes
0% similitudes entre comillas
< 1% entre las fuentes mencionadas
5% Idiomas no reconocidos (ignorado)
9% Textos potencialmente generados por la IA (ignorado)

Nombre del documento: dic14-Barcia-Formato - Informe-final-UPSE.pdf
ID del documento: 3270193321050542dd708b1b7ec626796c256c79
Tamaño del documento original: 6,29 MB
Autores: []

Depositante: Junior Rafael Figueroa Olmedo
Fecha de depósito: 15/12/2024
Tipo de carga: interface
fecha de fin de análisis: 15/12/2024

Número de palabras: 19.781
Número de caracteres: 144.385

TUTOR

Firmado electrónicamente por:
JUNIOR RAFAEL FIGUEROA OLMEDO

Ing. Junior Figueroa Olmedo, Mgtr.

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

AUTORIZACIÓN

Yo, **BARCIA AYALA ORLANDO GIOVANNI**

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación o parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la Institución.

Cedo los derechos en línea patrimoniales del Proyecto de titulación con componentes de investigación aplicada y/o de desarrollo con fines de difusión pública, además apruebo la reproducción de este proyecto de titulación dentro de las regulaciones de la Universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor.

Santa Elena, 13 de diciembre de 2024

EL AUTOR

Firmado electrónicamente por:
ORLANDO GIOVANNI
BARCIA AYALA

Barcia Ayala Orlando Giovanni

AGRADECIMIENTO

Agradezco a los docentes de la maestría, mi tutor profesor Junior Figueroa y mis compañeros de estudios por su guía. A mis autoridades de la Universidad Politécnica Salesiana y compañeros docentes de la Carrera de Electrónica y Automatización de la UPS sede Guayaquil.

Orlando Giovanni, Barcia Ayala

DEDICATORIA

Dedico el trabajo de titulación a mis hijos Giovanna, Orlandito, a mi madre Carmen, mi padre Orlando, mi esposa Bélgica y mi hermano Israel por su apoyo constante. El esfuerzo y sacrificio es por ellos.

Orlando Giovanni, Barcia Ayala

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La presente investigación referente a robótica móvil desarrolla un sistema colaborativo entre el robot humanoide NAO y el robot omnidireccional Robotino, enfocado en la ejecución y coordinación de tareas de locomoción en entornos controlados. Se utiliza OPC UA para la interacción entre ambos robots. El proyecto aplica técnicas de locomoción, navegación autónoma, evitación de obstáculos y monitoreo mediante visión artificial con herramientas como RobotinoView, Python, Naoqi, Choregraphe, y simuladores como Webots y Robotino Sim. Se efectúan diferentes actividades para moverse de un punto a un punto meta, evitando obstáculos. Los robots colaboran con el humano siguiendo por color a la persona. Los robots colaboran e interactúan entre ellos efectuando actividades al enviar señales de escritura y lectura de OPC UA entre uno y otro robot. La implementación y resultados reflejan un avance significativo en la integración y cooperación de robots móviles y humanoides en entornos controlados

Palabras claves: Robot Humanoide, Robot Omnidireccional, Navegación autónoma

ABSTRACT

This research, concerning mobile robotics, develops a collaborative system between the humanoid robot NAO and the omnidirectional robot Robotino, focused on the execution and coordination of locomotion tasks in controlled environments. OPC UA is used to interact with both robots. The project applies locomotion techniques, autonomous navigation, obstacle avoidance, and monitoring through artificial vision with tools such as RobotinoView, Python, Naoqi, and Choregraphe, and simulators such as Webots and Robotino Sim. Different activities are carried out to move from one point to a goal point, avoiding obstacles. The robots collaborate with the human by following the person by color. The robots collaborate and interact with each other by carrying out activities by sending OPC UA read and write signals between one robot and another. The implementation and results reflect a significant advance in the integration and cooperation of mobile robots and humanoids in controlled environments.

Keywords: Humanoid Robot, Omnidirectional Robot, Autonomous Navigation

INTRODUCCIÓN

En el campo de la robótica, la colaboración entre diferentes tipos de robots tiene relevancia debido a su potencial para mejorar la eficiencia y la capacidad de adaptación en tareas complejas y de automatización. Los robots colaborativos -cobots- han cambiado la forma en que se realizan las tareas industriales y de servicio (Benitti, 2012). Estos permiten una interacción segura y eficiente entre humanos y robots, donde pueden compartir el espacio de trabajo con personas, realizar tareas difíciles y adaptarse a diferentes situaciones sin comprometer la seguridad (Djuric et al., 2018).

La interacción entre robots heterogéneos, como los robots humanoides y los robots móviles de ruedas omnidireccionales, permiten la combinación de habilidades complementarias, para mejorar la eficiencia y versatilidad de los sistemas robóticos lo que es útil en entornos controlados como fábricas, laboratorios y centros educativos (Othman & Yang, 2023).

El robot NAO es un robot móvil humanoide con 25 grados de libertad que es capaz de realizar tareas de locomoción bípeda autónoma. interactuar con humanos y otros robots (Amirova et al., 2021). Robotino es un robot móvil omnidireccional que destaca por su movilidad, y movimiento en espacios reducidos para realizar tareas de navegación autónoma y transporte en entornos industriales (Ali & Ali, 2015). Ambos robots son herramientas para el estudio de la robótica colaborativa debido a su capacidad para realizar tareas de locomoción, manipulación y navegación autónoma en entornos controlados, No obstante, sigue siendo difícil integrar estos robots en un sistema colaborativo de manera efectiva (Huang & Lin, 2018; Mekonnen et al., 2016; Sabattini et al., 2019).

Huang & Lin (2018) manifiestan sobre un sistema de varios robots para realizar tareas colaborativas, enfatizando la coordinación y comunicación eficiente entre varios robots. Al-Dahhan et al. (2020)describen métodos de control para robots omnidireccionales y evitación de obstáculos con un enfoque en el Robotino en entornos desconocidos, destacando su utilidad en campos educativos y de investigación (Andrakhanov & Belyaev, 2017)

El proyecto se centra en el desarrollo e implementación de un sistema de interacción colaborativa entre el robot humanoide NAO y el robot omnidireccional Robotino para la ejecución y coordinación de tareas de locomoción en entornos controlados (Jin et al., 2022; Kashyap et al., 2020; Sahni et al., 2019) . La propuesta contribuye al desarrollo de nuevas metodologías que permitan integrar y mejorar las capacidades de estos robots en aplicaciones industriales y educativa, impactando directamente en la eficiencia de procesos, abriendo nuevas posibilidades e investigaciones en el campo de la robótica y la automatización en sistemas autónomos y colaborativos. (Banaeian & Gilanlioglu, 2021; Gardenghi & Gherardi, 2024; Ouafa et al., 2024).

La integración de estos dos robots involucra desafíos técnicos en la electrónica, la automatización, ejecución de tareas de locomoción y la utilización de protocolos de comunicación para interactuar con sensores y actuadores de los sistemas embebidos de cada robot en entornos controlados e implementar algoritmos de navegación autónoma y telemática (Al-Dahhan et al., 2020; Andrakhanov & Belyaev, 2017; Bragança et al., 2019) .

En el robot NAO se utiliza el framework Naoqi creado por Softbank Robotics que controla el robot, siendo requerida la programación paralela y sincronización. El robot contiene sensores ultrasónicos, de contacto y ultrasónico. El middleware Naoqi actúa como el cerebro del robot NAO, gestionando sus capacidades sensoriales, motoras y de comunicación para ofrecer una plataforma coherente para el desarrollo de aplicaciones robóticas avanzadas (SoftBank, 2024). En el robot Robotino se utiliza el Software de Festo Robotino View, Robotino SIM para interactuar con el Robot, controlar su movimiento y el uso de sensores (Al-Dahhan et al., 2020).

Hay pocas experiencias previas en la Universidad Politécnica Salesiana sede Guayaquil, Carrera de Electrónica referente a la integración de tecnologías como robots colaborativos y locomoción robótica en la interacción entre el robot NAO y Robotino en entornos controlados

El tema es relevante de forma académica, por su impacto tecnológico y aplicaciones en el mundo real y social. El desarrollo de un sistema que permita que NAO y Robotino trabajen juntos puede cambiar la forma en que se utilizan tanto en la industria como en la educación.

Planteamiento de la investigación (Fundamentación de la investigación)

La robótica colaborativa es un área de rápido crecimiento que se enfoca en desarrollar sistemas donde varios robots trabajan juntos de manera segura y efectiva. La integración del robot humanoide NAO, con el robot móvil omnidireccional Robotino, ha demostrado de forma individual ser una estrategia para mejorar la ejecución de tareas como la navegación autónoma y la interacción con entornos dinámicos o controlados (Abdo et al., 2020; Ciurea et al., 2014; Yandún & Sotomayor, 2012)

Los robots al colaborar de manera coordinada pueden aumentar significativamente la eficiencia y la productividad en tareas de locomoción. Este tipo de innovación puede impulsar la automatización y la inteligencia artificial a utilizar robots colaborativos en varios sectores. En el aula pueden ayudar a los estudiantes a aprender sobre robótica y programación. En la industria estos robots pueden colaborar en tareas de manufactura, ensamblaje y logística, aumentando la productividad y reduciendo los errores humanos (Bragança et al., 2019; Gardenghi & Gherardi, 2024). El uso de robots colaborativos también tiene ventajas sociales. Pueden ayudar en tareas de cuidado y rehabilitación brindando apoyo en entornos donde la interacción humana es limitada(Cooper et al., 2021; Moro et al., 2019). Además, la implementación de estos sistemas en la educación puede motivar a las próximas generaciones a interesarse por la robótica y las tecnologías emergentes (Latip et al., 2020).

La fundamentación de esta investigación radica en su impacto académico y práctico, dado que contribuye a la evolución de la robótica colaborativa y ofrece nuevas oportunidades para mejorar la automatización en diversos sectores. En el ámbito educativo, el sistema permitirá a los estudiantes experimentar con robots avanzados, potenciando su aprendizaje en áreas como la programación, locomoción, navegación, monitoreo de sensores y uso de actuadores (Djuric et al., 2018). En el ámbito industrial, el desarrollo de este sistema contribuirá a la mejora de la eficiencia en tareas como la logística y la manufactura, reduciendo errores humanos y optimizando los procesos (Bragança et al., 2019). El trabajo de titulación se estructura en varios capítulos, se inicia con el marco teórico referencia que expone la robótica colaborativa, la locomoción robótica, la arquitectura y control del robot humanoide NAO y el omnidireccional Robotino.

Posteriormente se presenta la metodología que se utiliza para lograr la locomoción e interacción colaborativa entre estos robots, describiendo los desafíos técnicos y las soluciones implementadas, como la sincronización de tareas y la integración de sensores. A continuación, se detallan los experimentos, aplicación de algoritmos realizados en entornos controlados, resultados obtenidos y discusión. Finalmente se concluye con una evaluación del impacto de esta colaboración en el ámbito educativo e industrial, abriendo nuevas oportunidades de investigación y aplicación en el campo de la robótica autónoma.

Formulación del problema de investigación

En los últimos años, la robótica colaborativa ha avanzado mucho, lo que permite que varios tipos de robots trabajen juntos para realizar tareas complejas de manera segura y eficiente. Los robots humanoides, como NAO, y los robots omnidireccionales, como Robotino, son herramientas versátiles que han encontrado uso en la industria, la educación y la investigación. No obstante, sigue siendo difícil integrar estos robots en un sistema colaborativo de manera efectiva.

En referencia al contexto indicado, uno de los principales desafíos en la robótica colaborativa es desarrollar e implementar sistemas que permitan la interacción efectiva entre diferentes tipos de robots en entornos controlados. En sí, la comunicación del robot humanoide NAO con la movilidad y flexibilidad del robot omnidireccional Robotino presenta varias desafíos técnicos y operativos. Estas incluyen la comunicación eficiente entre robots, la coordinación de tareas, la navegación autónoma precisa y la interacción con los usuarios humanos.

La formulación del problema es:

¿Cómo desarrollar un sistema de interacción colaborativo entre el robot humanoide NAO y el robot omnidireccional Robotino en entornos controlados, que permita una comunicación telemática, autónoma, ejecución y coordinación de tareas de locomoción?

Objetivo General:

Desarrollar e implementar un sistema de interacción colaborativo entre el robot humanoide NAO y el robot omnidireccional Robotino con protocolos de comunicación, algoritmos y técnicas de navegación autónoma para la ejecución y coordinación efectiva de tareas de locomoción en entornos controlados.

Objetivos Específicos:

1. Aplicar algoritmos de comunicación, coordinación y planificación de tareas de locomoción que permitan a NAO y Robotino trabajar de manera colaborativa en entornos controlados utilizando el framework Naoqi, Python, Choregraphe y robotinoView.
2. Implementar un sistema de navegación autónoma y evitación de obstáculos para Robotino y NAO para efectuar la locomoción y comportamiento de movimiento del robot omnidireccional y bípedo.
3. Implementar el sistema de monitoreo con visión artificial para robot Nao y Robotino.
4. Simular la navegación autónoma y evitación de obstáculos del Robot NAO con Webots, Choregraphe y Robotino Sim.
5. Parametrizar y configurar el sistema embebido del robot NAO y Robotino por plataforma web y local.
6. Ejecutar interfaces de usuario que permitan monitorear el sistema colaborativo entre NAO y Robotino.

Planteamiento hipotético

La implementación de un sistema de interacción colaborativo entre el robot humanoide NAO y el robot omnidireccional Robotino con protocolos de comunicación, algoritmos y técnicas de navegación autónoma, permitirá una ejecución y coordinación efectiva de tareas de locomoción en entornos controlados.

CAPÍTULO 1. MARCO TEÓRICO REFERENCIAL

1.1 Revisión de literatura

Magallán-Ramírez et al. (2022) describe la navegación en laberintos del robot Nao utilizando visión por computadora, planificación de rutas y aprendizaje colaborativo, para lograr comunicación y mejorar la interacción entre robots.

El robot móvil necesita mecanismos para moverse de un lugar a otro en un entorno, por lo cual Goualilier et al. (2010) expone el modelo de aminada omnidireccional en lazo cerrado. Mientras que Kofinas (2012) muestra el modelo cinemática y parámetros DH del robot nao.

El trabajo de R et al. (2023) explica las fortalezas de interacción del robot NAO por reconocimiento de voz y gestos mediante Python y mejorar la HRI Human Robot Interaction o interacción humano robot para diferentes aplicaciones como la educación, salud y entretenimiento.

Según Yang et al. (2022) describe en su artículo el grasping o agarre robótico utilizando atributos de objetos para detección, no está orientado para la locomoción, la navegación o comunicación del robot Nao.

K. Chen et al. (2020) describe el control del robot NAO en términos de visión, integrando técnicas avanzadas de procesamiento de imágenes para el reconocimiento de objetos, personas y ambientes. El robot NAO integrar sistemas de visión para reconocimiento de objetos, evitación de obstáculos, navegación utilizando sensores de sonar, algoritmos SLAM (simultaneous localization and mapping) y algoritmo mejorado de colonia de abejas artificiales para una búsqueda óptima de trayectorias (Jin et al., 2022).

Titov et al. (2017) describe la locomoción y navegación autónomas del robot antropomórfico AR-600E pero se aplican estrategias similares al Robotino.

Mahgoub & Sanhoury (2018) describe un planificador de navegación adaptativo para controlar relativo a la locomoción de un robot móvil con ruedas y asi lograr el seguimiento efectivo de la trayectoria.

Rudin et al. (2022) indica en su artículo las habilidades avanzadas para la locomoción y la navegación con aprendizaje de refuerzo profundo, mejorando las capacidades del robot en terrenos complejos con modos de andar energéticamente eficientes.

Según Diaz et al. (2022) expresa en su trabajo como Robotino emplea un sistema móvil con actuadores limitados, centrándose en aplicaciones educativas. De forma similar indica la interacción con elementos externos como el Lynxmotion AL5 para ampliar la funcionalidad de Robotino.

Relativo a navegación y control Karydis et al. (2017) describe las estrategias de navegación para sistemas robóticos similares que implican enfoques basados en modelos que mejoran el seguimiento de rutas y la adaptabilidad.

1.2 Desarrollo teórico y conceptual

1.2.1 Robótica

Existen diferentes definiciones de robot, Figueroa et al. (2023) indican que es un dispositivo reprogramable que interactúa con su entorno para llevar a cabo diversas funciones, tareas, a través de acciones autónomas (inteligentes) producidas a partir de los datos obtenidos por sus sensores (Lynch & Park, 2017).

La robótica es multidisciplinaria, por la aplicación de diferentes áreas del conocimiento en electrónica, informática, inteligencia artificial, matemáticas, física, teoría de control (Kabir et al., 2023). Los robots manipuladores industriales tienen un gran éxito comercial tienen desventajas en su falta de amplia movilidad, por su limitado rango de movimiento (Zafar & Mohanta, 2018). Los robots móviles tienen como una de sus funciones moverse a algún lugar, que puede ser especificados en función de alguna característica del entorno como un coordenada geométrica o mapa. Existen variedades de robots móviles que pueden moverse sobre el suelo, por el aire, por el agua. En sí, el robot debe tomar un camino para llegar a su destino y tendrá desafíos como obstáculos que pueden bloquear su camino, no tener algún mapa o un mapa (Craig, 2017; Siegwart et al., 2011).

En un robot móvil, una de las metas es que su movimiento sea autónomo para cumplir una actividad (W. Chen et al., 2024) . Por lo tanto, se deben efectuar algoritmos para que el robot realice:

- Locomoción. Moverse de forma física y mecánica de un punto determinado a otro

- Percepción. Detectar o identificar objetos
- Localización. Determinar donde se encuentra el robot
- Navegación y mapeo. Seleccionar la mejor ruta para llegar a un determinado lugar

Los robots móviles autónomos han transformado tanto la percepción como la interacción con la automatización, siendo la navegación y la evitación de obstáculos desafíos clave en su implementación en diversos entornos (Dhananji & Sharmilan, 2024).

Los robots en su parte mecánica se componen por elementos conocidos como eslabones o links, que se unen mediante articulaciones o joints. Estas articulaciones permiten un movimiento contiguo entre dos eslabones (Spont et al., 2006).

Un Grado de libertad (DOF Degree of Freedom) se describe como cada uno de los movimientos independientes que cada una de las articulaciones realiza, referente a la articulación anterior (Barrientos et al., 2007; Niku, 2020).

1.2.2 Locomoción

En robótica, la locomoción se refiere a los métodos y tecnologías que se utilizan para permitir que los robots se muevan de manera eficiente y se adapten a diversos entornos. Los avances recientes en este campo se han centrado en inspirarse tanto en la biología como en los sistemas mecánicos para mejorar el movimiento de los robots. Se han desarrollado robots humanoides, diseñados con sistemas de control de piernas que imitan la locomoción humana, para lograr una mayor agilidad y equilibrio en diversos terrenos mediante la integración de unidades de medición inercial (IMU) y sistemas de control motor para movimientos precisos de las piernas (Muntean & Leba, 2024) .

La tierra es el medio natural de locomoción de robots terrestres. Estos se clasifican en robots de patas, ruedas u orugas. Existen robots de dos patas (bípedos), de cuatro patas (cuadrúpedos), de seis patas (hexápodos) y más. Los robots de ruedas uno de ellos son los de tipo omnidireccional (Niku, 2020; Siegwart et al., 2011) .

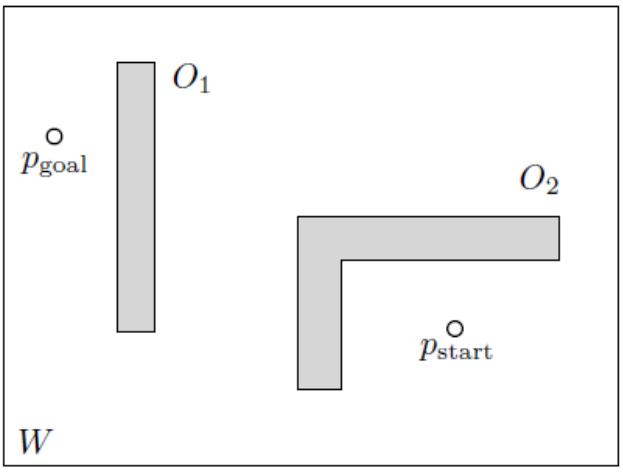
1.2.3 Planificación del movimiento (Motion Planning)

La planificación de movimientos radica en mover un robot desde una posición de inicio A hasta una posición fin o de destino B, evitando obstáculos. La planificación basada en sensores (Sensor-based Planning) para mover un punto en el plano bidimensional, considera que el robot tiene sensores y según los resultados de la medición del sensor, se

planifica el movimiento desde el inicio hasta la meta. En la solución se requiere conocer las capacidades e información que tiene el robot. La tarea es planificar el movimiento del robot desde el punto de partida hasta el punto destino, con una serie de pasos a ejecutar (no secuencia precalculada) para enfrentar los probables obstáculos que el robot encuentra en el camino (Bullo & Smith, 2016; Klancar, Zdesar, Blazic, & Skrjanc, 2017)

En la Figura 1 se describe el espacio de trabajo W que es subconjunto de $\mathbb{R}^2$ *o* $\mathbb{R}^3$, con obstáculos $O_1, O_2, .., O_n,$ O1, un punto de inicio p_{start} pstart y una meta p_{goal} y el robot descrito como un punto en movimiento (robot de tamaño 0)

Figura 1
Entorno de una planificación basada en sensores



Nota: (Bullo & Smith, 2016)

El robot debe cumplir las siguientes suposiciones

- conoce la dirección hacia la meta,
- conoce la distancia en línea recta entre él y la meta (goal)
- no conoce nada sobre los obstáculos (número, ubicación, forma, etc.),
- tiene un sensor de contacto que le permite detectar obstáculos localmente,
- puede moverse en línea recta hacia la meta o puede seguir un límite de obstáculos (probablemente usando su sensor de contacto)
- tiene una memoria limitada en la que puede almacenar distancias y ángulos

Referente al entorno se indican los siguientes supuestos sobre el espacio de trabajo y obstáculos.

- el espacio de trabajo está delimitado
- solo hay un número finito de obstáculos
- los puntos de inicio y de destino están en el espacio de trabajo libre Wfree, y
- cualquier línea recta dibujada en el entorno, traspasa el límite de cada obstáculo un número finito de veces

La evitación de obstáculos su fin es cambiar la trayectoria del robot según lo indicado por los sensores durante el movimiento del robot. El movimiento generado es una función de las lecturas actuales de posición objetivo y su ubicación relativa a la posición objetivo. Estos algoritmos dependen de un mapa global y conocimiento que tenga el robot de su ubicación en el mapa (Corke, 2011).

1.2.3.1 Algoritmo Bug 0

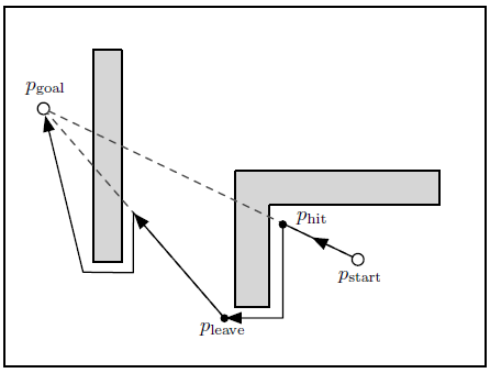
Si encuentra un obstáculo, se describe el algoritmo para evitar el obstáculo y avanzar hacia la meta (goal). (Bullo & Smith, 2016; Klancar, Zdesar, Blazic, & Skrjanc, 2017).

Tabla 1
Algoritmo Bug 0

```
1: while not meta:
2:          move towards goal
3:          if hit obstaculo :
4:                    while no se puede avanzar hacia la meta:
5:                              seguir el límite del obstáculo moviéndose hacia la izquierda
```

El algoritmo requiere solo los valores más actuales de los sensores del robot y datos aproximados del punto meta. La dirección del movimiento izquierda o derecha es fija pero no importante. El algoritmo Bug 0 no siempre obtiene un camino hacia la meta. Hay situaciones que, si existe solución para llegar a la meta, pero el algoritmo no la encuentra.

Figura 2
Ejecución de algoritmo Bug 0



Nota: Ejecuciones sucesivas del algoritmo bug 0

1.2.3.2 Algoritmo Bug 1

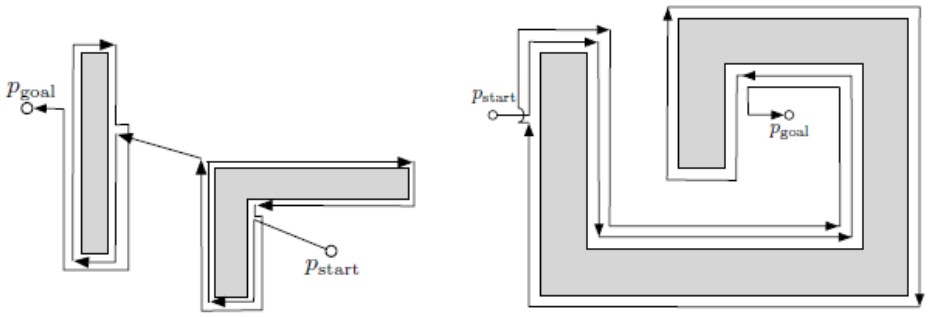
El algoritmo basado en sensores bug 0, no utiliza todas las capacidades del robot como la distancia hacia la meta ni memoria, por lo cual se indica el Algoritmo Bug 1 (Bullo & Smith, 2016; Klancar, Zdesar, Blazic, & Skrjanc, 2017).

Tabla 2
Algoritmo Bug 1

1:	while not goal :
2:	move towards goal
3:	if hit an obstacle :
4:	circunnavega (mover a la izquierda o derecha no es importante). While la circunnavegación, almacene en memoria la distancia mínima desde el límite del obstáculo al objetivo.
5:	Seguir el límite hasta el punto límite con la distancia mínima hasta la meta.

Nota: La diferencia entre Bug 0 y Bug 1 es la reacción al obstáculo detectado en el bloque if.

Figura 3
Algoritmo Bug 1



Nota: Dos sucesivas ejecuciones de algoritmo Bug 1.

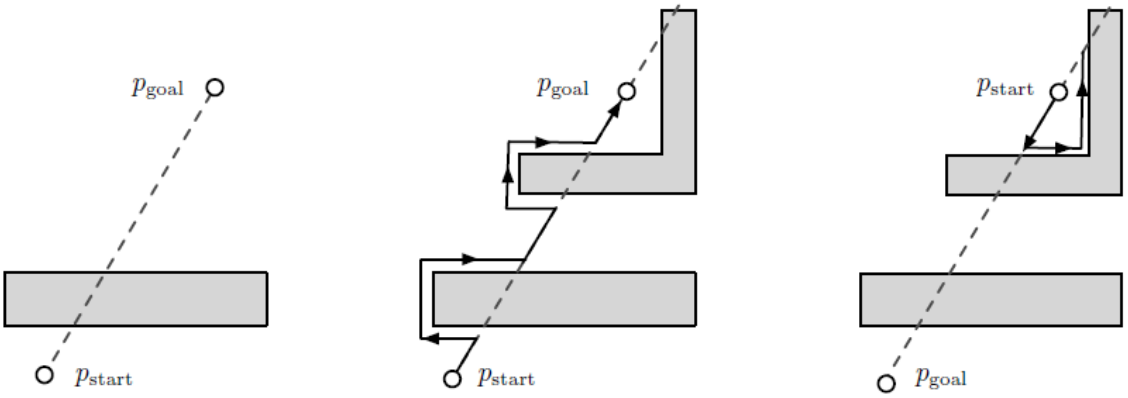
1.2.3.3 Algoritmo Bug 2

En bug 1 cada obstáculo debe ser explorado totalmente antes de que el robot pueda avanzar a la meta, el bug 2 establece una línea recta entre el inicio y final. Se describe el algoritmo y un escenario se muestra en la Figura 4 1 (Bullo & Smith, 2016; Klancar, Zdesar, Blazic, & Skrjanc, 2017).

Tabla 3
Algoritmo Bug 2

1:	while not goal :
2:	move towards goal (lo largo de la línea de salida-meta)
3:	if hit obstáculo :
4:	Seguir el límite del obstáculo (moviéndose hacia la izquierda o hacia la derecha) hasta que encuentre nuevamente la línea de inicio y pueda avanzar hacia la meta.

Figura 4
Ejecución de Bug 2



1.2.4 Movimiento Holonómico

En robótica el movimiento holonómico y no holonómico es relativo a las restricciones en los movimientos que puede efectuar el robot, fundamentado entre los grados de libertad del sistema y las capacidades de los actuadores (Ma et al., 2024).

Un sistema es holonómico si los grados de libertad (DOF) que tiene para moverse es igual a la cantidad de variables independientes que puede controlar directamente. Los robots omnidireccionales que pueden moverse de forma libre en todas las direcciones del plano como adelante, atrás, izquierda, derecha y giro sobre su eje. Las restricciones de los actuadores, no limitan las direcciones posibles de movimiento (W. Chen et al., 2024) .

Un sistema es no holonómico, si el número de grados de libertad es mayor que la cantidad de variables independientes controlables. Por lo tanto, no puede moverse directamente en todas las

direcciones posibles, el movimiento está restringido a varias trayectorias o combinación de movimientos. Los robots de ruedas estándar, no puede moverse directamente de lado a lado. Para moverse debe efectuar combinación de movimientos y giros.(Festo, 2024)

1.2.5 Navegación autónoma

La navegación robótica es guiar a un robot hacia una meta. Se pueden utilizar mapas, pero muchas tareas robóticas se logran sin ningún mapa, pero utilizando la navegación reactiva (reactive navigation), tal como seguir una luz, una línea blanca en el suelo, siguiendo una pared (Corke, 2011).

La navegación autónoma es la capacidad de un robot para moverse de manera independiente, utilizando sensores y algoritmos avanzados para planificar su trayectoria y evitar colisiones con obstáculos. La evitación de obstáculos permite a los robots detectar objetos en su entorno y ajustar su camino para prevenir accidentes. Este proceso se basa en técnicas de control híbrido y algoritmos que fusionan datos de múltiples sensores (como cámaras RGB-D, IMU y GPS) para mejorar la eficiencia y seguridad en la navegación robótica (Katona et al., 2024).

1.2.6 Interacción o colaboración de robots.

La interacción de robots implica la capacidad de colaborar entre sí o con humanos, compartiendo información y coordinando sus movimientos en tareas complejas. Los avances en la fusión de datos y la detección de personas en tiempo real, como los indicados por Souza Bezerra et al. (2023) que han permitido que los robots autónomos eviten colisiones incluso con objetos en movimiento, mejorando su tasa de éxito en misiones complejas. Estos desarrollos continúan impulsando la innovación en la robótica, logrando que los robots sean más adaptables y seguros en entornos dinámicos (Cao et al., 2024) .

La interacción colaborativa con robots humanoides y omnidireccionales implica la cooperación entre robots y humanos en un entorno de trabajo conjunto, donde los robots complementan las capacidades humanas para ejecutar tareas complejas. Estos robots, diseñados para imitar la forma y el comportamiento humano, permiten interacciones más naturales y eficientes, mejorando la colaboración en diversos contextos. Es importante utilizar simulaciones para garantizar la seguridad y eficiencia en la interacción para verificar el comportamiento colaborativo (W. Chen et al., 2024; Zagirov et al., 2023).

1.2.7 Robot Nao

El robot NAO es un androide humanoide desarrollado por SoftBank Robotics, tiene 25 grados de libertad, diseñado inicialmente para la investigación y la enseñanza en el ámbito de la robótica y la inteligencia artificial. El sistema operativo que lo controla es NAOqi, un software que permite gestionar la interacción entre el hardware del robot y sus capacidades de software, tales como el reconocimiento de voz, visión, y movimientos coordinados. Este sistema es ejecuta los algoritmos que permiten al robot NAO llevar a cabo tareas complejas y personalizables. A su vez, el software Choregraphe es una herramienta gráfica que permite programar y simular comportamientos del robot NAO sin necesidad de conocimientos avanzados de programación. Mediante una interfaz visual, los usuarios pueden crear, probar y modificar scripts que controlan las acciones del robot, facilitando su uso en entornos educativos y de investigación. La combinación de estas tres tecnologías hace que NAO sea accesible y flexible para una amplia gama de aplicaciones, desde la enseñanza de robótica hasta la investigación en interacción humano-robot. Las articulaciones en el robot NAO son de rotación (SoftBank, 2024).

1.2.7.1 Naoqi

NAOqi es un sistema operativo diseñado específicamente para el robot NAO, desarrollado por SoftBank Robotics. Su principal diferencia con otros sistemas operativos de robots, como ROS (Robot Operating System), radica en su diseño cerrado y específico para hardware propietario. NAOqi está optimizado para controlar las funcionalidades de los robots NAO y Pepper, brindando acceso directo a sus sensores, actuadores y capacidades cognitivas de manera integrada, lo que lo hace más fácil de usar para usuarios con menos experiencia en programación (Softbank, 2018).

El uso de robots sociales, como el Pepper que es de la misma línea del robot NAO destaca la necesidad de que los robots inicien interacciones sociales para facilitar la colaboración natural entre humanos y robots en entornos sociales (Bal et al., 2024). La implementación de conversaciones vocales naturales mejora significativamente la eficiencia en tareas colaborativas, permitiendo que los robots realicen preguntas o proporcionen actualizaciones para una mejor coordinación. Estos avances tecnológicos están impulsando el desarrollo de sistemas de interacción colaborativa más efectivos y seguros (Ferrari et al., 2023).

1.2.8 Robotino Festo

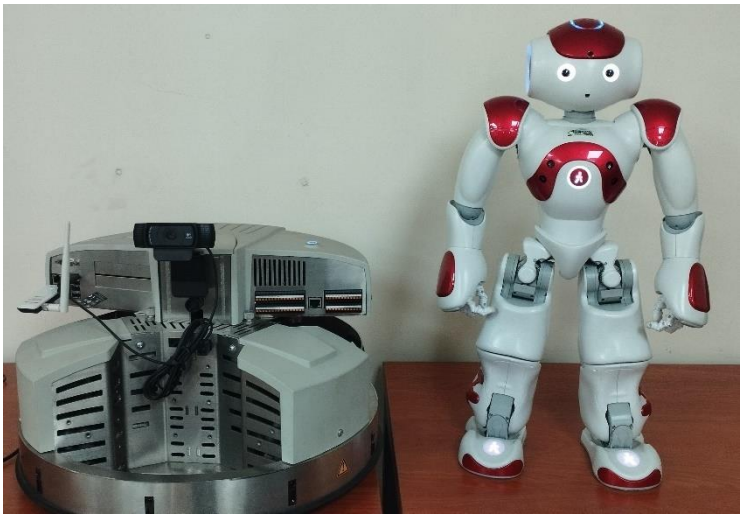
Robotino de FESTO es un robot móvil omnidireccional utilizado para la enseñanza y la investigación en áreas como la robótica y la inteligencia artificial. Está equipado con sensores, cámaras y controladores para navegar de manera autónoma en entornos complejos, permitiendo la simulación y el desarrollo de algoritmos avanzados de control y planificación de movimiento. Robotino View es el software de programación gráfica utilizado para controlar Robotino, ofrece una interfaz visual basada en bloques que permite diseñar algoritmos y configuraciones sin necesidad de conocimientos avanzados de programación. Robotino SIM es un simulador que permite replicar los entornos en los que opera Robotino, proporciona un espacio virtual donde los usuarios pueden probar y ajustar algoritmos antes de implementarlos en el robot real. Estos simuladores son útiles para verificar entre la simulación y la realidad, permitiendo realizar pruebas sin riesgos físicos y reduciendo el tiempo de desarrollo (FESTO, 2022; Gucwa & Cheng, 2015)

Tabla 4
Síntesis de características Robotino y Nao

Característica	Robotino	Nao
Versión	V3	V5
Peso	20 Kg, carga 30 Kg max	5.3 Kg
Alto	altura incluida la carcasa del mando a 29 cm.	57.4 cm
Ancho	diámetro: 45 cm	27.5 cm
Profundidad		31.1 cm (brazos extendidos&
Sensor	9 sensores de distancia infrarrojos, Bumper	Ultrasónico 2 Infrarrojos bumper
Sensor. Cámara	Cámara en color con resolución Full HD 1080p	Tipo webcam Hd
Procesador	Intel Atom, 1,8 GHz, Dual-Core	Intel Atom z530 1.6 GHz
Memoria	4 GB RAM, 32 GB SSD	1 GB Ram
Conectividad	WLAN como cliente o Access Point	Ethernet y Wifi
Autonomía	1 hora	60 minutos activo 90 minutos normal

El detalle de las demás características se describe en los anexos.

Figura 5 Robot Omnidireccional Robotino y bípedo NAO



Nota: Se muestran los robots Reales Robotino Festo y Robot Nao (FESTO, 2022; SoftBank, 2024).

1.2.9 Grafcet

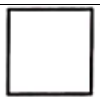
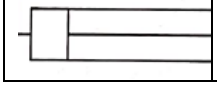
GRAFCET (Graphe Fonctionnel de Commande Étape/Transition) es un lenguaje gráfico que se utiliza para el diseño y representación de sistemas de control que pasan de un estado a otro en función de ciertas condiciones. Es normalizado por IEC 60848 , siendo una extensión del diagrama de estados y transiciones, aplicados a los sistemas de control programables PLC y otros (Yuste, 2017).

Los diagramas son estructurados para representar el comportamiento del sistema, utilizado etapas (steps, transiciones (transitions) y condiciones.

- Etapas (steps) representan condiciones o estados en las que está el sistema
- Transiciones (transitions) , son condiciones que permiten ir de una etapa a otra
- Acciones (actions), son operaciones que se efectúan cuando el sistema se encuentra en una etapa determinada
- Condiciones lógicas, determinan si una transición se puede ejecutar

En la Tabla 5, se describe los símbolos, nombre y función

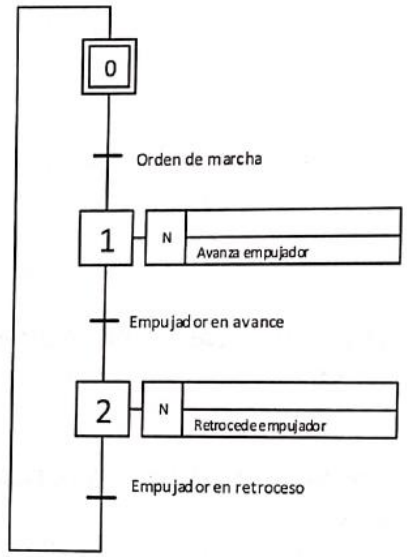
Tabla 5
Símbolos de Grafcet

Símbolo	Nombre	Descripción
	Etapa inicial	Inicio de la secuencia y modo ejecutar en el autómata
	Etapa	La activación inicia una acción o espera
	Unión	Unir varias etapas
	Transición	Condición para desactivar etapa en curso y activarse etapa siguiente
	Direccionamiento	Indica la activación de una u otra etapa en función de las condiciones. Se conoce como convergencia O
	Proceso simultáneo	Indica la activación o desactivación de varias etapas a la vez. Se conoce como convergencia Y
	Acciones asociadas	Acciones que se realizan al activarse la etapa

Nota: (Yuste, 2017)

Se muestra en la Figura 6 un ejemplo de grafcet

Figura 6.
Símbolos de GRAFCET



1.2.10 Comunicación y protocolos

Entre los protocolos y estándares de comunicación utilizados se describe:

1.2.10.1 TCP/IP (Transmission Control Protocol/Internet Protocol)

TCP/IP es un protocolo estándar utilizado en la mayoría de las redes y permite la comunicación confiable entre dispositivos conectados. Se basa en la transmisión y recepción de paquetes de datos de manera confiable. Entre las ventajas es la fiabilidad en la transmisión de datos, ya que garantiza la entrega de los mensajes sin errores. Soportado de forma nativa por casi todas las plataformas y dispositivos, incluidos NAO y Robotino. Es fácil de implementar y altamente adaptable. Es recomendado para aplicaciones donde la fiabilidad en la entrega de mensajes es más importante que la velocidad (Goralski, 2017). Puede ser ideal para intercambiar información crítica entre NAO y Robotino, como el estado de finalización de tareas o la sincronización de eventos importantes. Entre las desventajas es una latencia más alta en comparación con otros protocolos como MQTT, debido al proceso de confirmación de recepción y corrección de errores (Pulver, 2019).

1.2.10.2 WebSocket

WebSocket es un protocolo de comunicación en tiempo real que permite una comunicación bidireccional sobre un único socket TCP. Es ideal para aplicaciones web, pero también puede ser útil en sistemas robóticos. Entre las ventajas es la comunicación en tiempo real y bidireccional, lo que permite a NAO y Robotino intercambiar información de forma continua sin la sobrecarga de múltiples conexiones. Bajo consumo de recursos y eficiente para transmitir datos en tiempo real. Fácil de implementar si hay un entorno web involucrado (por ejemplo, interfaces de usuario basadas en navegador). Es recomendado para proyectos donde se necesita una comunicación bidireccional continua entre los robots y donde la interacción con interfaces web es importante. Entre las desventajas no tiene tantas características avanzadas de manejo de errores como otros protocolos como TCP/IP (Blokdyk, 2022).

1.2.10.3 OPC (Open Platform Communications) y OPC UA

OPC (OLE for Proccess Control) es un conjunto de estándares para la comunicación entre sistemas y dispositivos en la automatización industrial. OPC UA (Unified Architecture) es una evolución de OPC, su arquitectura unificada lo hace independiente del fabricante. Esto permite la interoperabilidad entre varios sistemas y/o dispositivos. Utiliza tecnologías de internet TCP/IP, HTTP, Websockets, servicios web (Li et al., 2020)

En robótica, OPC UA integra varios sistemas robóticos y facilita la comunicación e intercambio de datos entre robots y otros elementos de la automatización industrial (InfoPLC, 2024; OPC Foundation, 2024).

1.2.11 Matriz de transformación homogénea

Es una matriz T que representa la transformación de un vector de coordenadas de un sistema a otro. Su dimensión de 4x4 (Craig, 2017).

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} Rotación & Traslación \\ Perspectiva & Escalado \end{bmatrix} \quad (1)$$

La matriz homogénea T, está formada por cuatro submatrices de diferentes tamaños. En robótica se utiliza solo R_{3x3} y p_{3x1} considerando los otros elementos nulos.

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Rotación & Traslación \\ 0 & 1 \end{bmatrix} \quad (2)$$

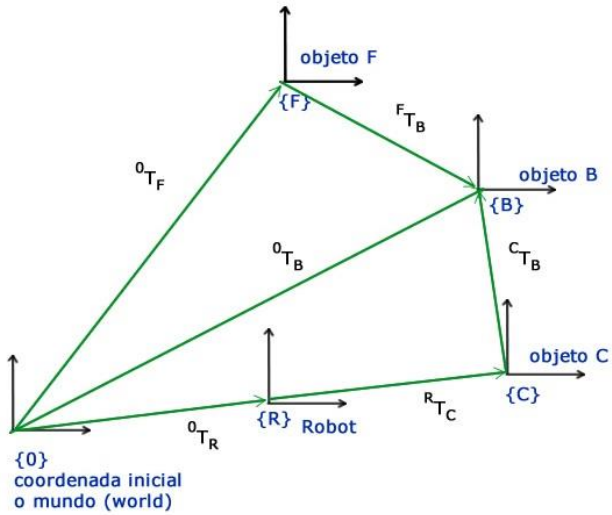
La Matriz de rotación R es ortonormal $R^{-1}=R^{T}$.

En robótica se utiliza se utilizan la composición de matrices homogénea para obtener la posición en base a otras posiciones relativas, así como en el cálculo de parámetros Denavit-Hartenverg. La notación para una transformación de A hacia B, o B relativo a es ${}^{A}T_{B}$.

Al existir composición de varias transformaciones, se representa con un gráfico de transformación o direct grahp, y se obtiene la relación entre el elemento A y B. Se debe ir multiplicando las matrices de transformación relativas al arco del gráfico, desde el punto A y seleccionar los arcos que lleguen a B. Si la relación es en sentido contrapuesto a las flechas, se debe utilizar la matriz inversa o cambiando el orden del índice en la transformación. Toda relación se puede obtener del gráfico.

En la Figura 7 se indican los objetos F, B ,R y C respecto al Frame 0. Estos tienen posiciones relativas a otros objetos. Con la composición de matrices de transformación se obtienen las posiciones respecto a otros Frames.(Barrientos et al., 2007)

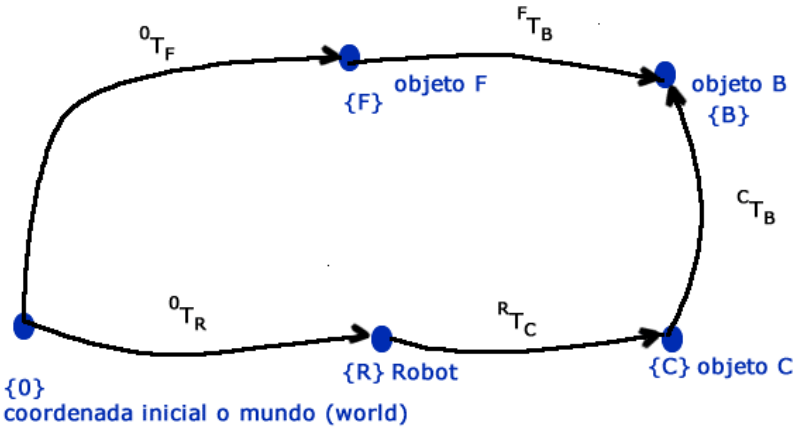
Figura 7.
Posiciones absolutas a F0 y relativas a otros objetos.



Aplicando un gráfico directo o de transformación se representa un comportamiento. Si se desea calcular la ruta desde el frame O mundo de F a R, se realiza por una multiplicación de Matrices Homogéneas. Se debe considerar que las operaciones no son conmutativas.

$$^{F}T_{R} = \left(^{O}T_{F}\right)^{-1} {}^{O}T_{R} \qquad (3)$$

Figura 8.
Gráfico de transformación o directo



Nota:(Barrientos et al., 2007; Craig, 2017; Lynch & Park, 2017)

1.2.12 Cinemática del robot móvil

Se debe obtener el comportamiento mecánico del robot para el diseño y ejecución de tareas o comprensión del sistema de control (Spong et al., 2020).

El espacio de trabajo de un robot móvil define el rango posibles de poses que el robot puede tener para moverse en este entorno.

La cinemática es el estudio del movimiento de sistemas mecánicos sin tener presente las fuerzas que originan el movimiento, por lo cual no existen ecuaciones diferenciales como ocurren en el caso de la dinámica. La cinemática es un punto fundamental en el control del robot (Barrientos et al., 2007; Craig, 2017; Lynch & Park, 2017; Niku, 2020).

1.2.13 Robotino. Cinemática del robot móvil Omnidireccional

El modelo cinemático del robotino se fundamenta por ser omnidireccional, por lo que se puede mover en cualquier dirección en un plano 2D, sin requerir cambiar orientación. Se utiliza ruedas omnidireccionales para obtener movimientos flexibles y suaves (FESTO, 2022).

El modelo cinemático directo describe cómo las velocidades de las ruedas afectan la velocidad lineal y angular del robot. Robotino tiene tres ruedas omnidireccionales dispuestas en un ángulo de 120° entre sí (Siegwart et al., 2011).

Parámetros

- R: Radio de las ruedas
- L: Distancia del centro geométrico del robot al eje de cada rueda.
- ωi Velocidad angular de cada rueda (i=1,2,3)
- vx : Velocidad lineal del robot en el eje x (frontal)
- vy: Velocidad lineal del robot en el eje y (lateral)
- ωz Velocidad angular del robot alrededor de su eje vertical (z)

La ecuación del modelo es (Klancar, Zdesar, Blazic, & Škrjanc, 2017; Ma et al., 2024)

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \frac{R}{3}\begin{bmatrix} -\sin(\theta_1) & -\sin(\theta_2) & -\sin(\theta_3) \\ \cos(\theta_1) & \cos(\theta_2) & \cos(\theta_3) \\ \frac{1}{L} & \frac{1}{L} & \frac{1}{L} \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \quad (4)$$

Con $\theta_1 = 0°, \theta_2 = 120°, \theta_3 = 240°$. Los ángulos de las ruedas con relación al eje z.

El modelo cinemático inverso calcula las velocidades de las ruedas, conociendo la velocidad deseada del robot.

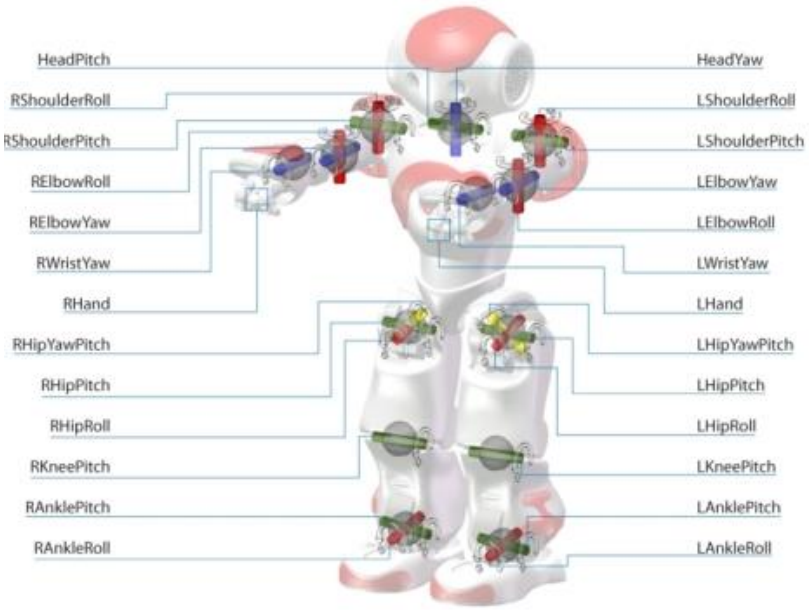
La ecuación del modelo es:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \frac{1}{R}\begin{bmatrix} -\sin(\theta_1) & cos(\theta_1) & L \\ -\sin(\theta_2) & \cos(\theta_2) & L \\ -\sin(\theta_3) & \cos(\theta_3) & L \end{bmatrix}\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \quad (5)$$

1.2.14 Nao. Cinemática del robot móvil bípedo

Considerando las características del robot, se obtienen los datos cinemáticos mediante los links (eslabones), joints (articulaciones), chains (cadenas) y mass (masa) . El robot está formado por 15 motores con cuatro tipos de articulación con un torque de 4.9 mN/m a 16.1 mN/m

Figura 9. 25
Joints (Articulaciones) del Robot Nao.



Nota: Las articulaciones formadas por roll, pitch, yaw

En la Tabla 6 se describen las características del robot real, articulaciones y su identificación junto con los sensores

Tabla 6.
Características y valores del Robot Real

Características	Valor
Model Type	naoH25
Head Version:	VERSION_50
Body Version:	VERSION_50
Laser	False
Legs	True
Arm Version:	VERSION_50
Number of Legs: 2	2
Number of Arms:	2
Number of Hands:	2

Al utilizar una articulación, cadena, parte del cuerpo X, se debe obtener la identificación, con nombre de la articulación, parte del cuerpo y grupo de articulación. El chain (cadena) formada por joints (articulaciones), eslabones con actuadores. El body (cuerpo) estructurada por todas las cadenas que componen (actuadores y articulaciones). Los efectores son para head (cabeza), arm (brazo), leg (pierna) y torso. Los giros roll, pitch , yaw se realizan con los ejes asignados como referencia.(Aldebaran, 2016)

Tabla 7
Roll, pitch, Yaw , Actuadores y Efectores

Cadena (chain)	Articulación (joint)	Movimiento	Actuadores	Efectores
		Body (cuerpo)		
Head(cabeza)	HeadYaw HeadPitch			Head
LArm (brazo izquierdo)	LShoulderPitch LShoulderRoll	Hombro		LArm
	LElbowYaw LElbowRoll	codo		
	LWristYaw		LHand	
RArm (brazo derecho)	RShoulderPitch RShoulderRoll	Hombro		RArm
	RElbowYaw RElbowRoll	codo		
	RWristYaw		RHand	
LLeg (pierna izquierda)	LHipYawPitch LHipRoll LHipPitch	Cadera		LLeg
	LKneePitch	rodilla		
	LAnklePitch LAnkleRoll	tobillo		
RLeg (pierna derecha)	RHipYawPitch RHipRoll RHipPitch	Cadera		RLeg
	RKneePitch	rodilla		
	RAnklePitch RAnkleRoll	tobillo		
				Torso

De acuerdo con Kofinas (2012) se describe la cinemática directa e inversa del robot Nao. Se aplica el método de Denavit-Hartengber DH y se obtienen las ecuaciones de transformación de coordenadas articulares a las cartesianas de posición y orientación (Barrientos et al., 2007) Los parámetros DH se obtienen de la cadena Head (cabeza) y Leg (pierna).

Se aplican las matrices Rx de Rotación en x, Ry de rotación en y, Rz de rotación en z.

$$Rot\ x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\text{sen}(\alpha) \\ 0 & \text{sen}(\alpha) & \cos(\alpha) \end{bmatrix} \quad (6)$$

$$Rot\ y(\emptyset) = \begin{bmatrix} cos(\emptyset) & 0 & sen(\emptyset) \\ 0 & 1 & 0 \\ -sen(\emptyset) & 0 & cos(\emptyset) \end{bmatrix} \quad (7)$$

$$Rot\ z(\theta) = \begin{bmatrix} cos(\theta) & -sen(\theta) & 0 \\ sen(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (8)$$

El torso del robot NAO es el punto en el cual las cadenas cinemáticas comienzan y están localizadas en el centro del cuerpo del robot. La base es el inicio de la cadena cinemática y el efector es el fin de la cadena. En la matriz A se expresa el desplazamiento o traslación.

Se obtienen los puntos del espacio 3D (p_x, p_y, p_z) y las orientaciones del punto final (a_x, a_y, a_z). Por lo cual

$$\begin{aligned} p_x &= T_{(1,4)} \\ p_y &= T_{(2,4)} \\ p_x &= T_{(3,4)} \end{aligned} \quad (9)$$

Con las transformaciones $R_z R_y R_x$ se obtiene la rotación y la orientación del punto final (a_x, a_y, a_z)

$$a_x = arctan2\left(T_{(3,2)}, T_{(3,3)}\right) \quad (10)$$

$$a_y = arctan2\left(-T_{(3,1)}, \sqrt{T_{(3,2)}{}^2 + T_{(3,3)}{}^2}\right)$$

$$a_x = arctan2\left(T_{(2,1)}, T_{(1,1)}\right)$$

Los parámetros DH del robot NAO se describen para la cadena Head

Tabla 8 .
Parámetros DH para la cadena Head.

Frame (articulación)	**a**	$\boldsymbol{\alpha}$	**d**	**θ**
Base		A(0,0, offset de la nuca Z)		
HeadYaw	0	0	0	θ_1
HeadPitch	0	$-\frac{\pi}{2}$	0	$\theta_2 - \frac{\pi}{2}$
Rotación		$R_x\left(\frac{\pi}{2}\right) R_y\left(\frac{\pi}{2}\right)$		
Cámara superior		A(cámara superiorX, 0, cámara superior Z)		
Cámara inferior		A(cámara inferiorX, 0, cámara inferior Z)		

Nota: Cámara superior X=53.9mm. Cámara superior Z =67.9mm. Cámara inferior X=48.8mm . Cámara inferior Z=23.8 mm

Con las matrices de transformación, se obtiene el punto del efector

$$^{Base}T_{End} = {}^{Base}A_0 \ {}^0T_1 \ {}^1T_2 \ R_x\left(\frac{\pi}{2}\right) R_y\left(\frac{\pi}{2}\right) \ {}^2A_{End} \qquad (11)$$

En la cual $^0T_1$ es la matriz de transformación DH de la articulación HeadYaw, y $^1T_2$ es de HeadPitch. $^2A_{End}$ es una de las dos matrices de translación de la cámara superior e inferior. El punto del efector final se obtiene de $^{Base}T_{End}$. Referente a la pierna derecha en la Tabla 9 se calculan los siguientes parámetros DH:

Tabla 9.
Parámetros DH para la pierna derecha.

Frame (articulación)	**a**	**α**	**d**	**θ**
Base		A(0,-hip offsetY, -hip offsetZ)		
RHipYawPitch	0	$-\frac{\pi}{4}$	0	$\theta_1 - \frac{\pi}{2}$
RHipRoll	0	$-\frac{\pi}{2}$	0	$\theta_2 - \frac{\pi}{4}$
RHipPitch	0	$\frac{\pi}{2}$	0	θ_3
RKneePitch	-ThighLength	0	0	θ_4
RAnklePitch	-TibiaLenght	0	0	θ_5
RAnkleRoll	0	$-\frac{\pi}{2}$	0	θ_6
Rotation		$R_z(\pi)R_y\left(-\frac{\pi}{2}\right)$		
End efector		A(0,0, -FootHeight)		

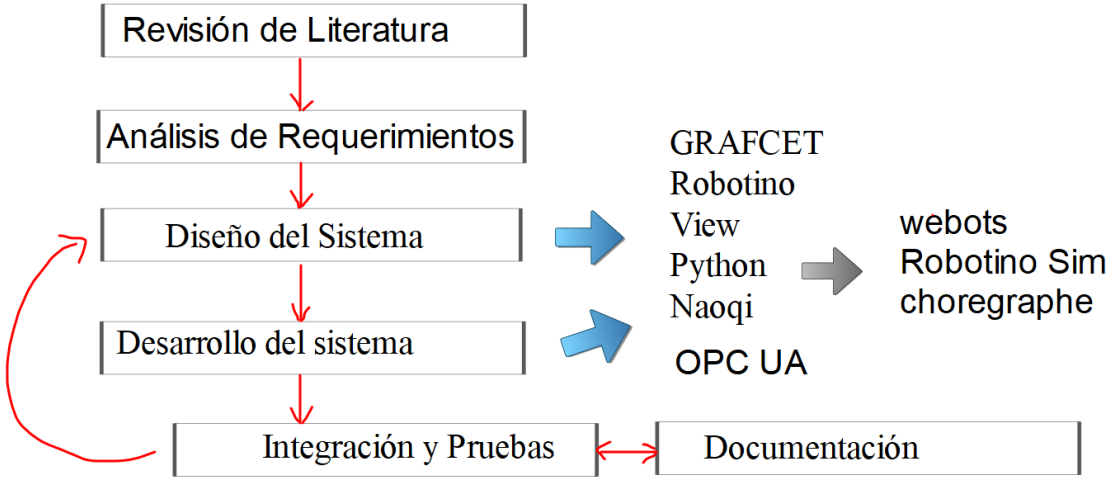
La cinemática de la pierna izquierda se calcula considerando que la cinemática de la pierna derecha es simétrica a la pierna izquierda. La diferencia entre ambas piernas son las articulaciones que rotan sobre el eje y , así como la distancia entre el eje y

$$^{Base}T_{End} = {}^{Base}A_0 \ {}^0T_1 \ {}^1T_2 \ {}^2T_3 \ {}^3T_4 \ {}^4T_5 \ {}^5T_6 \ R_z(\pi)R_y\left(-\frac{\pi}{2}\right) \ {}^6A_{End} \qquad (12)$$

CAPÍTULO 2. METODOLOGÍA

Se realiza el diseño metodológico, fundamentada en varias fases secuenciales, paralelas e iterativas que abarca desde la investigación inicial hasta la validación del sistema:

Figura 10
Metodología del proyecto



2.1 Contexto de la investigación

El trabajo de titulación se efectúa en la Universidad Politécnica Salesiana sede Guayaquil. Laboratorio de Fabricación Flexible como lugar o entorno controlado.

Figura 11
Laboratorio de fabricación flexible

Nota: Lugar de operación de los robots Nao y Robotino en laboratorio de fabricación flexible. Universidad Politécnica Salesiana

Se utilizan los robots bípedo NAO y robotino como se muestra en la Figura 12

Figura 12.
Robot NAO rojo, NAO azul, Robotino 1 y Robotino 2 en área de experimentos

Nota: Existen dos robots NAO y dos robots robotino Festo

2.2 Diseño y alcance de la investigación

El trabajo es según su naturaleza y alcance es de investigación aplicada, su enfoque es mixto (cuantitativo y cualitativo), su diseño es no experimental con cohorte longitudinal en el año 2024.

El proyecto es de tipo académico, orientado al aula Universitaria y se propone como un insumo de algunos proyectos del Grupo de Investigación NANOTECH de la UPS orientado a robótica móvil, localización y navegación.

Se establen condiciones en un entorno bidimensional, controlado o indoor como es un laboratorio. Se utilizan de los dos robots NAO, el robot rojo y de los dos Robotino el robotino1. Por motivo de que varios servomotores y conexiones están en permanente mantenimiento.

Para efectuar pruebas antes de utilizar los robots, se utiliza los simuladores robotinoSim para el Robotino y choregraphe, webots para el robot NAO.

El robot NAO utiliza su sensor ultrasónico y cámara de visión nativa, similar el robotino, utiliza sus sensores infrarrojos, de golpe y cámara de visión artificial. No se agregan sensores adicionales a ambos robots.

El resultado es un sistema teleoperado y autónomo de locomoción, navegación e interacción de un robot bípedo NAO con el robot omnidireccional Robotino.

Se considera por los objetivos y movimientos empleados, que la locomoción, la localización es en el plano 2d. Otras características e investigaciones no son consideradas y quedan fuera del alcance del trabajo.

2.3 Tipo y métodos de investigación

El tipo de investigación es mixto. Adicional a la metodología propuesta por el autor. Se utilizan los métodos de investigación, Inductivo- deductivo, hipotético-deductivo y Analítico- sintético.

2.4 Población y muestra

Población.

- Dos Robots NAO y
- dos robots Robotino de la Universidad Politécnica Salesiana.

Muestra:

- 1 robot NAO rojo versión 5
- 1 Robotino versión 3

2.5 Técnicas e instrumentos de recolección de datos

Para contrastar la información generada por los robots se utilizan Instrumentos y técnicas de Recolección de Datos numéricas, medición manual, documentación técnica, pruebas de laboratorio, simulaciones, pruebas de campo.

2.5.1 Materiales y recursos del proyecto

Se describen los elementos a utilizar en el proyecto

Tabla 10
Materiales del proyecto

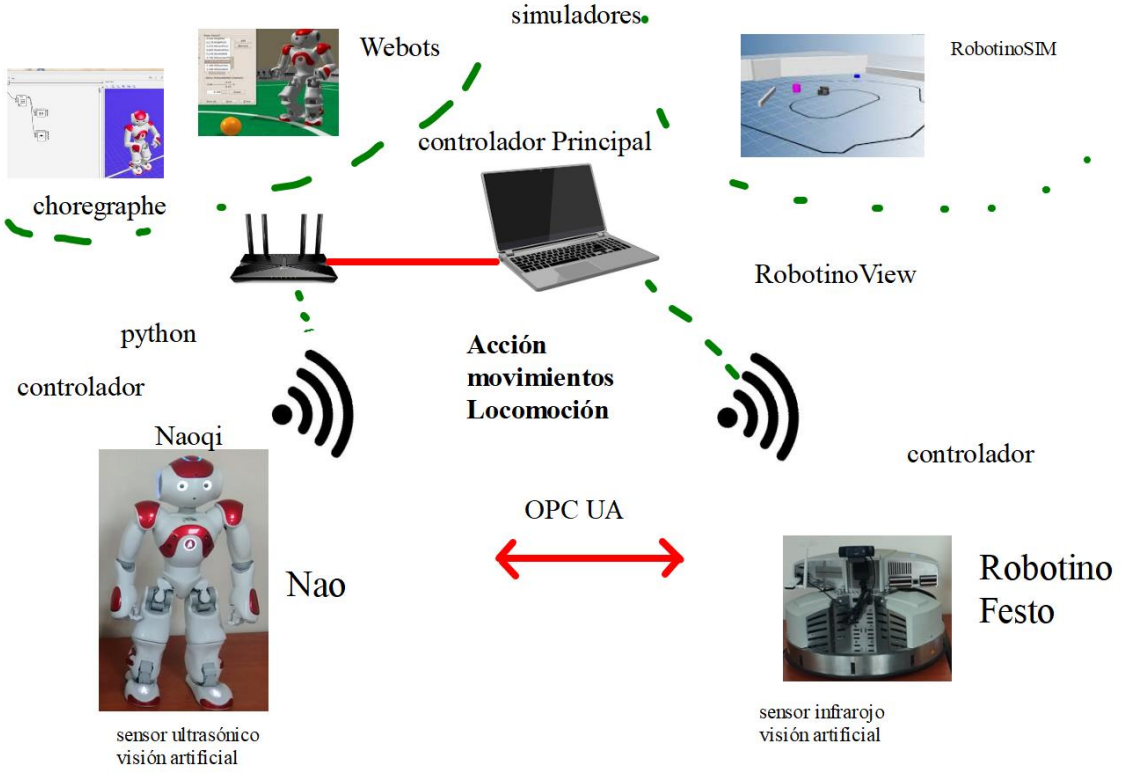
Dispositivo o Equipo	Acción a realizar
Computador principal	Computador que envía las ordenes a ambos robots
Robot Real Robotino v3.0	
Robotino View 4.0	Software Programar Robotino Festo
GRAFCET	Programar robotino View
Robotino Sim Demo 1.4.1	simular Robotino Festo
Robotino sensores infrarrojo, golpe bumper y visión artificial	percepción robotino
Python 2, python3	Lenguajes de programación Nao/Robotino
Pycharm IDE	Entorno de Desarrollo Integrado Python
UAExpert 1.7.1	Herramienta para explorar el servidor OPC UA y los NodeIds
Librería opcua Python 3	Comunicar Python 3 con OPC UA
Libreria subprocess	Comunicar python 2 con python 3
Libreria opencv	Libreria de vision artificial
Robot Real Nao v5.0 (2.143 image)	
Nao sensor ultrasónico, visión artificial	
Framework NAOqi 2.14 de 32 bits	Intermediario para comunicación robot NAO
Pynaoqi 2.1.4.13	para comunicación con Python con el framework naoqi
Soporte opencv 2.4 y 3.2.	en el proyecto 4.2.0.32 con Python 2.7 de 32 bits solamente.

2.5.2 Diseño del Sistema de Interacción colaborativo, arquitectura, modelado y simulación.

El diseño utiliza un computador central que se conecta por ethernet al router A. El robot nao se conecta por wifi al router A. El computador se conecta al wifi del Robotino.

La metodología aplicada es que se realiza la programación en RobotinoView , se experimenta en el simulador robotinoSim con los sensores y visión artificial. En el robot NAO, se programa en Python con librerías de visión artificial openCv en el simulador Webots. El documento de Grafcet de Robotino como el código de Python es similar al robot real. Esto se efectúa cambiando la Ip en Robotino, e Ip y puerto en robot NAO

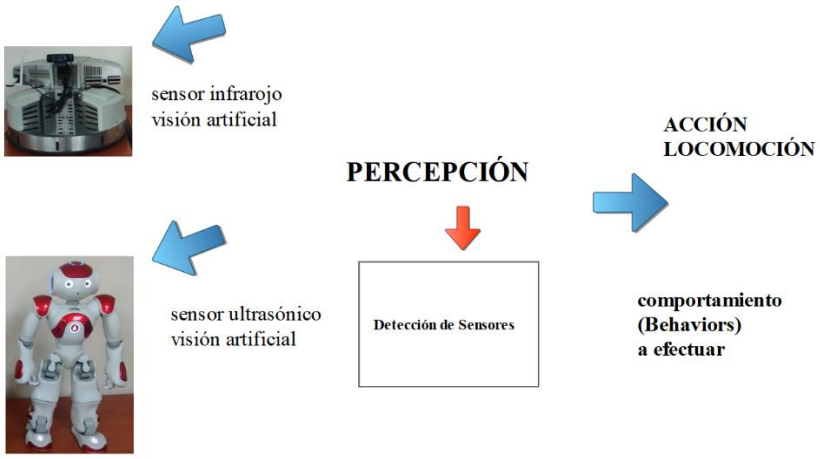
Figura 13.
Diseño del sistema



Nota: proceso del diseño del sistema la comunicación de cada robot,

Para el desarrollo del sistema de interacción se efectúa la aplicación de algoritmos de comunicación con OPC UA, desarrollo de algoritmos de coordinación y programación de la navegación autónoma. La percepción se efectúa mediante sensores infrarrojos, sensores ultrasónicos y cámara de visión artificial.

Figura 14
Diseño de las acciones según la percepción



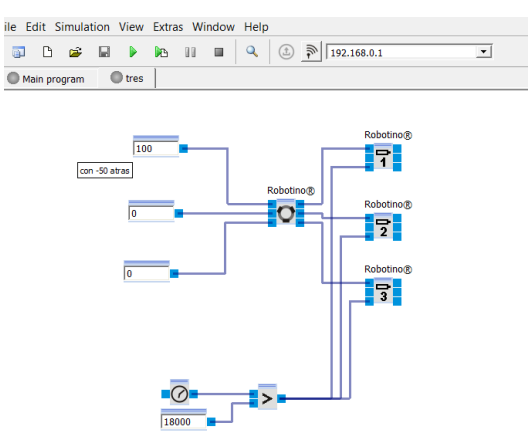
2.5.3 Herramientas de Programación para Robotino y NAO

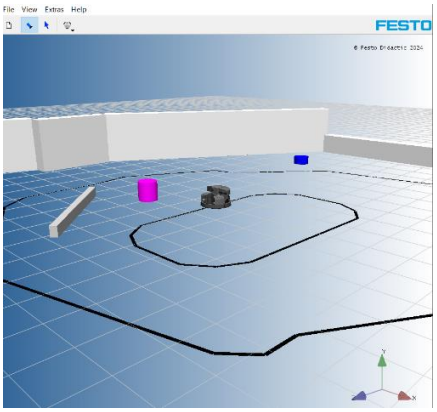
2.5.3.1 RobotinoView, Robotino SIM Demo y GRAFCET.

Una de las formas de programar a Robotino Festo es con RobotinoView utilizando GRAFCET. Se utiliza la versión 4.0 que contiene el servidor OPC UA. También se puede programar con Python y Matlab, pero hay que tener actualizado el core de Robotino a la versión 4.0, acción no autorizada para la propuesta de titulación.

Se considera para simulación como entorno ideal el Robotino SIM Pro. Por motivos de presupuesto se utiliza la versión Robotino SIM Demo, que permite simular elementos puntuales y lograr obtener el comportamiento mínimo requerido para el robot.

Figura 15.
RobotinoView 4 y Sim 1.4.1



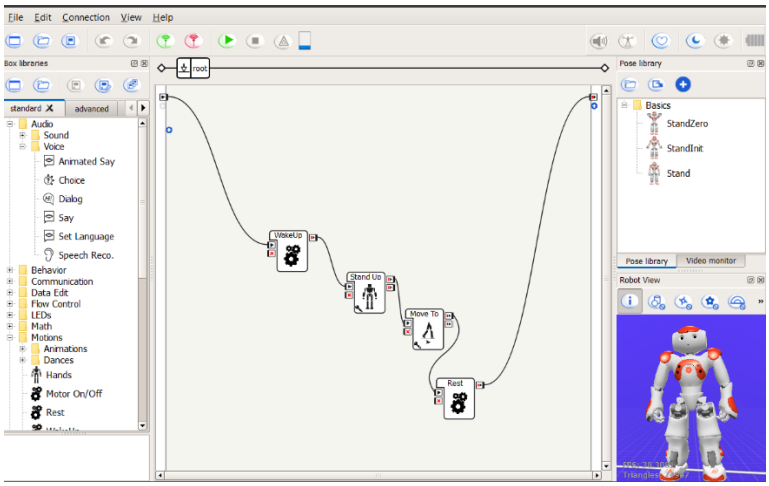


2.5.3.2 NAO, Choregraphe, Python, IDE Pycharm

La programación del robot NAO se puede efectuar de diferentes lenguajes tales como C, C++, Python, java, Matlab. Su entorno nativo es Python.

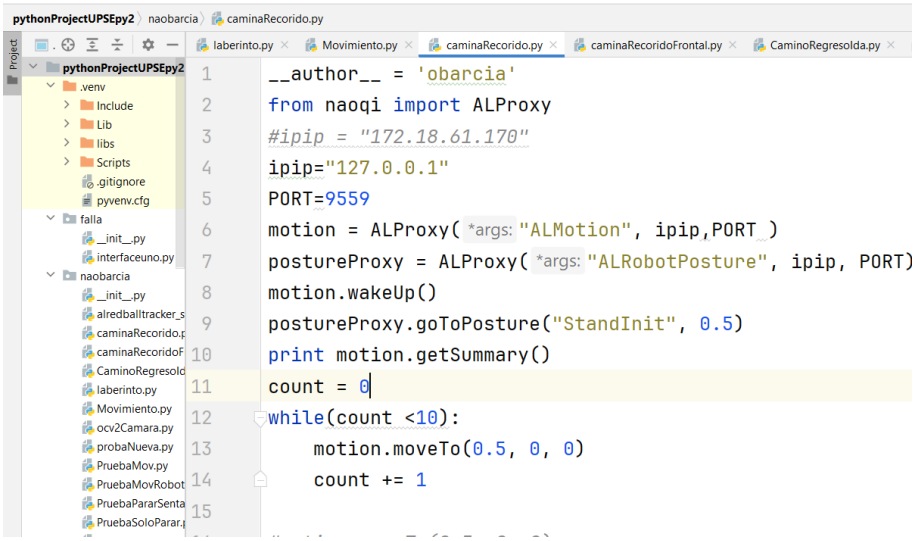
Para efectuar la programación se utiliza Choregraphe que es la versión a nivel gráfica para programar el robot. A nivel intermedio se programa con Python 2.7, el framework Naoqi y pynaoqi.

Figura 16.
Choregraphe. Programación por bloques de NAO



Para un desarrollo adecuado se utiliza el IDE Integrated Development Environment (Entorno de Desarrollo Integrado) Pycharm edición comunitaria. Se utilizará el IDE para cargar los entornos en Python 3 y Python 2

Figura 17.
IDE Pycharm



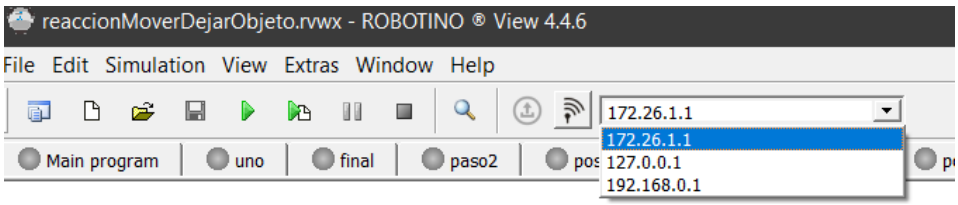
2.5.3.3 Entorno de simulación, entorno real y transferencia al robot real.

Un robot simulado, no existe en el entorno real, se ejecuta en el computador y realiza similares acciones o comportamientos que el robot real. Una de las metas es efectuar las pruebas o experimentos antes de cargarlo o efectuar la comunicación con el robot real, de esta manera detectar inconvenientes o acciones correctas y minimizar riesgos con el robot físico. Los simuladores efectuarán un entorno bastante aproximado al real. En el entorno existirán objetos, pisos, elementos. Después de experimentar con los robots simulados, se recomienda con la carga y comunicación con el robot real. El escenario o entorno puede ser similar para que tanto el espacio virtual y real sean similares a los experimentos.

En Robotino a nivel general, la programación que se efectúa en GRAFCET del robot real Robotino es similar al simulado. Lo que hay que actualizar es las direcciones IP.

Dependiendo de las licencias permiten agregar elementos y todo el entorno (webots) como solo mantener lo que ya están (RobotinoSim Demo)

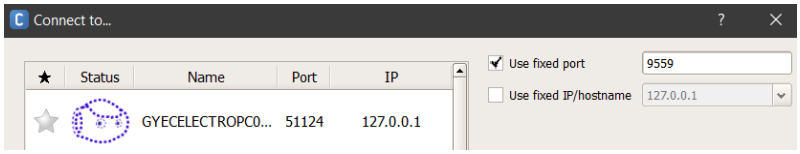
Figura 18.
RobotinoView , selección de IP local y Real



De forma idéntica, con el robot Nao se debe actualizar la dirección IP y puerto del robot. Para simular a nivel bloques se utiliza el software Choregraphe. Para nivel bloques o código Python se sugiere el simulador Webots 8.5 (versiones superiores no compatibles).

Figura 19.
Interfaces de Python y Choregraphe para cambio de IP robot virtual y real

```
from naoqi import ALProxy
#ipip = "172.18.61.170"
ipip="127.0.0.1"
PORT=9559
motion = ALProxy( *args: "ALMotion", ipip,PORT )
```



Las direcciones IP del robot local y simulado se mencionan en la Tabla 11

Tabla 11.
Direcciones IP robot simulado y real

Tipo de Robot	Dirección IP	Acción
Robotino Robot Virtual	127.0.0.1	Simular con Robotino SIM Carga de acción en robot Virtual
Robotino Robot Real	192.168.0.1 172.26.1.1	Carga de acción en robot real
NAO Robot Virtual	127.0.0.1 Puerto: 9559	Simular con choregraphe o webots
NAO Robot Real	169.254.30.68 conectado con cable para configuración 192.168.0.101 Conectado al wifi	Se presiona el botón del pecho para que indique la dirección ip

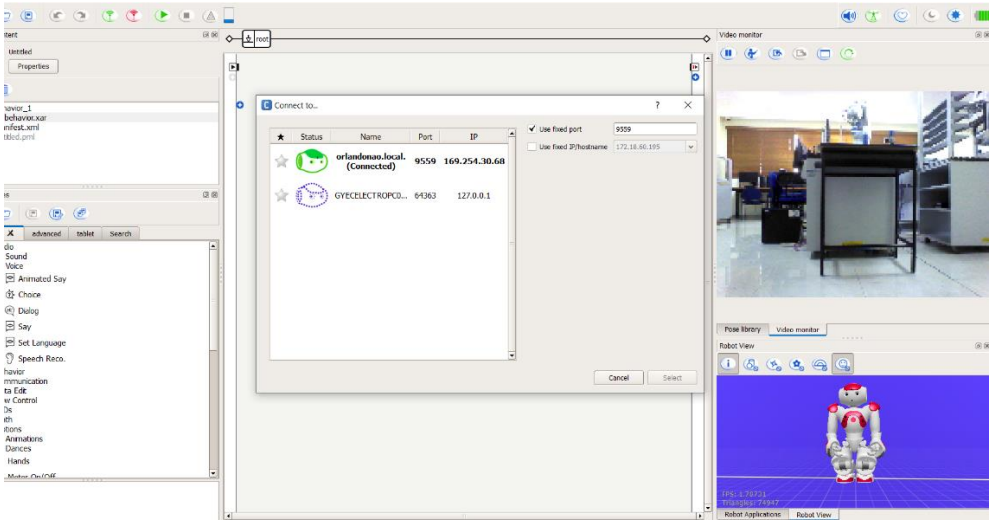
2.5.3.4 Comunicación, conexión al robot virtual o real.

Conociendo las direcciones IP, se puede acceder a los servidores que están en los robots.

- En robotino en un explorador 172.26.1.1
- En nao la IP generada al presionar el botón del pecho del robot, 192.168.0.101

Otra opción para efectuar la conexión es desde Choregraphe, sea con la IP local del robot virtual o con la IP del robot real.

Figura 20
Acceso al robot real desde Choregraphe

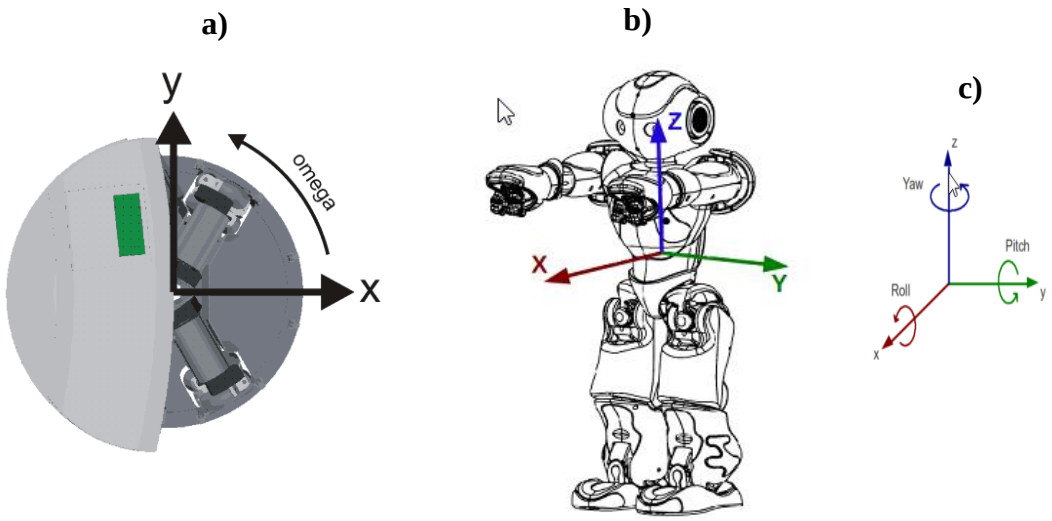


2.6 Definición de ejes de referencia Nao y Robotino

Un eje de coordenadas o frame se representa como {F}. Los ejes x, y, z, respecto al frame se denominan X_F, Y_F, Z_F. Una pose es la posición y orientación de frame de coordenadas. Un frame puede estar desplazado y rotado respecto a otro frame. El primer frame de coordenadas es el de referencia o world (mundo) (Corke, 2011; Craig, 2017). Un punto del robot en el espacio tridimensional 3D (pose world), requiere de 6 coordenadas en el frame, 3 para la posición y 3 para la orientación. Para el espacio bidimensional se requiere de 3 coordenadas en el frame, dos para la posición x , y y una para la orientación (Barrientos et al., 2007; Niku, 2020).

En Robotino utiliza la siguiente relación de referencia. En x se utiliza para velocidad lineal vx en mm/s. En eje y es vx en mm/s. En Eje z, se utiliza omega como velocidad angular en degree/s.

Figura 21.
Eje de referencia del robot ROBOTINO y NAO



Nota: a) ejes de referencia robotino, b) ejes de referencia NAO, c) roll, pitch, yaw

2.6.1 Robotino, técnicas y estrategias de locomoción y percepción

2.6.1.1 Motores del robotino y Ejes de referencia

El robotino tiene tres motores M1, M2 y M3 al que se asignan un Set Point de la velocidad en rpm. Las entradas y salidas se muestran en la Figura 22 e indican en la Tabla 12.

Figura 22.
Bloque de Motores en RobotinoView

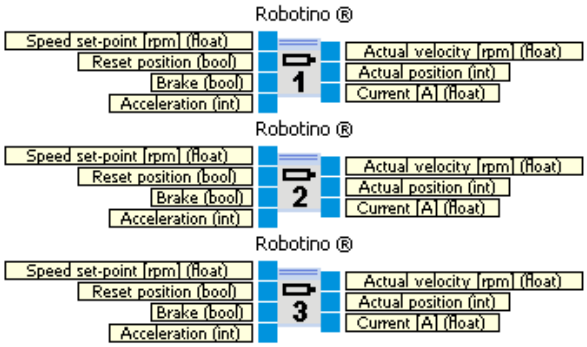


Tabla 12
Parámetros de entradas y salidas de Motor del robotino

Entradas	Tipo	Unidad	Descripción
set-point velocidad	float	rpm	Set Point de velocidad del control del motor en revoluciones por minuto. La relación de transmisión es 16:1 entre el motor y la rueda del robotino.
Reset position	bool		En caso de ser verdadero , el tick counter se reestablece a 0
Brake	bool		Si es verdad el motor para If true the motor is stopped.
Acceleration	int		Acoplamiento del setpoint SP de velocidad en la entrada y el set point SP de velocidad real transmitida
Salidas			
Velocidad actual	float	rpm	La velocidad actual del robot
posición Actual	int		Número de ticks contados desde el encendido del Robotino o desde que se establece "Reset position" como verdadera y falsa. Los ticks son generados por el codificador del motor . Estos son 2000 ticks por vuelta.
Corriente	float	A	. La corriente i medida en Amperios en el puente H del motor

Con el Control PID se utiliza para controlar la velocidad y el posicionamiento preciso de los robots en sus trayectorias planificadas. En el caso de Robotino, su capacidad omnidireccional requiere una regulación precisa de las ruedas, y el PID permite un movimiento suave y eficiente. Se aplican las constantes proporcional kp, integral ki, y derivativo kd. Los valores por defecto son kp=25, ki=25, kd=25.

$$\mu(t)_i = \kappa_p \left(e(t) + \frac{1}{T_n} \int_0^t e(t')dt' \right) + \kappa_d \dot{e}(t) \qquad (13)$$

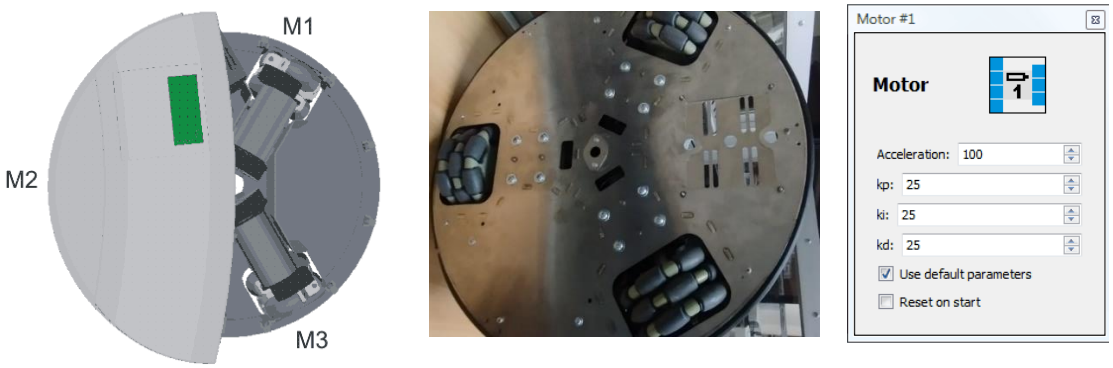
$$k_i = \frac{1}{T_n}$$

Los parámetros se calculan de acuerdo con las constantes

$$\boldsymbol{k_p} = \frac{\boldsymbol{k_p}}{2} \ , \boldsymbol{k_i} = \frac{\boldsymbol{k_i}}{1024} \ , \boldsymbol{k_d} = \frac{\boldsymbol{k_d}}{2} \qquad (14)$$

La ubicación de vista frontal e inferior de los motores se muestra en la Figura 23

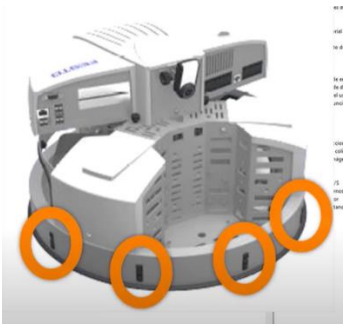
Figura 23
Ubicación de los motores en robotino y control por PID



2.6.1.2 Sensores infrarrojos

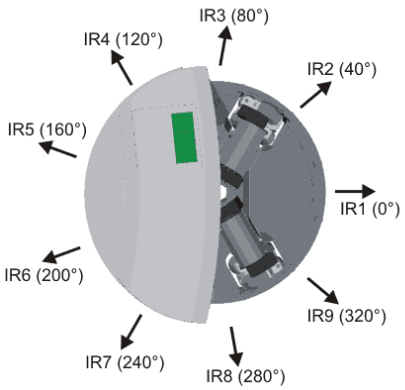
Está formado por 9 sensores infrarrojos separados por 40°. Donde 0 Voltios es sin obstáculos y 2.5 Voltios es cerca al obstáculo

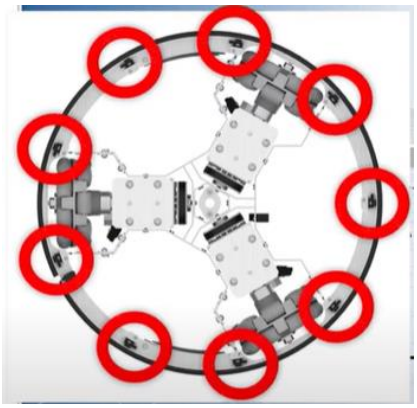
Figura 24.
Sensores infrarrojos de robotino



En Figura 25 se muestra la vista superior e inferior de los sensores

Figura 25
Vista superior y posterior de los sensores infrarrojos





De acuerdo al voltaje se procede a efectuar la siguiente conversión, denominada en robotinoView Función de transferencia. Esta relaciona el voltaje con la distancia a detectar

Para determinar el grado se utiliza de acuerdo a los números de IRnumber desde i=1 a 9

$$\text{Heading} = 40° \times (\text{IRNumber} - 1) \quad (15)$$

La Tabla 13 muestran la conversión de voltaje a distancia y la

Tabla 13. Conversión voltaje del sensor a distancia en cm

Voltaje (Volts)	Distancia (cm)
0.3	40
0.39	35
0.41	30
0.5	25
0.75	18
0.8	16
0.95	14
1.05	12
1.3	10
1.4	9
1.55	8
1.8	7
2	6
2.35	5
2.55	4

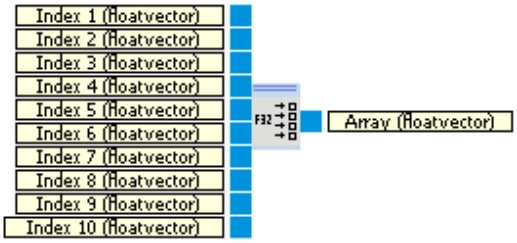
2.6.1.3 Transformaciones matemáticas

Para efectuar las operaciones de transformación para localización, se emplean en RobotinoView los bloques de vectores, conversiones, rotaciones y traslaciones.

Figura 26
Transformaciones matemáticas en RobotinoView

a)

b)

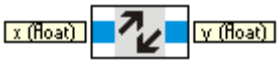


c)

d)

e)

f)

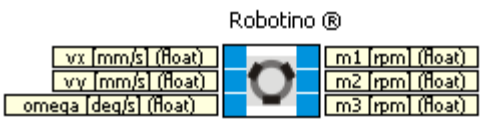


Nota: a) conversión a vectores, b) Array, c) Polar a vector , d) vector a cartesiano, e) Rotación, f) función de transferencia

2.6.1.4 Omnidrive

El bloque omnidrive recibe las velocidades y efectúa el movimiento en los motores. Se puede asignar directamente velocidades. Para utilizar de forma precisa se debe agregar otros bloques de control como PositionDriver y Odometria. El bloque establece el set point de velocidad vx,vy,wv que debe asignarse a los motores MI, M2 , M3.

Figura 27
Bloque Omnidrive del Robotino



Se describen los datos para ser establecidos en el movimiento de los motores

Tabla 14
Entradas y salidas del OmniDrive

Entrada	Unidad	Descripción
Vx	Mm/s	Establece la velocidad en x
Vy	Mm/s	Establece la velocidad en y
W	Deg/s	Establece la velocidad rotacional
Salidas		
M1	rpm	Set point del motor 1
M2	rpm	Set point del motor 2
M3	rpm	Set point del motor 3

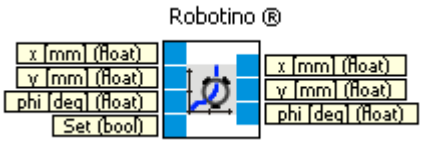
2.6.1.5 Odometría

La odometría es el uso de datos del movimiento de los actuadores para estimar el cambio de posición a lo largo del tiempo. La rotación de las ruedas se mide con la mayor resolución temporal posible. En cada paso de tiempo, la distancia recorrida por el vehículo se calcula a partir de la velocidad de rotación de las ruedas. Estas distancias muy pequeñas de los pasos de tiempo individuales se integran a lo largo del tiempo. Esto conduce a la posición real relativa a la posición inicial (Barrientos et al., 2007; Craig, 2017).

Figura 28
Bloque Odometría en Robotino

En distancias largas o en condiciones adversas (ruedas que patinan debido al polvo en el suelo, deslizamiento debido a la dirección preferencial de la alfombra) este método conduce a errores muy grandes. Por este motivo la odometría siempre se combina con otros métodos para compensar los errores descritos (Barrientos et al., 2007).

Figura 29
Bloque Odometria de Robotino

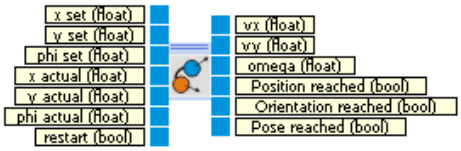


Las entradas del bloque son x, y, phi de la posición referencia, al inicio debe ser x=0, y=0, phi=0. Si se actualiza la nueva ubicación es el nuevo punto de referencia (posición relativa) . Las salidas son la actual posición x,y, phi en coordenadas globales o la nueva ubicación si está fue actualizada.

2.6.1.6 Navegación. Position Driver

El controlador de posición permite conducir a Robotino a una posición determinada, generando valores establecidos de velocidad y velocidad angular, partiendo desde la posición real a la posición establecida.

Figura 30
Bloque position Driver de Robotino

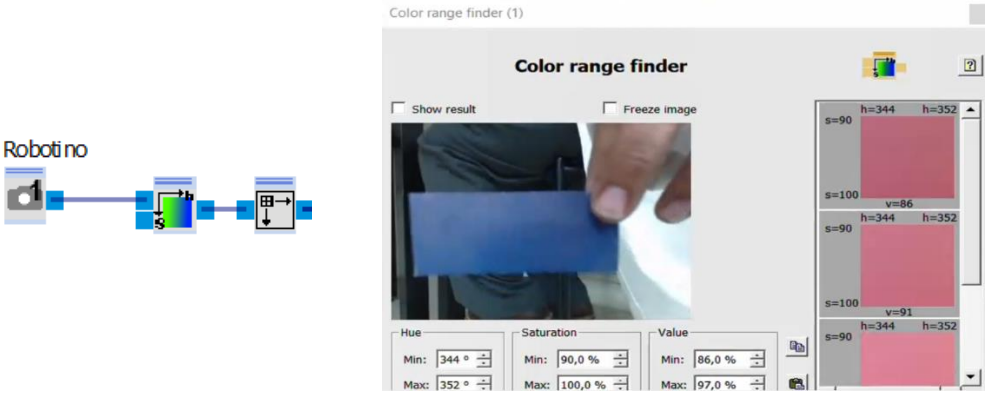


Los datos de entrada son x,y,phi que son los puntos objetivos en base al eje de referencia. x actual, y actual, phi actual que son los puntos actuales donde el robot se encuentra (por lo general se obtienen de la odometría). La salida es la velocidad vx,vy y omega. La salida posición, orientación y pose, si llegaran a ser alcanzadas se convierten en Verdadero.

2.6.1.7 Visión artificial.

Se utiliza el bloque ColorRangeFinder. Se procede a detectar el Hue, Saturation, Value HSV de los colores y se procede a efectuar la acción requerida.

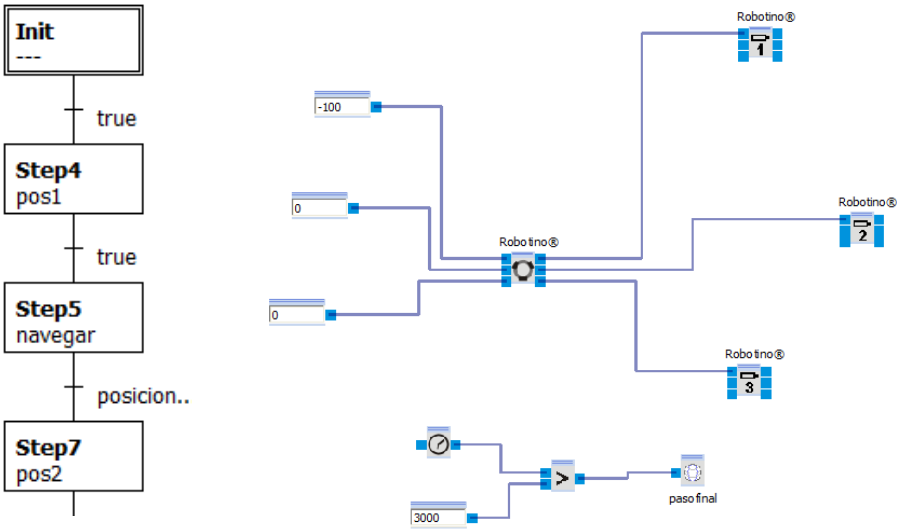
Figura 31
Bloque ColorRangeFinder de Robotino



2.6.1.8 Robotino. Control de Locomoción

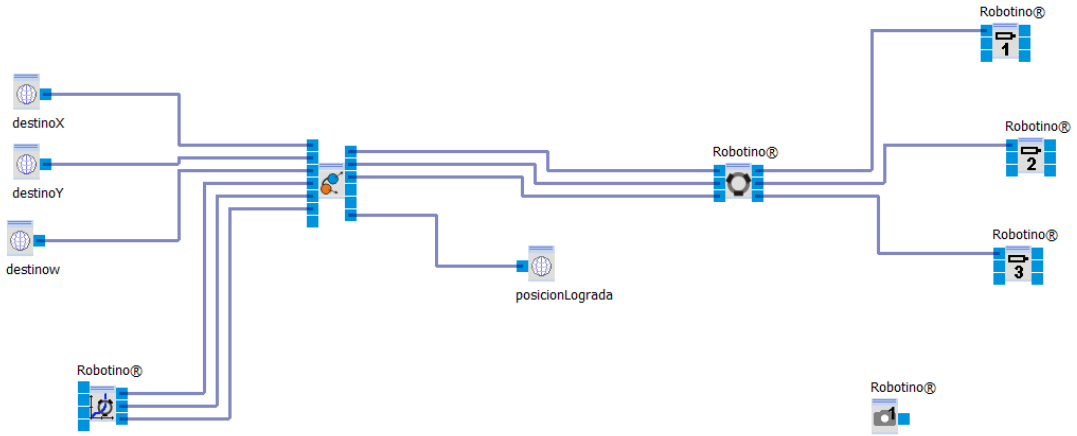
El control de locomoción se realiza con el OmniDrive con velocidades y la navegación con el PositionDriver, la lógica es con el GRAFCET del comportamiento o movimiento del robot. Utilizando velocidades lineales Vx, Vy y velocidad angular con temporizador para cambio de etapa.

Figura 32
Navegación utilizando velocidades lineales, angulares y paso de estado por tiempo



Para efectuar una locomoción precisa, en la Figura 33 se muestra el uso del bloque odometría y positionDriver

Figura 33
Navegación utilizando odometría y PositionDriver



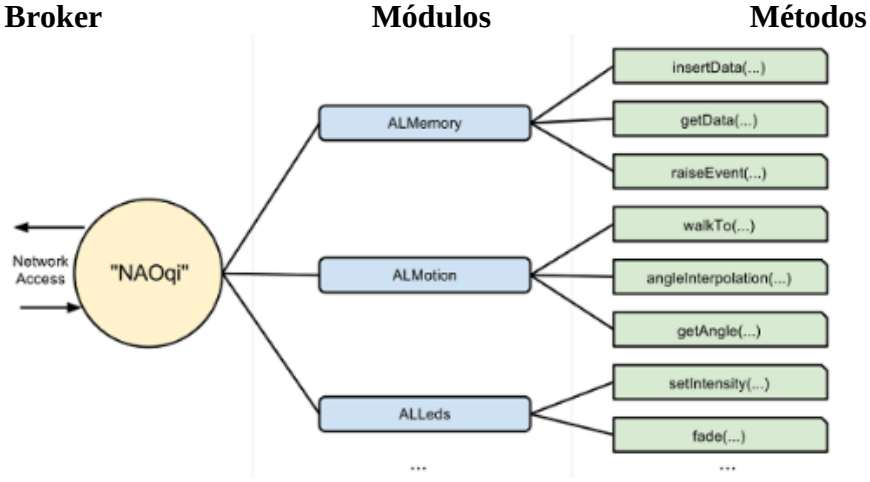
2.6.2 Robot NAO, técnicas y estrategias de locomoción y percepción

2.6.2.1 Comunicación con el Robot. Framework NAOqi

El framework NAOqi, es el software que permite controlar el Robot. Se utiliza en el robot virtual como en el real, ambos configurados con NAOqi 2.14 para reducir incompatibilidad

Bróker y proxy. NAOqi se ejecuta en el robot y se denomina bróker o agente-corredor. El bróker permite acceder a los módulos del framework, y está atento a recibir los comandos remotos en base a una dirección ip y un puerto 9559 (Softbank, 2018).

Figura 34.
Proceso de comunicación Framework NAOqi



Un proxy es un objeto que se comporta como el módulo que desea utilizar.

En el framework NAOqi al proxy se denomina ALProxy y es el objeto principal. En la Tabla 15 se indicas los módulos utilizados en el trabajo

Tabla 15.
Módulos de NAOqi utilizados.

Módulo	Operación a realizar
ALMotion	Control del movimiento caminar, parar, girar
ALRobotPosture	Control de la postura del robot
ALVideoDevice	Acceso a la cámara
ALSonar	Acceso al sonar
ALMemory	Acceso a los datos del sensor inercial y a los datos del sonar
ALTextToSpeech	Acceso para que el robot hable

En la Tabla 16 se describe el código básico para utilizar ALProxy con los módulos ALMotion para movimientos, ALRobotPosture para posición , ALTextToSpeech para sintetización de voz. Todos los llamados, requieren la dirección IP y puerto del robot.

Tabla 16.
Instrucciones básicas para el movimiento del robot.

```
varmovimiento = ALProxy("ALMotion", ip, puerto)
varposture =    ALProxy("ALRobotPosture", ip, puerto)
tts =           ALProxy("ALTextToSpeech", ip, puerto)

varmovimiento.wakeUp() # despertar al robot
varposture.goToPosture("StandInit", 1)#ir a una posición de inicio del robot

varmovimiento.moveTo(0.5,0,0) # mover a una posición de 0.5 metros en X

tts.setLanguage("Spanish")
tts.post.say("Hola Bienvenido a mi tesis")

varmovimiento.rest()   # descansar
```

2.6.2.2 Nao. Locomoción del robot bípedo y control del movimiento

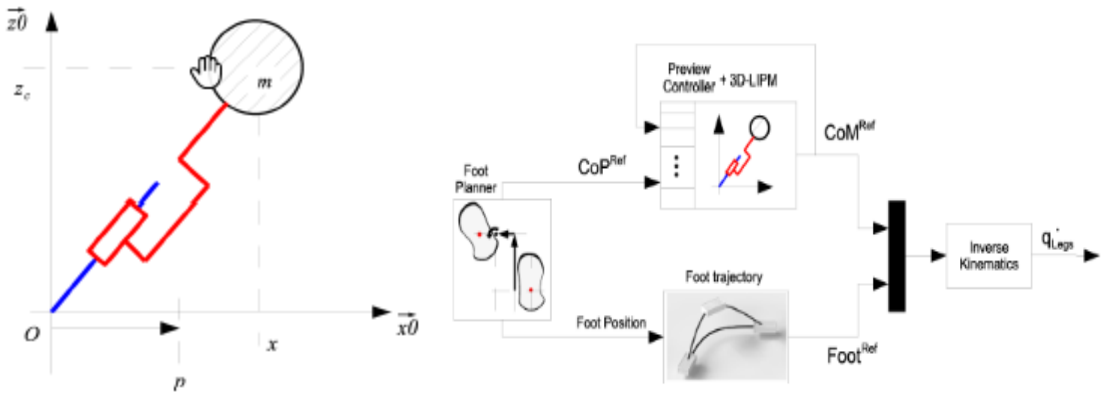
Para trabajar con la locomoción, es necesario entender el mecanismo del robot, la cinemática, la dinámica y la teoría del control. El robot NAO humanoide, se aplican pruebas en el laboratorio para que su movimiento sea de forma suave. La movilidad con patas demanda más niveles de libertad y más complejidad en términos mecánicos que la movilidad con ruedas de robotino.

El robot NAO tiene en su pierna izquierda y derecha 5 DOF, más 1 DOF compartido. En total para la locomoción en la parte inferior tiene 11 DOF.

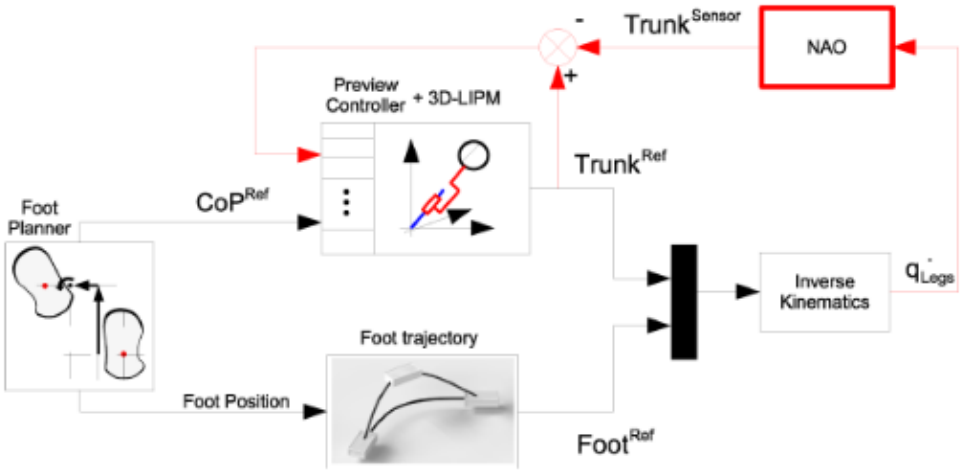
La eficiencia de locomoción bípeda depende de la masa de las piernas y la masa del cuerpo, debido a que es lo que tiene que soportar para realizar el movimiento. Una de las desventajas de la locomoción bípeda es la potencia requerida y la complejidad.

Realizar el movimiento del robot a un punto exacto es complicado, pero el robot NAO tiene comportamientos para caminar preconfigurados. Al caminar el robot, de forma interna crea patrones de pasos. De esos se obtiene el punto de momento cero o ZMP (Zero Moment Point) mediante muestreo, el cual posteriormente se transforma en una trayectoria del COM centro de masa y a los ángulos de las articulaciones. La caminada del robot NAO (Kajita & Tani, 1995) utiliza un péndulo lineal invertido. El modelo a lazo abierto y cerrado para la caminada lo expone (Wieber, 2006)

Figura 35.
Modelo de caminada a lazo abierto y cerrado



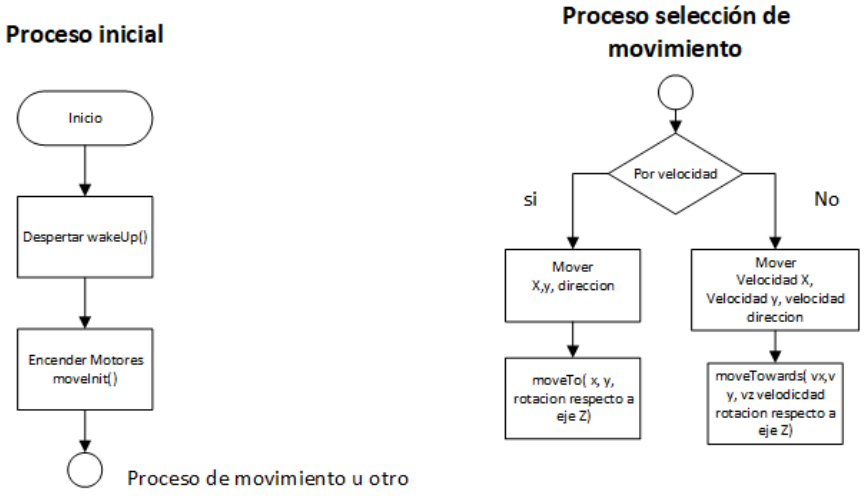
a) péndulo invertido lineal. b) Modelo a lazo abierto del motor de caminado.



Nota: Modelo a lazo cerrado del motor de caminado (Wieber, 2006).

En la Figura 36 se muestra el procedimiento para la locomoción que se muestra en la

Figura 36.
Proceso para la locomoción.



El método despertar wakeUp () acciona los motores y ubica al robot en una posición lista para que los pies pueden moverse sin inconvenientes.

Tabla 17
Proceso de inicialización del robot antes del movimiento. Ejm con robot virtual

```
movimientoProxy  = ALProxy("ALMotion","127.0.0.1", 9559)
posturaProxy = ALProxy("ALRobotPosture", ","127.0.0.1", 9559)
movimientoProxy .wakeUp() #despierta al robot
postureProxy.goToPosture("StandInit", 0.5)
movimientoProxy
motionProxy.moveInit() # inicializa el proceso de movimiento, chequea la pose del robot
```

Con wakeUp () enciende los motores e intenta ir a una posición inicial. Con moveInit () se verifica y establece que la posición del robot sea la correcta antes del movimiento, pero no realiza el movimiento. La orden goToPosture () establece una posición adecuada ideal para iniciar el movimiento.

Para efectuar la locomoción de Caminar y mover se utilizan los siguientes métodos:

moverPosicion(x, y,θ).

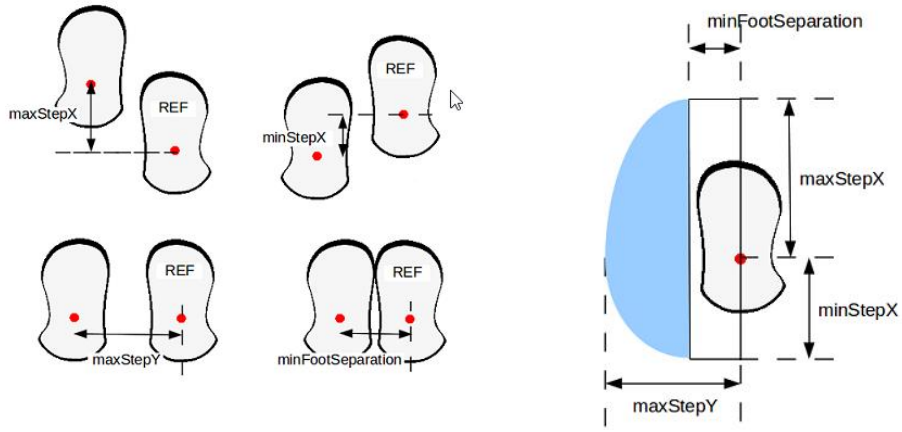
moverPosicion(velocidadX, velocidadY, velocidad θ).

Tabla 18
Comandos para caminar, parar y moverse

```
Movimimiento a posición estimada en x, y y ang en radianes
# primero sentencias para obtener datos del sonar y vision
movimientoProxy.moveTo(x,y,angr)

Movimiento en velocidad
        frecuencia = 1.0
        motionProxy.moveToward(vx, vy, vang, [["Frequency", frecuencia]])

# detección de obstáculos o parada de emergencia
        motionProxy.Killmove()  # No se considera el estado de balance del robot el cual puede caer
```

La configuración del movimiento para los pies, con esto se disminuye los errores de posición al caminar y a la vez se evita caídas del robot.

Figura 37.
Parámetros de configuración para la caminada.



Nota: (Aldebaran, 2016)

2.6.2.3 Masa del robot

Se calcula la masa de cada cadena y cada articulación. Para todas las cadenas:

cadenamasa= motionProxy.getMass(nombre)

lista articulaciones = motionProxy.getBodyNames(nombre)

Para todas las articulaciones

jointmasa= motionProxy.getMass(njoints)

Se describe la masa de la cadena cabeza, brazo izquierdo y pierna izquierda.

Tabla 19
Masa de las cadenas y articulaciones del Robot NAO (Kg).

Body mass 5.30535030365
Com mass 5.30535030365
cadenas ['Head', 'LArm', 'LLeg', 'RLeg', 'RArm']
Torso mass 1.04955995083
Es cadena Head
cadena: Head mass 0.683749973774
------ lista joints ['HeadYaw', 'HeadPitch']
--------- joint: HeadYaw mass = 0.0784199982882
--------- joint: HeadPitch mass = 0.605329990387
Verificacion masa de esta cadena Head suma de los joints 0.683749988675
cadena: LArm mass 0.578580021858
------ lista joints ['LShoulderPitch', 'LShoulderRoll', 'LElbowYaw', 'LElbowRoll', 'LWristYaw', 'LHand']
--------- joint: LShoulderPitch mass = 0.093039996922
--------- joint: LShoulderRoll mass = 0.157769992948
--------- joint: LElbowYaw mass = 0.0648299977183
--------- joint: LElbowRoll mass = 0.077610000968
--------- joint: LWristYaw mass = 0.1853300035
Verificacion masa de esta cadena LArm suma de los joints 0.578579992056
cadena: LLeg mass 1.20744001865
------ lista joints ['LHipYawPitch', 'LHipRoll', 'LHipPitch', 'LKneePitch', 'LAnklePitch', 'LAnkleRoll']
--------- joint: LHipYawPitch mass = 0.0698100030422
--------- joint: LHipRoll mass = 0.140530005097
--------- joint: LHipPitch mass = 0.389679998159

```
--------- joint:  LKneePitch  mass =  0.301420003176
--------- joint:  LAnklePitch  mass =  0.134159997106
--------- joint:  LAnkleRoll  mass =  0.171839997172
Verificacion masa de esta cadena LLeg suma de los joints  1.20744000375
```

2.6.2.4 Centro de masa

Se obtiene el Centro de Masa COM (Center Of Mass) del solido (S) en metros. Se aplica la matriz Inercial con valores relativos al eje coordenado (o,R).

$$COM = \begin{bmatrix} X_G \\ Y_G \\ Z_G \end{bmatrix}_{(o,R)} \quad (16)$$

$$[I_o S]_R = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}_R$$

Se procede a obtener el COM.

Tabla 20
Comandos para obtener el COM del robot NAO

```
nombreCadenas = motionProxy.getBodyNames(name)
combody = motionProxy.getCOM("Body",frame, useSensors)
comtorso = motionProxy.getCOM("Torso",frame, useSensors)
# recorrido por cada cadena y cada joint
        comjoint = motionProxy.getCOM(njoints,frame, useSensors)
```

Se obtiene la cadena Body y Head el centro de masa es:

Tabla 21
Centro de Masa COM de la cadena Body y Head.

```
COM Body ---------- x = 0.0129609489813 y = 0.00105522957165 z = -0.0328637994826
cadenas  ['Head', 'LArm', 'LLeg', 'RLeg', 'RArm']
COM Torso ---------- x = -0.00412999978289 y = 0.0 z = 0.0434199981391
Es cadena  Head
COM cadena Head x = 0.0119333053008 y = -0.000421705743065 z = 0.168357789516
------ lista joints  ['HeadYaw', 'HeadPitch']
```

2.6.2.5 Control de los efectores.

Utilizando cinemática inversa, se realiza el control de los efectores con orientación y posición se obtiene las coordenadas articulares. Aldebaran (2016) describe el modelo geométrico del robot y dada la posición de los efectores:

$$X = [P_x, P_y, P_z, P_{wx,} P_{wy}, P_{wz}] \quad (17)$$

Estos efectores relativos al espacio absoluto en función de todas las articulaciones:

$$q = [q_1, q_2, \ldots\ldots, q_n] \tag{18}$$

$$X = f(q) \tag{19}$$

El modelo cinemático es derivado con relación al tiempo.

$$\dot{X} = \frac{\delta}{\delta t} f(q)\dot{q} \tag{20}$$

$$\dot{X} = J(q)\dot{q}$$

En la cual $J(q)$ es la matriz Jacobiana. Para controlar un efector y estimar la posición de la articulación se utiliza el modelo cinemático inverso:

$$\dot{q} = J^{-1}\dot{X} \tag{21}$$

2.6.2.6 Visión artificial en el robot NAO y reconocimiento de objetos

Se utiliza visión artificial con la libreria OpenCV y se adapta al reconocimiento para que el robot identifique los diferentes elementos de una imagen, describirla y realizar una decisión en base a esta descripción.

Se complementa el procesamiento con el uso de la librería numpy de Python para la operación a nivel matricial de las imágenes.

En el computador la imagen se representa en forma de matriz de dimensión mxn elementos, m filas y n columnas. Cada elemento o contenido de esta matriz está formada por valores enteros ubicado en localizaciones espaciales (i,j) denominadas pixel de picture element. Por lo tanto una imagen es una función de intensidad bidimensional I(i,j) donde i,j representan las coordenadas espaciales y el valor de I es proporcional a la intensidad o nivel de gris. La resolución espacial es representada por la cantidad de filas y columnas (pixel). Cada pixel tiene un valor relacionado con el nivel de luminosidad que se obtiene por la cuantización del nivel de gris o intensidad términos usados forma indistinta. Este rango de valores de niveles de intensidad se denomina resolución en amplitud Resolución de niveles de gris o intensidad de 0 a 255, el cual el 0 representa el negro absoluto (intensidades bajas) y el 255 el blanco (Howse & Minichino, 2020).

Esta imagen se representa con modelo o espacio de color. Existen varios espacios de colores. En el proyecto se utiliza el Gray, BGR, RGB, HSV y para obtener la imagen desde el robot el YUV.

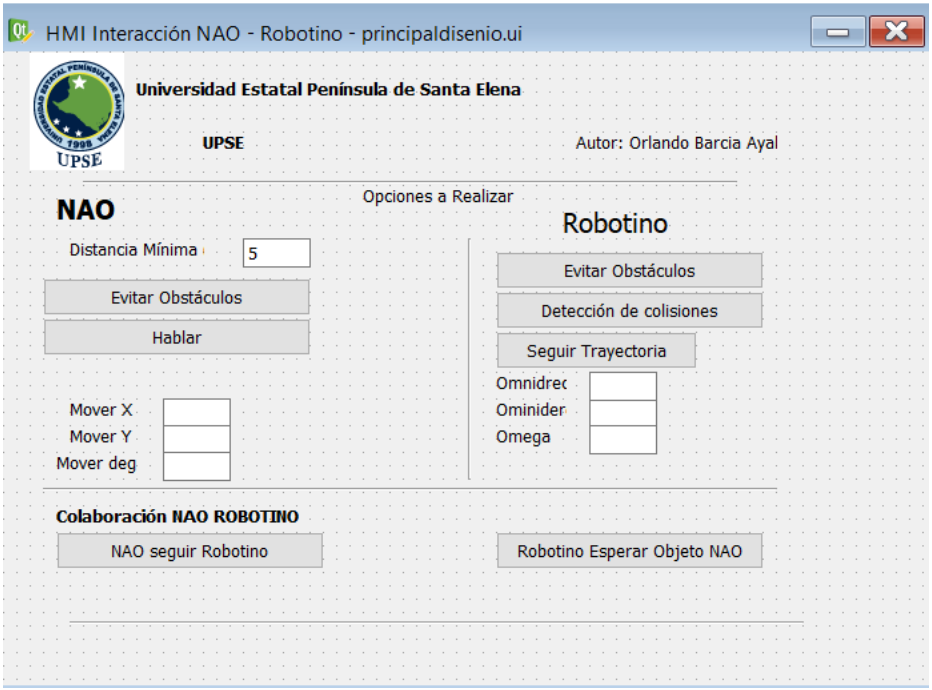
Para la Integración y pruebas en entorno controlado.

Se integran todos los algoritmos desarrollados en RobotinoView y Python 3, Python 4. Se efectúan pruebas en los simuladores. Se efectúan las pruebas en el entorno real del laboratorio de fabricación flexible. Se identifican errores y se procede a realizar los ajustes correctivos.

2.6.3 Interfaz de usuario

Utilizando Python 3 y la librería PyQt5 se realiza la interfaz

Figura 38
HMI de Interacción Robotino y NAO



Nota: Es la GUI que se crea en pyQt5 para interactuar con NAO y Robotino

La GUI realiza el envío de los comandos para que efectué el comportamiento indicado Robotino, NAO y colaboración entre los robots.

2.7 Procesamiento de la evaluación: Validez y confiabilidad de los instrumentos aplicados para el levantamiento de información.

Se ha realizado medidas de posiciones de las ubicaciones del robot, con la posición y orientación esperada y la conseguida por el robot. En sensores se efectúan las pruebas de detección de obstáculos. En sensor de visión artificial se efectúan la prueba de detección de colores. En la prueba de interfaz de usuario, se realiza la prueba de envío de mensajes. Similar en la comunicación OPC UA y los robots.

Se efectúa mediciones con instrumentos cuantitativos, captura de datos de los sensores y medir la eficiencia del sistema, tasa de éxito, productividad que permita cumplir los resultados esperados. Se tabulan los datos y se genera resultados, para medir el desempeño del sistema.

Figura 39
Medidas de posición del robot posterior al movimiento

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN

Con los materiales, métodos y metodología propuesta se describe los resultados y discusión del trabajo para cumplimiento de los objetivos establecidos.

Los videos de los comportamientos, acciones se muestran para revisión se encuentran en:

https://bit.ly/robotinoNao

Figura 40 QR de carpeta de videos de pruebas del proyecto

3.1 Comportamientos (Behaviors) y pruebas

Cada comportamiento es simulado, efectuado en el robot real y monitoreado por las cámaras del robot Robotino y NAO. Los videos se encuentran en los anexos.

ROBOTINO

- Control Teleoperado. Locomoción omnidireccional en Robotino. Mover x,y, w
- Seguidor de pared, con sensores
- Detección de colisiones. Bumper (parachoques)
- Detección de distancias, sensor infrarrojo. Evitar obstáculos Robotino
- Detección de colores

Planificación basada en sensores. Robotino

- Algoritmo Bug0
- Algoritmo Bug2

ROBOT NAO

- Teleoperado, parar, sentar, descansar
- locomoción bípeda en NAO. mover en x,y,w, en NAO

- Locomoción y hablar NAO
- Detección de distancias. Ultrasónico. Evitar obstáculos en NAO

COLABORACIÓN ROBOTS - HUMANOS

- Comunicación OPC UA Robotino
- Colaboración humana. Seguidor de color en robotino
- Colaboración humana. Seguimiento de color en NAO
- Colaboración Robots OPC UA. Nao - Robotino se desplazan
- Colaboración Robots. Nao Robotino se acercan e interactúan
- Experimentos Roberta LAB. Nao y Robotino

3.2 Simuladores versus Robots Real

Utilizando de 2.5.3 las herramientas de programación para Robotino y Nao, los simuladores de cada Robot son independientes tanto webots, choregraphe, robotinoSim y RobertaLab, por lo cual la interacción es individual. A pesar de esto, se pueden efectuar similares comportamientos en el robot real y simulado. En robotinoSim hay limitantes debido a los objetos fijos y no configuración del entorno, se sugiere robotinoSIM Pro, pero el costo de licencia queda fuera del alcance del proyecto. El simulador Webots permite agregar objetos al entorno.

Cambiando las IP tanto en robotinoView y NAO, se experimenta los mismos algoritmos con cambios pequeños y su comportamiento funciona en el entorno real.

3.3 Comunicación Python 2 desde Python 3

Debido a compatibilidad, se utiliza el lenguaje de programación Python 2 para el robot Nao. Para interacción se utiliza Python 3, con OPC UA y robotino. Se usa como IDE Pycharm community 2024

Tabla 22
Proyecto IDE Pycharm y versiones de Python

Proyecto	Ruta Local	Versión Python
pythonProjectUPSEpy2	proyectPy2UPSE	2.7.14
pyUPSEV3python	proyectUPSEv3	3.1.2

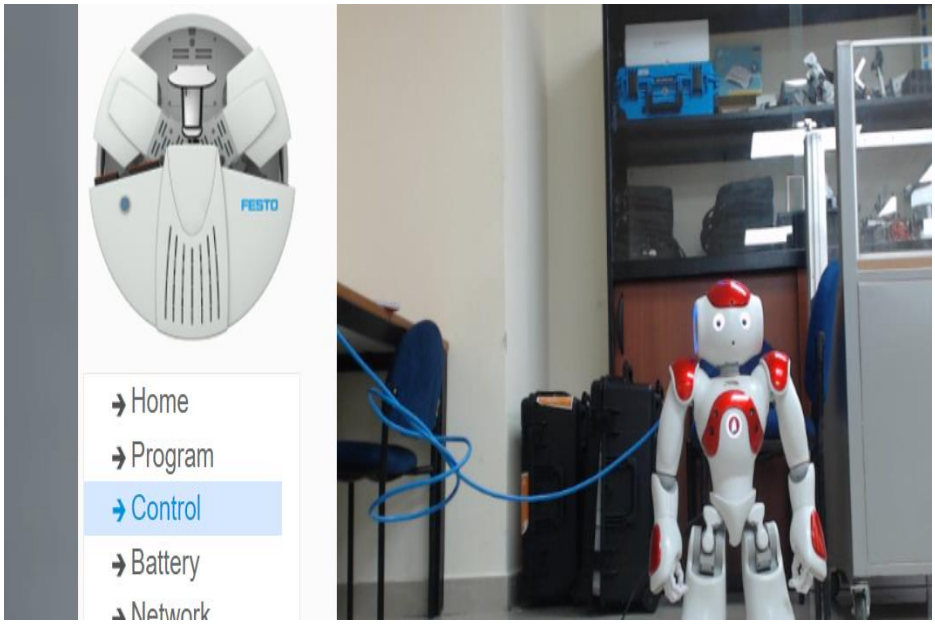
La librería opcua actual no es compatible con Python 2 que utiliza el robot NAO. Para efectuar la comunicación se procede a enviar comandos desde Python 3 y mediante la librería subprocess envía la orden a Python 2 , la cual realiza los movimientos o algoritmo del robot NAO que está en python2.

```
import subprocess

# Ejecuta el script de Python 2.7 desde Python 3

python2_interpreter = "c:/python27/python"
python2_archivos = "c:/python27/codbarcia/"

result = subprocess.run( args: [python2_interpreter, python2_archivos+"LaPruebaMov.py"]

print(result.stdout)
print(result.stderr)
```

3.4 Robotino y Nao control por web

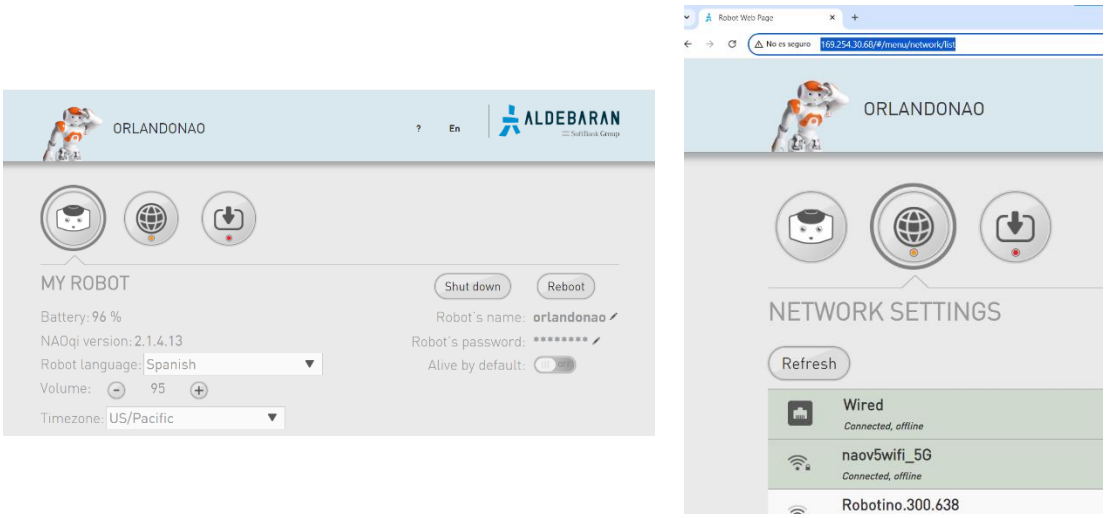
Se puede acceder a la interfaz web del servidor de robotino por su dirección IP.

Figura 41
Interfaz Web de robotino



Para NAO se puede acceder con la IP del robot, a la interface web para configurar características del robot y redes de conexión

Figura 42
Interfaz web de RobotNAO

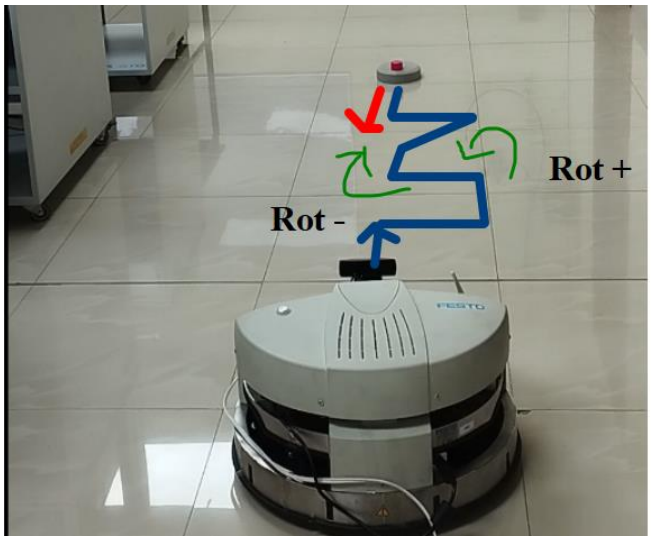


3.5 Robotino pruebas y Resultados

Las pruebas y resultados se evidencian con el requerimiento, proceso realizado y la grabación de una cámara externa y el sensor de cada robot de visión artificial. En robotino se utiliza GRAFCET, variables de transición para realizar el cambio de estado

3.5.1 Robotino Tele operado. Locomoción omnidireccional. Mover x,y,w

Requerimiento de la prueba: Robot debe moverse a varias posición, x,y y orientación w, mediante el control de un operador humano.



Proceso: El humano mediante el control panel de robotino mueve a los posición y orientación en tiempo real. Se efectúa el proceso de con un panel de control, se utiliza

solo el omnidriver y se envían las velocidades a cada motor. Se utiliza la cámara para mostrar lo que la percepción de visión artificial. El algoritmo propuesto es:

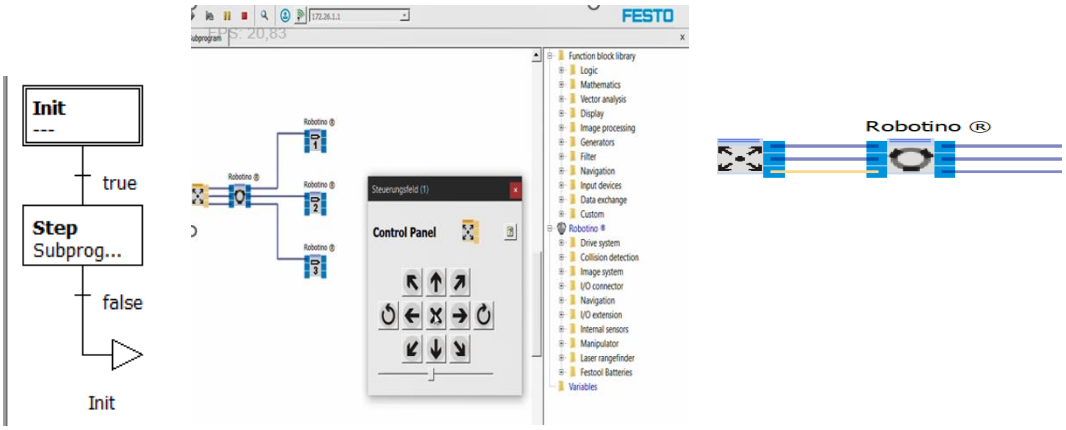
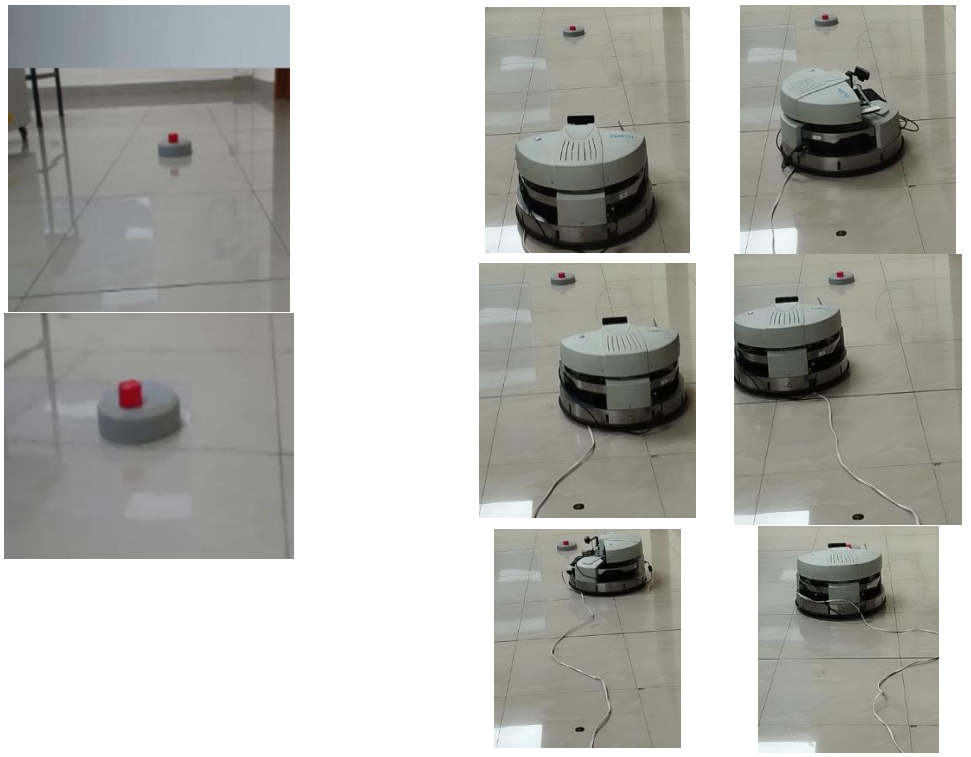


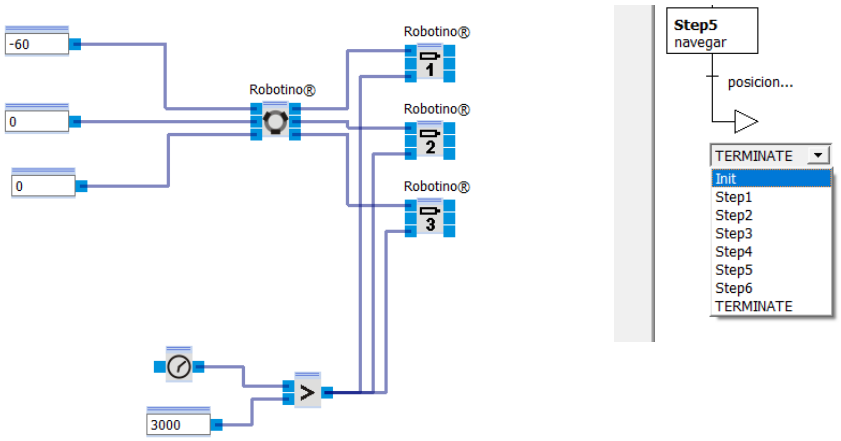
Figura 43
Robotino Teleoperado Resultado

Cámara robot **Cámara externa**



simulado

También se puede realizar movimientos directamente enviando las velocidades vx, vy en mm/s y w deg/s y que en un tiempo t en ms se detenga, Se puede efectuar en GRAFCET que el algoritmo termine o continue, donde repetir es init o al bloque requerido, sino se utiliza Terminate



3.5.2 Seguidor de pared, con sensores

Requerimiento de la prueba: Se debe realizar el seguimiento de pared lateral a una distancia x sin golpearla

Proceso: con los sensores infrarrojos , validaciones y ganancias se efectúa el seguimiento de los bordes de la pared. Cuando llega una distancia x, se aleja. El algoritmo propuesto es:

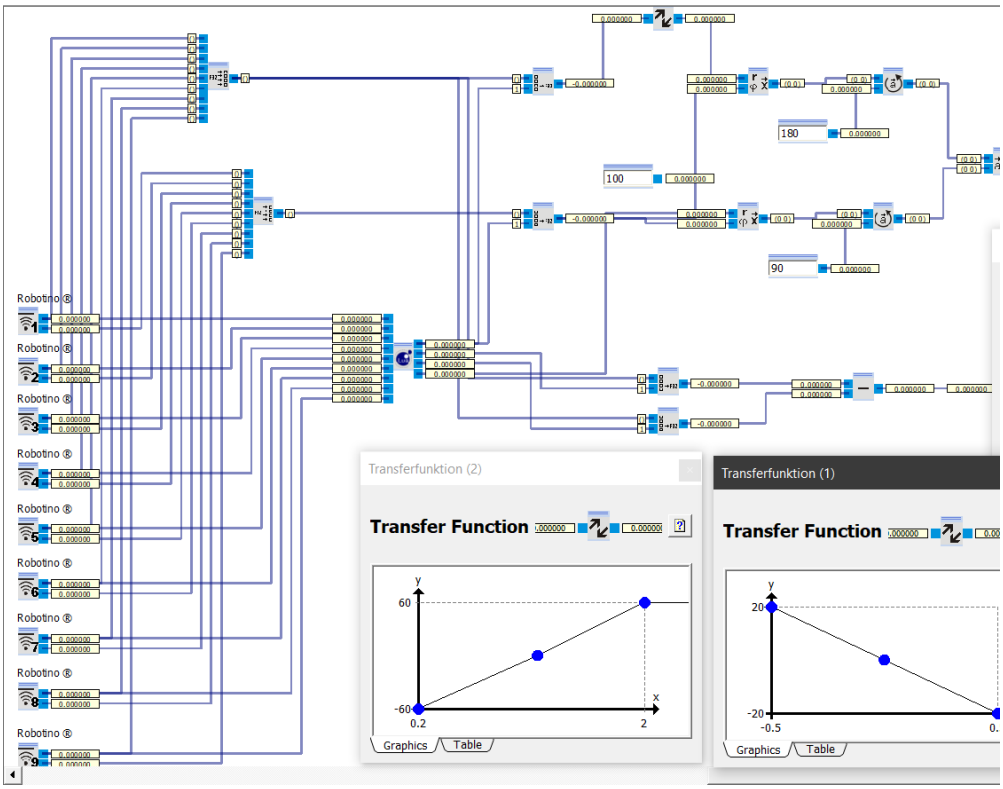
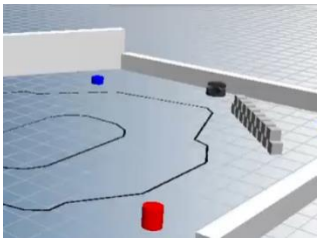
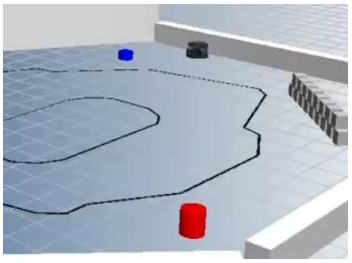
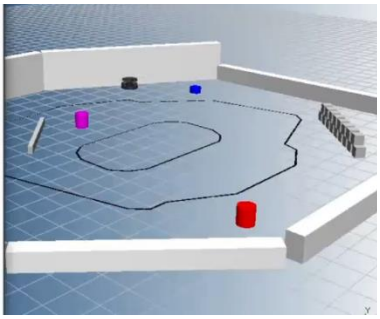


Figura 44
Robotino Seguidor de Pared

Cámara robot

Cámara externa

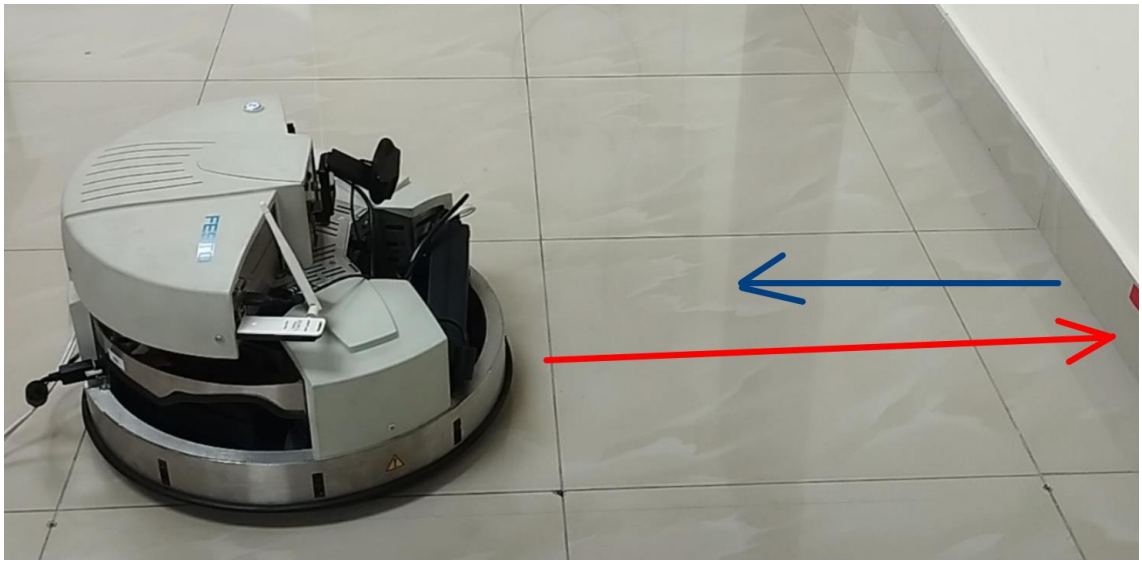
simulado

También se puede realizar movimientos directamente enviando las velocidades vx, vy

3.5.3 Detección de colisiones. Bumper (parachoques)

Requerimiento: Toca la pared y retrocede



Proceso: Mediante el sensor de bumper se detecta el tocar la pared. Se activa y se efectúa que retrocede. Por motivo de seguridad se realiza el movimiento a una velocidad lineal suave.

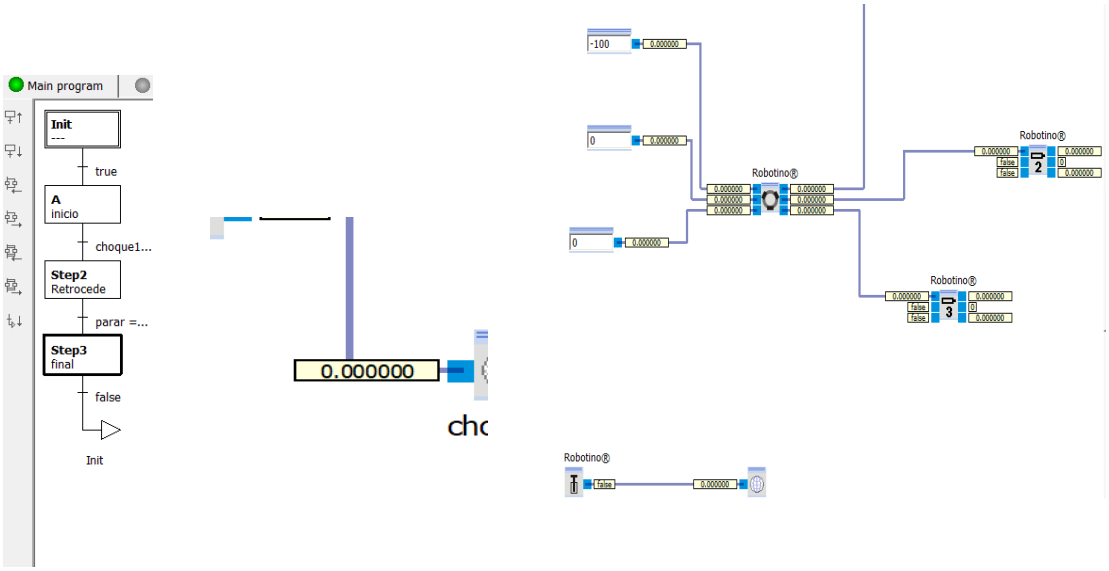
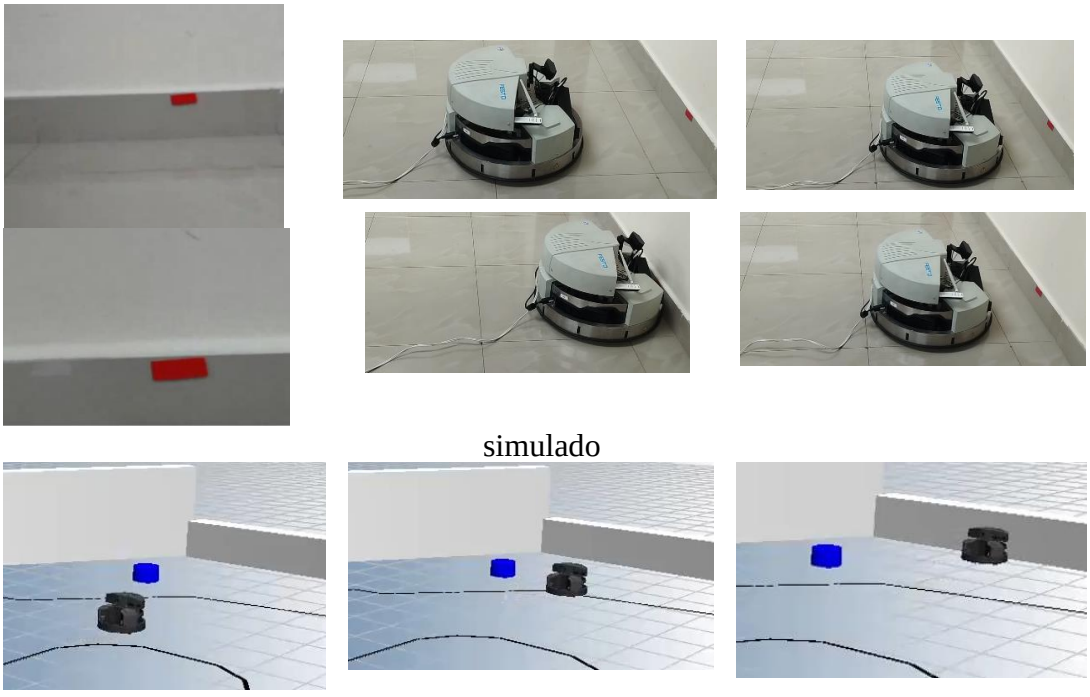


Figura 45
Robotino Detección de colisión bumper

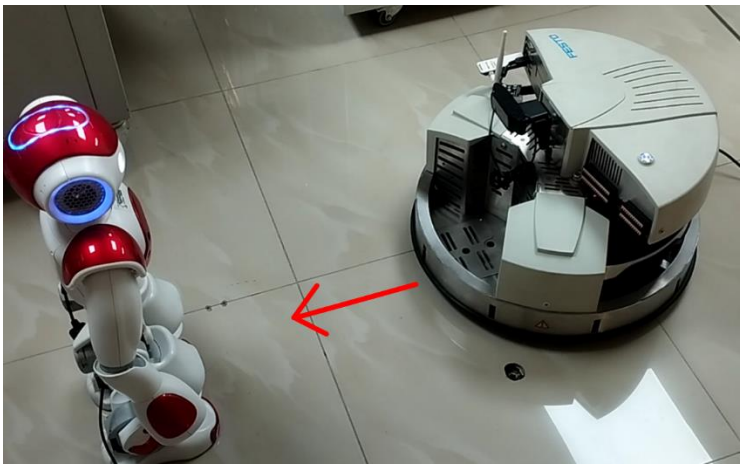
Cámara robot **Cámara externa**



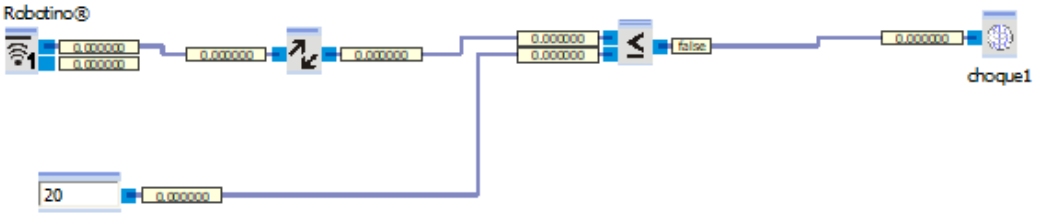
simulado

3.5.4 Detección de distancias, sensor infrarrojo. Evitar obstáculos

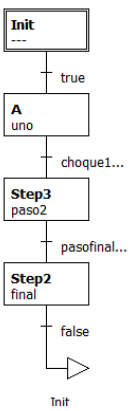
Requerimiento: Evitar obstáculo por sensor infrarrojo.



Proceso: utilizando los sensores el robot evita obstáculos. Se realiza la conversión de voltaje a distancia. Una vez que la distancia es la indicada se cambia el valor de la variable de transición, para que cambie de estado



Para Evitar obstáculos, se debe efectuar la conversión voltaje a distancia como lo indica la Tabla 13



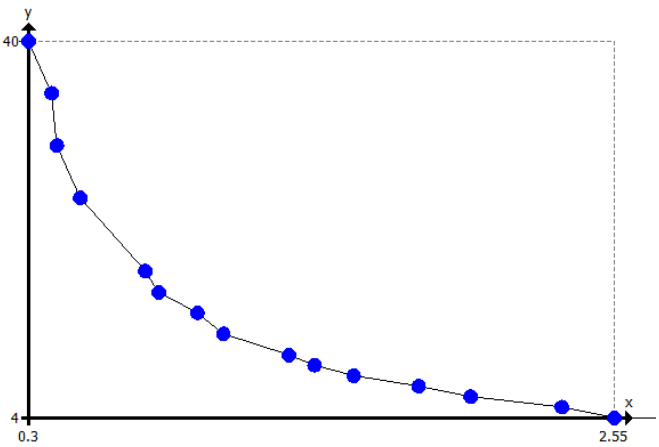
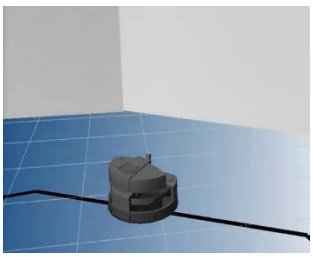
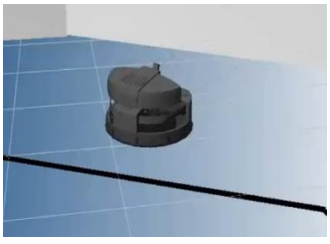


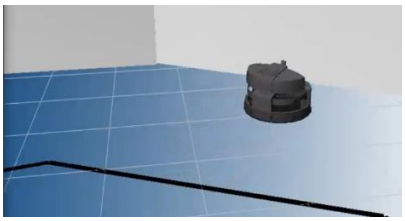
Figura 46
Robotino Detección de distancias infrarrojo

Cámara externa

simulado



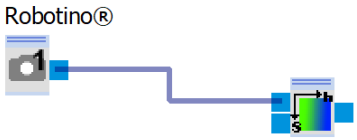




3.5.5 Detección de colores.

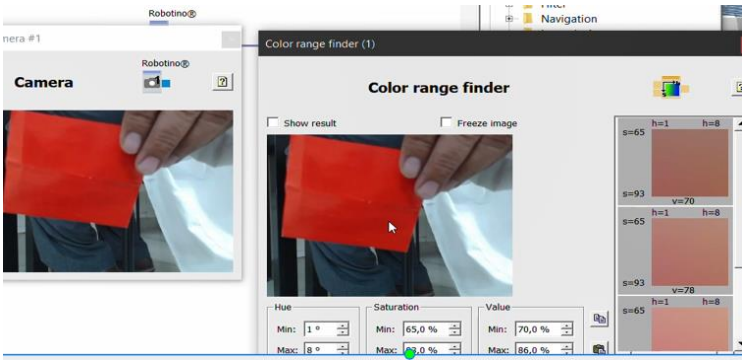
Requerimiento: obtener el color del objeto mediante visión artificial, para posterior locomoción del robot

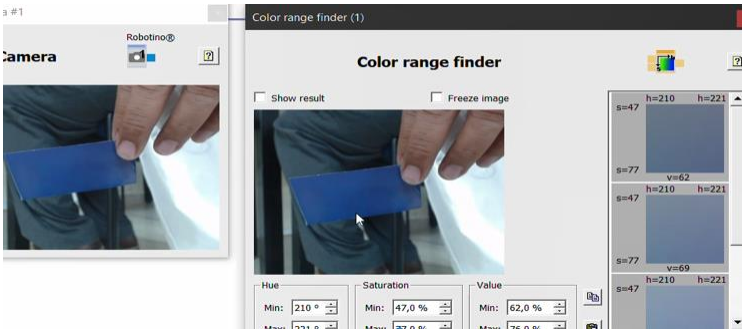
Proceso: Se utiliza el bloque color Range Finder para seleccionar el color y detectar los valores HSV. Como prueba se utiliza el rojo, verde y azul.

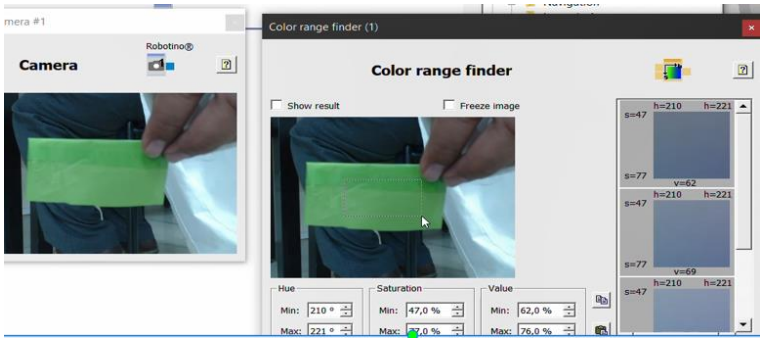


Los valores que se obtiene para color rojo, azul, verde son:

Rojo (1 8 70 94 65 84) ; Azul (211 221 58 82 52 61) ; Verde (100 106 36 45 60 66)



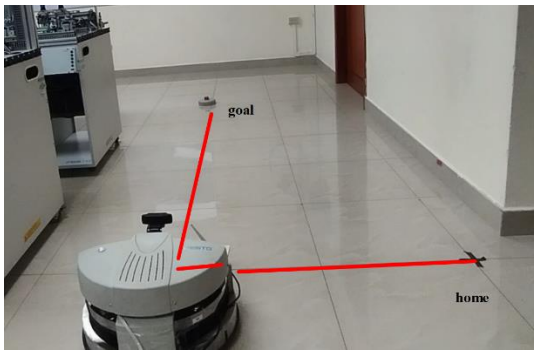




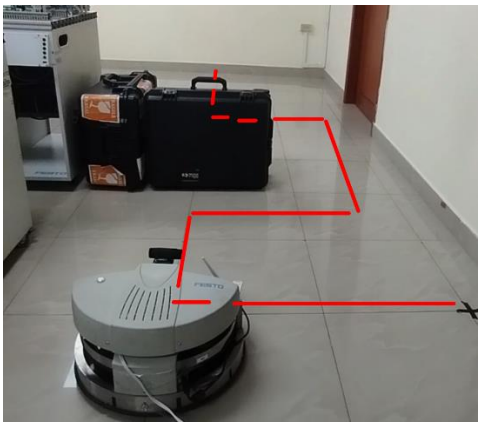
3.5.6 Algoritmo Bug0

Requerimiento: Efectuar el algoritmo bug0 en un entorno sin obstáculo y con obstáculo frontal al robot.

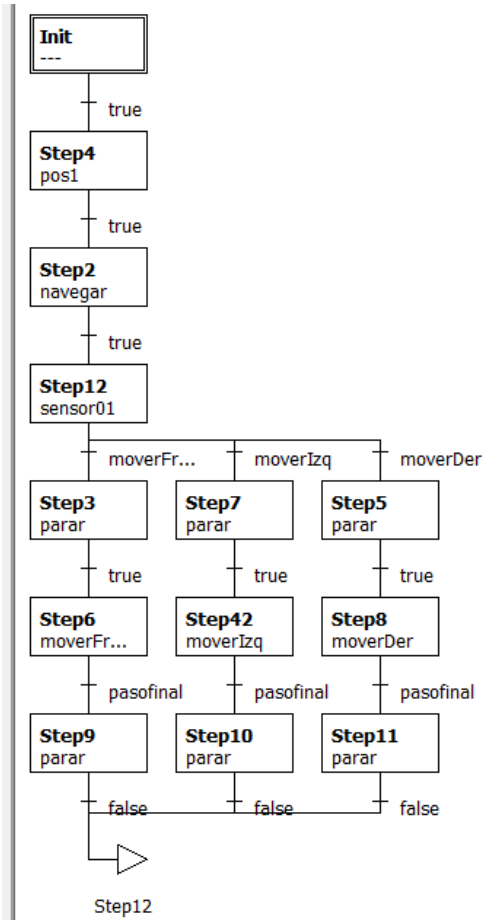
Sin obstáculo



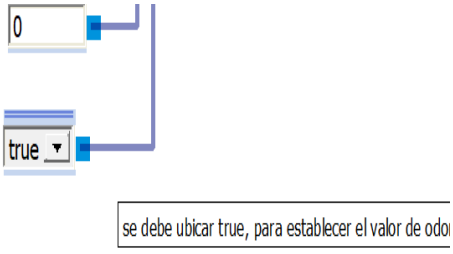
Con obstáculo



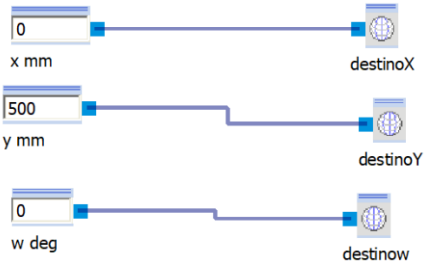
Proceso: El robot inicia desde su posición home(0,0,0), se conoce la posición y orientación destino x,y,w. Se aplica el algoritmo en grafcet, utilizando variables de transición de estados, sensores infrarrojos, odometria, positionDriver y omnidrive.



En el bloque de odometría se establece el punto de referencia global, que puede actualizarse a punto de referencia local. Al ubicar 0,0,0 true el movimiento del robot es el nuevo 0,0,0 local



Se establece la posición y orientación deseada.



Se procede a efectuar la navegación según la respuesta de los sensores y hacer la locomoción al punto asignado.

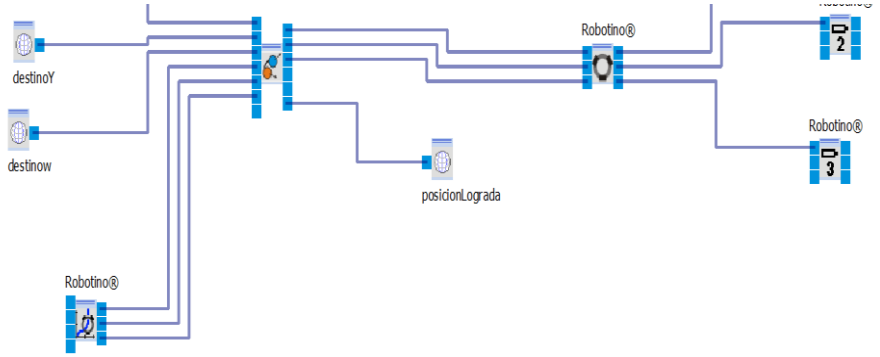
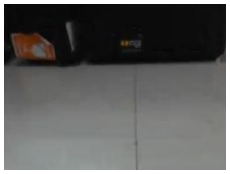


Figura 47
Robotino Algoritmo Bug0

Cámara robot **Cámara externa**





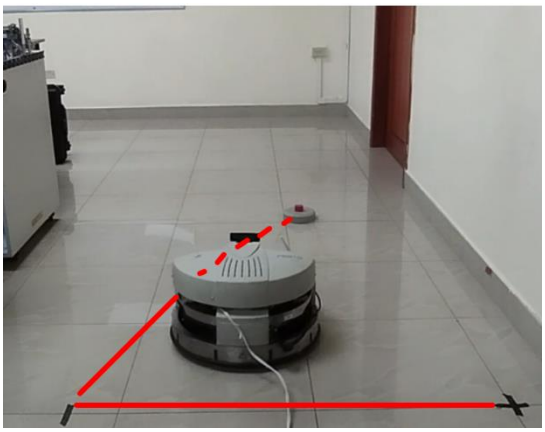




3.5.7 Algoritmo Bug2

Requerimiento: Realizar el algoritmo de bug2 sin obstáculos y con obstáculos frontal al robot

Sin obstáculo

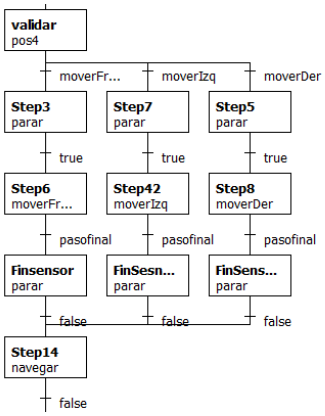


Con obstáculo

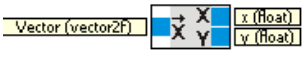
Proceso: Se aplica el algoritmo bug 2. Se genera una línea recta imaginaria al encontrar un obstáculo, se utilizan sensores, positionDriver, omniaccionamiento, odometria. Dependiendo los sensores, se actualiza la posición actual para efectuar la linea imaginaria lm al seguir la frontera aplicando la distancia entre un punto y una línea formada por dos puntos.

$$distancia_{puntoAlinea} = \frac{[(x_2 - x_1)(y_1 - y_0) - (x_1 - x_0)(y_2 - y_1)]}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \quad (22)$$

En la cual (x_0, y_0) es un punto A en el espacio y (x_1, y_1) con (x_2, y_2) es el punto que define a la linea. Se actualizan los valores en cada movimiento en los bloques.



Se utilizan los bloques matemáticos.



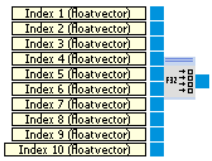
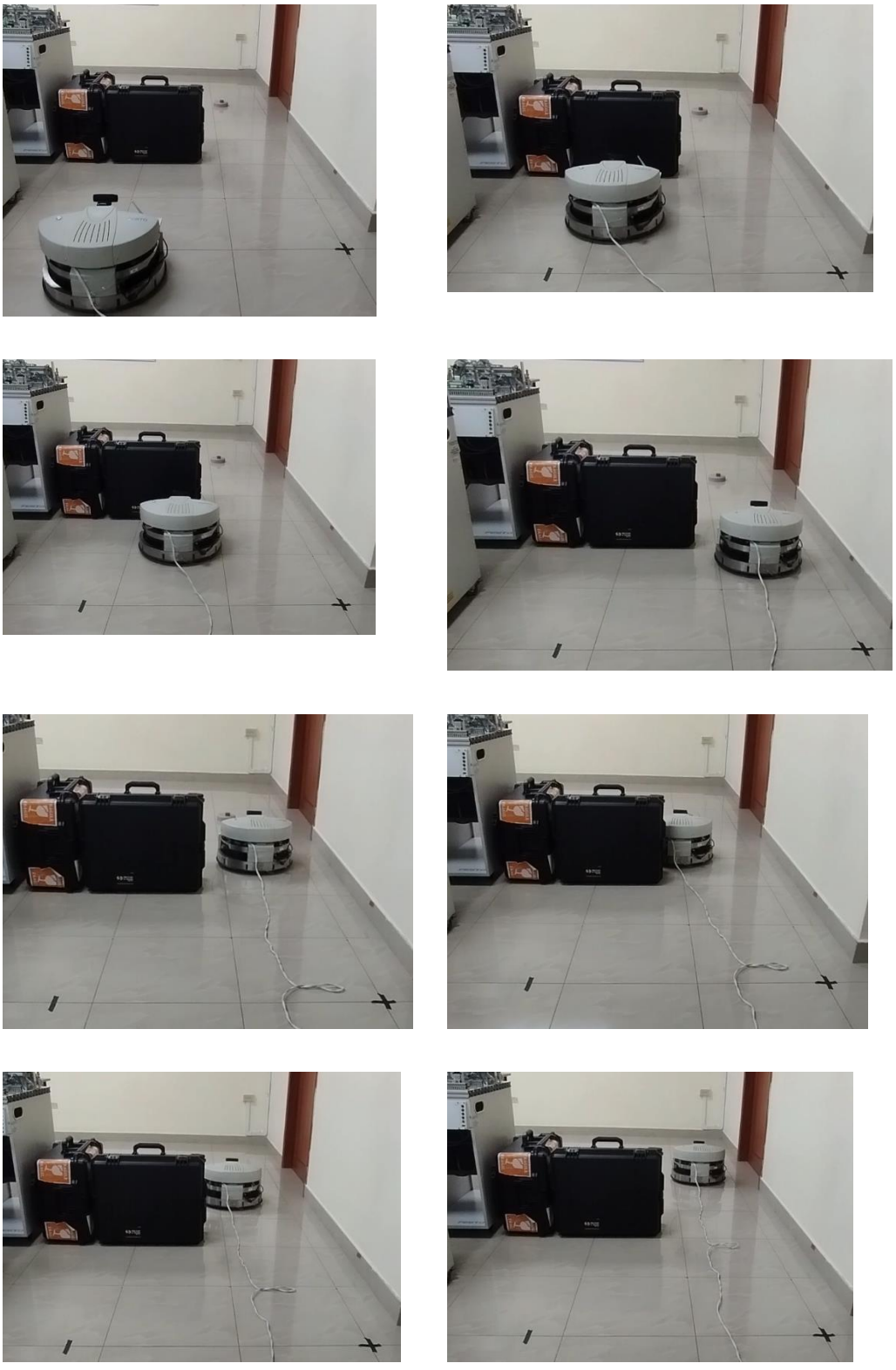


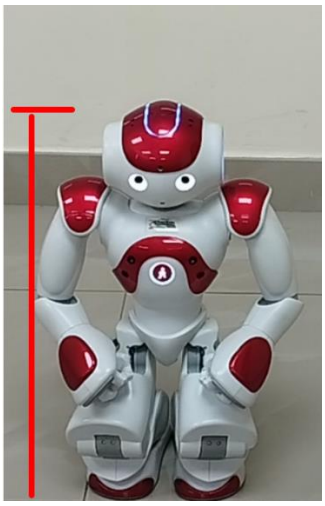
Figura 48
Robotino Algoritmo Bug2



3.6 NAO pruebas y resultados

3.6.1 Teleoperado, parar, sentar, descansar

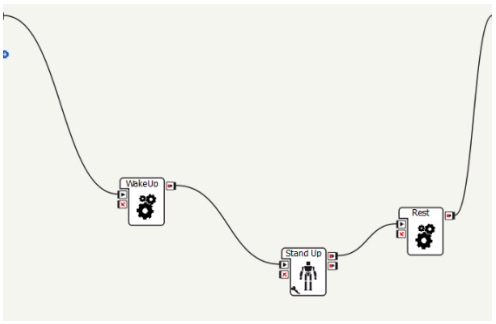
Requerimiento: El robot bípedo debe levantarse y sentarse, utilizar la visión artificial para monitorear.



Proceso: Se puede efectuar con Python o con choregraphe. Se debe realizar la conexión. En las pruebas se utiliza en la mayoría código python

Figura 49 NAO, levantar , sentar

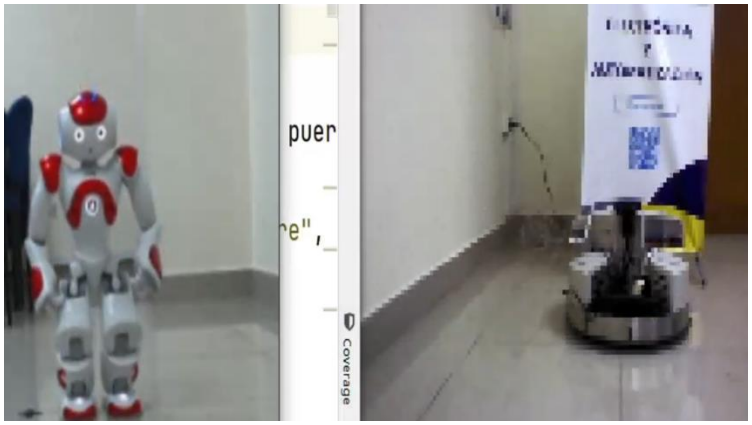
choregraphe



Python

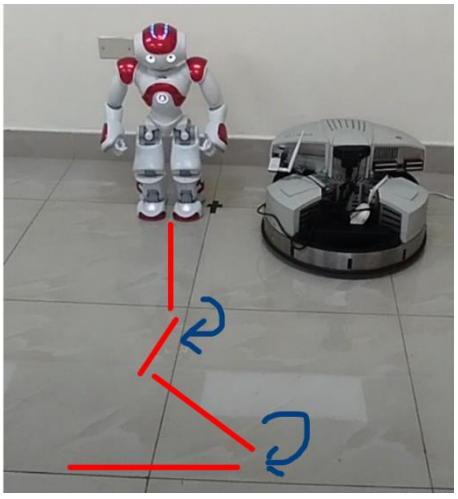
```python
from naoqi import ALProxy
# Direccion IP y puerto de NAO
ip = "192.168.0.101"  # ipreal
#ip="127.0.0.1" # ipvirtual
puerto = 9559  # Puerto por defecto

# Creacion de un proxy para moverse
movimiento = ALProxy( *args: "ALMotion", ip, puerto)
postureProxy = ALProxy( *args: "ALRobotPosture", ip, puerto)

# Wake up robot
movimiento.wakeUp()
print movimiento.getSummary()
# enviar a pose standInit
postureProxy.goToPosture("StandInit", 0.5)
print '\n-----------------------------'

# Go to rest position
movimiento.rest()
```

El robotino monitorea por visión artificial la levantada de NAO



3.6.2 locomoción bípeda mover en x,y,w, en NAO

Requerimiento: mover a un punto x,y y orientación w. Robotino observa el movimiento

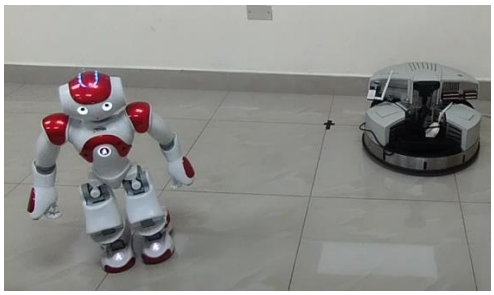


Proceso: se efectúan diferentes movimientos y rotaciones

```
#import naoqi
from naoqi import ALProxy
# Direccion IP y puerto de NAO
#ip="127.0.0.1"
ip = "192.168.0.101"  # IP ROBOT REAL
puerto=9559
# Creacion de un proxy para moverse
mimovim = ALProxy( *args: "ALMotion", ip, puerto)
# Conectarse y hacer que el robot NAO se ponga de pie
mimovim.wakeUp()
# Probar un comando de movimiento basico
mimovim.moveTo(0.5, 0, 0)  # Moverse 0.5 metros hacia adelante
mimovim.moveTo(0, 0, -10)
```

Figura 50
Locomocion mover X,Y , w

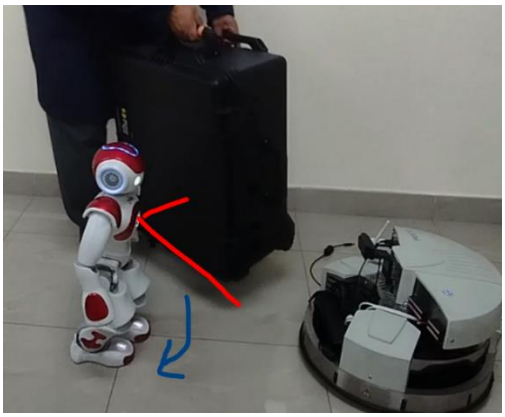
Cámara externa



Visión artificial de ambos robots

3.6.3 Detección de distancias. Ultrasónico. Evitar obstáculos en NAO

Requerimiento: Detecta obstáculos y cambia de posición



Proceso: utilizando los sensores ultrásonicos. El robot cambia de posición y orientación

Se describe el bloque de código donde se realiza la detección de obstáculos

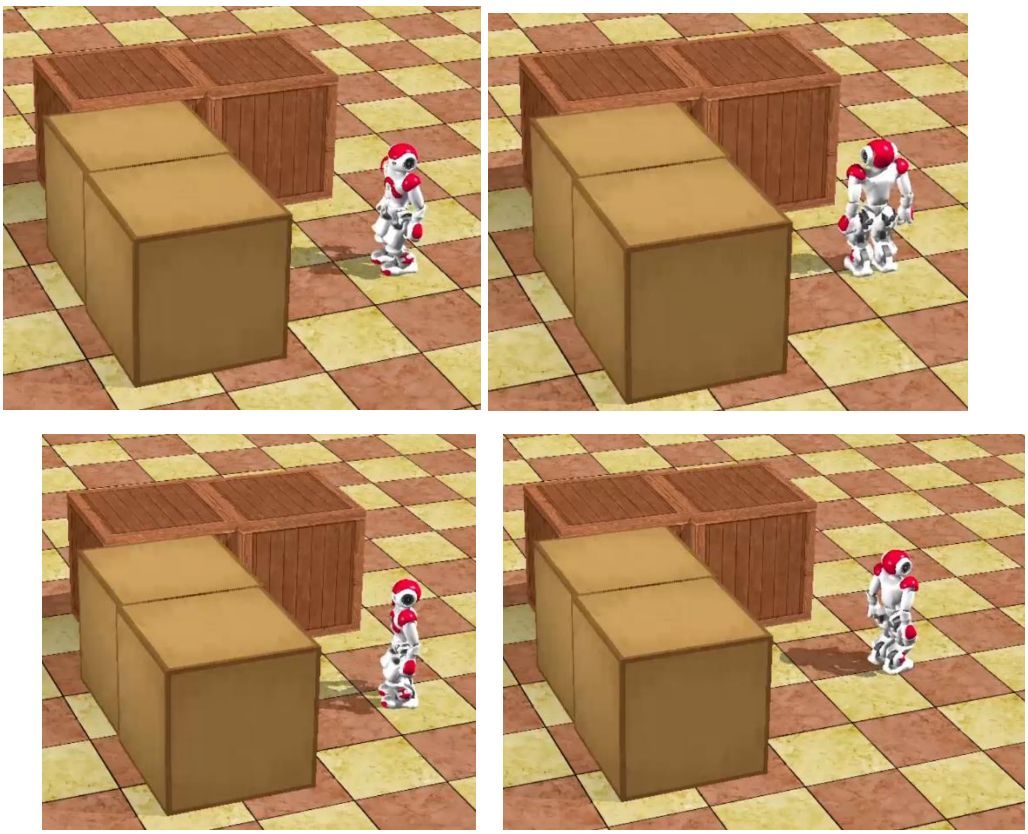
```
from naoqi import ALProxy
import sys
import math
import time
```

```
def sonar(robotIP,puerto=9559):
    # conectar al modulo sonar
    sonarProxy = ALProxy( *args: "ALSonar", robotIP, puerto)

    # subscribir los sonares para correr a nivel hw  y empezar adquisicion datos
    sonarProxy.subscribe("miupsapp")

    # crear modulo de memoria
    memoryProxy= ALProxy( *args: "ALMemory",robotIP,puerto)

    # obtener el valor del sonar derecho -distancia en metros al primer objetivo

    izq = memoryProxy.getData("Device/SubDeviceList/US/Left/Sensor/Value")
    izq1= memoryProxy.getData("Device/SubDeviceList/US/Left/Sensor/Value1")
    izq2 = memoryProxy.getData("Device/SubDeviceList/US/Left/Sensor/Value2")
    izq3 = memoryProxy.getData("Device/SubDeviceList/US/Left/Sensor/Value3")
    izq4 = memoryProxy.getData("Device/SubDeviceList/US/Left/Sensor/Value4")
    izq5 = memoryProxy.getData("Device/SubDeviceList/US/Left/Sensor/Value5")
    izq6 = memoryProxy.getData("Device/SubDeviceList/US/Left/Sensor/Value6")
```

```
def caminar(motion,dist):
    # iniciar(robotIP,puerto)
    #motionProxy = ALProxy("ALMotion", robotIP, puerto)
    motionProxy = motion
    segur= 0.6 # distancia segur
    if dist > segur:
        ladist = 0.25*(dist-segur)
        print "distancia ", ladist
        if ladist >= 0.1:
            motionProxy.moveTo(ladist,0,0)  # con move el algoritmo continua ,se us
            print "mover distancia >=0.1"
        else:
            print "distancia muy pequena para mover ", ladist, "mover 0.1"
            motionProxy.moveTo(0.1,0,0)
    else:
        print "no me muevo por seguridad"
```

Figura 51 Evitación de obstáculos con ultrasonico NAO

Simulado en webots



3.7 COLABORACIÓN ROBOTS – HUMANOS

En las interacciones o robots colaborativos se utiliza la detección de variables de transición en robotinoView, señales de sensores, finalización de movimientos, se envía los valores a cada robot que actúa dependiendo el movimiento requerido. Se utiliza los movimientos individuales efectuados en las pruebas realizadas con Robotino y NAO

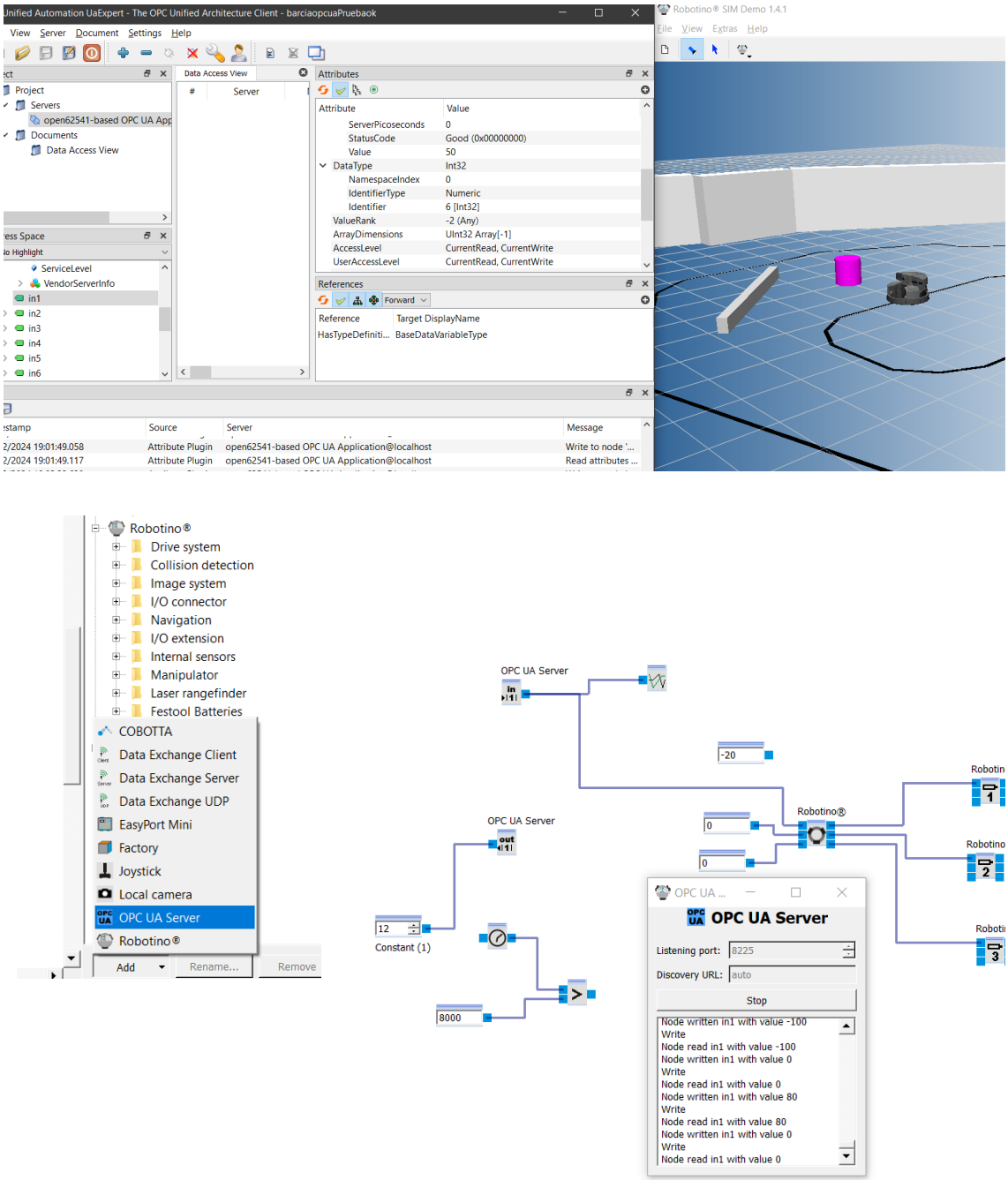
3.7.1 Comunicación OPC UA Robotino Nao

Requerimiento: El robot se comunica con OPC UA escribiendo o realizando lecturas desde un servidor OPC

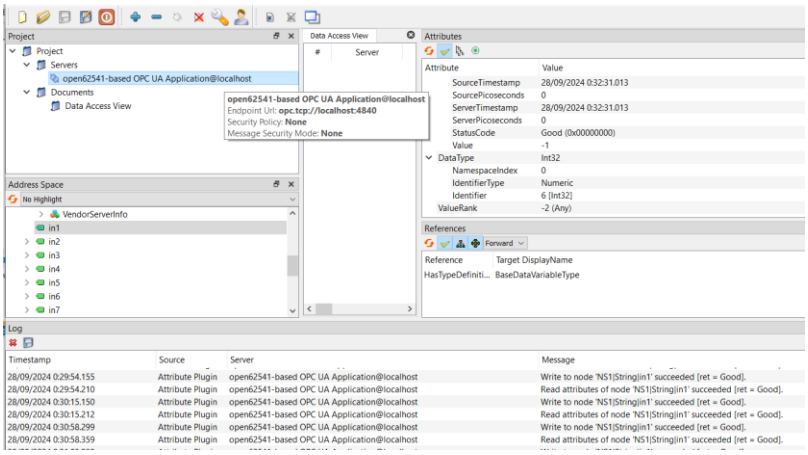
Proceso: Se establece un servidor OPC en robotino

opc.tcp://localhost:8225/" . Hay 8 entradas in, y 8 out.

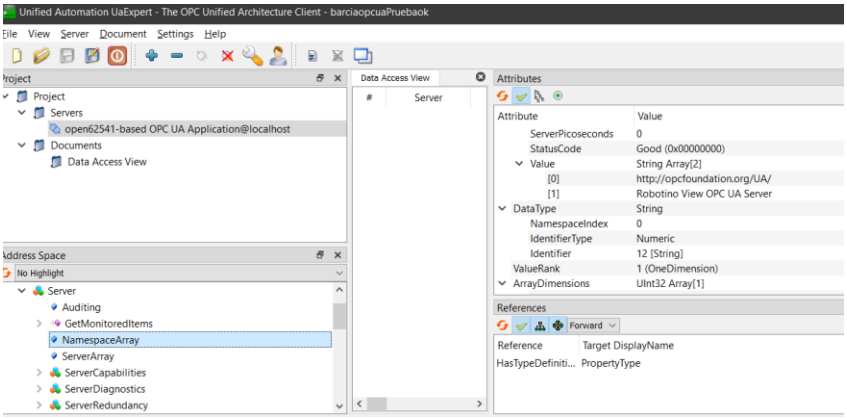
Figura 52
Comunicación servidor OPC UA Robotino y Python



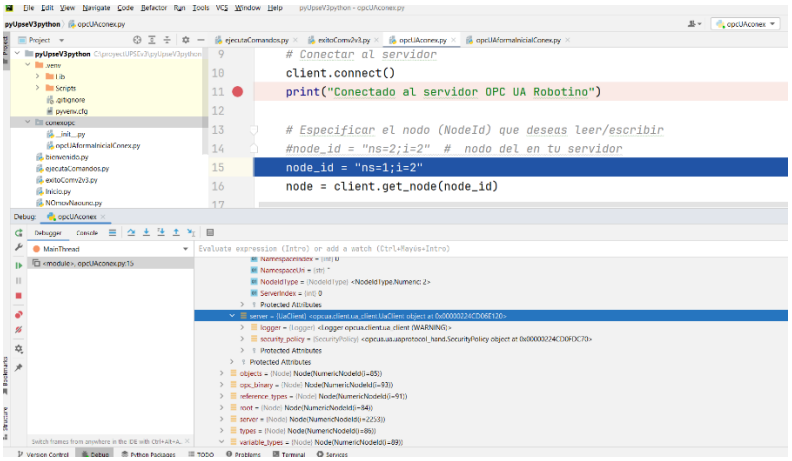
Para prueba de conexión se utiliza UAExpert, para establecer comunicación con el servidor



Muestra conexión. Robotino View OPC UA Server



En Python se utiliza la librería opcua. El cliente se conecta al servidor de Robotino



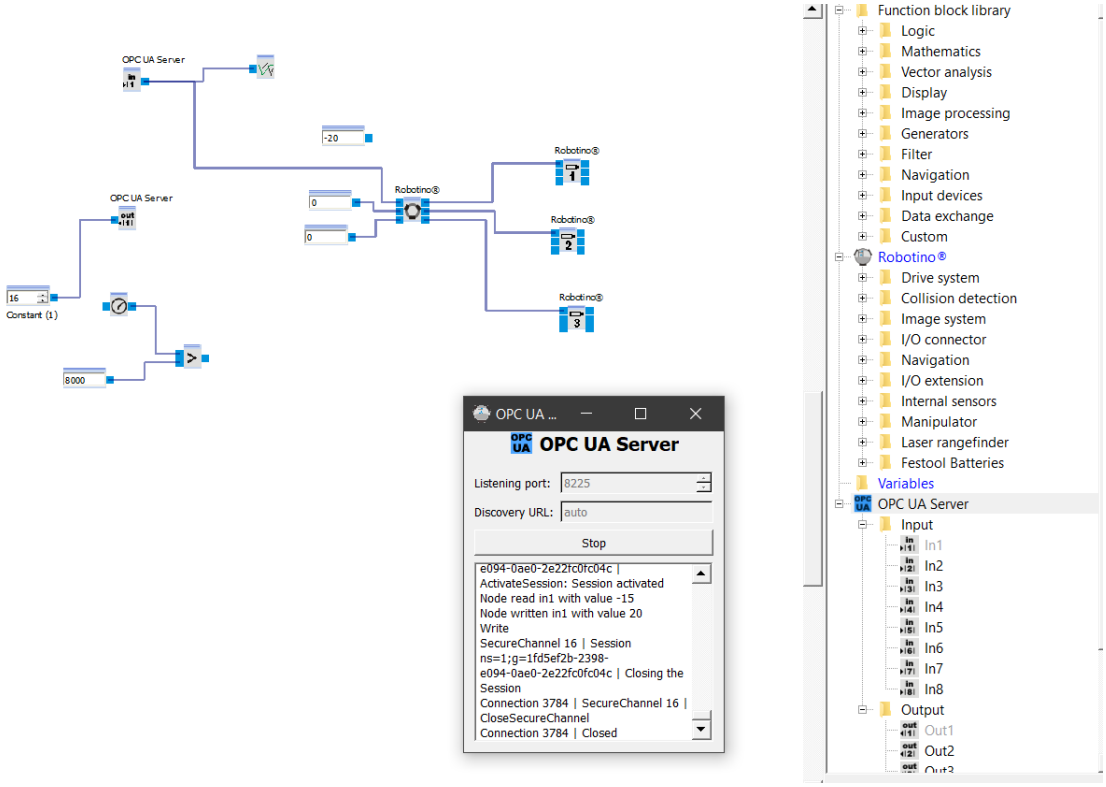
El código guía es:

```python
from opcua import Client, ua
# Dirección del servidor OPC UA
#url = "opc.tcp://localhost:4840/freeopcua/server/"
url = "opc.tcp://localhost:8225/"
client = Client(url)

try:
    # Conectar al servidor
    client.connect()
    print("Conectado al servidor OPC UA Robotino")

    # Especificar el nodo (NodeId) a leer/escribir
    #node_id = "ns=2;s=in1"  #  nodo del  servidor
    node_idIn1 = "ns=1;s=in1"
    node = client.get_node(node_idIn1)
```

La lectura/ escritura desde Python a servidor opc ua in, out

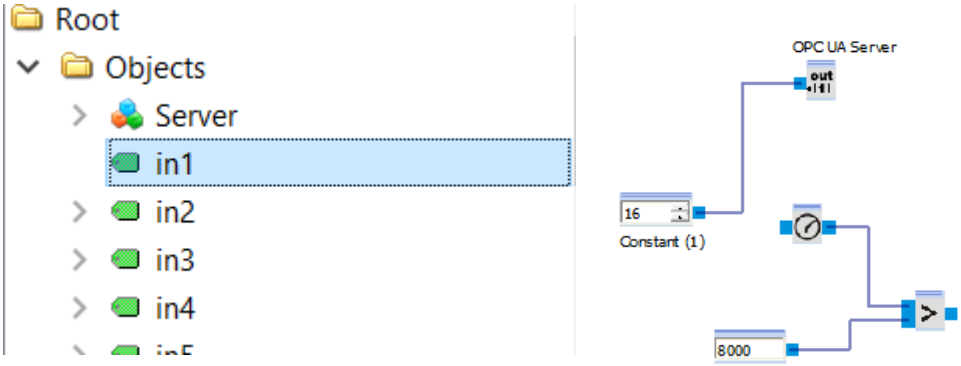
```python
node_idIn1 = "ns=1;s=in1"
node = client.get_node(node_idIn1)

node_idout1="ns=1;s=out1"
nodeOut=client.get_node(node_idout1)

valorOut= nodeOut.get_value()
print(f"Valor OUT leído del nodo {node_idout1}: {valorOut}")
# Leer el valor del nodo
valor = node.get_value()

print(f"Valor leído del nodo {node_idIn1}: {valor}")
# Escribir un nuevo valor al nodo
valornuevo = 20  # Cambia esto al valor que deseas escribir
datovalor= ua.DataValue(ua.Variant(valornuevo, ua.VariantType.Int32))
node.set_data_value(datovalor)
#node.set_value(nuevovalor)
print(f"Se escribe el nuevo valor {valornuevo} al nodo {node_idIn1}")
```

En robotinoView debe estar levantado el servidor y las variables in , out conectadas



Para lectura desde robotino, el valor se lee desde el servidor al cliente. En out1 . Esto leer desde el servidor, pero out1 escribe un dato en un elemento final de robotinoView

Con los objetos a leer o escribir

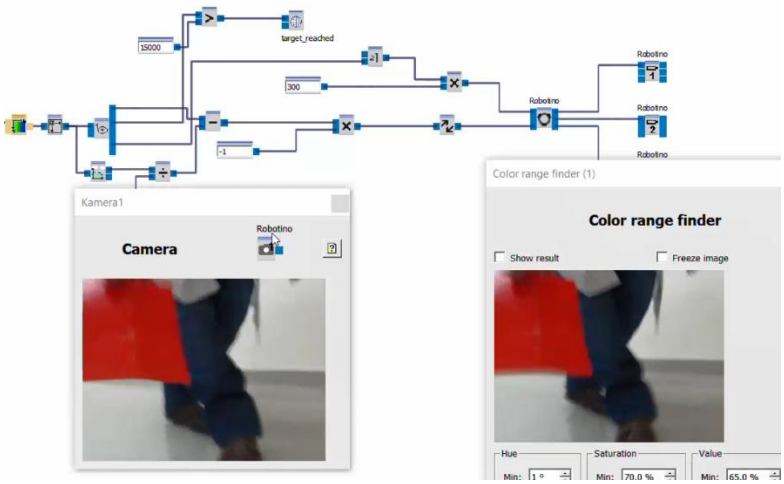


3.7.2 Colaboración humana. Seguidor de color en robotino

Requerimiento: Seguir a un humano, que tiene un landmark de color rojo.

Proceso: Se efectúa desde robotinoView, la detección de colores con ColorRangeFinder, se efectúa el movimiento con el omnidriver.

Figura 53
Colaboración humana seguidor de Color Robotino



Cámara externa

3.7.3 Colaboración humana. Seguimiento de color en NAO

Requerimiento: Seguir a un humano con landmark color rojo.

Proceso: El robot mediante Python, librería y proxy AlTracker , detecta el color rojo , aplica locomoción y sigue a la persona.

Figura 54
Colaboración humana. seguidor de color NAO

```
def main(IP, PORT, ballSize):
    motion = ALProxy( *args: "ALMotion", IP, PORT)
    posture = ALProxy( *args: "ALRobotPosture", IP, PORT)
    tracker = ALProxy( *args: "ALTracker", IP, PORT)
```

```
tracker.registerTarget(targetName, diameterOfBall)
et mode
mode = "Move"
tracker.setMode(mode)
tracker.track(targetName)
tim = 1
try:
    # while True:
    while tim <= 60:
        time.sleep(1)
        tim = tim + 1
        print tim
```

Cámara de robotino y NAO

Cámara externa

3.7.4 Colaboración Robots OPC UA. Nao - Robotino se desplazan

Requerimiento: Nao camina , robotino rueda. Se localizan, giran y retroceden.

Proceso: Se efectúa el siguiente algoritmo que ejecute el movimiento

- Se levanta NAO
- Robotino se mueve desde punto 0,0,0
- NAO camina frontal y robotino mueve frontal. se acercan
- NAO se detiene y envía señal robotino se detiene
- NAO gira, envía señal robotino gira
- NAO camina, robotino camina envía señal parar NAO

Figura 55
Colaboración Robotino- RobotNao

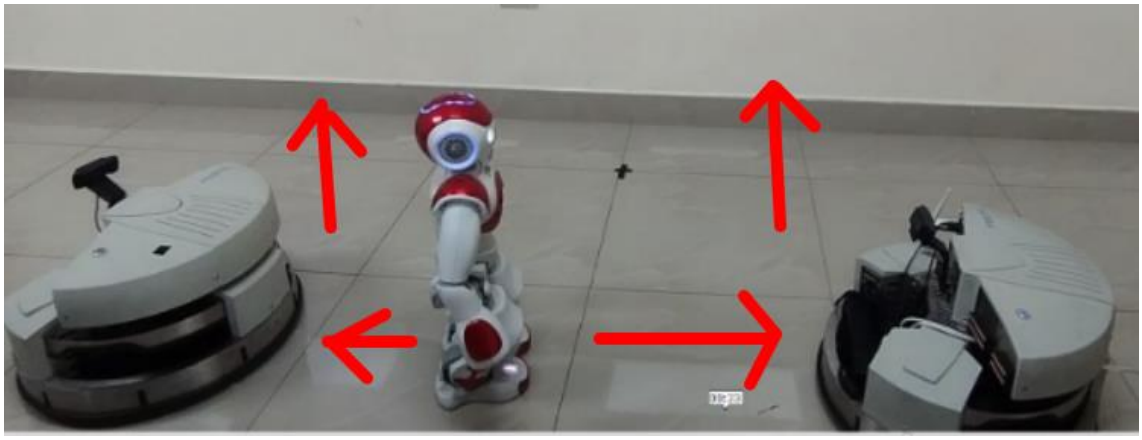
```
def principal():
    try:
        # Conectar al servidor

        conectarCliente()
        # Especificar el nodo (NodeId)
        # node_id = "ns=2;s=in1"  #  nodo del servidor
        estado=1
        r = moverNao()
        #escribirRobotinoIn("ns=1;s=in1", 20, 0, 0)
        if r.returncode == 0:
            escribirRobotinoIn( nodoIdIn: "ns=1;s=in1",  x: 50,  y: 0,  omega: 0)
```

Cámara externa.




3.7.5 Colaboración Robots. Nao Robotino se acercan e interactúan

Requerimiento: NAO y Robotino interactúan con OPC UA simulando el llevar un elemento desde robotino2 a robotino1



Proceso: El comportamiento es similar a la interacción de 3.7.4 sino que se envían a movimientos de mover la mano

Figura 56
Colaboracion NAO Robotino UPC UA Levantar DescargarMano

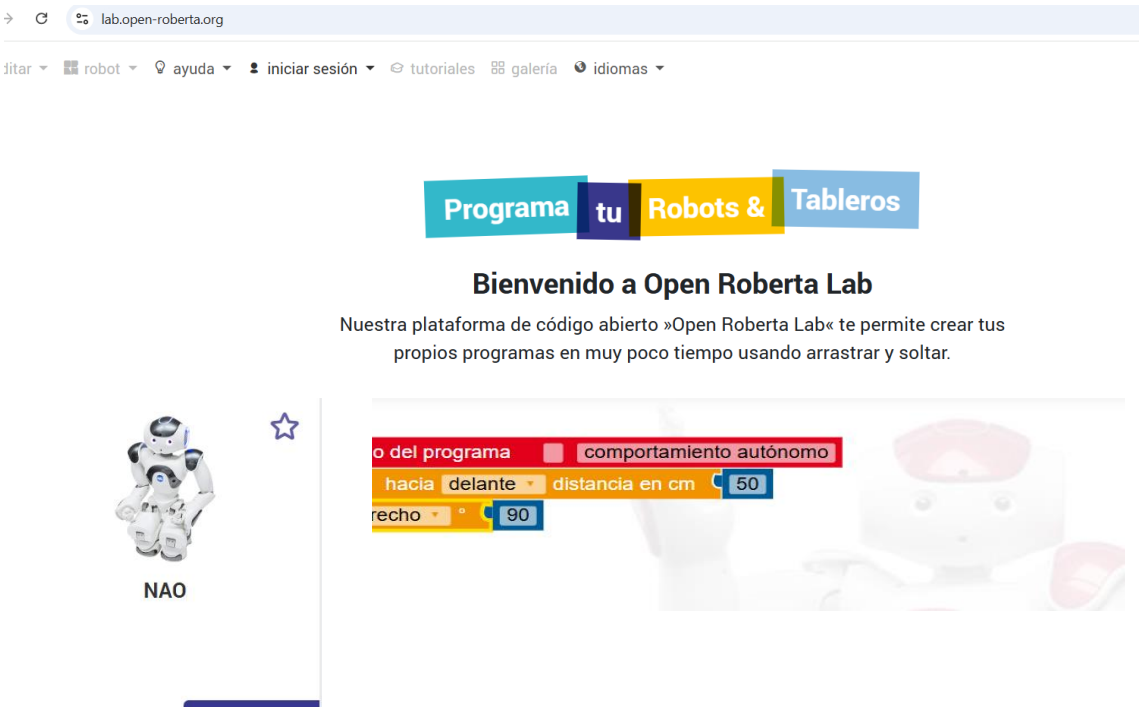
```
izquierdaAgarrar(motion)
motion.moveTo(-0.2, 0, 0)
```

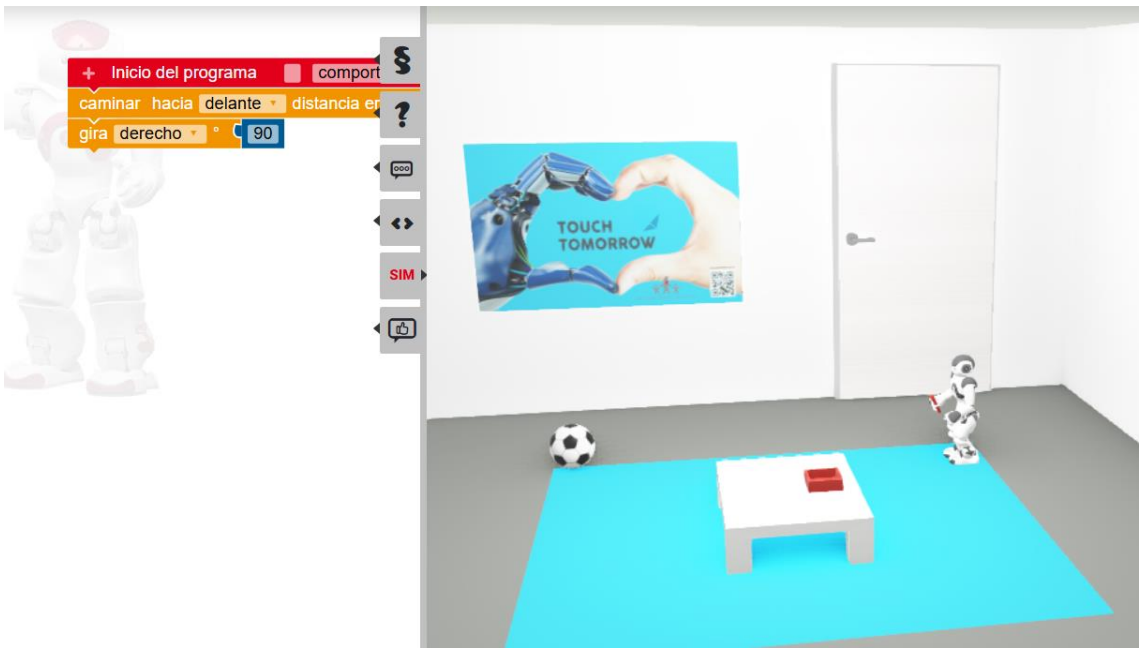
```
def izquierdaAgarrar(motion):
    # Choregraphe bezier export in Python.

    names = list()
    times = list()
    keys = list()

    names.append("HeadPitch")
    times.append([0.04])
    keys.append([[-0.181053, [3, -0.0133333, 0], [3, 0, 0]]])

    names.append("HeadYaw")
    times.append([0.04])
    keys.append([[0, [3, -0.0133333, 0], [3, 0, 0]]])
```

Cámara externa

3.7.6 Experimentos Roberta LAB. Nao y Robotino

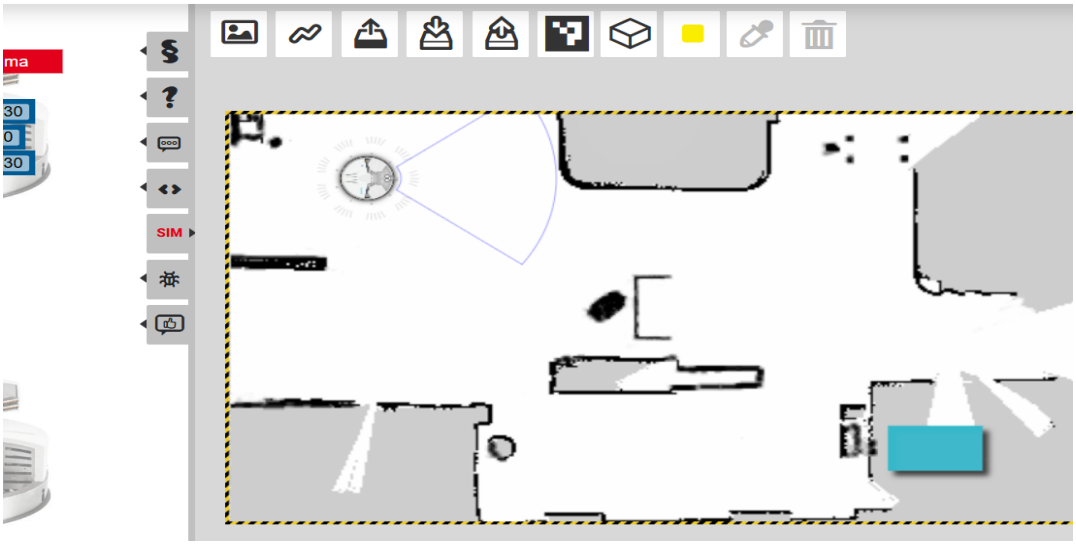
Se efectúan pruebas con el software online Roborta LAB para NAO y Robotino





```python
#!/usr/bin/python

import math
import time
import random
from roberta import Hal
h = Hal()


def run():
    h.setAutonomousLife('OFF')
    h.walk(100, 0, 0)
    h.walk(0, 0,-90)

def main():
    try:
        run()
    except Exception as e:
        h.say("Error!" + str(e))
    finally:
        h.myBroker.shutdown()

if __name__ == "__main__":
    main()
```

OpenRobertaLAB Robotino




```python
#!/usr/bin/env python3
import math, random, time, requests, threading, sys, io

def driveForDistance(RV, x, y, distance):
    angle = math.atan2(y, x)
    angle += RV.readFloatVector(1)[2] * math.pi / 180
    targetX = RV.readFloatVector(1)[0] / 10 + distance * math.cos(angle)
    targetY = RV.readFloatVector(1)[1] / 10 + distance * math.sin(angle)
    resultingSpeed = math.sqrt(math.pow(x, 2) + math.pow(y, 2))
    driveToPosition(RV, targetX, targetY, resultingSpeed)

def driveToPosition(RV, x, y, speed):
    global currentSpeed
    speed = abs(speed)
    RV.writeFloatVector(2, [x * 10, y * 10, RV.readFloatVector(1)[2], 1])
    time.sleep(0.5)
    while RV.readFloatVector(2)[3] != 1 and not isBumped():
        returnedSpeedX = RV.readFloatVector(2)[0] * 10
        returnedSpeedY = RV.readFloatVector(2)[1] * 10
        speedX = (returnedSpeedX * speed / 100)
        speedY = (returnedSpeedY * speed / 100)
        if abs(speedX) < 0.005:
            speedX = toSlowCheck(returnedSpeedX, speedX)
        if abs(speedY) < 0.005:
            speedY = toSlowCheck(returnedSpeedY, speedY)

        currentSpeed = [speedX, speedY, 0]
        time.sleep(0.05)
    currentSpeed = [0, 0, 0]

def toSlowCheck(rawSpeed, percentSpeed):
```

3.8 Discusión

El sistema desarrollado, que permite la interacción colaborativa entre NAO y Robotino, cumplió los objetivos planteados, mostrando mejoras en la locomoción, navegación autónoma y coordinación de tareas en entornos controlados.

Los resultados logrados son:

1. Coordinación y comunicación efectiva:

La implementación de algoritmos de comunicación y planificación de tareas fue exitosa, logrando una colaboración entre los robots. NAO y Robotino intercambiaron información a través de protocolos de comunicación basados en el framework Naoqi y RobotinoView, con OPC UA permitiendo la sincronización de tareas. Las pruebas mostraron que ambos robots pudieron adaptarse a las condiciones del entorno y realizar ajustes en sus patrones de movimiento, logrando una sincronización con un margen de error mínimo en la finalización de tareas colaborativas.

2. Navegación autónoma y evitación de obstáculos:

El sistema de navegación autónoma implementado para ambos robots permitió una locomoción precisa y eficiente. Los algoritmos de evitación de obstáculos resultaron efectivos en múltiples escenarios controlados, donde se evaluaron factores como la velocidad de reacción ante obstáculos y la capacidad de mantener trayectorias. NAO, con su locomoción bípedo, mostró estabilidad de sus movimientos, mientras que Robotino, gracias a su diseño omnidireccional, demostró una excelente maniobrabilidad y capacidad de ajuste de trayectoria en tiempo real. La tasa de éxito en evitar obstáculos fue superior al 90%, cumpliendo con los criterios de navegación autónoma planteados.

3. Visión artificial para el monitoreo:

La integración de un sistema de monitoreo basado en visión artificial permitió a los robots NAO y Robotino identificar objetos de color, así como detectar su entorno. Este monitoreo es para el usuario, es tipo supervisor por el humano. Las métricas de éxito en la detección y respuesta visual superaron el 95%, demostrando un alto nivel de eficiencia en las tareas colaborativas que involucraban reconocimiento visual.

4. Simulación en Webots y Robotino Sim:

Las simulaciones realizadas en Webots y Robotino Sim permitieron probar diferentes configuraciones de navegación y evitación de obstáculos antes de la implementación física. Estos simuladores facilitaron la optimización de algoritmos y el ajuste fino de los parámetros de locomoción. Los resultados obtenidos en la simulación se correlacionaron con los experimentos físicos, con un margen de error menor al 5%, demostrando la eficacia de las simulaciones para replicar escenarios reales.

5. Parametrización y configuración del sistema embebido:

La parametrización y configuración de los sistemas embebidos de NAO y Robotino se realizaron tanto en plataformas locales como web. Mediante robotinoView cambiando parámetros y Python cambiando valores de variables, se facilitó el acceso y monitoreo remoto permitió mejorar la eficiencia del sistema y la adaptabilidad del entorno controlado.

6. Interfaces de usuario:

La interface desarrollada en pyQt permite una comunicación centralizada de las operaciones pre-establecidas, optimizando el rendimiento del sistema y facilitando el ajuste dinámico de los parámetros en tiempo real. Se pudo enviar ordenes interactuar entre los robots, y realizar la locomoción. Esto facilitó el análisis y la optimización del sistema, logrando una integración eficiente entre el usuario y el entorno robótico.

A continuación, se describen las tablas de métricas que describen los resultados obtenidos en el desarrollo e implementación del sistema colaborativo entre NAO y Robotino efectuada en 20 experimentos

Las métricas utilizadas

- Métricas de Coordinación y Comunicación
- Métricas de Navegación y Evitación de Obstáculos
- Métricas de Visión Artificial
- Métricas de Simulación vs Implementación Física

Métricas de Coordinación y Comunicación:

Calcula el tiempo de sincronización en milisegundos, precisión en la comunicación y eficiencia de coordinación entre los robots

Tabla 23
Métricas de Coordinación y Comunicación

Aspecto Evaluado	NAO (%)	Robotino (%)
Tiempo de sincronización ms	98	99
Precisión en comunicación	95	96
Eficiencia de coordinación	94	95

Métricas de Navegación y Evitación de Obstáculos:

Evalúa la tasa de éxito en la evitación de obstáculos, precisión en locomoción y adaptación a cambios del entorno.

Tabla 24
Métricas de Navegación y evitación de obstáculos

Aspecto Evaluado	NAO (%)	Robotino (%)
Tasa de éxito en evitación de obstáculos	92	94
Precisión de locomoción	95	98
Adaptación a cambios del entorno	89	92

Métricas de Visión Artificial: abarca la precisión en que se muestra un objeto en la cámara y velocidad de respuesta visual

Tabla 25
Métricas de visión artificial tipo monitoreo

Aspecto Evaluado	NAO (%)	Robotino (%)
Precisión en monitoreo de objetos	97	98
Velocidad de respuesta visual	95	98

Métricas de Simulación vs Implementación Física: compara la precisión de simulaciones en Webots y Robotino Sim respecto a los experimentos físicos, así como la correlación de los resultados.

Tabla 26
Precisión de simulación respecto a experimentos físicos

Aspecto Evaluado	Webots (%)	Robotino Sim
Precisión de simulación respecto a experimentos físicos	95	96

CONCLUSIONES

Cumplimiento de los objetivos:

El sistema desarrollado cumple los objetivos generales y específicos planteados, en términos de coordinación, navegación y monitoreo. La integración de algoritmos de planificación, comunicación y monitoreo de visión artificial resultó en una interacción entre NAO y Robotino.

El uso combinado de algoritmos de planificación de rutas, evitación de obstáculos permitió a NAO y Robotino navegar de forma autónoma en entornos controlados. La integración de estos enfoques, con la posibilidad de ajustarlos a través de simulaciones en Webots y Robotino Sim garantizó una alta eficiencia en la locomoción y colaboración entre los robots.

El sistema colaborativo desarrollado entre NAO y Robotino, utilizando protocolos de comunicación, algoritmos de navegación autónoma y herramientas como Naoqi, Python, Choregraphe y RobotinoView, logró ejecutar las tareas de locomoción de manera efectiva y coordinada. La comunicación entre los robots fue fluida, con una tasa de sincronización cercana al 98% y un tiempo de respuesta adecuado para evitar colisiones y obstáculos en un entorno controlado.

En la coordinación y planificación de Tareas, los algoritmos de comunicación y planificación de tareas permitió una colaboración entre NAO y Robotino. La sincronización en la ejecución de tareas fue precisa, con una eficiencia en la coordinación que superó el 94%. La arquitectura basada en la OCP UA permite una comunicación fluida entre ambos equipos, siempre y cuando estén conectados y no existan procesamientos adicionales en el computador principal

La aplicación de los algoritmos de navegación autónoma y de evitación de obstáculos demostraron ser efectivos. NAO alcanzó una tasa de éxito del 92% en la evitación de obstáculos, mientras que Robotino logró un 94%. Ambos robots demostraron una locomoción precisa y estable en diferentes escenarios, con una adaptación adecuada a cambios en el entorno, permitiendo una ejecución autónoma confiable

El monitoreo y Visión Artificial, permitió mediante la cámara de los robots mostrar objetos y obstáculos para el humano.

La simulación de las capacidades de navegación en Webots y RobotinoSim permitió probar y ajustar los algoritmos de navegación y evitación de obstáculos antes de la implementación física. Los resultados obtenidos en simulaciones tuvieron similitud con los experimentos reales, lo que confirma la efectividad de las simulaciones para optimizar las interacciones entre los robots. La capacidad de ajustar parámetros en tiempo real y realizar simulaciones precisas contribuyó a una integración robusta.

La parametrización y configuración del sistema embebido de NAO y Robotino fue exitosa tanto a nivel local como remoto mediante plataformas web, permitiendo una supervisión eficiente de los robots. Esta funcionalidad permitió ajustes en tiempo real para mejorar el desempeño de las tareas.

La interfaz de usuario desarrollada facilitó el envio de acciones preestablecidas entre NAO y Robotino. El sistema permitió a los usuarios ajustar configuraciones en tiempo real, lo que contribuyó a la optimización del desempeño de los robots.

Este proyecto ofrece un enfoque innovador en la interacción colaborativa de robots humanoides y omnidireccionales, sentando bases importantes para futuras investigaciones y aplicaciones en entornos industriales y educativos.

Con base en los resultados obtenidos, se concluye que la interacción entre NAO y Robotino es efectiva para tareas de locomoción colaborativa

RECOMENDACIONES

Aunque los resultados fueron positivos, futuras mejoras deben en sistema de visión artificial, así como en la optimización de algoritmos de aprendizaje que permitan una mayor capacidad de adaptación en entornos dinámicos.

Para lograr una interacción colaborativa efectiva entre NAO y Robotino, se recomienda utilizar una combinación de sensores que proporcionen una visión completa del entorno, eviten obstáculos y permitan la navegación autónoma. Mientras que NAO se beneficiaría principalmente de cámaras, sensores de presión y ultrasónicos, Robotino puede aprovechar su capacidad omnidireccional con sensores como LIDAR y cámaras de profundidad, complementados por IMU y sensores infrarrojos. Esta combinación proporcionará datos precisos para la toma de decisiones autónoma y la colaboración efectiva en tareas de locomoción en entornos controlados.

Mejora en la estabilidad de la Locomoción de NAO que mostró buen desempeño en la locomoción autónoma, pero se observó que en algunos casos la estabilidad podría mejorarse en terrenos ligeramente irregulares. Se recomienda realizar ajustes en los algoritmos de equilibrio y control de locomoción, o incluso integrar sensores adicionales de presión o IMU más precisos para mejorar el equilibrio en situaciones dinámicas.

Si el sistema se implementa en entornos más complejos o de acceso remoto, se sugiere mejorar la seguridad de la comunicación entre los robots mediante el uso de cifrado TLS/SSL en el protocolo de comunicación (en caso de usar **MQTT**) o reforzar la autenticación entre nodos en **ROS**. Lamentablemente por permisos, no se pudo actualizar la versión 3 a 4 del SO del robotino

Debido a que la comunicación se realiza en un entorno de red local bien controlado, la combinación de **TCP/IP** y **OPC UA** es efectiva para equilibrar fiabilidad y velocidad en las tareas requeridas.

Para futuros trabajos, se recomienda ampliar la capacidad del sistema para soportar la interacción con más robots o dispositivos. Esto puede lograrse utilizando un protocolo de

comunicación como MQTT para escalabilidad en un entorno distribuido, o mejorando la infraestructura de red si se opta por ROS

Incorporar algoritmos de aprendizaje automático o inteligencia artificial podría mejorar la capacidad de los robots para aprender de sus interacciones y optimizar su rendimiento en tareas colaborativas. Esto permitiría una mayor autonomía y adaptación en entornos no controlados o cambiantes.

Si se consideran las recomendaciones mencionadas, el sistema podría mejorar aún más en términos de autonomía, escalabilidad y adaptabilidad a entornos más dinámicos.

REFERENCIAS

Abdo, A., Ibrahim, R., & Rawashdeh, N. A. (2020, abril 14). *Mobile Robot Localization Evaluations with Visual Odometry in Varying Environments Using Festo-Robotino*. https://doi.org/10.4271/2020-01-1022

Al-Dahhan, R. R., Rabeea Hashim AL-Dahhan, M., & Jebur, M. H. (2020). Target Seeking and Obstacle Avoidance of Omni Robot in an Unknown Environment. *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 1–7. https://doi.org/10.1109/HORA49412.2020.9152864

Aldebaran. (2016). *NAOqi Documentación*. http://doc.aldebaran.com/2-1/index.html

Ali, T. Y., & Ali, M. M. (2015). Robotino obstacles avoidance capability using infrared sensors. *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, 1–6. https://doi.org/10.1109/AEECT.2015.7360562

Amirova, A., Rakhymbayeva, N., Yadollahi, E., Sandygulova, A., & Johal, W. (2021). 10 Years of Human-NAO Interaction Research: A Scoping Review. *Frontiers in Robotics and AI*, *8*. https://doi.org/10.3389/frobt.2021.744526

Andrakhanov, A., & Belyaev, A. (2017). Navigation learning system for mobile robot in heterogeneous environment: Inductive modeling approach. *2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, *1*, 543–548. https://doi.org/10.1109/STC-CSIT.2017.8098846

Bal, F., Tekerek, M., GÖK, M., & ŞİMŞİR, R. (2024). Human Robot Interaction with Social Humanoid Robots. *El-Cezeri Fen ve Mühendislik Dergisi*. https://doi.org/10.31202/ecjse.1390219

Banaeian, H., & Gilanlioglu, I. (2021). Influence of the NAO robot as a teaching assistant on university students' vocabulary learning and attitudes. *Australasian*

Journal of Educational Technology, *37*(3), 71–87. https://doi.org/10.14742/ajet.6130

Barrientos, A., Peñín, L. F., Balaguer, C., & Aracil, R. (2007). *Fundamentos de robótica* (2a ed.). McGrawHill.

Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, *58*(3), 978–988. https://doi.org/https://doi.org/10.1016/j.compedu.2011.10.006

Blokdyk, G. (2022). *Websocket* (5STARCooks, Ed.; 2a ed.).

Bragança, S., Costa, E., Castellucci, I., & Arezes, P. (2019). A Brief Overview of the Use of Collaborative Robots in Industry 4.0: Human Role and Safety. En J. S. and B. M. P. and C. P. and C. P. and C. N. and M. R. B. and M. A. S. and P. G. Arezes Pedro M. and Baptista (Ed.), *Occupational and Environmental Safety and Health* (pp. 641–650). Springer International Publishing. https://doi.org/10.1007/978-3-030-14730-3_68

Bullo, F., & Smith, S. (2016). *Lectures on Robotic Planning and Kinematics*.

Cao, Y., Zhou, Q., Yuan, W., Ye, Q., Popa, D., & Zhang, Y. (2024). Human-robot collaborative assembly and welding: A review and analysis of the state of the art. *Journal of Manufacturing Processes*, *131*, 1388–1403. https://doi.org/10.1016/j.jmapro.2024.09.044

Chen, K., Ai, Q., Chen, J., Zhou, S., & Zhao, Y. (2020). NAO Robot Walking Control System Based on Motor Imagery. *Journal of Physics: Conference Series*, *1453*(1), 012123. https://doi.org/10.1088/1742-6596/1453/1/012123

Chen, W., Chi, W., Ji, S., Ye, H., Liu, J., Jia, Y., Yu, J., & Cheng, J. (2024). A survey of autonomous robots and multi-robot navigation: Perception, planning and collaboration. *Biomimetic Intelligence and Robotics*, 100203. https://doi.org/10.1016/j.birob.2024.100203

Ciurea, C.-F., Duka, A.-V., & Oltean, S.-E. (2014). Automatic Mapping of an Enclosure Using a Mobile Robot. *Procedia Technology*, *12*, 50–56. https://doi.org/https://doi.org/10.1016/j.protcy.2013.12.455

Cooper, S., Di Fava, A., Villacanas, O., Silva, T., Fernandez-Carbajales, V., Unzueta, L., Serras, M., Marchionni, L., & Ferro, F. (2021). Social robotic application to support active and healthy ageing. *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, 1074–1080. https://doi.org/10.1109/RO-MAN50785.2021.9515432

Corke, P. (2011). *Robotics, Vision and Control* (Springer, Ed.; 2a ed., Vol. 73). https://doi.org/10.1007/978-3-319-54413-7

Craig, J. (2017). *Introduction to Robotics: Mechanics and Control* (4a ed.).

Dhananji, A. N., & Sharmilan, T. (2024). Autonomous Mobile Robot Navigation and Obstacle Avoidance: A Comprehensive Review. *European Modern Studies Journal*, *7*(6), 260–267. https://doi.org/10.59573/emsj.7(6).2023.25

Diaz, B., Pacheco, N., & Vinces, L. (2022). Integration of a robotic arm Lynxmotion to a Robotino Festo through a Raspberry Pi 4. *2022 IEEE International Conference on Automation/XXV Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, 1–5. https://doi.org/10.1109/ICA-ACCA56767.2022.10005932

Djuric, A., Rickli, J., Sefcovic, J., Hutchison, D., & Goldin, M. M. (2018, noviembre 9). Integrating Collaborative Robots in Engineering and Engineering Technology Programs. *Volume 5: Engineering Education*. https://doi.org/10.1115/IMECE2018-88147

Ferrari, D., Alberi, F., & Secchi, C. (2023). Facilitating Human-Robot Collaboration through Natural Vocal Conversations. *2023 I-RIM Conference*.

FESTO. (2022). *Robotino FESTO*. https://ip.festo-didactic.com/InfoPortal/Robotino3/Software/Programming/EN/index.html

Festo. (2024, junio 5). *Robotino Festo Manual*. https://ip.festo-didactic.com/InfoPortal/Robotino3/Overview/EN/index.html

Figueroa, J., Montalvo, W., & Bayas, M. (2023). *Cinemática y Dinámica de robots móviles con ruedas* (A. Poveda, Ed.).

Gardenghi, C., & Gherardi, L. (2024). Teaching With the Nao Robot: Teacher - Users' Attitudes. *ItalianJournalofSociologyofEducation*, *16*(1), 71–86. https://doi.org/10.14658/PUPJ-IJSE-2024-1-4

Goralski, W. (2017). *The Illustrated Network: How TCP/IP Works in a Modern Network.*

Goualilier, D., Collette, C., & C, K. (2010). Omni-directional closed loop walk for nao. *IEEE-RAS International Conference on Humanoid Robots.*

Gucwa, K. J., & Cheng, H. H. (2015, agosto 2). An Interactive Virtual Environment for Programming Modular Robots. *Volume 9: 2015 ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications*. https://doi.org/10.1115/DETC2015-47705

Howse, J., & Minichino, J. (2020). *Learning OpenCV 4 Computer Vision with Python 3* (Packt, Ed.; 3a ed.).

Huang, Y.-C., & Lin, H.-Y. (2018). Development and Implementation of a Multi-robot System for Collaborative Exploration and Complete Coverage. *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, 472–479. https://doi.org/10.1109/SITIS.2018.00078

InfoPLC. (2024). *Robots ya hablan OPC*. https://www.infoplc.net/actualidad-industrial/item/106959-opc-ua-robotics

Jin, Y., Wen, S., Shi, Z., & Li, H. (2022). Target Recognition and Navigation Path Optimization Based on NAO Robot. *Applied Sciences*, *12*(17), 8466. https://doi.org/10.3390/app12178466

Kabir, H., Tham, M.-L., & Chang, Y. C. (2023). Internet of robotic things for mobile robots: Concepts, technologies, challenges, applications, and future directions. *Digital Communications and Networks*, *9*(6), 1265–1290. https://doi.org/10.1016/j.dcan.2023.05.006

Kajita, S., & Tani, K. (1995). Experimental study of biped dynamic walking in the linear inverted pendulum mode. *IEEE Int. Conf. on Robotics and Automation*, 2885–2891.

Karydis, K., Poulakakis, I., & Tanner, H. G. (2017). A Navigation and Control Strategy for Miniature Legged Robots. *IEEE Transactions on Robotics*, *33*(1), 214–219. https://doi.org/10.1109/TRO.2016.2623343

Kashyap, A. K., Parhi, D. R., Muni, M. K., & Pandey, K. K. (2020). A hybrid technique for path planning of humanoid robot NAO in static and dynamic terrains. *Applied Soft Computing*, *96*, 106581. https://doi.org/10.1016/j.asoc.2020.106581

Katona, K., Neamah, H. A., & Korondi, P. (2024). Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot. *Sensors*, *24*(11), 3573. https://doi.org/10.3390/s24113573

Klancar, G., Zdesar, A., Blazic, S., & Skrjanc, I. (2017). *Wheeled mobile robotics* (ElSevier, Ed.).

Klancar, G., Zdesar, A., Blazic, S., & Škrjanc, I. (2017). *Wheeled mobile robotics* (El Sevier, Ed.).

Kofinas, N. (2012). *Forward and Inverse Kinematics for the NAO Humanoid Robot*.

Latip, A., Andriani, Y., Purnamasari, S., & Abdurrahman, D. (2020). Integration of educational robotic in STEM learning to promote students' collaborative skill. *Journal of Physics: Conference Series*, *1663*(1), 012052. https://doi.org/10.1088/1742-6596/1663/1/012052

Li, Y., Jiang, J., Lee, C., & Hong, S. H. (2020). Practical Implementation of an OPC UA TSN Communication Architecture for a Manufacturing System. *IEEE Access*, *8*, 200100–200111. https://doi.org/10.1109/ACCESS.2020.3035548

Lynch, K., & Park, F. C. (2017). *Modern Robotics: Mechanics, Planning, and Control*.

Ma, H., Song, A., Li, J., Ge, L., Fu, C., & Zhang, G. (2024). Legged odometry based on fusion of leg kinematics and IMU information in a humanoid robot. *Biomimetic Intelligence and Robotics*, 100196. https://doi.org/10.1016/j.birob.2024.100196

Magallán-Ramírez, D., Martínez-Aguilar, J. D., Rodríguez-Tirado, A., Balderas, D., López-Caudana, E. O., & Moreno-García, C. F. (2022). Implementation of NAO Robot Maze Navigation Based on Computer Vision and Collaborative Learning. *Frontiers in Robotics and AI*, *9*. https://doi.org/10.3389/frobt.2022.834021

Mahgoub, W. M. E., & Sanhoury, I. M. H. (2018). Adaptive Navigation Planner for Autonomous Locomotion Control of Nonholonomic Wheeled Mobile Robot. *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 1–5. https://doi.org/10.1109/ICCCEEE.2018.8515856

Mekonnen, G., Kumar, S., & Pathak, P. M. (2016). Wireless hybrid visual servoing of omnidirectional wheeled mobile robots. *Robotics and Autonomous Systems*, *75*, 450–462. https://doi.org/10.1016/j.robot.2015.08.008

Moro, C., Lin, S., Nejat, G., & Mihailidis, A. (2019). Social Robots and Seniors: A Comparative Study on the Influence of Dynamic Social Features on Human–Robot Interaction. *International Journal of Social Robotics*, *11*(1), 5–24. https://doi.org/10.1007/s12369-018-0488-1

Muntean, E., & Leba, M. (2024). Advancing Humanoid Robotics: Leg Control and Locomotion. *2024 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 1–5. https://doi.org/10.1109/AQTR61889.2024.10554215

Niku, S. B. (2020). *Introduction to Robotics: Analysis, Control, Applications* (Wiley, Ed.; 3a ed.).

OPC Foundation. (2024). *Basics OPC UA*. https://reference.opcfoundation.org/Robotics/v100/docs/4.2.2/

Ouafa, S., Nadir, Y., Qbadou, M., & Mansouri, K. (2024). Comparative Study of the Impact of Robot-Human Integration in the Context of Learning Scenarios: Literature Review. *2024 4th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, 1–7. https://doi.org/10.1109/IRASET60544.2024.10548633

Pulver, T. (2019). *Hands-On Internet of Things with MQTT* (Packt Publishing, Ed.).

R, C., G, D., J, S., S, R., & A, R. (2023). Human Robot Interaction with Nao. *2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, 1–5. https://doi.org/10.1109/ICAECA56562.2023.10200502

Rudin, N., Hoeller, D., Bjelonic, M., & Hutter, M. (2022). Advanced Skills by Learning Locomotion and Local Navigation End-to-End. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2497–2503. https://doi.org/10.1109/IROS47612.2022.9981198

Sabattini, L., Villani, V., Secchi, C., & Fantuzzi, C. (2019). A General Approach to Natural Human-Robot Interaction. En F. and F. A. Ficuciello Fanny and Ruggiero (Ed.), *Human Friendly Robotics* (pp. 61–71). Springer International Publishing. https://doi.org/https://doi.org/10.1007/978-3-319-89327-3_5

Sahni, Y., Cao, J., & Jiang, S. (2019). Middleware for Multi-robot Systems. En *Mission-Oriented Sensor Networks and Systems: Art and Science* (Vol. 2, pp. 633–673). https://doi.org/10.1007/978-3-319-92384-0_18

Siegwart, R., Nourbakhsh, I., & Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots* (2a ed.). MIT Press.

Softbank. (2018). *Framework NAOqi manual 2.1*. http://doc.aldebaran.com/2-1/dev/naoqi/index.html#naoqi-framework-overview

SoftBank. (2024). *Aldebaran Robotics Solutions*. https://www.softbankrobotics.com/solution/

Souza Bezerra, C., Teles Vieira, F. H., & Queiroz Carneiro, D. P. (2023). Autonomous Robotic Navigation Approach Using Deep Q-Network Late Fusion and People Detection-Based Collision Avoidance. *Applied Sciences*, *13*(22), 12350. https://doi.org/10.3390/app132212350

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2020). *Robot Modeling and Control* (Wiley, Ed.; 2a ed.).

Spont, M., Hutchinson, S., & Vidyasagar, M. (2006). *Robot Modeling and control* (Wiley, Ed.).

Titov, A., Markov, A., Skorikov, A., Tarasov, P., Andreev, A. E., Alekseev, S., & Gorobtsov, A. S. (2017). *Autonomous Locomotion and Navigation of Anthropomorphic Robot* (pp. 242–255). https://doi.org/10.1007/978-3-319-65551-2_18

Wieber, P.-B. (2006). Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. *IEEE-RAS International Conference on Humanoid Robots*. https://doi.org/10.1109/ICHR.2006.321375

Yandún, A., & Sotomayor, N. (2012). *Planeación y seguimiento de trayectorias para un robot móvil*. https://www.researchgate.net/publication/277756099_Planeacion_y_seguimiento_de_trayectorias_para_un_robot_movil

Yang, Y., Lou, X., & Choi, C. (2022). Interactive Robotic Grasping with Attribute-Guided Disambiguation. *2022 International Conference on Robotics and Automation (ICRA)*, 8914–8920. https://doi.org/10.1109/ICRA46639.2022.9812360

Yuste, R. L. (2017). *Autómatas programables SIEMENS Grafcet y Guía Gemma con TIA Portal* (Primera). Marcombo.

Zafar, Mohd. N., & Mohanta, J. C. (2018). Methodology for Path Planning and Optimization of Mobile Robots: A Review. *Procedia Computer Science, 133*, 141–152. https://doi.org/10.1016/j.procs.2018.07.018

Zagirov, A., Chebotareva, E., Tsoy, T., & Martínez-García, E. A. (2023). A New Virtual Human Model Based on AR-601M Humanoid Robot for a Collaborative HRI Simulation in the Gazebo Environment. *2023 7th International Conference on Information, Control, and Communication Technologies (ICCT)*, 1–4. https://doi.org/10.1109/ICCT58878.2023.10347048

ANEXOS

Anexo 1: Listas de Videos

ROBOTINO

- Teleoperado.Locomoción omnidireccional en Robotino. Mover x,y, w
- Robotinoteleoperado01
- RobotinoTeleoperado02
- RobotinomoverXyWsimulado

Seguidor de pared, con sensores

- RobotinoseguidorParedCerradaExito.mp4
- RobotinoseguidorParedDOSsimulado.mp4
- RobotinoseguidorParedEnL.mp4
- RobotinoseguidorParedSimuladoAnterior.mp4

Detección de colisiones. Bumper (parachoques)

- robotinogolpeRetrocedeParar Real.mp4
- robotinogolpeRetrocedeParaSimulado.mp4
- CamaraRobotinGolpeRetrocede - Trim.mp4

Detección de distancias, sensor infrarrojo. Evitar obstáculos Robotino

- RobotinosensorDistanciayAtras.mp4
- RobotinopuntoRegresa.mp4
- CamaraposicionDistanciaprueba2.mp4

Detección de colores.

- RobotinoDeteccioncolorDeteccion.mp4

Planificación basada en sensores

Motion Planning. Planificación del movimiento

Algoritmo Bug1

- robotinobug1LineaDirecta.mp4
- Robotinobug1DirectoCamara.mp4
- bug1TotalObstaculos.mp4
- CamaraBug1Finalizado - Trim.mp4

Algoritmo Bug2

- bug2directoPunto.mp4
- Robotinobug2DirectoPuntoChoque.mp4
- CamaraBug2Directo - Trim.mp4

Comunicación OPC UA Robotino

- OPCUAEscrituraSimulado.mp4
- opcUALectura.mp4

****

ROBOT NAO

- Teleoperado, parar, sentar, descansar.mp4
- naolevantarsimulado.mp4

locomoción bípeda en NAO. mover en x,y,w, en NAO

- CamaraPosicionvoRecorridoNaoExito.mp4
- CaminaPuntosXYang - Trim.mp4

Locomoción y hablar NAO

- BienvenidaTesisMaestria- Trim.mp4

Detección de distancias. Ultrasónico. Evitar obstáculos en NAO

- CamaradeteccionObstaculos.mp4
- NaoEvitaObstaculos - Trim.mp4
- NaoEvitarobstsculosDos- Trim.mp4

****

COLABORACIÓN ROBOTS - HUMANOS

Colaboración humana. Seguidor de color en robotino

- robotinoseguidorColorRojoTodoCuerpo.mp4
- RobotinoseguidorColorRojoReal.mp4
- RobotinoseguidorColorRojoExito.mp4

Colaboración humana. Seguimiento de color en NAO

- CamaraseguidorColorNaoFinalFinal.mp4
- SeguidorNaoFinalRojo - Trim.mp4

Colaboración Robots OPC UA. Nao - Robotino se desplazan

- NaoFrenteRobotino.mp4
- NaoRobiotinoColabRetroceder.mp4

Colaboración Robots. OPC UA Nao Robotino se acercan e interactúan

- NaoFinalColaboracion – Trim.mp4

****

Experimentos Roberta LAB. Nao y Robotino

- robertaLabNao.mp4
- robertaRobotino.mp4

Anexo 2. Robotino. Características del sistema de robot móvil omnidireccional

- Diámetro: 450 mm, altura incluida la carcasa del mando: 290 mm
- Peso total: aprox. 20 kg (sin torre de montaje), carga: máx. 30 kg
- Chasis redondo de acero inoxidable con actuador omnidireccional
- Regleta protectora de goma con sensor de protección de colisiones integrado
- 9 sensores de distancia infrarrojos, 1 sensor inductivo, 2 sensores ópticos
- Cámara en color con resolución Full HD 1080p y puerto USB

Control e interfaces

- Computador: Intel Atom, 1,8 GHz, Dual-Core, 4 GB RAM, 32 GB SSD
- WLAN conforme a la especificación 802.11g/802.11b como cliente o Access Point
- Regulación del motor con microcontrolador de 32 bit y conexión libre de motor
- 2 Ethernet, 6 USB 2.0 (HighSpeed), 2 ranuras PCI Express, 1 VGA
- 1 interfaz I/O para la integración de otros componentes eléctricos

Software

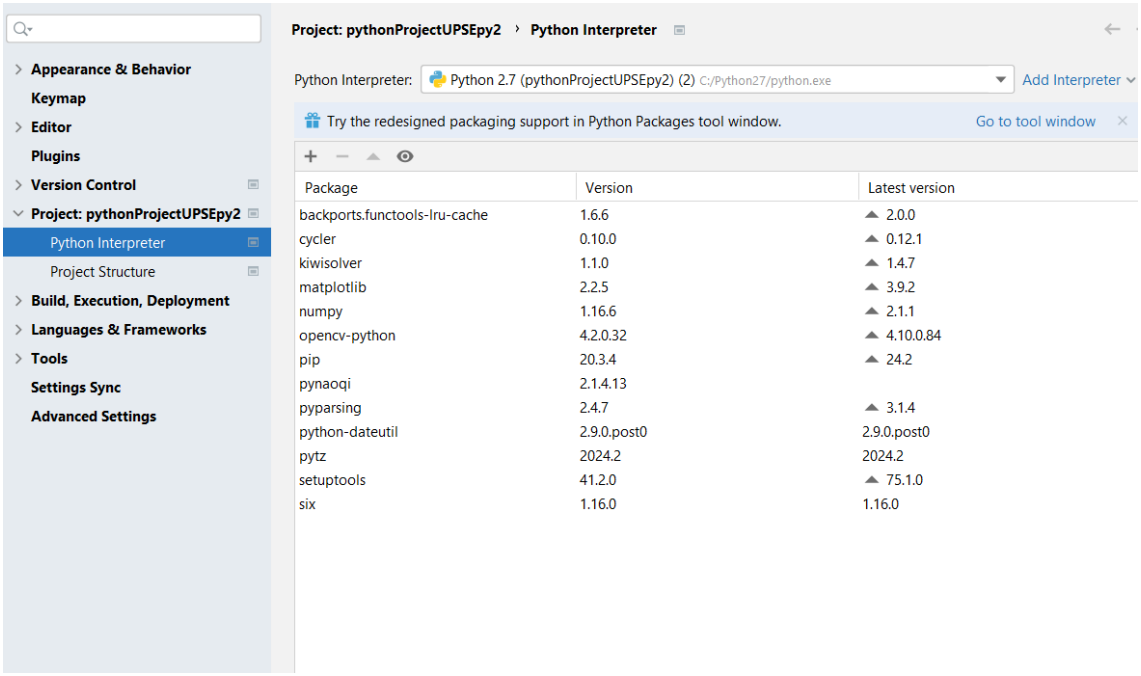
- RobotinoView y posibilidad de Matlab y Python con SO de imagen 4, que no está disponible en el robot.

Anexo 3 . NAO. Características del robot Móvil bípedo

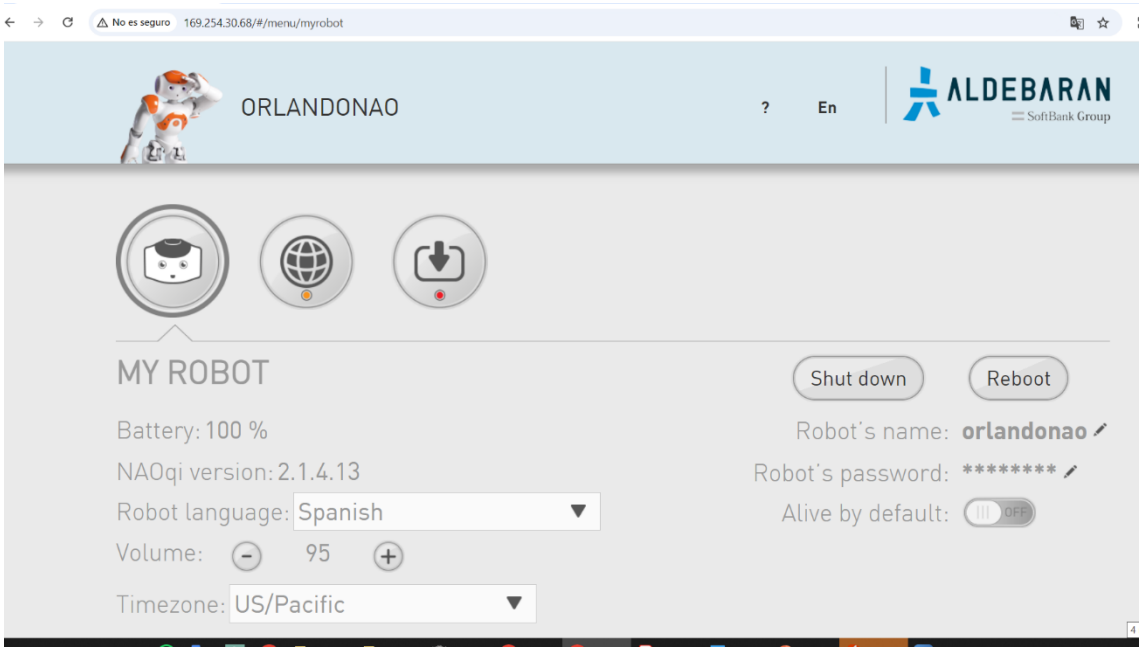
Existen versiones del robot NAO-H25, NAO-H21, NAO-T14 (solo parte superior), NAO-T2 (solo torso y cabeza).

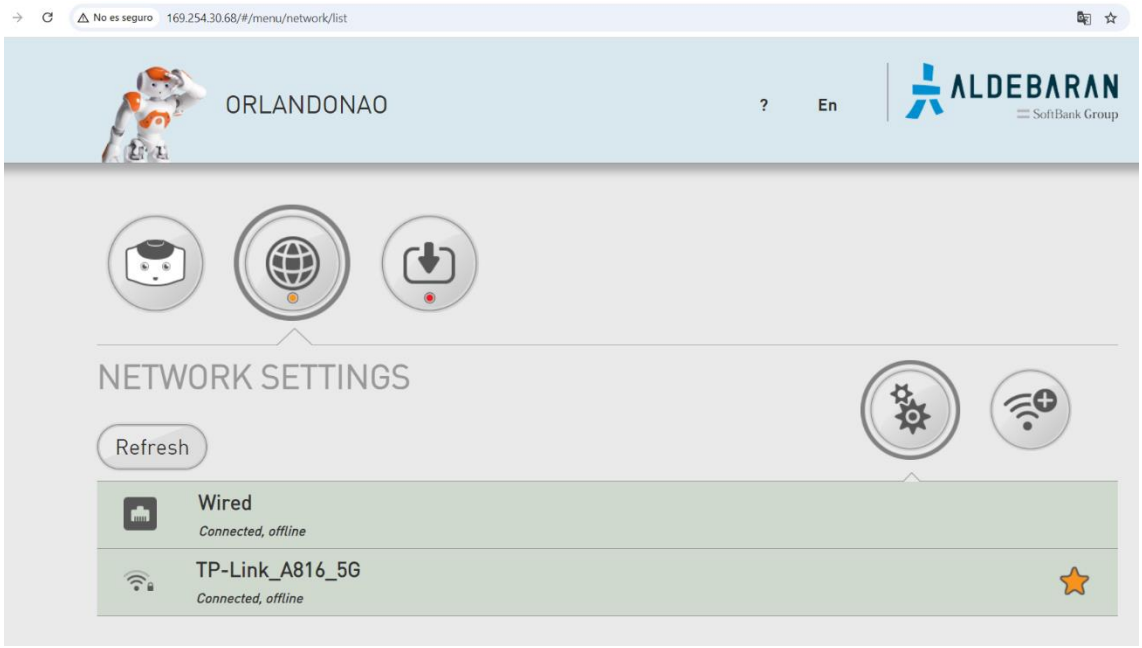
- El robot Nao versión 5. Tiene 25 grados de libertad es el utilizado en el proyecto
- Es creado por la empresa Aldebaran del grupo SoftBank Robotic
- Alto 27.4 cm, Profundidad 31.1 cm con los brazos extendidos y 27.5 cm de ancho.
- Peso de 5.3 Kg.
- Los sensores del robot están formados por dos cámaras CMOS tipo Webcam HD de máximo 1280x960 pixeles a 30 fps.
- Con dos sensores ultrasónicos, sensores resistivos de fuerza FSR. sensor inercial, infrarojos.
- Tiene micrófonos, altavoces (parlantes) y LED para obtener retroalimentación del robot.
- Existen cámaras monoculares, estéreo y omnidireccional, panorámica, RGB-D. Las cámaras que tiene el robot nao son de tipo monocular. Estan en cabeza , vista al frente y otra vista al piso.
- El computador es ATOM Z530 1.6 GHz CPU con 1 GB RAM
- para acceso externo 2 GB Flash memory con 8 GB Micro SDHC.
- es compatible con Windows, Linux y Mac Os.
- La batería tiene una autonomía de 60 minutos con uso activo y de 45 minutos con sensores de visión y ultrasonido. Para uso normal es de 90 minutos (1 hora y media). Capacidad de 48.6 Wh.
- La conectividad para computador y configuración es con un puerto ethernet 1×RJ45 - 10/100/1000 base T
- WIFI con especificación IEEE 802.11 a/b/g/n.
- puerto USB para actualizar o conexión con otros dispositivos.

Anexo 4: Librerías de Python 2

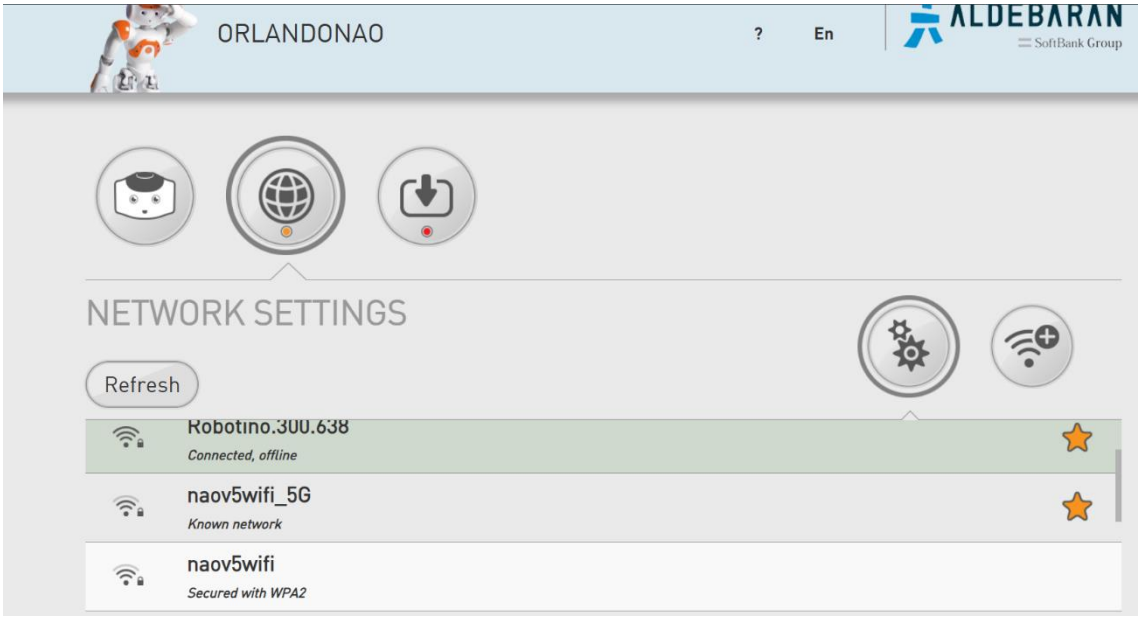


Anexo 5. Configuración Web Nao

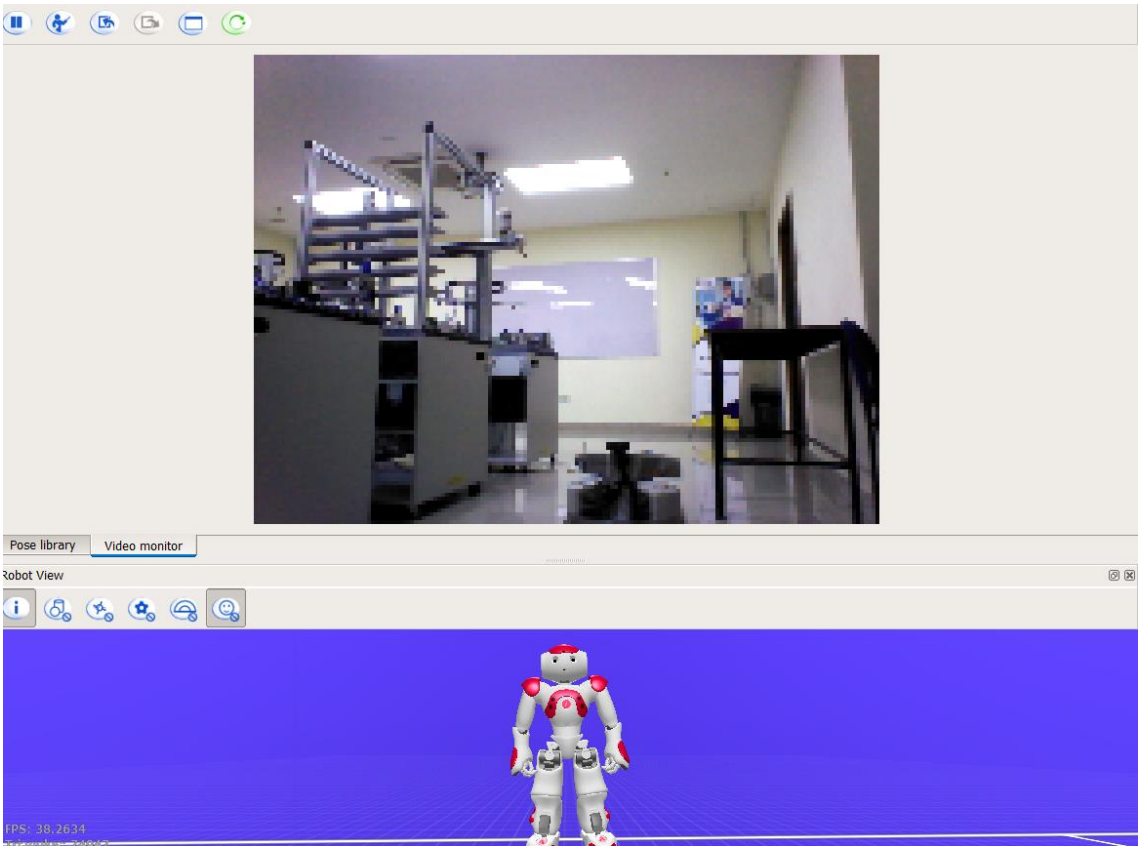




Conexión

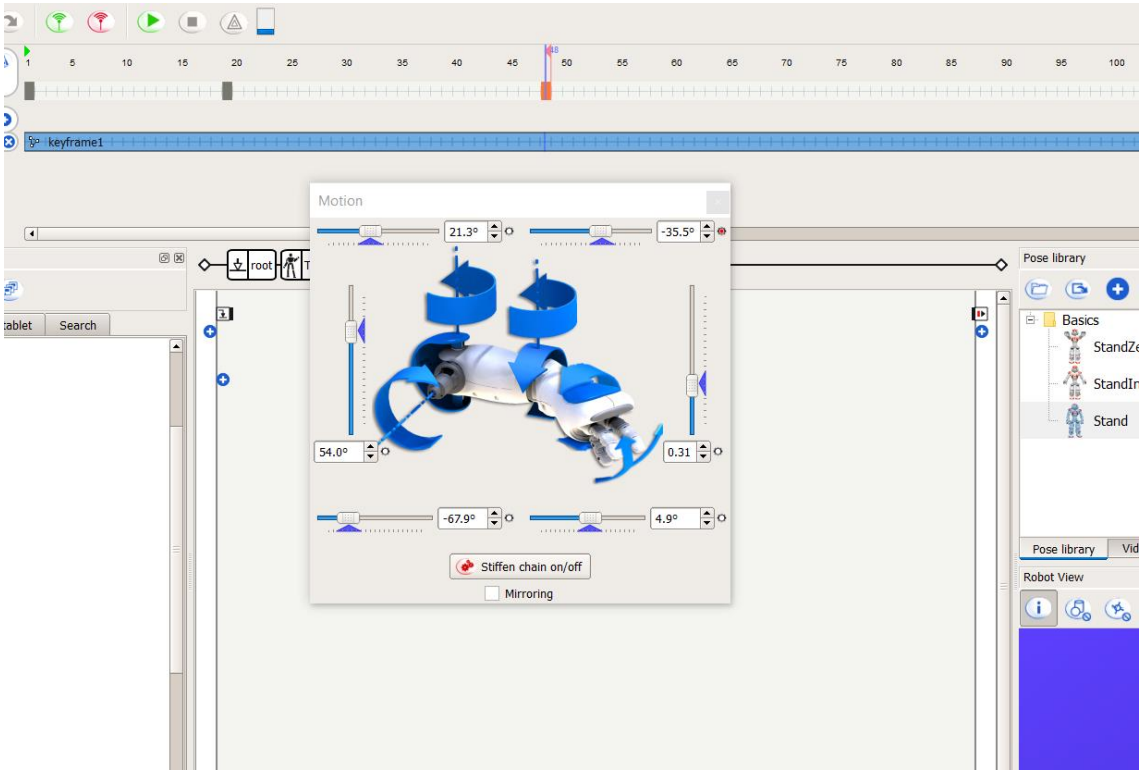


Conexión con la red wifi



Anexo 6 Generación de código de movimiento de NAO

Se utiliza choregraphe para realizar movimientos personalizados del robot , sea manos, cabeza, pierna

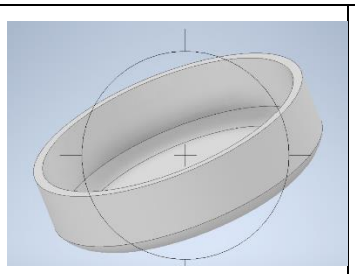


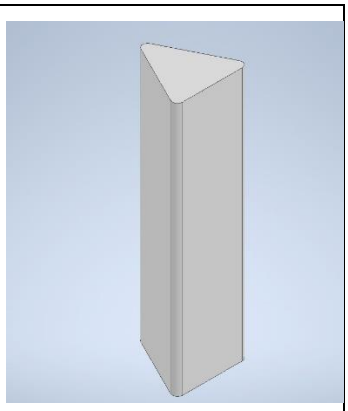
Se exporta los movimientos realizados en choregraphe a NAO Python

```
def descarga(motion):
    # Choregraphe bezier export in Python.
    from naoqi import ALProxy
    names = list()
    times = list()
    keys = list()

    names.append("HeadPitch")
    times.append([0.04])
    keys.append([[-0.170316, [3, -0.0133333, 0], [3, 0, 0]]])

    names.append("HeadYaw")
    times.append([0.04])
    keys.append([[0, [3, -0.0133333, 0], [3, 0, 0]]])
```

Anexo7 Diseño de Torre de montaje

Debido a que el robot real no tiene la torre de montaje





Está se ubica en la parte superior del robot