ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO Y SIMULACIÓN DE UN SISTEMA DE TELEOPERACIÓN DE UN ROBOT HUMANOIDE NAO

DISEÑO Y SIMULACIÓN DE ALGORITMOS DE EVASIÓN DE OBSTÁCULOS PARA UN HUMANOIDE NAO

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

ALEX JAVIER YUGCHA NAVARRETE

alex.yugcha@epn.edu.ec

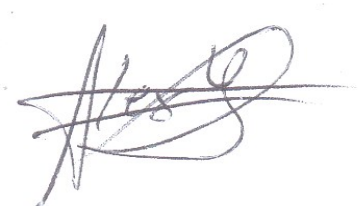
DIRECTOR: DANILO GEOVANNY CHÁVEZ GARCÍA

danilo.chavez@epn.edu.ec

DMQ, abril 2023

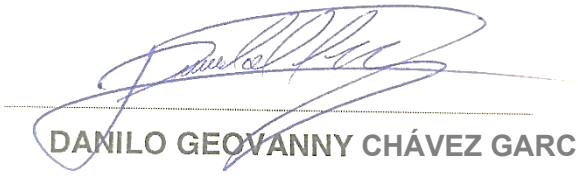
CERTIFICACIONES

Yo, Alex Javier Yugcha Navarrete declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



ALEX JAVIER YUGCHA NAVARRETE

Certifico que el presente trabajo de integración curricular fue desarrollado por Alex Javier Yugcha Navarrete, bajo mi supervisión.



DANILO GEOVANNY CHÁVEZ GARCÍA

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

NOMBRE_ESTUDIANTE
ALEX JAVIER YUGCHA NAVARRETE

NOMBRE_DIRECTOR

GEOVANNY DANILO CHÁVEZ GARCÍA

NOMBRE_COLABORADOR(ES)

VIVIANA ISABEL MOYA GONZÁLEZ
KLEBER DARIO PATIÑO CAIZA

DEDICATORIA

A mi madre: gracias por ser mi roca durante toda mi vida. Gracias por tu amor incondicional, tu apoyo y tu paciencia. Siempre has sido mi inspiración y mi motivación para seguir adelante, incluso cuando las cosas se ponían difíciles. Este logro es tuyo tanto como mío.

A mi padre: gracias por enseñarme la importancia del trabajo duro y la perseverancia. Gracias por ser mi mentor, mi consejero y mi amigo. Tu guía ha sido fundamental para mi éxito en la vida. Espero hacer que te sientas orgulloso.

A mis hermanos: gracias por ser mis compañeros en este viaje llamado vida. Gracias por sus palabras de ánimo y sus risas que siempre me ayudaron a mantener la cabeza en alto. Este logro es nuestro y espero que podamos seguir apoyándonos mutuamente en todo lo que hagamos.

A mi director de tesis: gracias por su dedicación, su paciencia y su sabiduría en guiarme a través de este proyecto. Gracias por creer en mí y por desafiarme a ser la mejor versión de mí mismo.

AGRADECIMIENTO

Quiero aprovechar esta oportunidad para agradecerles desde lo más profundo de mi corazón por el apoyo que me brindaron durante mi carrera universitaria y en la investigación de mi tesis.

Papá y mamá, Marco y Carmen, ustedes son mis héroes. Desde el primer día en que decidí seguir una carrera universitaria, estuvieron ahí para apoyarme y motivarme. Recuerdo que había momentos en los que me sentía abrumado y desanimado, pero siempre estuvieron ahí para recordarme lo lejos que había llegado y lo mucho que valía la pena seguir adelante. Gracias por su amor incondicional, su paciencia y su dedicación. Sé que fue difícil para ustedes verme luchar y esforzarme más allá del tiempo requerido para terminar la carrera, no puedo describir como me sentía, pero a pesar de todo su apoyo nunca vaciló. Todo lo que he logrado hasta ahora es gracias a ustedes. Los amo mucho, en verdad.

Dr. Danilo Chávez, quiero agradecerle por ser una gran ayuda y por ser tan paciente conmigo. Desde el momento en que comenzamos a trabajar juntos, supe que había encontrado al mejor mentor que podría haber tenido. Su experiencia, conocimiento y orientación fueron fundamentales para mi éxito en mi tesis. Siempre estuviste dispuesto a responder mis preguntas, aclarar mis dudas y a guiarme a través de las dificultades. Gracias por creer en mí y por desafiarme a ser la mejor versión de mí mismo.

A pesar de las dificultades que encontré en mi camino, nunca me sentí solo gracias al apoyo incondicional de mis padres y el gran respaldo de mi director de tesis. Este logro no habría sido posible sin ustedes. Hoy me siento muy orgulloso de haber terminado mi carrera universitaria y mi tesis, y lo hice gracias a su amor, paciencia y dedicación.

Estimados Klever Patiño y Viviana Moya, expreso mi sincero agradecimiento por el valioso apoyo y los consejos que me brindaron durante mi tesis. Gracias a su ayuda y orientación, pude superar muchos obstáculos y completar mi proyecto con éxito.

Realmente aprecio su tiempo y dedicación, y nunca olvidaré su generosidad y amabilidad.

De nuevo, les agradezco y anhelo de todo corazón retribuir todo lo que han hecho por mí y continuar haciéndolos sentir orgullosos.

INDICE DE CONTENIDOS

RESUMEN

En el siguiente documento se presenta el diseño y simulación del algoritmo de evasión de obstáculos obtenido a partir del modelo de un robot diferencial y la posterior aplicación al para un humanoide NAO con el método de fuerzas ficticias. Para esto es necesario la implementación de un control en cascada por lo cual el controlador de posición se lo realizo mediante un controlador basado en Lyapunov, mientras que para la evasión de obstáculos se realizó mediante el algoritmo de impedancias.

Para las pruebas de funcionamiento del algoritmo de evasión de obstáculos se realizó en el entorno de simulación CoppeliaSim, herramienta que ofrece condiciones cercanas a la realidad

Posteriormente se revisa sobre la librería de NAOqi, la cual es de mucha utilidad para ejecutar los comandos que dispone el software de Choregraphe usando el software Python. Este último software que a su vez servirá como elemento comunicante entre Matlab y Coppelia, usando comunicación UDP. Se muestra además el desarrollo del controlador de posición implementado con control basado en Lyapunov para un modelo de un robot uniciclo. Además, se muestra el controlador para la evasión de obstáculos con el algoritmo de fuerzas ficticias.

Para las pruebas del controlador mencionado anteriormente se simula en el entorno Coppelia, para el cual es necesario modelar los sensores del robot puesto que no están disponibles.

Finalmente se muestra el desarrollo de la interfaz de usuario realizada con Matlab Guide para la apreciación de variables seteadas en el sistema.

PALABRAS CLAVE: obstáculos, evasión, humanoide NAO, fuerzas ficticias, Lyapunov.

ABSTRACT

The following document presents the design and simulation of the obstacle avoidance algorithm obtained from the model of a differential robot and the subsequent application to NAO humanoid with the method of fictitious forces. For this it is necessary the implementation of a cascade control for which the position controller was performed by a Lyapunov-based controller, while for the obstacle avoidance was performed by the impedance algorithm.

The obstacle avoidance algorithm was tested in the CoppeliaSim simulation environment, a tool that offers conditions close to reality.

Subsequently, the NAOqi library is reviewed, which is very useful to execute the commands provided by the Choregraphe software using the Python software. This last software will serve as a communicating element between MATLAB and Coppelia, using UDP communication. The development of the position controller implemented with Lyapunov-based control for a model of a unicycle robot is also shown. In addition, the controller for obstacle avoidance with the fictitious forces algorithm is shown.

For the tests of the controller mentioned above, it is simulated in the Coppelia environment, for which it is necessary to model the robot sensors since they are not disponible.

Finally it is shown the development of the user interface made with MATLAB Guide for the appreciation of variables set in the system.

KEY WORDS: obstacles, evasion, NAO humanoid, fictitious forces, Lyapunov.

1 INTRODUCCIÓN

La robótica móvil está en constante evolución de manera que se puede encontrarlos en diferentes áreas del desarrollo de las actividades del ser humano, incluyendo desde actividades domesticas hasta operaciones de alto riego como minas y reactores. Igualmente se puede encontrar diferentes tamaños y principalmente una gran diversidad en la estructura física, llegando al punto que la robótica ofrece robots con una apariencia amigable y muy similar a un ser humano, conocidos estos como robots humanoides, razón por la cual se ha prestado mayor atención a estos robots debido a su rendimiento y también que pueden ser aplicados en plataformas humanas. Entonces para que los robots realicen estas actividades es necesario realizar un análisis de algoritmos de evasión de obstáculos y proponer la mejor solución para aplicar en estos tipos de robots. Una alternativa optada es partir de un robot más simple, un robot diferencial, en el cual se pueden aplicar modelos cinemáticos para su posterior control de posición y posterior implementación de un algoritmo de evasión de obstáculos.

Considerando lo anterior mencionado, para el desarrollo de este proyecto se ha propuesto partir de un robot diferencial, el control de posición en este implementado con un controlador Lyapunov y la búsqueda de ajuste de parámetros para la implementación en el robot humanoide NAO V6, es importante considerar aspectos como que este tipo de robots presentan un movimiento oscilante al momento de desplazarse, lo cual puede llegar a alterar su desplazamiento. Posteriormente se trabaja en un algoritmo para la evasión de obstáculos, llamado control basado en impedancia. En el cual se tiene un conocimiento del entorno que rodea al robot para planificar toda la trayectoria que debe seguir para alcanzar el objetivo. Este enfoque, sin embargo, pierden eficacia si aparece un obstáculo imprevisto en la trayectoria del robot: al no estar incluido en el mapa del entorno, no es posible garantizar que no se produzca una colisión.

Entonces el robot tiene que buscar un objetivo evitando cualquier obstáculo que aparezca repentinamente en su camino, proceso que se puede dividir en una tarea de dos pasos. El primer paso consiste en acercarse a la meta y el segundo paso consiste en cambiar el ángulo de dirección del robot para evitar el obstáculo más cercano. Después de que el robot deje atrás un obstáculo, se reanuda el primer paso, y la distancia entre el robot y su objetivo se reduce continuamente, hasta que alcanza dicho objetivo.

La prueba del presente proyecto se ha realizado en un ambiente de simulación con condiciones cercanas sin que exista riesgo de dañar su estructura física, por lo cual se

puede evitar problemas físicos como daños en el robot, lo cual puede implicar perdida de capital económico y tiempo en su reparación.

1.1. Objetivo general

Diseñar, simular un controlador de evasión de obstáculos, para su control de posición con control basado en Lyapunov mientras que para el algoritmo de evasión de obstáculos mediante un control basado por impedancia usando fuerzas ficticias.

1.2. Objetivos específicos

1. Realizar un análisis de la teoría y documentación respecto al robot humanoide NAO V6, reconocimiento del entorno de simulación CoppeliaSim, fundamentación del lenguaje de programación Python con sus respectivos algoritmos para la comunicación Matlab-Python y Python -Coppelia.
2. Diseñar un controlador basado en base a la teoría de estabilidad de Lyapunov para la evasión de obstáculos y para el controlador de impedancia basado en la generación de fuerzas artificiales que dependen de la distancia de los obstáculos.
3. Simular el desempeño del controlador desarrollado en el entorno CoppeliaSim con diferentes disposiciones de escenarios para la prueba del mismo.
4. Desarrollar una interfaz gráfica en Matlab mediante GUIDE, en la cual se observara y verificara el funcionamiento del sistema implementado.
5. Comprobar el funcionamiento del controlador y del sistema elaborado mediante la evaluación de índices de desempeño como el ISE e ISU.

1.3. Alcance

Revisión e inmersión bibliográfica de las siguientes temáticas: características generales del robot humanoide nao v6 y revisión especifica respecto a sonares para replicar de la forma mas fiel en el software Coppelia Sim, revisión de librerías de los API para definir la interacción entre los software comunicantes, además de fundamentación en la programación Python, revisión del funcionamiento básico del software CoppeliaSim, revisión de la teoría Lyapunov para el controlador de evasión de obstáculos.

- Diseño de esquema de control basado en Lyapunov para el control de posición en el cual se considera al robot NAO inicialmente como un robot diferencial.

- Implementación de un algoritmo en Python parte medular de la comunicación UDP Matlab-Python y Python-Coppelia, en la cual se verificara la comunicación bidireccional de envío y recepción de datos..
- Análisis e implementación del controlador el cual será desarrollado en Matlab en el cual el resultado de las señales de control hacia el robot NAO serán de características de programación a alto nivel para el robot NAO.
- En el software Coppelia Sim se realizara la simulación y se observara los resultados del desempeño del controlador, al poner a prueba en varias disposiciones de los obstáculos.

1.4. Marco teórico

1.4.1. Revisión teórica del robot humanoide nao v6

NAO es el primer robot creado por SoftBank Robotics mostrado en la Figura 1.1. Famoso en todo el mundo, NAO es una tremenda herramienta de programación y se ha convertido especialmente en un estándar en la educación y la investigación. [1]

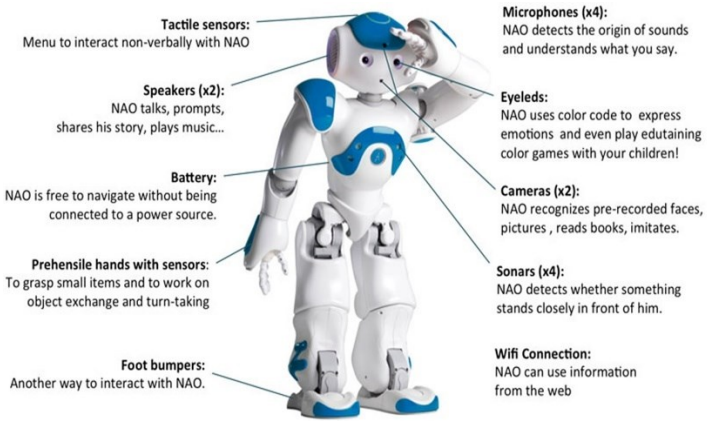


Figura 1.1 Componentes principales del robot NAO V6 [2]

1.4.1.1. Características generales del robot NAO

Entre sus características generales principales están

- 25 grados de libertad que le permiten moverse y adaptarse a su entorno.
- 7 sensores táctiles situados en la cabeza, las manos y los pies, sonares y una unidad inercial para percibir su entorno y situarse en el espacio.

- 4 micrófonos direccionales y altavoces para interactuar con los humanos.
- Reconocimiento de voz y diálogo disponible en 20 idiomas (Inglés, francés, español, alemán, italiano, árabe, holandés, portugués, checo, finlandés, ruso, sueco, turco...)
- Dos cámaras 2D para reconocer formas, objetos e incluso personas.
- Plataforma abierta y totalmente programable.

Las principales especificaciones técnicas del robot se detallan a continuación en la Tabla 1.1 [2].

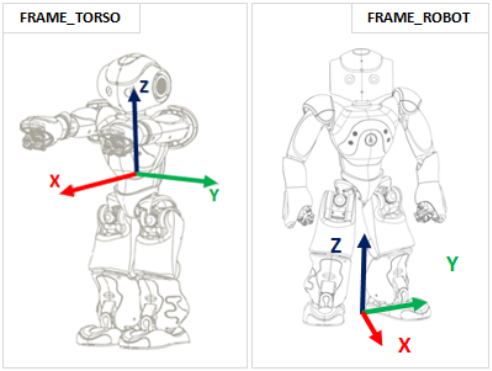
Tabla 1.1 Características del robot NAO

DIMENSIONES		
Alto (mm)	Grosor (mm)	Ancho (mm)
574	311	275
CPU		
Procesador	Atom E3845	
CPU	Quad core	
Velocidad de reloj	1.91 GHz	
RAM	4 GB DDR3	
BATERIA		
Tipo	Lithium-Ion	
Voltaje nominal/ capacidad.	21.6V / 2.9Ah	
Máximo voltaje	25.2V	
Recomendada corriente de carga.	1.8 ~ 2.1A	
Max carga/descarga corriente	2.1A / 2.0A	
Energy	62.5Wh	
Duración de la carga	90 min	
CONECTIVIDAD		
Ethernet	WIFI	USB
Uso principal	Uso principal	Uso principal
Configuración de la conexión WiFi.	Comunicación entre Choregraphe y el robot.	para actualizar el sistema del robot, también puede usar el puerto USB para conectar dispositivos externos como: Sensor Kinect o Asus 3D, o dispositivo Arduino.
Especificación	Especificación	Especificación

1×RJ45 - 10/100/1000 base T.	IEEE 802.11 a/b/g/n Security: 64/128 bit: WEP, WPA/WPA2.	

El robot NAO V6 tiene tres marcos de referencia, como se muestra en la Figura 1.3

- **Marco de referencia del torso:** cuyo marco se mueve con el robot mientras camina y cambia de orientación cuando se inclina. Este marco se podría considerar muy útiles para tareas muy locales.
- **Marco de referencia del robot:** este marco se define como el promedio de las posiciones de los pies proyectadas alrededor de un eje z vertical. Razón por la cual se considera muy útil este marco es debido a que el eje x siempre se ubica hacia adelante.
- **Marco de referencia Mundo:** con este marco se fija un origen y nunca se modifica. Se deja atrás cuando el robot camina y será diferente en la rotación z después de que el robot haya girado. Este espacio es útil para cálculos que requieren un marco de referencia absoluto y externo.



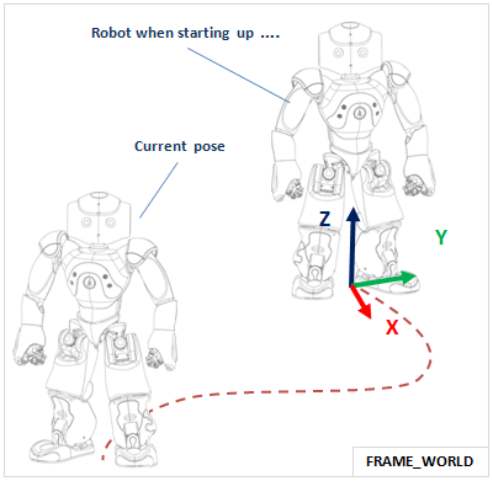


Figura 1.2 Marcos de referencia del robot NAO [2]

1.4.2. NAOqi Framework (Cuadro de trabajo de NAOqi)

El software principal que se ejecuta en el robot nao V6 y lo controlar, es llamado NAOqi, el cual responde a las tareas comunes en la robótica, incluyendo: recursos, paralelismo, sincronización y eventos.

Este software permite una comunicación homogénea entre los diferentes módulos, en el robot, ya sean movimiento, audio, vídeo, una programación homogénea y un intercambio de información homogéneo.

A continuación, se detallan algunas características de este.

- **Multiplataforma,** lo cual permite ejecutar en Windows, Linux o Mac.
- **Multilenguaje,** posee una API idéntica tanto para C++ como para Python. Para lo casos mencionados anteriormente, los métodos de programación son exactamente los mismos, toda la API que existe puede ser llamada indistintamente desde cualquier lenguaje soportado:

  Si se crea un nuevo módulo C++, las funciones de la API de C++ pueden ser llamadas desde cualquier lugar, Así también si se definen las funciones de la API del módulo de Python pueden ser llamadas desde cualquier lugar.

  De acuerdo a [3] , la mayoría de las veces se desarrollara los comportamientos en Python y los servicios en C++.

  En la Figura 1.3. se puede observar una representación detallada.

- **Proporciona introspección,** lo que significa que el framework sabe qué funciones están disponibles en los diferentes módulos y dónde. La introspección es la base de la API del robot, las capacidades, la supervisión y la acción sobre las funciones supervisadas. El robot conoce todas las funciones de la API disponibles.

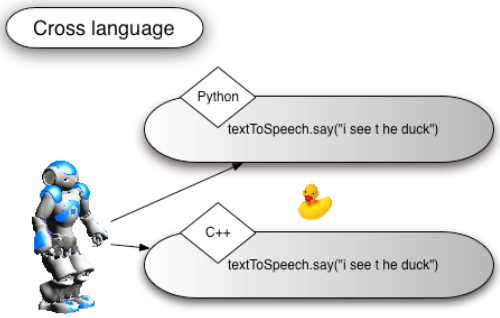


Figura 1.3 Soporte multilenguaje robot NAO V6

1.4.3. Python

Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Sus estructuras de datos de alto nivel, combinadas con la tipificación y la vinculación dinámicas, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripting o a manera a de unión para conectar componentes existentes. La sintaxis de Python, sencilla y fácil de aprender, hace hincapié en la legibilidad y, por tanto, reduce el coste de mantenimiento de los programas. Python admite módulos y paquetes, lo que fomenta el modularidad del programa y la reutilización del código. El intérprete de Python y la extensa biblioteca estándar están disponibles en forma de código fuente o binario sin coste alguno para las principales plataformas, y pueden distribuirse libremente. [4]

1.4.3.1. Python SDK

Al escribir código para NAO, la API de NAOqi está disponible actualmente en al menos 8 idiomas. Se puede mencionar que algunas diferencias menores específicas de cada idioma, la API es en su mayor parte la misma en todos los idiomas, lo que le permite llevar los conocimientos de un idioma a otro.

Se puede escribir únicamente módulos NAOqi en C++ y Python, pero se tiene acceso a la API completa del cliente en todos los idiomas. Sólo C++ y Python son soportados en el robot, otros lenguajes sólo son soportados en el ordenador para acceder remotamente a NAO.

La API de Python para NAO permite utilizar toda la API de C++ desde una máquina remota, o crear módulos de Python que puedan ejecutarse de forma remota o en el robot. El principal motivo del uso de este es porque el uso de Python es una de las formas más fáciles de programar con NAO. [5]

Broker: este elemento se encarga de gestionar un servicio de búsqueda de módulos .

Proxy: al definirlo como un objeto el cual adquiere todas las características del modulo

1.4.4. CoppeliaSim (Vrep)

El simulador de robótica CoppeliaSim posee con entorno de desarrollo integrado, se basa en una arquitectura de control distribuido, en la que cada objeto/modelo puede ser controlado individualmente a través de un script integrado, un plugin, un nodo ROS, un cliente API remoto o una solución personalizada. Esto hace que CoppeliaSim sea muy versátil e ideal para aplicaciones multi-robot. Los controladores pueden escribirse en C/C++, Python, Java, Lua, Matlab u Octave.

CoppeliaSim se utiliza para el desarrollo rápido de algoritmos, simulaciones de automatización de fábricas, creación rápida de prototipos y verificación, educación relacionada con la robótica, monitorización remota, doble comprobación de seguridad, como gemelo digital, y mucho más. [6]

A continuación, se citan A continuación, algunas de las aplicaciones de CoppeliaSim:

- Simulación de sistemas de automatización de fábricas
- Supervisión remota
- Control de hardware
- Creación rápida de prototipos y verificación
- Supervisión de la seguridad
- Desarrollo rápido de algoritmos
- Educación relacionada con la robótica

- Presentación de productos

CoppeliaSim puede utilizarse como una aplicación independiente o también se puede incluirse fácilmente en una aplicación cliente principal: su pequeña huella junto a su elaborada API hace de CoppeliaSim un candidato ideal para incluirla en aplicaciones de nivel superior. El intérprete de scripts Lua o Python hace de CoppeliaSim una aplicación altamente versátil, dejando la posibilidad al usuario de combinar las funcionalidades de tanto de bajo como de alto nivel para obtener nuevas funcionalidades de alto nivel. [6]

CoppeliaSim extiende la función API de Lua y Python y añade comandos específicos de CoppeliaSim que pueden ser reconocidos por sus prefijos sim (por ejemplo, sim.getObjectPosition). Para obtener una lista de todas las funciones API específicas de CoppeliaSim, [7]

1.4.5. Matlab/Simulink

MATLAB es un lenguaje de computación numérica y un ambiente de desarrollo integrado (IDE) para la resolución de problemas en ciencia, ingeniería, economía, etc. Se lo utiliza muchísimo en la visualización de datos, el análisis estadístico, la modelación matemática y el desarrollo de algoritmos. MATLAB también tiene una gran cantidad de bibliotecas y herramientas para el procesamiento de señales, la inteligencia artificial y el aprendizaje automático.

Simulink es un entorno de modelado y simulación gráfico desarrollado por MathWorks. Se lo utiliza para simular sistemas dinámicos y para diseñar sistemas de control. Simulink tiene una interfaz gráfica de bloques, que permite al usuario crear modelos mediante la conexión de bloques predefinidos para representar componentes matemáticos y de sistemas establecidos. Los bloques pueden representar funciones matemáticas, operaciones de señal, sistemas dinámicos y algoritmos de control. Con Simulink, los usuarios pueden simular y analizar el comportamiento temporal de sus sistemas, generar código automáticamente para su implementación en sistemas embebidos y realizar análisis estático y dinámico. Los bloques vitales que permiten la recepción y envío de datos o información desde Python a Matlab y viceversa son los siguientes mostrados.

UDP Send y Receive: cómo se puede apreciar en la Figura *1.4* este tipo de bloques son de gran utilidad ya que permiten enviar y recibir información de una aplicación a otra ya sea de Simulink a Python o viceversa mediante la configuración de una dirección IP remota y un puerto remoto

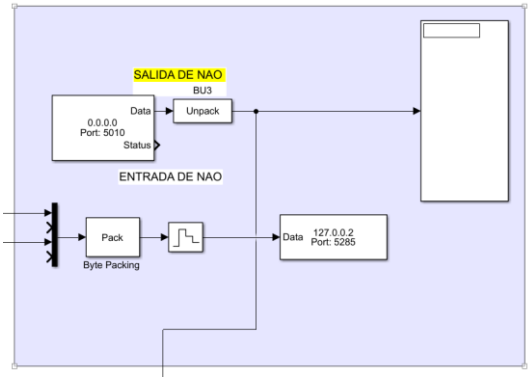


Figura 1.4 Bloque usado en Matlab para envío y recepción de datos mediante UDP

1.4.6. REVISION DE ROBOTICA MOVIL

Ahora se realizará un análisis sobre conceptos de robótica móvil, La robótica móvil es un sector industrial en rápido desarrollo que pretende encontrar soluciones, combinando ingeniería, informática y muchas otras disciplinas. La complejidad de los robots móviles sólo puede abordarse mediante la cooperación de todas estas disciplinas.

Las características de la robótica móvil: [8]

- Percepción del entorno gracias a los sensores
- Capacidad de adaptación a los cambios del entorno
- Navegación, planificación y acciones independientes
- Software/programación orientada a las tareas

Ahora se partirá del hecho de considerar a un robot móvil como una partícula en movimiento, como en la Figura 1.5 para poder obtener sus ecuaciones cinemáticas. Entonces a partir de un robot de tracción diferencial, con dos ruedas motrices y una rueda loca. Entonces se obtiene las variables de estado del robot diferencial las cuales son serán: posición en x, posición en y y su ángulo de orientación φ, como se ve en las ecuaciones (1.1),(1.2) y (1.3)con lo cual se obtiene el modelo básico del modelo cinemático cartesiano del robot móvil que se representa con variables de entrada son u (velocidad lineal) y w (velocidad angular)

$$\dot{x} = u\cos\ \theta \quad (1.1)$$
$$\dot{y} = u\,\mathrm{sen}\ \theta \quad (1.2)$$
$$\dot{\theta} = \omega \quad (1.3)$$

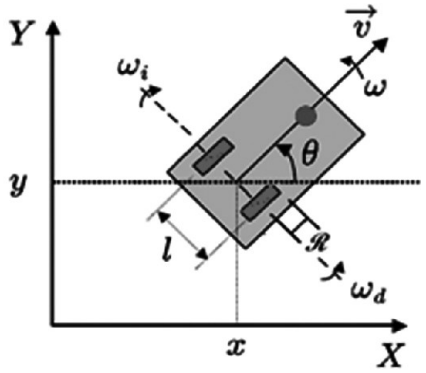


Figura 1.5 Orientación del robot en coordenadas cartesiana

1.4.7. Control basado en Lyapunov.

Para garantizar un comportamiento estable del robot móvil durante su movimiento, se utiliza un sistema de control diseñado en base a la teoría de estabilidad de Lyapunov. El controlador de movimiento consiste en un controlador de posición no lineal y un controlador de impedancia. El controlador de impedancia se basa en la generación artificial de una fuerza ficticia en función de la distancia a los obstáculos. Se usa esta fuerza ficticia para mantener al robot alejado de los obstáculos como módulo para la evasión de colisiones.

A continuación, se desarrollará de manera general sobre el análisis de estabilidad de un sistema, en el cual la teoría de Lyapunov será de mucha utilidad en este sistema descrito por ecuaciones en espacio de estado.

Para realizar un análisis de estabilidad mediante Lyapunov, existen el método directo y el indirecto. El método indirecto necesita de procedimientos en los cuales es necesario las formas explicitas de las soluciones de ecuaciones diferenciales o ecuaciones en diferencias. Mientras que el método directo no requiere de estas soluciones, lo cual se traduce que resulta mas útil al aplicarlo en la práctica.

Ahora bien enfocado al método directo de Lyapunov, la cual se puede definir como una extensión matemática de una observación física fundamental , lo cual menciona: si la energía total de un sistema mecánico o eléctrico es disipada en forma continua, entonces el sistema lineal o no lineal debe quedar eventualmente en un punto de equilibrio.

Entonces, mediante una función de Lyapunov $V(t,x)$ y el análisis de su variación se podrá determinar la estabilidad de un sistema.[12]

Ahora bien se realizara un listado de condiciones para establecer las condiciones en la cual un sistema tiene un estado de equilibrio en el origen y también este estado será uniforme y asintóticamente estable.

1. $V\ (t,x)$ es continua y tiene derivadas continuas.
2. $V\ (t,x)$ es positiva
3. $\dot{V}\ (t,x)\ es\ negativa$
4. $V\ (t,0) = 0$

Considerando las condiciones anteriormente explicadas para que cierto sistema este en equilibrio asintóticamente estable, se siguen los siguientes pasos para el diseño del controlador.

- Elegir una función Lyapunov $V\ (t,x)$ la cual se llama candidata y tiene que cumplir con las condiciones anteriormente detalladas.
- Derivar la función candidata obteniéndose $\dot{V}\ (t,x)$
- Después de esto se selecciona una ley de control de retroalimentación $u = u(x)$
- Con esto se asegura que$d\frac{V(x)}{dt} < 0$ para $x \neq 0$
- Por lo general, la función $u(x)$es una función no lineal con valores de ganancia y parámetros configurables de manera de establecer $dV\backslash dt < 0$ y así poder asegurar un sistema asintóticamente estable

1.4.8. Evasión de obstáculos control basado en impedancia

El control basado en la impedancia que da lugar a una rápida reacción ante la presencia de obstáculos, lo que lo convierte en un enfoque muy interesante. Utiliza el concepto de impedancia generalizada o extendida para caracterizar la relación entre un robot móvil que se mueve hacia un obstáculo y una fuerza de repulsión ficticia (Hogan, 1985; Secchi, 2001) proporcional a la distancia entre el robot y el obstáculo. Así, el objetivo de evitar el contacto robot-obstáculo se cumple si la fuerza de repulsión aumenta cuando el robot se acerca al obstáculo.

La Figura *1.6* muestra una situación en la que un obstáculo es detectado por los sensores del robot. Entonces, se genera una fuerza de repulsión F que provoca un desplazamiento temporal del punto de destino Xd, como se muestra. Como resultado de la búsqueda del nuevo objetivo se impone un cambio en el ángulo de dirección del robot, lo que le permite desviarse del obstáculo. Los componentes de la fuerza F (F_t,que esta alineada con el eje de movimiento del robot, y , F_r,que es perpendicular a éste) [13]

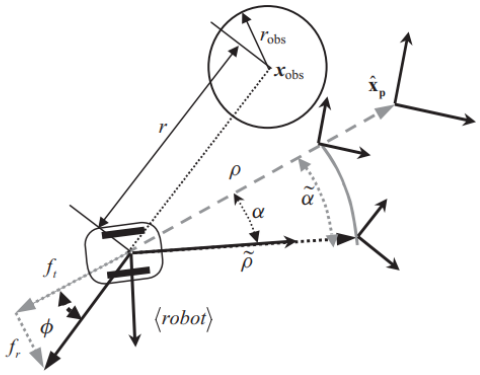


Figura 1.6 Posición y orientación del robot respecto al objetivo. [13]

La magnitud de la fuerza ficticia repulsiva generada por un obstáculo puede definirse como

$$F(t) = k_3 - k_4(t) \tag{1.4}$$

Donde: k3, k4 son constantes positivas tales que $k_3 - k_4 r_{\max} = 0\ y\ k_3 - k_4 r_{min} = 1$ La distancia del robot al obstáculo se denota por r, que satisface $r_{\min} \leq r \leq r_{max}$ siendo $r_{\min}\ y\ r_{max}$ las distancias mínima y máxima entre el robot y el obstáculo, respectivamente. Éstas definen la zona en la que actúa la fuerza actúa. La magnitud de f(t) se escala al intervalo [0,1] para que su valor pueda ser amplificado por el controlador de impedancia.

Por otro lado, el ángulo de la fuerza ficticia φ se define como la orientación del obstáculo respecto al robot

La fuerza ficticia tangencial y la fuerza ficticia normal se calculan como $f_t = f cos(\phi) y\ f_r = f sen(\phi)$ respectivamente. Cuando hay múltiples obstáculos, se calcula una fuerza total sumando todos los vectores de fuerza

El modelo de impedancia se define como.

$$[\rho_e \quad \alpha_e] = Z\ f_e \tag{1.5}$$

$$Z = \begin{bmatrix} K_\rho & 0 \\ 0 & K_\alpha \end{bmatrix} \quad (1.6)$$

Donde $f_e = [f_t \quad f_r]^T$ Y $K_\rho, K_\alpha > 0$ las cuales describen los parámetros de elasticidad, $f_t$es la fuera ficticia en la dirección del movimiento del robot y $f_r$es la fuerza en la dirección normal al movimiento del robot

Cuando el robot NAO navega e interactúa con el ambiente se define su espacio de estados como

$$[\tilde{p} \quad \tilde{\alpha}]^T = [\rho \quad \alpha]^T - [\rho_e \quad \alpha_e]^T \quad (1.7)$$

2 METODOLOGÍA

En el siguiente apartado, se desarrolla la metodología en el presente trabajo para describir explícitamente cual ha sido el procedimiento durante la investigación.

2.1. Desarrollo del entorno para simular el robot NAO (Coppelia sim)

Como actividades iniciales el proceso a seguir es implementar el robot NAO el cual se lo encuentra en el apartado de robots móviles como se muestra en la figura 2.1 , además de esto para fines de simulación debido a que el piso (Floor) por defecto resulta demasiado pequeño se ha decidido colocar un nuevo piso ajustable en medidas, con lo cual ahora el robot puede desplazarse por una superficie de 10m x 10m.

Para completar el modelado del robot es necesario incluir los sensores de distancia, los cuales no están incluidos en el cuerpo del robot mostrado por Coppelia, entonces se añaden manualmente y el próximo apartado se especificara la configuración del ángulo de visión y como anclar estos señores al cuerpo del robot.

Finalmente se incluyen los bloques con característica Detectable ON, lo que servirá para la simulación de obstáculos los cuales tendrá que evitar el robot como se muestra en la Figura 2.1.

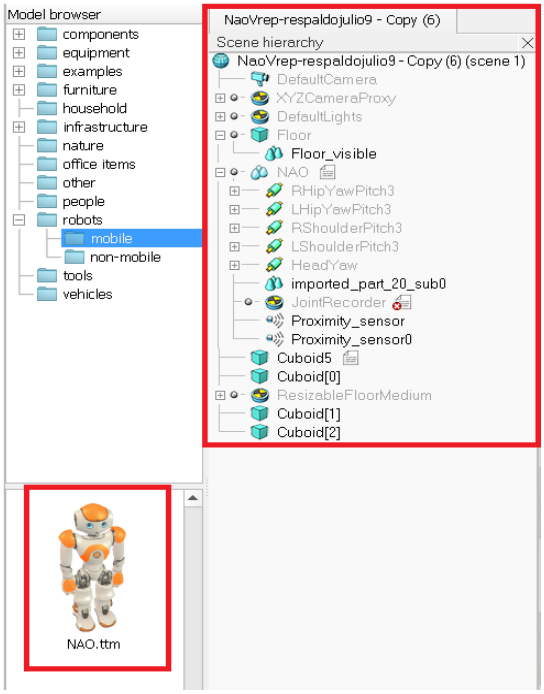


Figura 2.1 Implementación de la escena y paso elementos iniciales

Para el desarrollo de presente controlador se podrá visualizar e interpretar en el software de simulación Coppelia, por lo cual se establece el origen del robot en la posición marcada en la Figura *2.2* en la cual se obtiene además que cuada división mostrada en la rejilla corresponde a 0.5 m con los cuales se hará cálculos en el controlador.

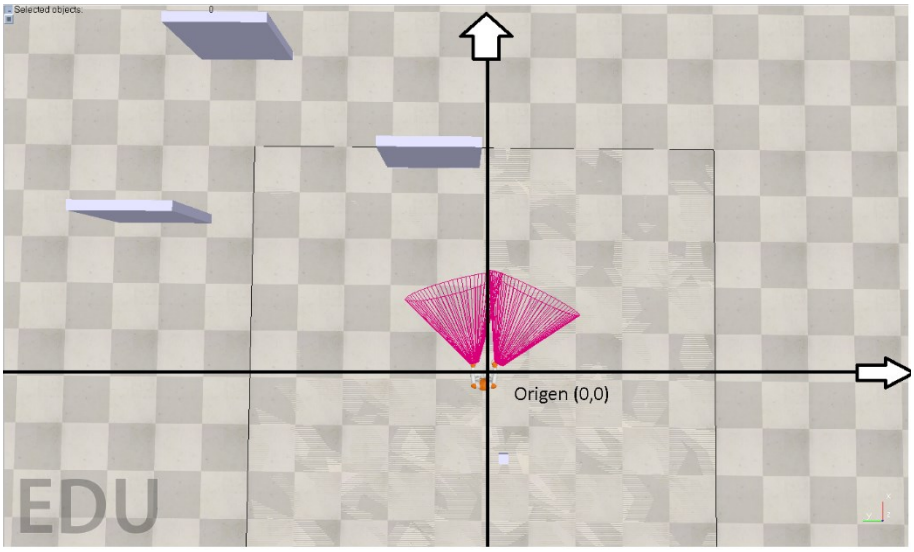


Figura 2.2 Entorno de simulación, consideración del punto inicial de trayectoria del robot.

2.2. Modelado de la escena y el robot NAO en Coppelia.

Ahora bien, una vez configurado la escena de simulación, con los componentes necesarios robot, obstáculos. Es importante mencionar que el software Coppelia no permite agregar al robot NAOv6 con sus sensores sonares incorporados por lo cual se los agrega manualmente a la base del robot.

Entonces para que la simulación en cuanto a alcance de los señores, ubicación y orientación de los sensores, se ha seguido fielmente las especificaciones de fabricante para agregar los sensores sonares como se observa en Figura *2.3* [14]

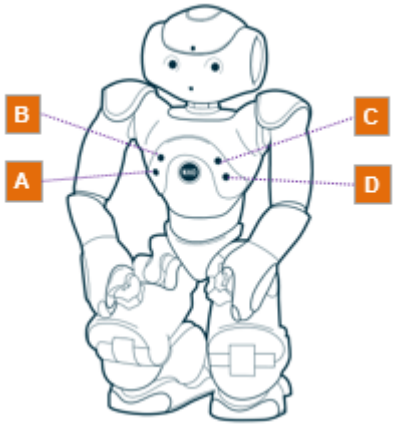


Figura 2.3 Sensores ultrasónicos del robot NAO [14]

En la Tabla 2.1 se describe la ubicación de cada uno de los sensores en la parte pectoral del robot NAO, los cuales se modelo de la manera más similar para obtener resultados que sean lo más fieles.

Tabla 2.1 Características y dimensiones de sensores para su modelado

Parte	**Nombre del sonar**		**Parte**	**Nombre del sonar**	
B	Sonar/Derecho	Transmisor	C	Sonar/Izquierdo	Transmisor
A		Receptor	D		Receptor

Además, algunas características que se pueden mencionar de los sensores sonares del robot NAO son:

- NAO está equipado con dos sensores ultrasónicos (o sonares) que le permiten estimar la distancia a los obstáculos de su entorno.

- Rango de detección: 0,20 m - 0,80 m Por debajo de 20 cm no hay información de distancia, el robot sólo sabe que hay un objeto. Por encima de 80 cm el valor devuelto es una estimación.

- Cono efectivo: 60°.

Considerando las características anteriores se configura en el cuadro de diálogo de configuración de cada uno de los sensores como se muestra en la Figura *2.4*

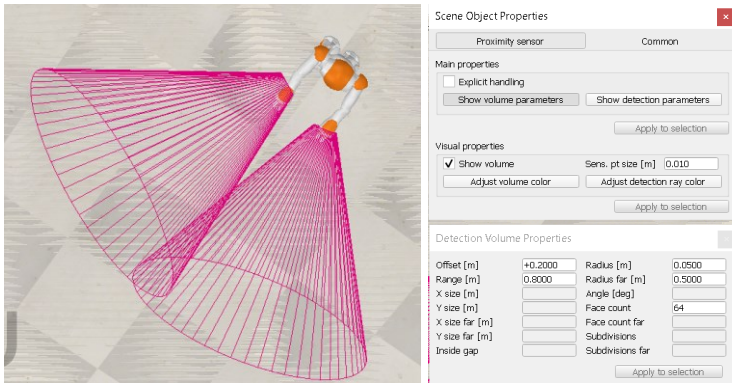


Figura 2.4 Configuración de parámetros de los sensores sonares agregados.

2.3. Recepción y transmisión de datos entre Matlab y Coppelia mediante Python.

Primeramente, cabe mencionar que para una comunicación exitosa entre Python y Coppelia Sim, se iniciara un API Remoto que se ejecutara dentro del entorno de simulación. Para lo cual se necesita agregar la línea de código (simRemoteApi.start(199999)), línea de código que se la puede agregar en algún objeto estático dentro del entorno de simulación. Como se observa en la figura se agregado en un objeto tipo Cuboid.

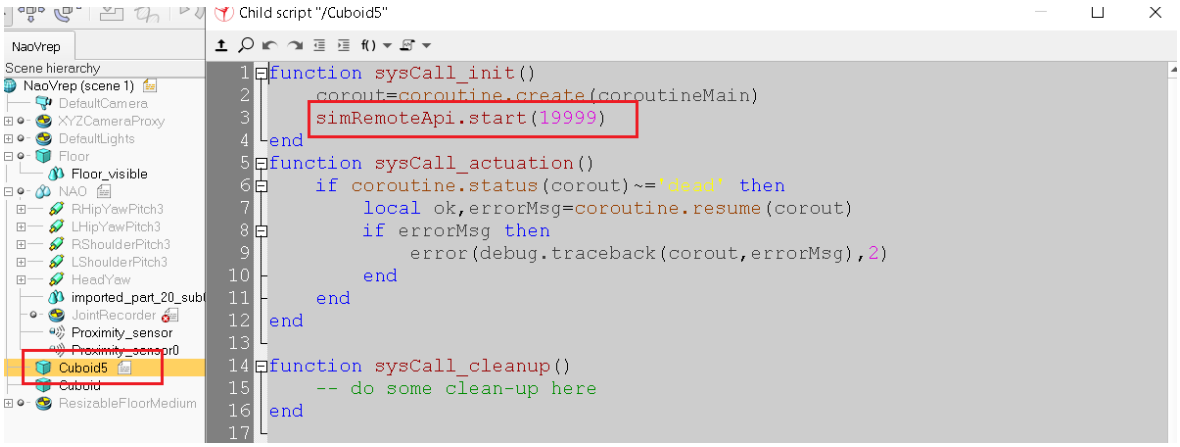


Figura 2.5 Inicialización API Remoto en Coppelia Sim

En la Figura 2.6 se puede observar los comandos de la librería NAOQi que hace posible la creación de un proxy mediante la dirección IP que se toma del robot virtual creado por NAO Choregraphe, con los cual se almacenas los valores de movimiento y posturas del robot NAO.

```
print('Conectado con Modulo Naoqi')
puerto=58067
movProxy = ALProxy("ALMotion", NaoIP, puerto)
postureProxy = ALProxy("ALRobotPosture", NaoIP, puerto)
```

Figura 2.6 Uso del proxy para obtener movimiento y posición del robot NAO

Mientras que para una correcta simulación y obtención del puerto del robot de Choreogrape, se puede encontrar esta dirección en Preferencias>Editar preferencias>Robot virtual, y en la parte inferior en letras de color verde se puede observar RunningPort, el cual es necesario actualizar en cada momento de realizar una nueva sesión en Choregraphe, esta dirección se la debe establecer en el programa en Python.

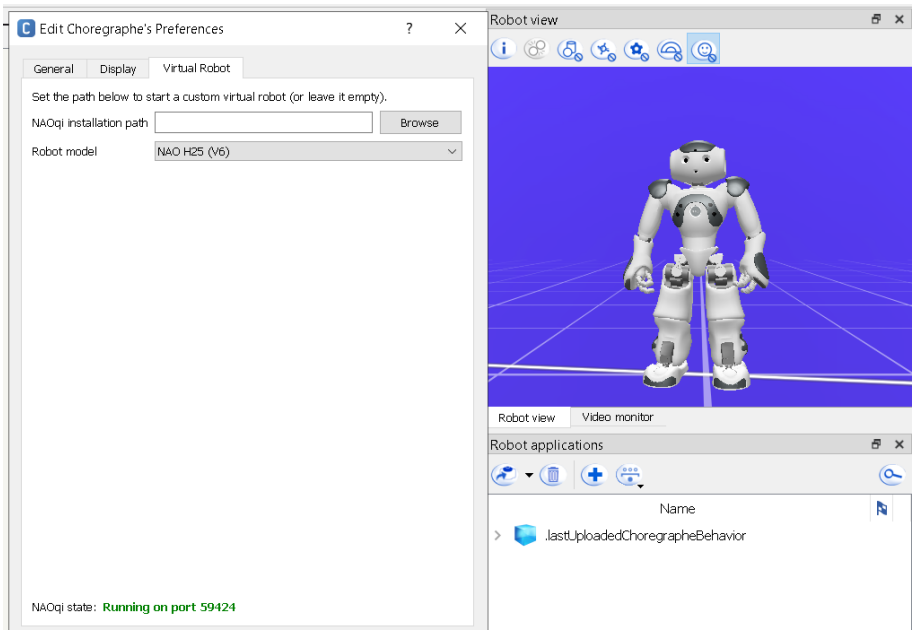


Figura 2.7 Verificación del puerto del robot virtual en Choregraphe.

Ahora se mostrarán los comandos importantes que se han usado en la programación de Python para lograr el envío y recepción de datos entre MATLAB y Simulink. Para lo cual se tiene un socket de recepción el cual tiene los valores de Matlab en forma de vector los cuales se procede a desempaquetar y usarlos para su fin especifico en Coppelia.

```
Recepcion, addr = sock.recvfrom(1024)
Recibir=unpack('>dddd', Recepcion)
print "Velocidad en x:", Recibir[0]
print "Velocidad en y:", Recibir[1]
print "Velocidad Angular :", Recibir[2]
```

Figura 2.8 Instrucciones para la recepción desde Matlab y desempaquetado de datos.

Mientras que para el envío de los datos adquiridos desde Coppelia, y sus valores importantes como posiciones distancias de cada uno de los sensores y el estado del sensor se empaqueta en un vector para enviar a Matlab

```
DATOS_PARA_MATLAB=[X,Y,ALPHA,a1,b1,c1,a2,b2,c2,detectionState,detectionStateB]
mensajebody=pack('>ddddddddddd',*DATOS_PARA_MATLAB)
sock.sendto (mensajebody, ("127.0.0.2",5010))
sock.shutdown(0)
sock.close
```

Figura 2.9 Instrucciones para envío hacia Matlab y desempaquetado de datos.

Para el control del desplazamiento del robot entre los puntos A y B. Se partirá del hecho de considerar al robot Nao como una caja negra, lo que quiere decir que su cinemática directa, cinemática inversa y variables en la dinámica del robot dinámica del robot ya están consideradas dentro de las diferentes librerías de NAOqi.

Figura 2.10 Robot NAO representado como una caja negra. [10]

A partir de los expuesto anteriormente, entonces para controlar el desplazamiento del robot Nao, se va a monitorear su orientación y posición cada tiempo de muestreo y, de existir algún error, es cuando se ejecuta las correspondientes acciones de control para velocidad lineal, y velocidad angular. Al revisar las librería NAOqi se accederá al método "move (x, y, w)", del módulo "ALMotion", donde x es la velocidad lineal del robot en el eje x, con respecto al eje de referencia del torso del robot, y es la velocidad lineal del robot en

el eje y con respecto al eje de referencia del torso del robot y w es la velocidad angular del robot en el eje z con respecto al eje de referencia del torso del robot. [10]

En la figura 2.11 se puede observar el procedimiento seguido para la comunicación entre Coppelia y Matlab, incluyendo proceso desde la declaración de los handler con el uso del SDK, para luego empaquetar o desempaquetar los datos de acuerdo las necesidades de envió o recepción respectivamente.

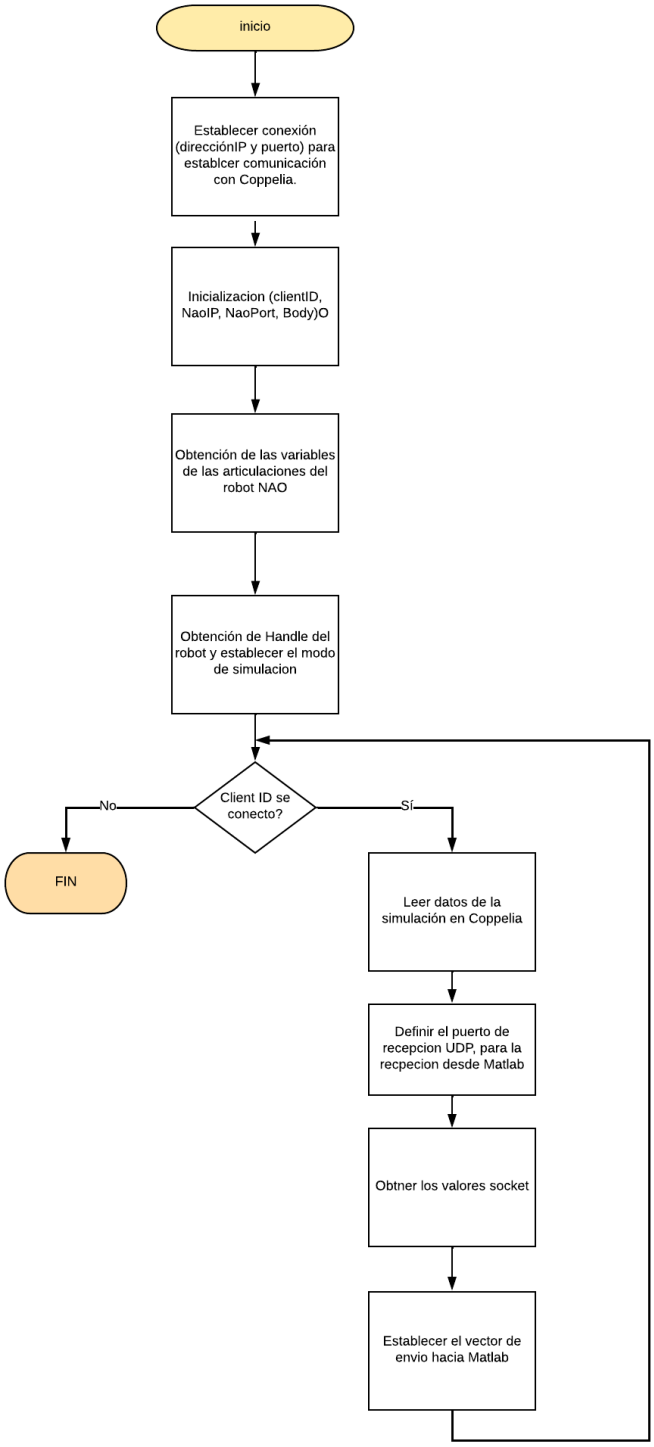


Figura 2.11 Algoritmo seguido en Python como elemento comunicante de variables.

2.4. Desarrollo del sistema de control

2.4.1. Desarrollo del sistema de control de posición basado en Lyapunov

A partir del modelo cinemático, expresado en coordenadas polares y el modelo en coordenadas cartesianas se puede obtener mediante relaciones matemáticas el siguiente conjunto de ecuaciones mostrado en las ecuaciones [9]

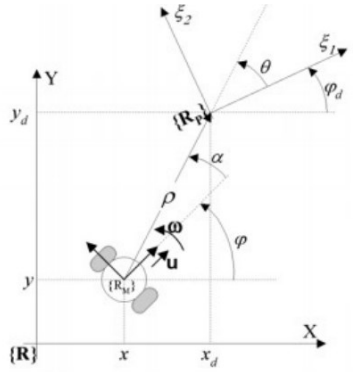


Figura 2.12 posición y orientación del robot móvil en coordenadas polares y coordenadas cartesianas[9]

$$\rho = \sqrt{(x_d - x)^2 + (y_d - y)^2} \tag{2.1}$$

$$\theta = \arctan\left(\frac{y_d - y}{x_d - x}\right) - \varphi_d \tag{2.2}$$

$$\alpha = \arctan\left(\frac{y_d - y}{x_d - x}\right) - \varphi \tag{2.3}$$

Al representar la posición del robot en coordenadas polares, en las ecuaciones (2.4) (2.5) y (2.6) y considerando el vector error e con orientación θ, respecto al eje X de referencia <g> y al definir $\alpha = \theta - \varphi$el angulo medido entre el eje principal del robot y el vector de distancia e, entonces las ecuaciones se puede reescribir como

$$\dot{\rho} = -u * cos\alpha \tag{2.4}$$

$$\dot{\alpha} = -\omega + u * \frac{sen(\alpha)}{\rho} \tag{2.5}$$

$$\dot{\theta} = u\frac{sen\alpha}{\rho} \tag{2.6}$$

En la siguiente sección revisa las variables que se van a controlar para desplazar al robot humanoide NAO y además se mostrará un desarrollo teórico de las estrategias de control a seguir. [11]

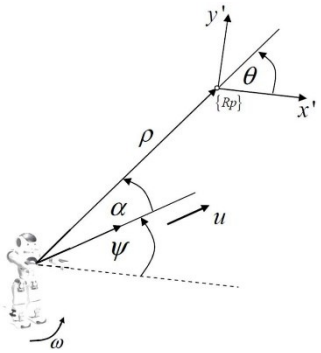


Figura 2.13 Posición y orientación del robot humanoide respecto a punto destino

El objetivo del control es hacer que ρ, α sean asintóticamente cero. Para logra esta condición, se puede empezar considerando la siguiente candidata Lyapunov. [12]

$$V(\rho, \alpha) = \frac{1}{2}\rho^2 + \frac{1}{2}\alpha^2, \tag{2.7}$$

Al derivar esta ecuación se obtiene:

$$\dot{V} = \rho\dot{\rho} + \alpha\dot{\alpha} \tag{2.8}$$

Considerando la cinemática del robot el valor $\dot{V}$ se obtiene

$$\dot{V} = \rho(-u\cos(\alpha)) + \alpha\left(-\omega + u\frac{\sin(\alpha)}{\rho}\right) \tag{2.9}$$

En las cuales se puede identificar $\dot{V} = \dot{V}1 + \dot{V}2$ donde V1 puede no ser positivo cuando la velocidad lineal u tenga la siguiente forma

$$u = K_u \tanh(\rho)\cos(\alpha) \qquad con\ \ K_u > 0 \tag{2.10}$$

Cuando el coeficiente $K_u = |U_{\max}|$ y es negativa definida si las variables de control u, ω

Son definidas de la siguiente forma

$$u = u_{max}\tanh\rho\cos\alpha, \tag{2.11}$$

$$\omega = k_\omega\alpha + u_{max}\frac{\tanh\rho}{\rho}\sin\alpha\cos\alpha, k_\omega > 0, \tag{2.12}$$

2.4.2. Desarrollo del sistema de control de evasión de obstáculos basado en fuerzas ficticias.

En la figura 2.14, se muestra el comportamiento del robot ante la presencia del obstáculo. En el cual se tiene la distancia respecto al obstáculo y el ángulo de rotación en función a la fuerza a aplicarse.

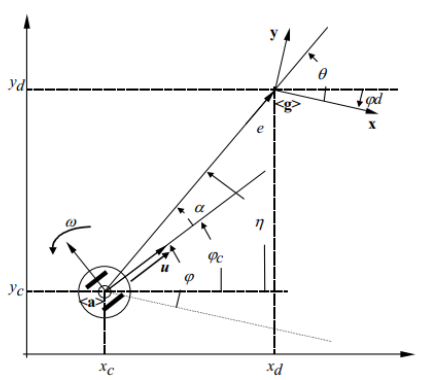


Figura 2.14 Descripción gráfica robot ante la presencia de obstáculo[12]

Consideremos el modelo cinemático del robot móvil dado como (2.4),(2.5)(2.6). Las principales características del problema de control son

1. El objetivo es alcanzar <g>. El problema de alcanzar el marco objetivo puede formularse de dos maneras diferentes: la primera en términos de una trayectoria de movimiento deseada y la segunda es en términos de la posición objetivo (en esta segunda situación, se puede considerar una orientación final).

2. La relación dinámica entre el error de posición y la fuerza de interacción F(t) que actúa sobre el robot móvil. En este trabajo, F(t) es una fuerza ficticia generada a partir de la información de distancia procedente de los sensores exteroceptivos (sensores ultrasónicos).

Entonces la tarea del robot se puede describir en dos importantes pasos ,el primer paso consiste en acercarse a la meta (siempre que no haya ningún obstáculo en las proximidades del robot), y el segundo paso consiste en cambiar el ángulo de dirección del robot para evitar el obstáculo más cercano (cuando se detectan obstáculos en las proximidades del robot).

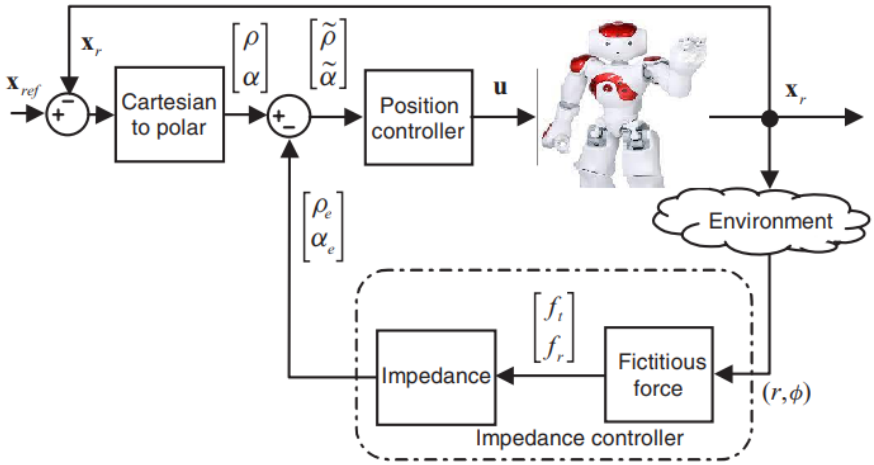


Figura 2.15 Esquema general del controlador de evasión de obstáculos [15]

Después de que el robot deje atrás un obstáculo, se reanuda el primer paso, y la distancia entre el robot y su objetivo se reduce continuamente, hasta que alcanza dicho objetivo (después de haber evitado todos los obstáculos en su camino).

Entonces acorde al razonamiento anterior, se presenta aquí para guiar al robot en el cumplimiento de su tarea es un sistema de control compuesto por dos bucles anidados. El interno se encarga de reducir la distancia robot-objetivo en ausencia de obstáculos, mientras que el exterior se encarga de desviar al robot del obstáculo más cercano a él.

Por lo cual, el problema de control de movimiento corresponde al diseño de un controlador que conduzca al robot móvil al punto de coordenadas $e = 0$ y $\alpha = 0$ partiendo de cualquier distancia no nula desde el objetivo <g>. Además el problema del control de impedancia corresponde al diseño de un controlador que, tras detectar obstáculos en el entorno de trabajo del robot, modifique momentáneamente la posición del objetivo para evitar dichos obstáculos.

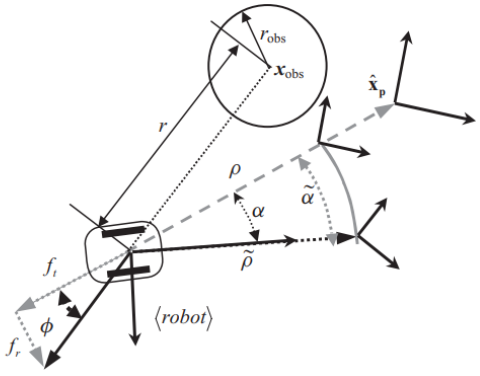


Figura 2.16 Posición y orientación del robot ante la presencia de un obstáculo. [12]

La magnitud de la fuerza ficticia repulsiva generada por un obstáculo puede definirse como

$$F(t) = k_3 - k_4(t) \tag{2.13}$$

Donde: k3, k4 son constantes positivas tales que:

$k_3 - k_4 r_{\max} = 0 \; y \; k_3 - k_4 r_{min} = 1$

La distancia del robot al obstáculo se denota por r, que satisface $r_{\min} \leq r \leq r_{max}$ siendo $r_{\min} \; y \; r_{max}$ las distancias mínima y máxima entre el robot y el obstáculo, respectivamente. Éstas definen la zona en la que actúa la fuerza actúa. La magnitud de f(t) se escala al intervalo [0,1] para que su valor pueda ser amplificado por el controlador de impedancia.

Por otro lado, el ángulo de la fuerza ficticia φ se define como la orientación del obstáculo respecto al robot

La fuerza ficticia tangencial y la fuerza ficticia normal se calculan como $f_t = f cos(\phi) y \; f_r = f sen(\phi)$ respectivamente. Cuando hay múltiples obstáculos, se calcula una fuerza total sumando todos los vectores de fuerza

El modelo de impedancia se define como.

$$[\rho_e \quad \alpha_e] = Z \; f_e \tag{2.14}$$

$$Z = \begin{bmatrix} K_\rho & 0 \\ 0 & K_\alpha \end{bmatrix} \tag{2.15}$$

Donde $f_e = [f_t \quad f_r]^T$ y $K_\rho, K_\alpha > 0$ las cuales describen los parámetros de elasticidad, $f_t$es la fuera ficticia en la dirección del movimiento del robot y $f_r$es la fuerza en la dirección normal al movimiento del robot

Cuando el robot NAO navega e interactúa con el ambiente se define su espacio de estados como

$$[\tilde{\rho} \quad \tilde{\alpha}]^T = [\rho \quad \alpha]^T - [\rho_e \quad \alpha_e]^T \tag{2.16}$$

2.4.3. Implementación del sistema en MATLAB

En esta sección se desarrollará la implementación que se ha realizado en el software Matlab, para el cual se tiene los diferentes componentes medulares del sistema los cuales son

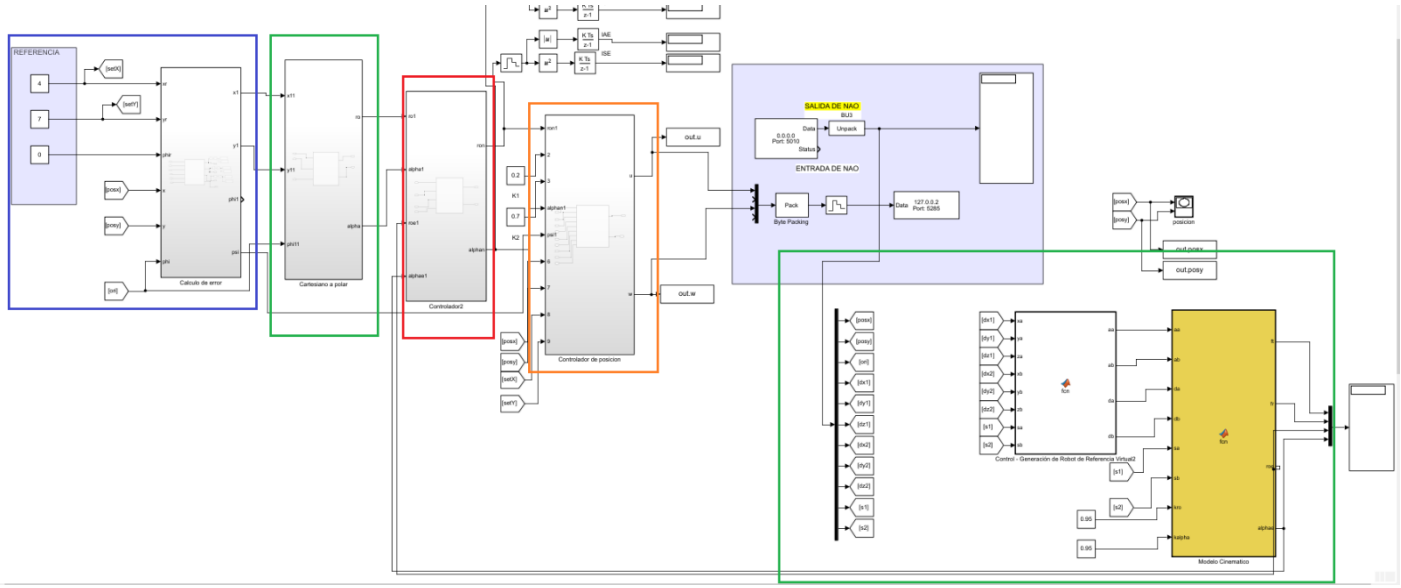


Figura 2.17 Implementación del controlador en Matlab.

2.4.3.1. Controlador de posición

El cual consiste en el recuadro en naranja en la Figura 2.17 en la cual se tiene como entrada los valores de las constantes del controlador, constante que se han ido ajustando para obtener un mejor resultado, y el resultado de la diferencia entre las coordenadas polares y al ángulo como resultado de las fuerzas ficticias. Mientras que a la salida resultan

las variables v, ω que serán enviadas hacia el bloque de comunicación mediante UDP que envía la información hacia Python.

Entonces para realizar una prueba del controlador implementado para la evasión de obstáculos. Se realizará primeramente una prueba del controlador de posición, es decir se probará el controlador con la existencia nula de obstáculos en su camino.

Cuando se a obtenido el controlador se procede a probar el funcionamiento de este controlador para una situación ideal y para tener una referencia de variación de los valores de las constantes del controlador para lo cual con el modelo cinemático del robot uniciclo, se ha obtenido la siguiente respuesta. [16]

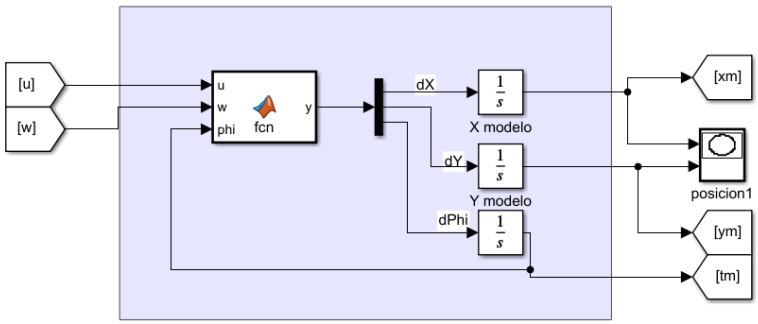


Figura 2.18 Modelo cinemático del robot implementado en Matlab

La respuesta ante una señal paso a t=1s se puede observar la velocidad de respuesta

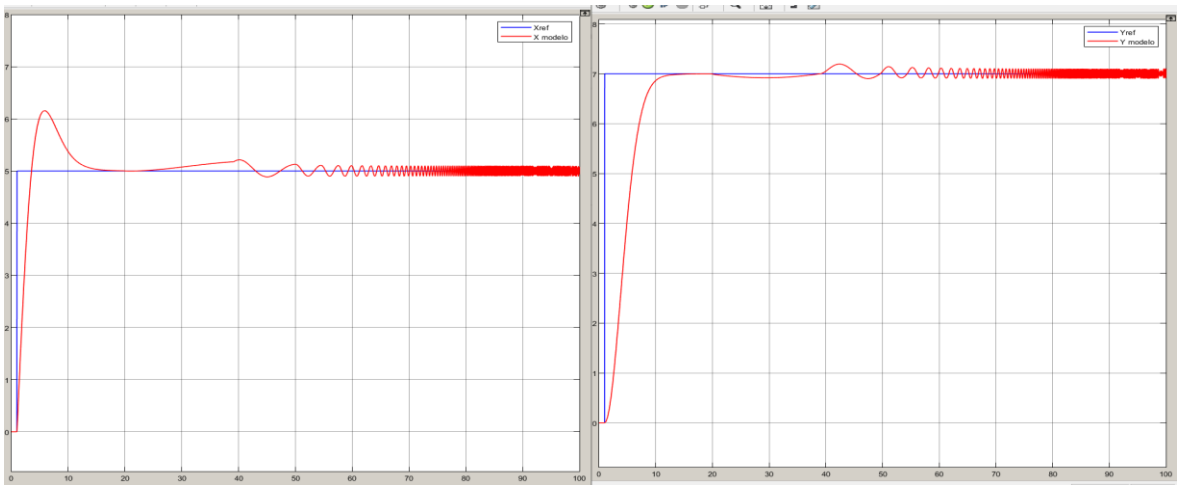


Figura 2.19 Respuesta ante una entrada paso para el controlador de Lyapunov con constantes k1=05 y k2=0.1

Para lo cual se observa una respuesta muy buena para una condición ideal, pero sucede que al implementar el controlador con el robot NAO se obtiene la siguiente respuesta.

Se puede verificar que el comportamiento ya no es el mismo ya no es el ideal que se obtuvo con una señal paso y usando el modelo cinemático, esto es debido a la cantidad de factores que hacen que el robot NAO en su conjunto ya no sea muy simple al considerarlo como una partícula en movimiento.

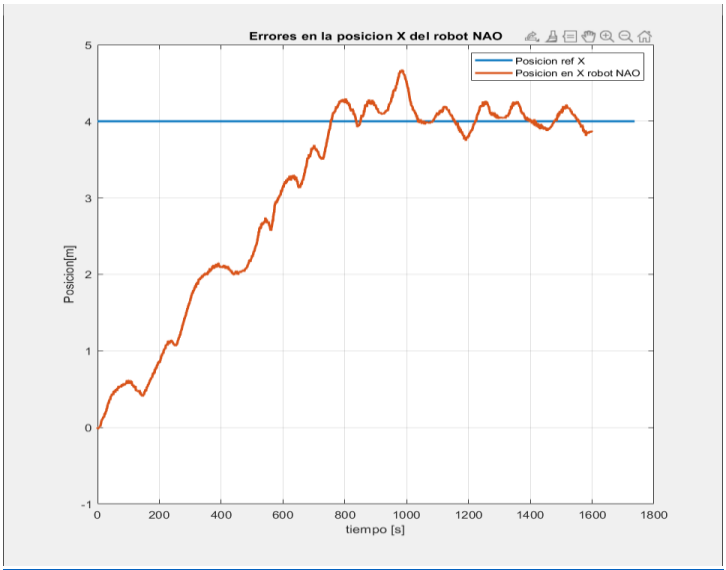


Figura 2.20 Posición en X para el controlador de posición con el robot NAO

Sucede de manera similar tanto en la posición en el eje X como en el eje Y, de manera que al ver el conjunto de estos en la figura se puede observar el comportamiento que se obtuvo con el controlador Lyapunov.

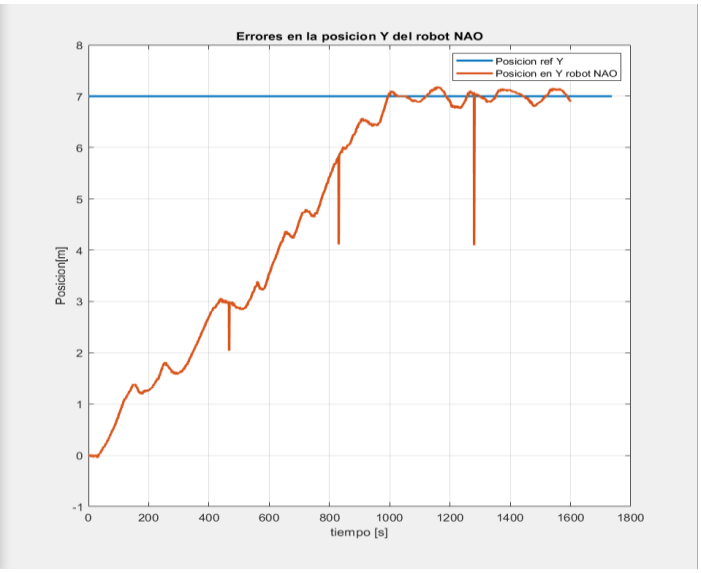


Figura 2.21 Posición en X para el controlador de posición con el robot NAO.

Al superponer las 2 figuras anteriores se puede evidenciar la variación en la respuesta, con lo cual es necesario realizar un ajuste en las constantes del controlador.

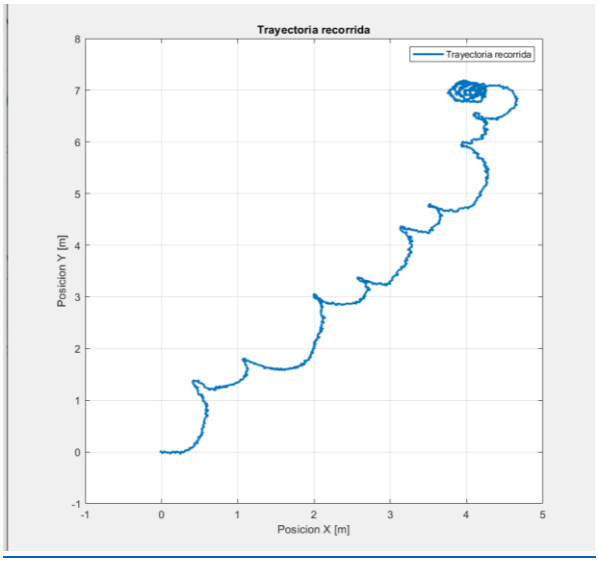


Figura 2.22 Posición en X vs Posición en Y para el controlador probado con el robot NAO.

Además, se ha obtenido los gráficos de la velocidad lineal y velocidad angular del robot durante su desplazamiento, figura en la cual se puede evidenciar ciertos periodos de tiempo en donde existen tramos de velocidad lineal negativa, que se dan por que el robot se ha salido de su curso y ha tenido que circular en sentido reversa para volver a su trayectoria hacia el objetivo deseado.

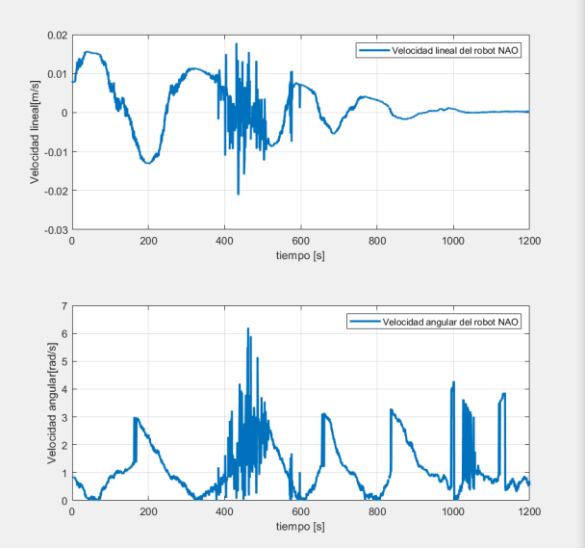


Figura 2.23 Gráficos de velocidad lineal y velocidad angular del robot NAO

Por lo tanto se realizó una serie de ajustes en el controlador para el cual el controlador presente una mejor respuesta tanto como planta, como elemento a controlar, el robot NAO.

En la siguiente secuencia se muestra el resultado obtenido al cambiar las variables a k1=0.2 y k2=0.5 con lo cual se puede evidenciar una mejora tanto en la mejora en la trayectoria que realiza el robot, por consecuente el esfuerzo que se tiene con el controlador el cual se analiza en el siguiente capitulo.

```
function [u,w] = fcn(ron,alphan,psi)
K1=0.2;  %%oficial 0.5
K2=0.7;
q2=0.5;
```

Figura 2.24 Variación de las constantes en el controlador de Lyapunov

Por lo cual para este caso se pueden observar que las graficas de posición en X ya no muestras las oscilaciones pronunciadas como se obtuvo en el primer caso.

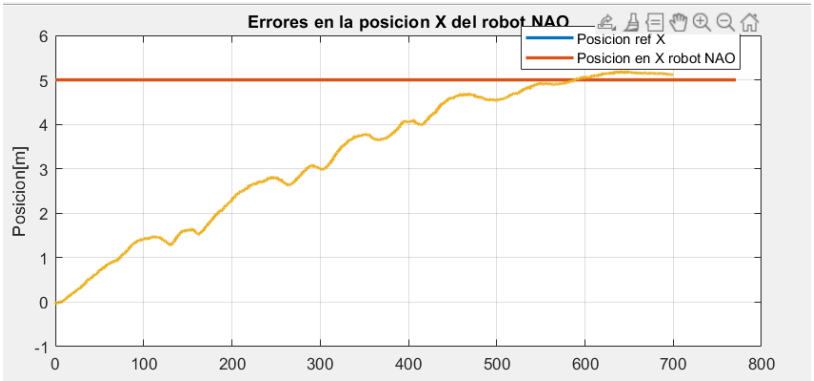


Figura 2.25 Posición en X para el controlador de posición con el robot NAO

De manera similar pasa con la posición del robot en el eje Y, como se puede observar en la figura

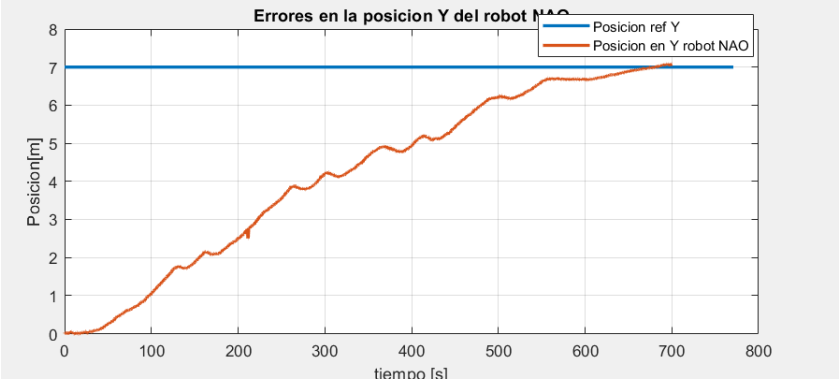


Figura 2.26 Posición en Y para el controlador de posición con el robot NAO

Se puede verificar al comparar el gráficos de posición en X vs posición en Y, aún existen algunos desvíos en los cuales el robot se sale de su trayectoria y luego vuelve a incorporase a esta, pero al comparar con al condición de simulación anterior se puede notar una cierta mejora en la respuesta de este controlador.

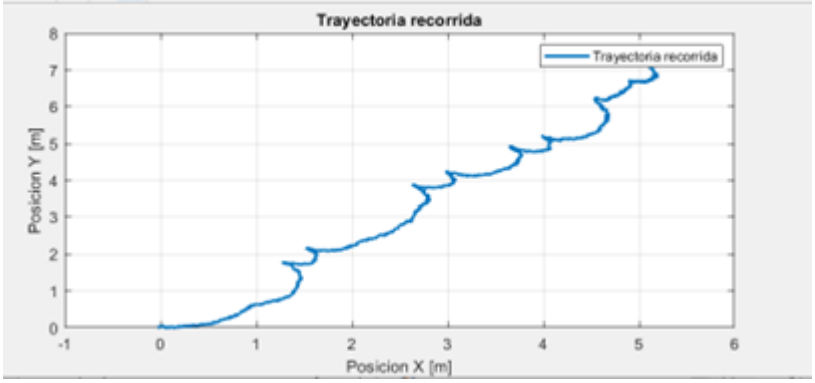


Figura 2.27 Posición en X vs Posición en Y para el controlador probado con el robot NAO.

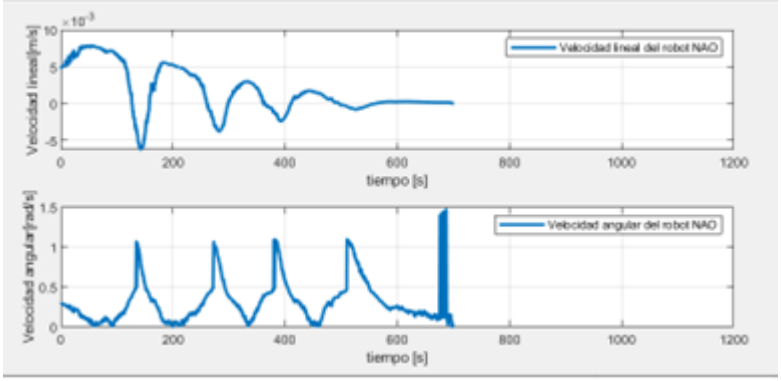


Figura 2.28 Gráficos de velocidad lineal y velocidad angular del robot NAO

2.4.4. Bloque de recepción y envío de datos en Matlab

Para la configuración de este bloque, el elemento medular es la configuración y acorde de la dirección remota la cual tiene que coincidir con la establecida en el puerto en Python. Además se configura un periodo de muestre de 20 ms, tiempo que corresponde al tiempo de respuesta del robot NAO, además de establecer el tipo de datos que se va a recibir en este bloque, para el presente proyecto se esta recibiendo las posiciones del robot, las mediciones obtenidas con los sensores y los estados de los sensores. Información que será tratada posteriormente con el controlador de fuerzas ficticias. [17]

De forma similar en la figura para la configuración del envío de datos desde Matlab hacia Python se configura el puerto de manera que acorde con el establecido en Matlab.

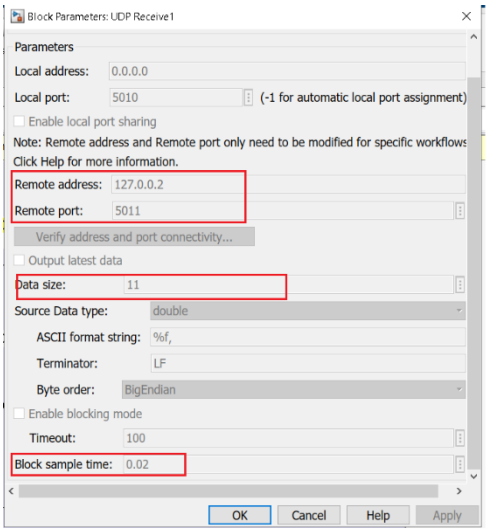


Figura 2.29 Configuración del bloque de recepción en Matlab desde Python.

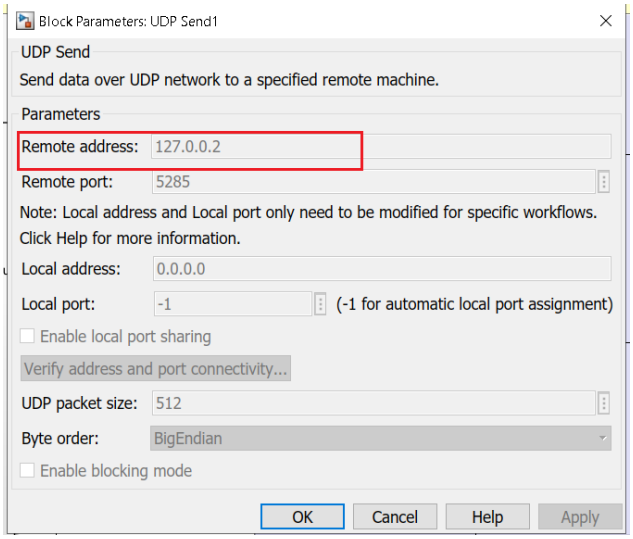


Figura 2.30 Configuración del bloque de envío desde Matlab hacia Python.

2.4.5. Controlador de evasión de obstáculos basado en fuerzas ficticias

Par el controlador en la esquina inferior derecha en la figura Figura ***2.17*** se tiene el controlador basado en fuerzas ficticias en el cual se ha implementado el procedimiento planteado en el diseño para el cual se estableció una distancia mínima de 10 cm con respecto al obstáculo y una distancia máxima de 20 cm, rango dentro del cual las fuerzas ficticias actúan en función de cuan cerca están del obstáculo.

Como se observa en la figura este controlador estar recibiendo las distancia en x, y y z de cada uno de los sensores, además el estado del sensor es decir si el sensor detecta o no detecta al obstáculo, además se incluye las constantes las cuales son una entrada que se puede variar desde la interfaz de usuario, y como salida de este controlador se tiene como resultado las fuerzas tangenciales y normales, que se transformara a valores de variación de posición y variación de orientación respecto al obstáculo.[18] [19]

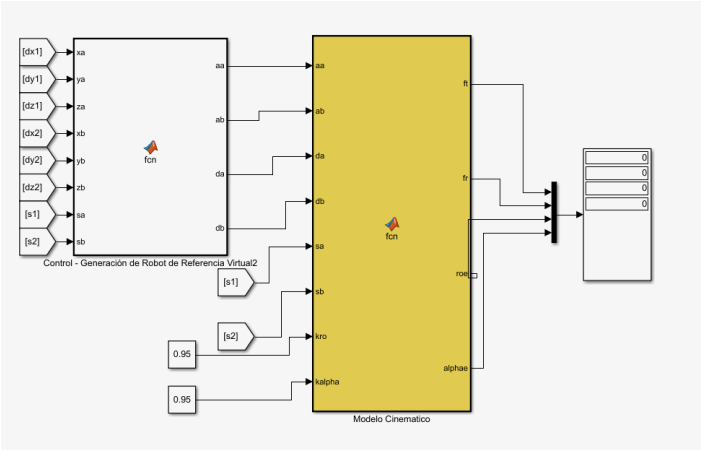


Figura 2.31 Controlador evasión de obstáculos con el método de fuerzas ficticias.

En la figura se muestra el recorrido que ha realizado el robot NAO en la cual se localiza como posición inicial la posición $(x, y) = (0, 0)$ el desplazamiento se realiza de forma suave hasta la posiocon $(x, y) = (2.3, 1.7)$ para luego volver a retomar su trayectoria y alcanzar al punto de referencia establecido en $(x, y) = (4, 4)$

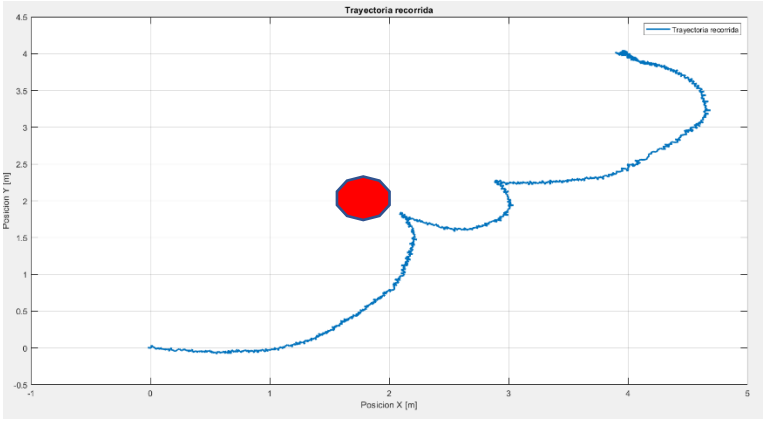


Figura 2.32 Grafico de posición en X vs posición en y del robot NAO, una vez implementado el controlador para la evasión de obstáculo con obstáculo en la pasión x=1.8 y =2.

Las gráficas muestran la posición en x junto con su posición de referencia y la posición en y junto con su posición de referencia en las cuales se observa que la posición del robot alcanza a su referencia en referencia en t = 250 s además se puede ver un sobre pico con un valor de 4.5 m para luego establecerse en t=325 en la posición de referencia deseada. Con lo cual se puede evidenciar resultados de mejora en comparación a los ensayos anteriores con las constantes indicadas.

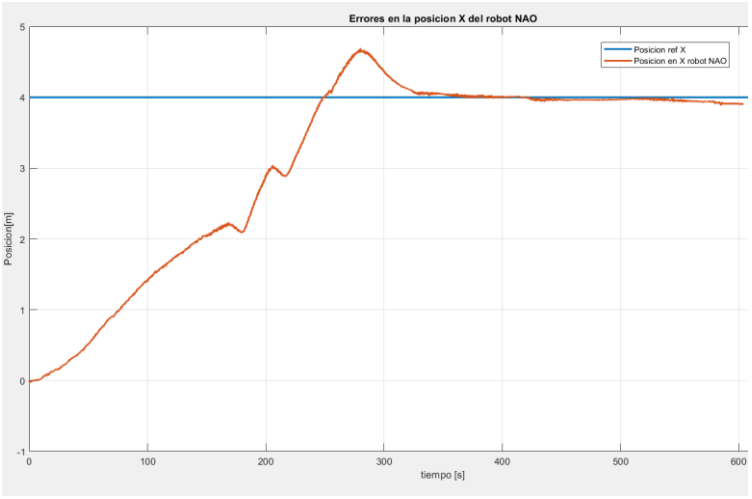


Figura 2.33 Grafico de posición en X del robot NAO, con controlador para la evasión de obstáculo y con obstáculo en la pasión x=1.8 y y =2.

Mientras que para la posición en Y se puede observar que se alcanzo su referencia en el tiempo t=320 s, instante a partir del cual se mantiene la posición de referencia deseada en el robot NAO.

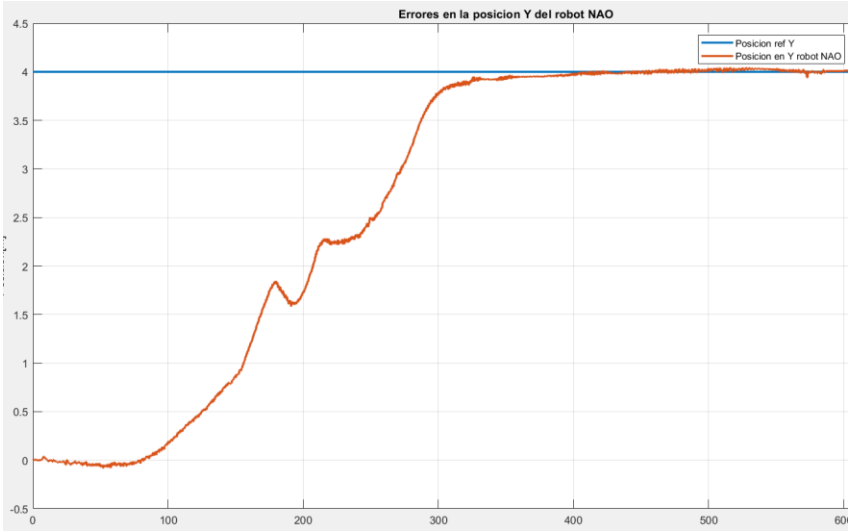


Figura 2.34 Grafico de posición en X del robot NAO, con controlador para la evasión de obstáculo y con obstáculo en la pasión x=1.8 y y =2.

En la figura siguiente se muestra el comportamiento de velocidad lineal y velocidad angular descrito por el robot NAO, se puede evidenciar que existen algunos periodos en los cuales existe velocidad negativa, en tramos donde el robot tiene que evitar al obstáculo se mueve por unos instantes de forma reversa para luego volverse a incorporarse a su camino en búsqueda del objetivo establecido.

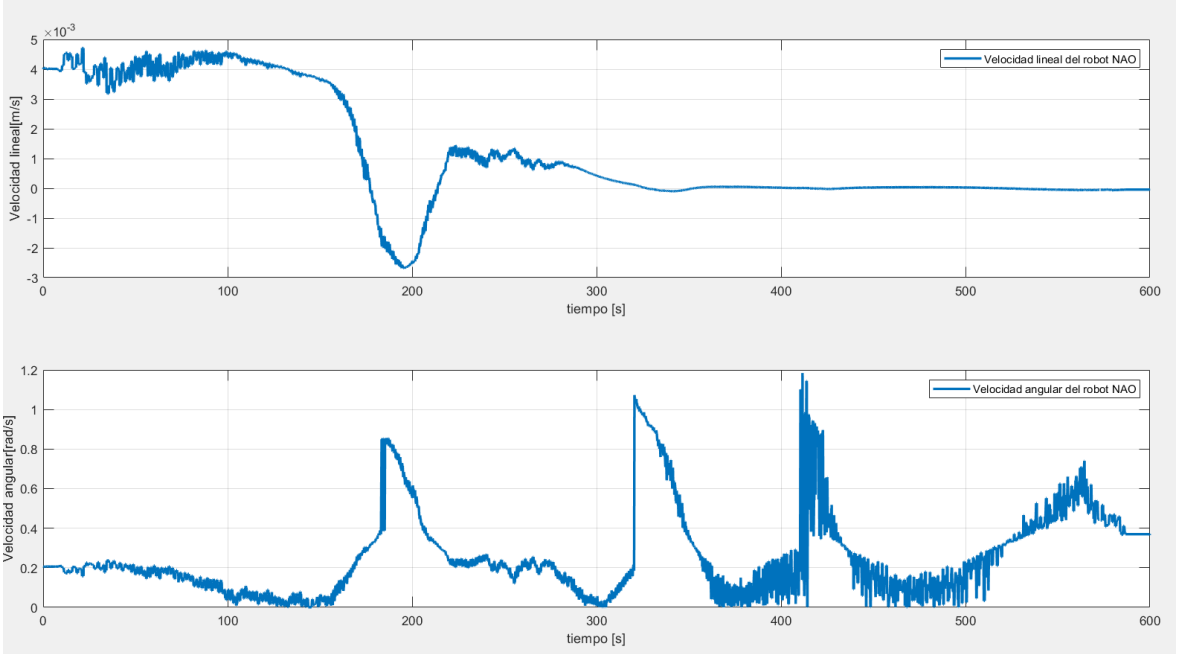


Figura 2.35 Grafico de velocidad lineal y velocidad angular del robot NAO, con controlador para la evasión de obstáculo y con obstáculo en la pasión x=1.8 y y =2.

De manera similar a lo hecho en el anterior caso con el controlador Lyapunov, se procede a realizar ciertos ajustes a las constantes de los controladores para obtener un mejor resultado, por lo cual se procedió.

Posteriormente se procede a probar con varios valores para las constantes en $K_\rho \; y \; K_\alpha$, con lo cual se obtiene ellos siguientes resultados cuando las $K_\rho \; y \; K_\alpha$,0,1 y 0.2, respectivamente, teniendo en cuenta que el obstáculo ha sido posicionado sobre la misma posición X=1.8 y Y=2

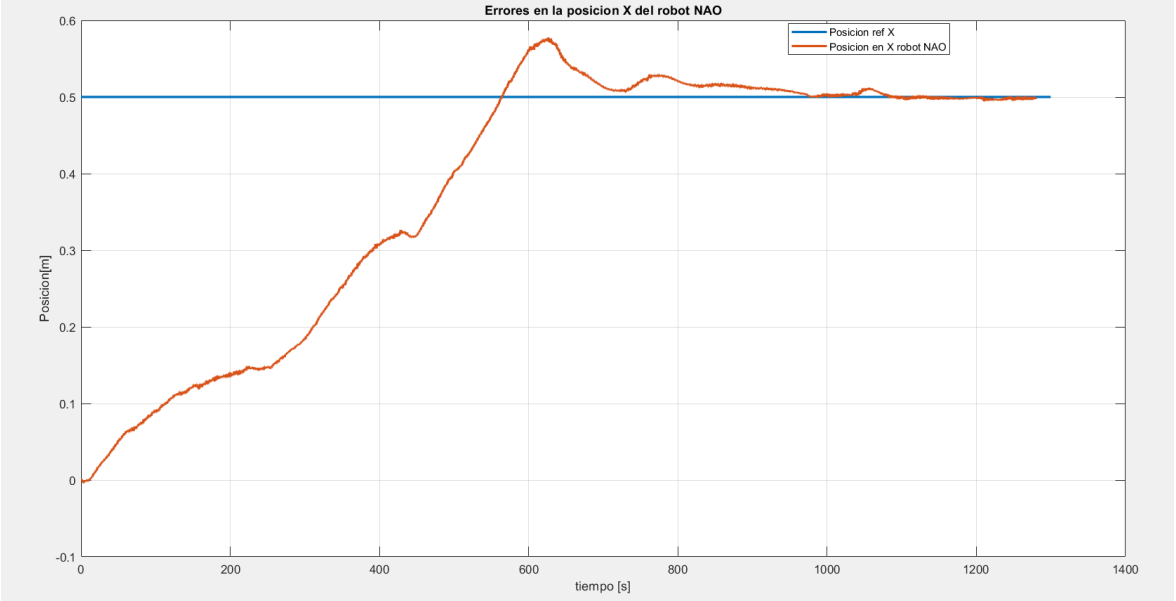


Figura 2.36 Posición en X del robot para los nuevos valores de las constantes K_ρ y K_α

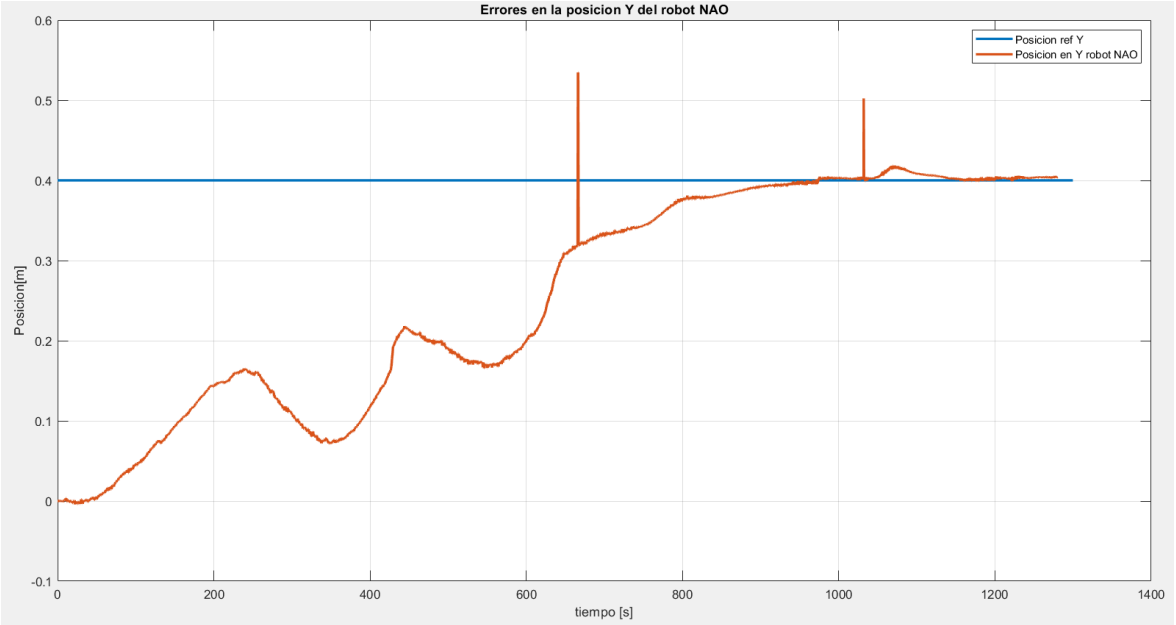


Figura 2.37 Posición en Y del robot para los nuevos valores de las constantes K_ρ y K_α

Condición en la que se ve una mejora , aunque existe un pico en l=el instante 525 , se da por ciertas anomalías con el software debido al muestreo que esta realizando el software.

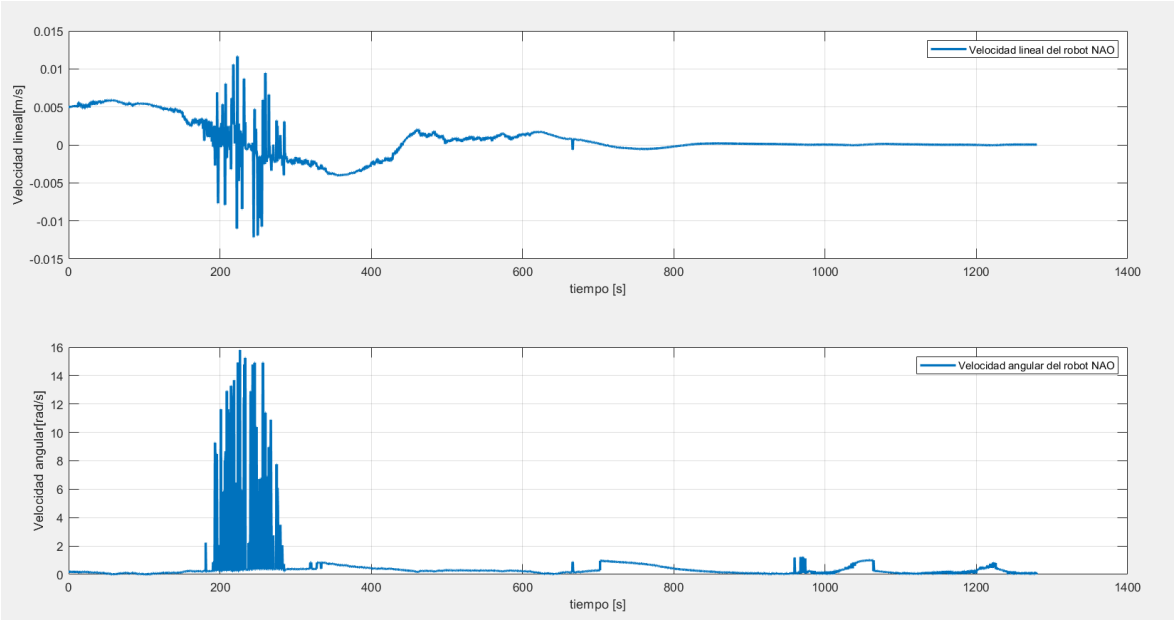


Figura 2.38 Gráficos de velocidad lineal y velocidad angula para el robot con los nuevos valores de las constantes K_ρ y K_α

En la siguiente figura se puede observar el comportamiento del robot NAO al experimentar el cambio de constantes, aunque se puede evidenciar aun hace falta realizar un ajuste debido a que se evidencia un remanente de las fuerzas que provocan esos giros abruptos cuando termina de evitar un obstáculo.

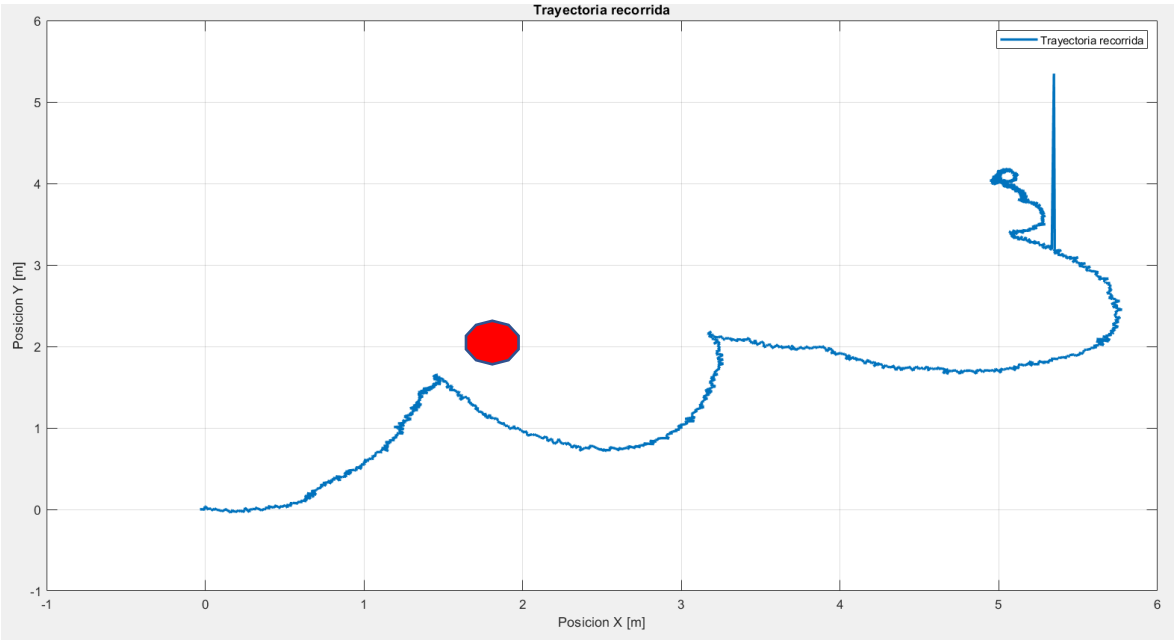


Figura 2.39 Gráficos de velocidad en X vs velocidad en Y para el robot con los nuevos valores de las constantes K_ρ y K_α

2.4.6. Desarrollo de la interfaz grafica

El objetico principal de la interfaz es mostrar de forma gráfica el proceso de este proyecto, con lo cual se a seguido recomendaciones de la norma ISO 9241-151 para el desarrollo de interfaces. [20]

2.4.6.1. Pantalla principal de maniobra de constantes

El desarrollo de la presente interfaz de usuario se la realizo es la cual será de mucha ayuda visual al usuario por lo cual posee los campos de establecimiento de setpoint, establecimiento de las constantes tanto del controlador de Lyapunov como de las constantes de las fuerzas variables.

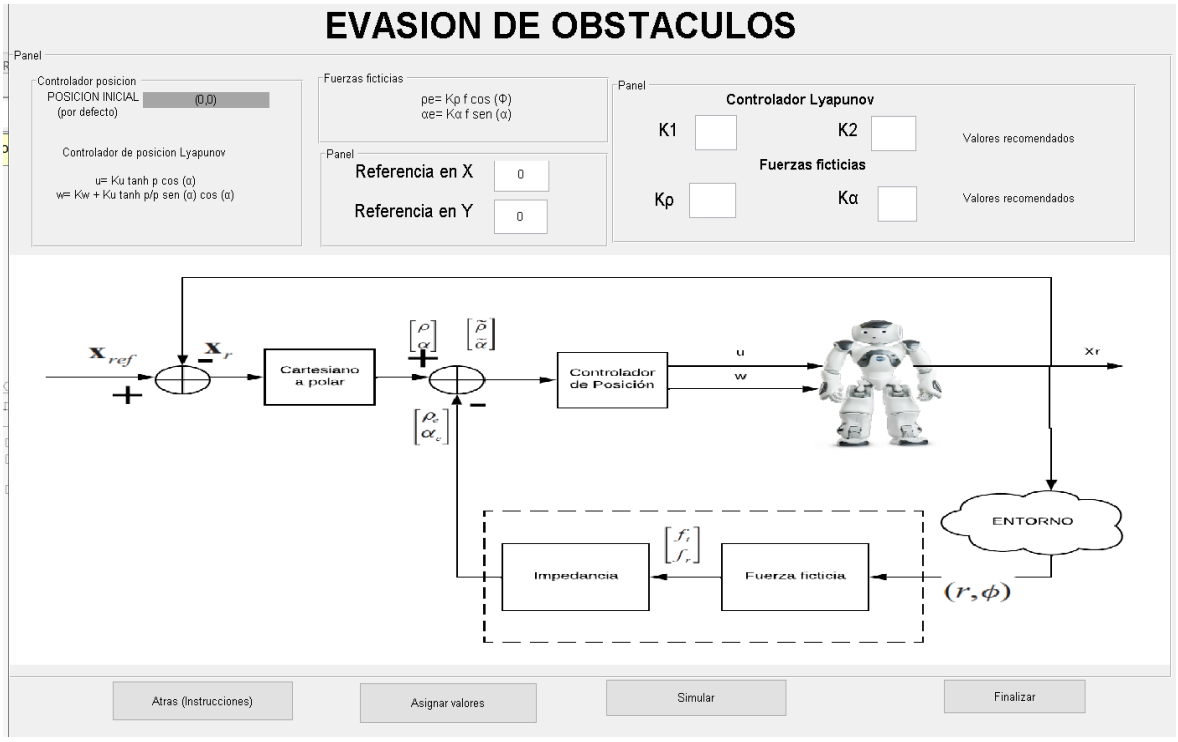


Figura 2.40 Pantalla de establecimiento de constantes

2.4.6.2. Pantalla de instrucciones

En esta pantalla se despliegan las instrucciones de uso de la presente interfaz, y las instrucciones en general para correr el proyecto.

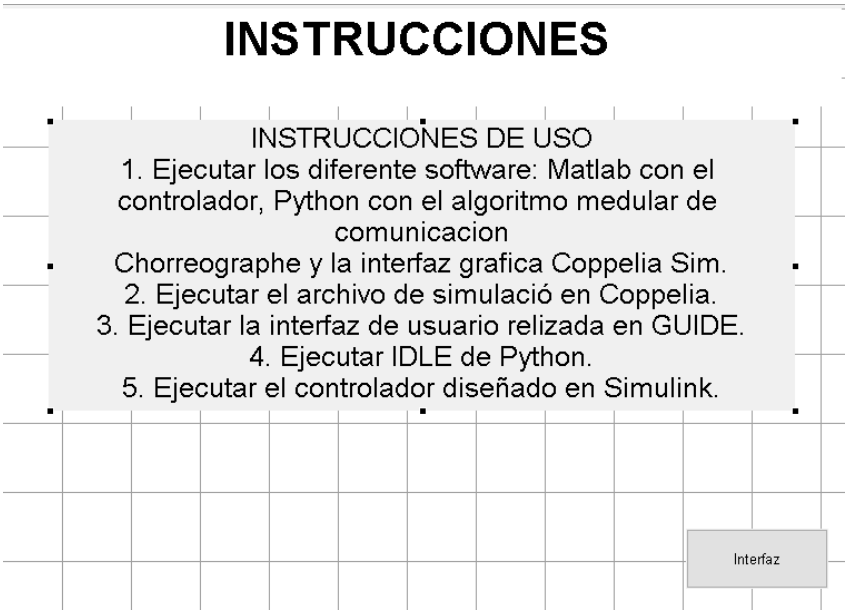


Figura 2.41 Pantalla de instrucciones de la interfaz y del sistema implementado.

2.4.6.3. Pantalla de presentación

En la presente pantalla se muestra una breve presentación con respecto a información del proyecto, información de director de tesis y el autor.

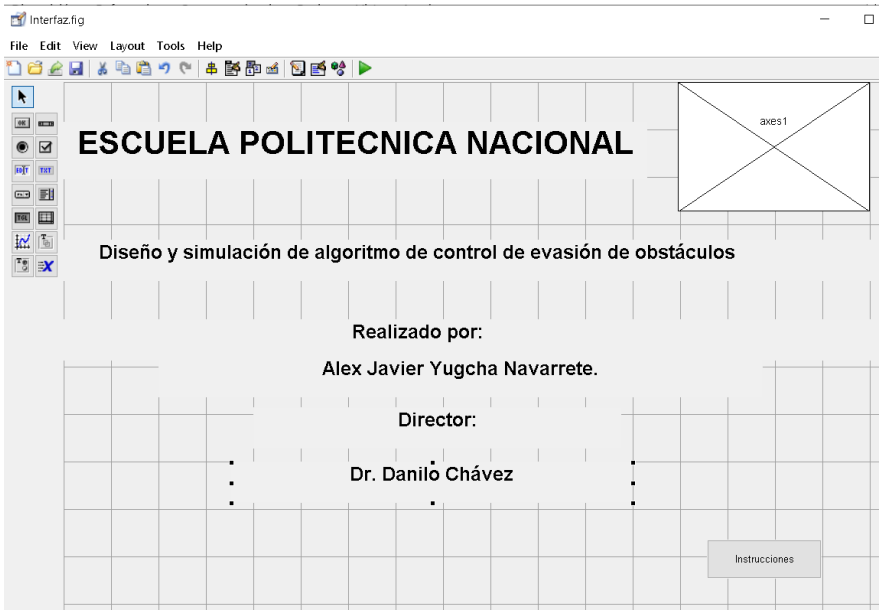


Figura 2.42 Pantalla de presentación

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1. RESULTADOS

En la tabal 3.1 se observa los valores con los cuales se realizó diferentes pruebas para obtener los valores para las constante del controlador de Lyapunov, controlador de posición del lazo interno en el presente sistema. Para lo cual se observara el comportamiento de los índices de desempeño ISU e ISE con lo cual se realizó variaciones en las constantes K2 manteniendo fija K1 con lo cual se observa que el mejor comportamiento para el set de valores K1 y K2, con valores 0.2 Y 0.7, valores con los cuales se muestra una respuesta menor para los valores ISU e ISE un valor menor.

Tabla 3.1 Resultados de las pruebas realizadas para el controlador oposición

Cont. cont. posición	K1	0.1	K1	0.2	K1	0.5	K1	0.9
	K2	0.5	K2	0.7	K2	0.5	K2	0.7
Índices	ISE	ISU	ISE	ISU	ISE	ISU	ISE	ISU
Eje X	100.9	89.8	65.9	56.9	82.3	98.9	123.9	89.9
Eje Y	98	99.8	69.8	76.9	81.7	100.34	101.256	99.12
Ts [s]	532		578		623		712	
Tiempo simulación [s]	823		872		829		885	

Para la siguiente etapa, una vez definidas las constantes para el controlador de posición, del lazo interno, ahora se procede a realizar un conjunto de pruebas para establecer las constantes Kρ y Kαdel controlador de fuerzas ficticias. Con los cuales se determinó un proceso similar.

Tabla 3.2 Resultados de las pruebas realizadas para el controlador de evasión e obstáculos (fuerzas ficticias)

Cont. cont. posición	K1	0.2	K1	0.2	K1	0.2	K1	0.2
	K2	0.7	K2	0.7	K2	0.7	K2	0.7
Const. fuerza ficticia	Kρ	0.5	Kρ	0.25	Kρ	0.95	Kρ	0.25
	Kα	0.25	Kα	0.75	Kα	0.95	Kα	0.75
Índices	ISE	ISU	ISE	ISU	ISE	ISU	ISE	ISU
Eje X	82.3	99.11	98.5	89.1	69.3	56.2	82.3	94.8
Eje Y	81.7	97.83	80.2	97.8	975.7	75.9	91.7	92.5
Ts [s]	850		720		750		900	
Tiempo simulación [s]	876		916		912		934	

Luego de la serie de pruebas realizadas con las contantes, se implementa un entorno mostrado en la figura, en la cual posee obstáculos que simulan paredes que el robot tiene que evitar.

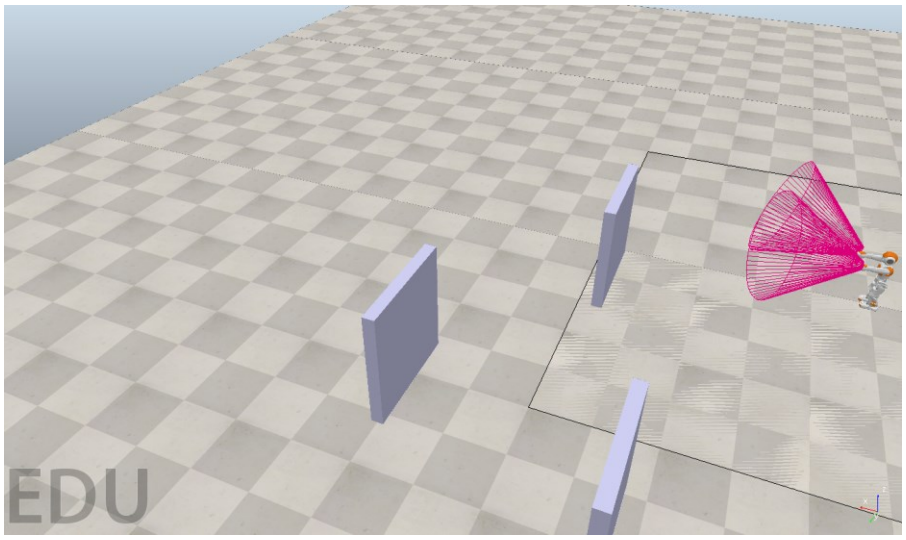


Figura 3.1 Entorno de simulación probado para el controlador, en el cual se muestran 3 paredes a manera de obstáculos

Se puede observar que el movimiento en X y en Y por separado al acercarse a su punto objetivo lo realizan en el eje X de manera mas fluida sin retrocesos considerables, mientras que para el eje Y se puede verificar como el robot retrocede el momento de evitar el obstáculo vuelve a su camino hacia el objetivo y para evitar el próximo obstáculo lo realiza con menor intensidad en su retroceso.

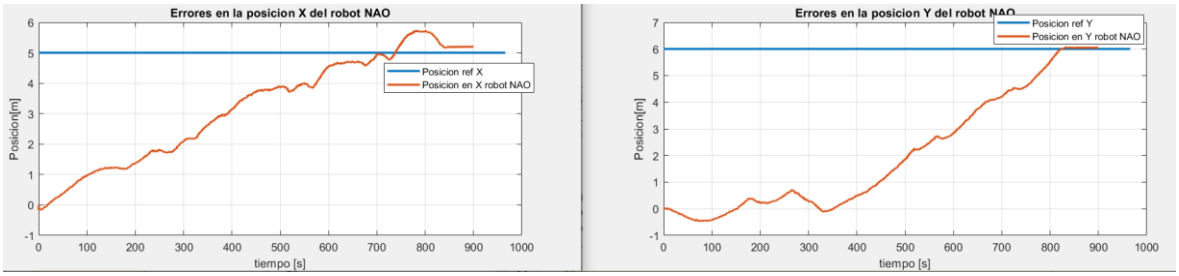


Figura 3.2. Gráficos de posición en X y posición en Y para el robot con el controlador de evasión de obstáculos para una escena con 3 paredes.

A continuación, se muestra el grafico del desplazamiento del robot en el plano X, Y en el cual se puede verificar como esquiva en las esquinas de cada una de las paredes, además se adjunta como responde la velocidad, lo que se puede resaltar es que en esos instantes del esquivo se genera estas velocidad negativas para el retroceso del robot y también se ven los sobre picos en la velocidad angular justamente para hacer girar al robot hacia la pascón de escape del obstáculo.

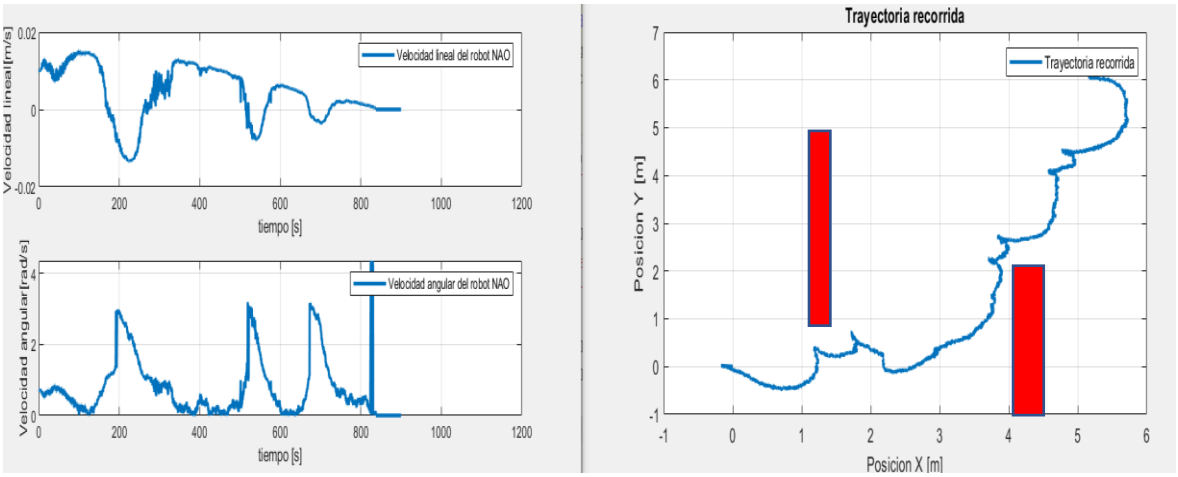


Figura 3.3 Grafico de evolución de la velocidad lineal y la velocidad angular
Desplazamiento del robot a través del plano X, Y con la escena con paredes.

Luego de la serie de pruebas realizadas con las contantes, se implementa un entorno mostrado en la figura, en la cual posee obstáculos que simulan paredes que el robot tiene que evitar.

En la Fig. 3.4. se observa la nueva disposición de los obstáculos en Coppelia Sim, se muestra el punto de partida en 0,0. y el punto de referencia deseado en $(x, y) = (4, -5)$

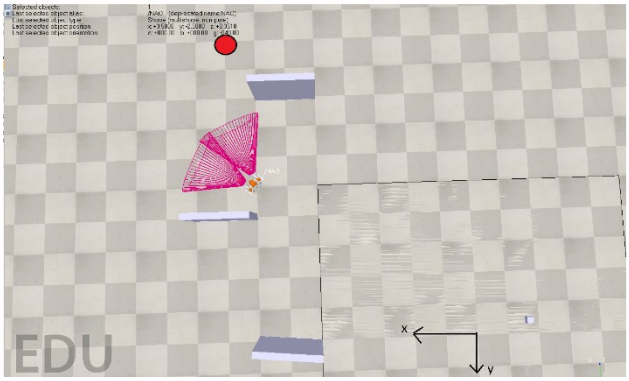


Figura 3.4 Entorno de simulación con nuevo punto de referencia en (4,-5)

Se puede observar la presencia de 3 obstáculos, los cuales son evitados por el robot NAO, en Fig. 3.5. se observa la trayectoria descrita por el robot, empieza a realizar la búsqueda del objetivo, para el primer obstáculo lo realizo con una curva suave, mientras que para el segundo existe un cambio notable tanto en la velocidad lineal como en la velocidad angular del robot, como se evidencia en la Fig. 3.6.

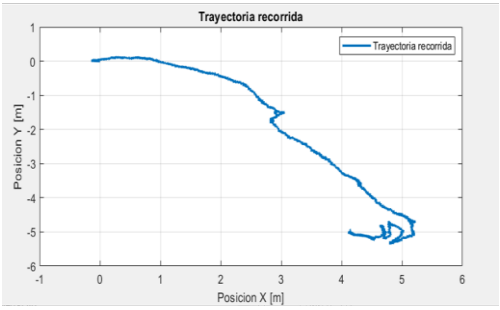


Figura 3.5 Trayectoria recorrida por el robot NAO.

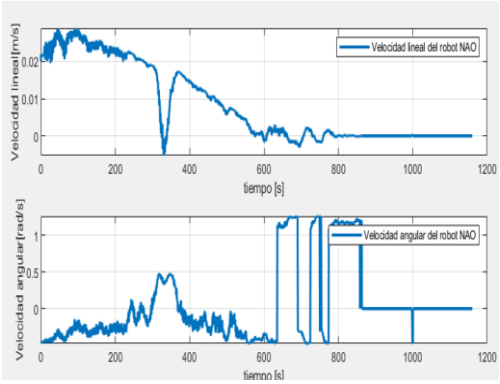


Figura 3.6 Velocidad lineal y velocidad angular descritas por el robot NAO

Se debe mencionar que para que se establezca como cumplida la búsqueda de su posición de referencia deseada, se establece un rango de ±0.10 m, para evitar que el robot permanezca circulando alrededor del punto de referencia, porque puede resultar difícil que el robot alcance justamente el punto (4.00, -5.00) para este caso, lo que se puede evidenciar en la Fig. 3.7. se nota en error en estado estable una vez que se aproximado y alcanzado la posición (4.1, -5.1)

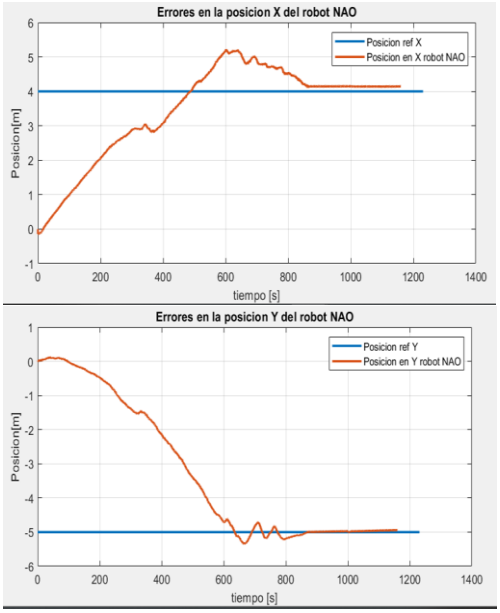


Figura 3.7 Errores en la posición X y en la posición Y del robot

3.2. CONCLUSIONES

El desarrollo del controlador se lo realizo suponiendo un modo sencillo, asumiendo la cinemática y dinámica como una caja negra y se consideró al robot como una partícula móvil, por lo cual fue posible aplicar las ecuaciones mostradas, posteriormente implementar los controladores obtenidos en estas condiciones dieron resultados aceptables con el robot NAO.

Se logro implementar el controlador Lyapunov, para el control de posición controlador que funciona adecuadamente para este robot ajustando las constantes en valores de K1 0.2 y K2=0.7, valores de constantes que se determine a partir de diferentes simulaciones los cuales resultaron con mayores resultados para los índices ISU e ISE.

Con el logro del controlador de posición se pasó a la etapa de evasión en la cual las constates del controlador de posición ya permanecen fijas, y se logró obtener valores de

$K\alpha$ Y $K\rho$ se logró verificar que a la existencia nula de obstáculos el aporte de las fuerzas de este controlador es cero.

Se implementó el controlador de posición, aunque en algunos tramos sucede que el robot empieza a avanzar hacia atrás por las componentes de las fuerzas ficticias, se realizó las pruebas necesarias para poder obtener un resultado aceptable.

La comunicación entre Python Y Matlab se logró mediante comunicación UDP, con el envío y recepción de datos requeridos valores que se discretizan a un tiempo de muestreo de 0.02s el cual se determinó en función al tempo de respuesta del robot NAO.

3.3. RECOMENDACIONES

Es aconsejable empezar al tratar al sistema como una caja negra es decir, en la cual la dinámica del sistema se da por conocida, entonces lo que resta es basarse en las ecuaciones cinemáticas a partir de las cuales se desarrolla el controlador, puede que el resultado no sea el de los mejores, puedo luego de los ensayos pruebas error para los valores de las constantes se tiene que buscar un resultado aceptable.

Se recomienda el uso de las librerías de NAOqi disponibles para Python, las cuales serán de mucha utilidad y permiten el uso de instrucciones a alto nivel, porque de no hacerlo de esta manera resultaría en programar en Coppelia en donde la sintaxis podría resultar compleja debido al tratamiento de articulación por articulación del robot NAO.

Prestar atención al funcionamiento de NAO Choreographe, puesto que al usar un robot virtual requiere de una dirección la cual se podría bloquear cada vez que se ejecuta el programa, entonces a partir de experiencia propia luego de sufrir reinicios constantes del ordenados, lo mejor es presionar el botón de ‘Turn automotous life on’ en Choregraphe cada vez que se va a realizar una nueva prueba con los controladores y sus modificaciones, cerrar el programa Coppelia y volverlo abrir.

Siempre es recomendable ajustarse al tiempo de muestreo que en este caso va a estar dado por la velocidad de respuesta del robot NAO siendo 20ms, ya que si se coloca un valor inadecuado podría derivar en la perdida de información y por lo tanto pérdida de control momentánea del robot.

Verificar el correcto funcionamiento del algoritmo de comunicación entre Matlab y Coppelia, puesto que es el elemento medular del proyecto, establecer las condiciones adecuadas de captura de datos para evitar falsa toma de datos, siempre desplegar en consola los

elementos de interés y verificar que estén en los rangos deseados para lo posteriores cálculos con los controladores. Este algoritmo debe únicamente servir de puente por lo cual no se deben incluir condiciones de control que van a ser realizadas en Matlab.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] “Nao - ROBOTS: Your Guide to the World of Robotics,” @robotsapp, May 18, 2018. https://robots.ieee.org/robots/nao/ (accessed Mar. 07, 2023).

[2] “NAO the humanoid and programmable robot | SoftBank Robotics,” Softbankrobotics.com, 2018. https://www.softbankrobotics.com/emea/en/nao (accessed Mar. 18, 2022).

[3] “NAOqi Framework — NAO Software 1.14.5 documentation,” Aldebaran.com, 2022. http://doc.aldebaran.com/1-14/dev/naoqi/index.html#:~:text=NAOqi%20is%20the%20name%20of,%2C%20resources%2C%20synchronization%2C%20events. (accessed Jul. 22, 2022).

[4] “What is Python? Executive Summary,” Python.org, 2022. https://www.python.org/doc/essays/blurb/ (accessed Jul. 22, 2022).

[5] “Python SDK — NAO Software 1.14.5 documentation,” Aldebaran.com, 2022. http://doc.aldebaran.com/1-14/dev/python/index.html (accessed Jun. 22, 2022).

[6] “Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics,” Coppeliarobotics.com, 2022. https://www.coppeliarobotics.com/ (accessed Jul. 22, 2022).

[7] “Scripts,” Coppeliarobotics.com, 2022. https://www.coppeliarobotics.com/helpFiles/en/scripts.htm (accessed Jul. 22, 2022).

[8] “Harmonic Drive SE - Glossaire,” Harmonic Drive SE, 2022. https://harmonicdrive.de/fr/glossaire/la-robotique-mobile (accessed Jul. 23, 2022).

[9] A. Fernando, “Modelamiento, simulación y control de posicionamiento automático de un robot móvil con tracción...,” ResearchGate, 2017. https://www.researchgate.net/publication/316734735_Modelamiento_simulacion_y_control_de_posicionamiento_automatico_de_un_robot_movil_con_traccion_diferencial_como_h

erramienta_para_apoyar_la_formacion_en_robotica_en_ambientes_de_aprendizaje_SENA (accessed Jul. 24, 2022).

[10] E. Roberto and J. Carlos, “Diseño, simulación y comparación de un controlador pid y un controlador basado en lyapunov para el desplazamiento de un robot humanoide nao v6 sobre un camino generado por un algoritmo de exploración rápida de árbol aleatorio - rrt,” Epn.edu.ec, 2021, doi: T-IE 5164/CD 10993.

[11] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, “Closed loop steering of unicycle like vehicles via Lyapunov techniques,” IEEE Robotics & Automation Magazine, vol. 2, no. 1, pp. 27–35, Mar. 1995, doi: 10.1109/100.388294.

[12] M. García and A. Barreiro, “Análisis de la Estabilidad según Lyapunov de un Control Borroso en Tiempo Discreto.” [Online]. Disponible: https://intranet.ceautomatica.es/old/actividades/jornadas/XXIV/documentos/incon/170.pdf

[13] A. Ferreira, Flávio Garcia Pereira, Raquel Frizera Vassallo, and Mário Sarcinelli-Filho, “A new approach to avoid obstacles in mobile robot navigation: tangential escape.,” ResearchGate, 2005. https://www.researchgate.net/publication/221645816_A_new_approach_to_avoid_obstacles_in_mobile_robot_navigation_tangential_escape (accessed Mar. 07, 2023).

[14] “Sonars | SoftBank Robotics Developer Center,” Softbankrobotics.com, 2022. https://developer.softbankrobotics.com/nao6/nao-documentation/nao-developer-guide/technical-overview/sonars (accessed Jul. 29, 2022).

[15] J. Nieto, E. Slawiñski, V. Mut, and B. Wagner, “Toward safe and stable time-delayed mobile robot teleoperation through sampling-based path planning,” Robotica, vol. 30, no. 3, pp. 351–361, Jul. 2011, doi: https://doi.org/10.1017/s0263574711000695.

[16] G. Tzafestas, “Introduction to Mobile Robot Control,” Academia.edu, Oct. 15, 2013. https://www.academia.edu/11985316/Introduction_to_Mobile_Robot_Control (accessed Jul. 28, 2022).

[17] R. Jalovecký and R. Bystřický, "On-line analysis of data from the simulator X-plane in MATLAB," 2017 International Conference on Military Technologies (ICMT), Brno, Czech Republic, 2017, pp. 592-597, doi: 10.1109/MILTECHS.2017.7988826.

[18] V. Mut, O. Nasisi, R. Carelli and B. Kuchen, "Tracking adaptive impedance robot control with visual feedback," Proceedings. 1998 IEEE International Conference on

Robotics and Automation (Cat. No.98CH36146), Leuven, Belgium, 1998, pp. 2002-2007 vol.3, doi: 10.1109/ROBOT.1998.680609.

[19] A.-C. Huang, K.-J. Lee, and W.-L. Du, “Contact force cancelation in robot impedance control by target impedance modification,” ResearchGate, Feb. 06, 2023. https://www.researchgate.net/publication/368313355_Contact_force_cancelation_in_robot_impedance_control_by_target_impedance_modification

[20] R. DeMoyer and E. E. Mitchell, "Use of the MATLAB graphical user interface development environment for some control system applications," FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No.99CH37011, San Juan, PR, USA, 1999, pp. 12B3/7-12B311 vol.2, doi: 10.1109/FIE.1999.841577.