ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

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

SISTEMA DE TELEOPERACIÓN DE UN ROBOT HUMANOIDE NAO

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

DENNIS ADRIAN CALAPAQUI OÑA

dennis.calapaqui@epn.edu.ec

DIRECTOR: DR. GEOVANNY DANILO CHÁVEZ GARCÍA

danilo.chavez@epn.edu.ec

DMQ, septiembre 2023

CERTIFICACIONES

Yo, DENNIS ADRIAN CALAPAQUI OÑA 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.

_____

DENNIS ADRIAN CALAPAQUI OÑA

Certifico que el presente trabajo de integración curricular fue desarrollado por DENNIS ADRIAN CALAPAQUI OÑA, bajo mi supervisión.

_____

DR. GEOVANNY DANILO 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.

DENNIS ADRIAN CALAPAQUI OÑA

DR. GEOVANNY DANILO CHÁVEZ GARCÍA

ING. KLEVER DARIO PATIÑO CAIZA

DEDICATORIA

Dedico este trabajo a mis abuelitos Aurelio y Rosa, quienes siempre estuvieron pendientes de mí, acompañándome en cada uno de mis pasos, no me vieron cumplir con esta meta, pero estoy seguro de que desde el cielo están muy felices por mí.

También está dedica a mi madre Laurita, por haberme guiado para ser la persona que soy en la actualidad, sus reglas, bendición, motivación y apoyo constante me ha permitido llegar a cumplir con cada uno de mis anhelos y objetivos.

Dennis

AGRADECIMIENTO

Agradezco principalmente a mi madre Laurita, gracias por sus enseñanzas, motivación, inspiración y valores que han forjado la persona que soy. Por cada sacrificio y esfuerzo realizado para que pueda cumplir cada objetivo que me he trazado, por su compresión, amor y apoyo incondicional en cada etapa de mi vida.

A la Escuela Politécnica Nacional por sus grandes maestros quienes aportaron con el conocimiento y las enseñanzas necesarias a lo largo de toda la carrera y en la realización de este trabajo.

En especial al Dr. Danilo Chávez y al Ing. Klever Patiño, quienes estuvieron pendientes en brindarme el apoyo y conocimiento necesario para que logre culminar con este trabajo.

A mis compañeros y amigos con quienes he compartido sus sueños, aspiraciones y largas jornadas de estudio, me han inspirado a cumplir cada una de mis metas.

Gracias a todos por formar parte de esta importante etapa de mi vida.

Dennis

ÍNDICE DE CONTENIDO

RESUMEN

El uso de sistemas de comunicación junto al desarrollo de la tecnología de los últimos años permite contar con robots teleoperados, es decir operados a distancia. De esta forma permite al ser humano operar equipos sin la necesidad de exponerse en ambientes que pueden significar un peligro al momento de realizar una actividad.

Es así, que en el siguiente documento se presenta el diseño y simulación de un sistema de teleoperación de un robot humanoide (NAO V6). En el primer capítulo se detalla la información de los programas a utilizar, la cinemática del robot humanoide y el protocolo de comunicación utilizado. Información obtenida a partir de la investigación realizada para la implementación del sistema de teleoperación.

En el segundo capítulo se presenta la metodología utilizada en la implementación del sistema teleoperado, la transmisión de datos entre la estación local, la estación remota y mediante el uso de Python la transmisión de datos entre CoppeliaSim Edu y MATLAB/Simulink para la operación del robot humanoide NAO V6. Además, se realiza el análisis de la confiabilidad en la recepción y transmisión de datos del protocolo de comunicación UDP (User Datagram Protocol).

Finalmente, se presenta la interfaz gráfica desarrollada en App Designer, herramienta proporcionada por MATLAB, para cada una de las estaciones. La misma permite al usuario operar de forma remota el robot humanoide con la ayuda de un joystick. La operación del robot se puede dar en tres escenarios diferentes, mismo que puede ser seleccionado por el usuario.

PALABRAS CLAVE: robot humanoide, protocolo UDP, NAO, teleoperación, joystick.

ABSTRACT

The use of communication systems together with the development of technology in recent years allows for teleoperated robots. In this way, it allows the human being to operate equipment without the need to be exposed to environments that may pose a danger when carrying out an activity.

Thus, the following document presents the design and simulation of a teleoperation system for a humanoid robot (NAO V6). The first chapter details the information on the softwares to be used, the kinematics of the humanoid robot and the communication protocol used. Information obtained from the research carried out for the implementation of the teleoperation system.

The second chapter presents the methodology used in the implementation of the teleoperated system, the data transmission between the local station, the remote station and with Python the data transmission between CoppeliaSim Edu and MATLAB/Simulink for the operation of the NAO V6 humanoid robot. In addition, the analysis of the reliability in the reception and transmission of data of the communication protocol UDP (User Datagram Protocol) is carried out.

Finally, the graphical interface developed in App Designer, a tool provided by MATLAB, for each of the stations is presented. It allows the user to remotely operate the humanoid robot with the help of a joystick. The robot's operation can take place in three different scenarios, which can be selected by the user.

KEYWORDS: humanoid robot, UDP protocol, NAO, teleoperation, joystick.

1. INTRODUCCIÓN

La robótica ha experimentado un crecimiento importante en los últimos años en el campo de la ingeniería, por las innumerables tareas que se están llevando a cabo en la industria, investigación, educación, etc. Tareas que son realizadas de forma automática o semiautomática basándose en algoritmos predeterminados y adaptados a las tareas requeridas, con la finalidad de facilitar la vida de los usuarios, mismos que son los encargados de supervisar las diferentes actividades realizadas por los robots. [1]

El desarrollo de la robótica y diversas herramientas tecnológicas han permitido reemplazar al hombre en el momento de ejecutar diversas actividades que son consideradas de alto riesgo, desde este punto de vista de seguridad, utilizar robots teleoperados se considera una ventaja ya que de esta forma se evita la exposición a escenarios que pueden ocasionar daños irreversibles en la integridad de las personas.[2]

Uno de los robots que son utilizados para la investigación y educación es el robot humanoide denominado NAO V6, este robot imita los comportamientos del ser humano, con el fin de tener la capacidad de ejecutar las mismas tareas que realiza una persona y utilizarlo en la vida cotidiana e industrial. Los algoritmos implementados en el robot NAO son utilizados para el control de posición, trayectoria, acciones y tareas a cumplir por el robot. [3]

Para recibir la información y esta sea ejecutada se debe implementar un sistema de comunicación entre el emisor y el receptor. En este trabajo se hace uso del protocolo de comunicación UDP, este permite tener una operación remota (teleoperación) del robot. La teleoperación es un área de la robótica, que consiste en el desarrollo de algoritmos que permiten el control del robot de forma remota por el usuario/operador. Este sistema permite controlar el movimiento y las diferentes acciones requeridas por el operador a ciertas distancias.[4]

Los sistemas teleoperados tienen ciertas limitaciones, los mismos que se dan por la capacidad de procesamiento de la información recibida, y con ello la precisión y coordinación del sistema hombre - robot. Sin embargo, estos sistemas son utilizados desde hace varios años. Uno de los principales sectores, donde se implementaron este tipo de sistemas, es el nuclear y con el paso de los años la aplicabilidad se expandió a sectores como el industrial, educación, investigación, medicina, etc. [5]

Los diferentes softwares utilizados (Python, MATLAB/Simulink, CoppeliaSim Edu, Choregraphe) en el desarrollo de este sistema permiten establecer la comunicación remota

entre el operador y el robot, con la finalidad de que este cumpla con las acciones requeridas por el operador.[6]

Se presentan tres escenarios diferentes, en los cuales el operador tiene la facultad de guiar al robot NAO, evadiendo obstáculos para llegar desde el punto origen a un punto final.

1.1 OBJETIVO GENERAL

Establecer la comunicación mediante el protocolo UDP (User Datagram Protocol) entre la estación local y estación remota para el control de un robot humanoide NAO V6 mediante el uso de varios programas de simulación como: MATLAB/Simulink, Python, Choregraphe y CoppeliaSim Edu; para que el robot sea capaz de recibir órdenes y cumplir con el trabajo deseado a distancia.

1.2 OBJETIVOS ESPECIFICOS

1. Revisar el contenido bibliográfico con respecto al manejo y simulación del robot NAO V6.
2. Revisar el contenido bibliográfico con respecto a la teleoperación y comunicación del robot NAO V6.
3. Establecer la comunicación entre MATLAB/Simulink y Python para el envío y recepción de datos a través del protocolo de comunicación UDP.
4. Establecer la conexión entre Python, Choregraphe y CoppeliaSim Edu para la ejecución de comandos.
5. Desarrollar un interfaz de usuario que permita enviar y recibir información de las ordenes que se envían para el NAO V6.

1.3 ALCANCE

El alcance del proyecto queda definido por los siguientes puntos:

- Se realizará la revisión bibliográfica de la información necesaria para implementar el sistema de comunicación.
- Se estudiará los diferentes softwares a ser utilizados para la comunicación y simulación como: MATLAB/Simulink, Python, Choregraphe y CoppeliaSim Edu.
- Se implementará la comunicación con el protocolo UDP para el envío y recepción de información y las órdenes a cumplir por parte del robot NAO V6.

- Se diseñará e implementará una interfaz gráfica que permita observar el funcionamiento del sistema desarrollado.
- Se comprobará el funcionamiento y desempeño del sistema de comunicación y el trabajo a distancia, con la implementación de diferentes órdenes.

1.4 MARCO TEÓRICO

A continuación, se presenta toda la información relacionada para el desarrollo del algoritmo del sistema de comunicación, el uso del robot humanoide con la finalidad de cumplir con tareas a distancia, los diferentes programas utilizados y el desarrollo de la interfaz para la estación local y remota que se presenta como parte del Trabajo de Integración Curricular.

1.4.1 ROBOT HUMANOIDE NAO V6

El Robot Humanoide Nao V6 es un robot autónomo y programable desarrollado por Aldebaran Robotics en el año 2004, empresa francesa con sede en París. En 2015 la empresa japonesa SoftBank Robotics adquiere los derechos. NAO es un robot humanoide pequeño que viene integrado con motores y sensores esto permite interactuar con las personas como caminar, bailar, hablar y reconocer rostros y objetos. En el caso de NAO en su sexta presentación se utiliza en la educación, investigación y atención médica. Las facilidades para adaptarse a diferentes escenarios han hecho que NAO presente una producción en masa y se encuentre al alcance del consumidor con valores accesibles, en este caso se pueden encontrar en un valor de 10 mil euros en el mercado.[7]

NAO es uno de los robots más famosos a nivel mundial, con presencia en más de 70 países, esto por las prestaciones que tiene y la facilidad que presenta a la hora de programarlo.

1.4.1.1 CARACTERÍTICAS PRINCIPALES

El robot NAO tiene 58 cm de altura, con un peso de 5.5 Kg, posee 25 grados de libertad en diferentes articulaciones distribuidos de la siguiente forma: 10 grados de libertad se encuentran en sus dos extremidades inferiores, 10 grados de libertad en sus dos extremidades superiores, 2 grados en el cuello, 2 grados por el conjunto de las 2 manos y 1 grado de libertad para la pelvis. Cada uno de los grados de libertad son comandados por un motor de corriente continua sin núcleo con escobillas Portescap, que son alimentados por una batería de iones de litio de 27,6 Wh, con autonomía de 90 minutos de funcionamiento. El material de construcción es de policarbonato, reforzado con fibra de carbono. En la **Figura 1.1** se puede observar el Robot NAO V6.[8]

Figura 1.1 Robot Humanoide NAO V6 [9]

El robot NAO cuenta con 2 cámaras de 5 megapíxeles, con acelerómetro de tres ejes y dos giroscopios, cuatro micrófonos omnidireccionales, dos sensores infrarrojos, nueve sensores táctiles y ocho sensores de presión. Para procesar toda la información recopilada por los sensores el robot cuenta con un CPU Intel Atom de 1,91 GHz de cuatro núcleos, 4 GB de RAM SSD de 32 GB, para la comunicación cuenta con Bluetooth, Wi-Fi y Ethernet. Para la programación y visualización se hace uso de Linux OS y Choregraphe Suite. NAO tiene la capacidad de reconocimiento de voz y diálogo en más de 20 idiomas incluidos inglés, francés, español, italiano, ruso, etc. [10]

1.4.1.2 GRADOS DE LIBERTAD ROBOT NAO

El robot NAO cuenta con articulaciones para realizar las rotaciones de las partes móviles del cuerpo, se coloca un marco de referencia en cada articulación, el robot en su posición inicial tiene la misma orientación. Las rotaciones de balanceo tienen lugar alrededor del eje X, las rotaciones de cabeceo alrededor del eje Y, y en el eje Z se tiene las rotaciones de guiñada. [11]

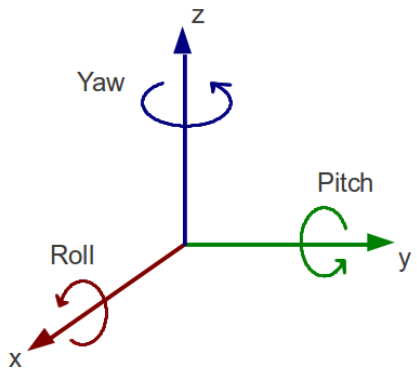

Figura 1.2 Plano de referencia para las rotaciones de las articulaciones del robot NAO V6[11]

En este trabajo se toma en cuenta los ángulos de giro de las dos extremidades inferiores, cada uno de ellos con el movimiento realizado, la acción que se visualiza y el rango en el cual se va a presentar el giro del conjunto mencionado. [11]

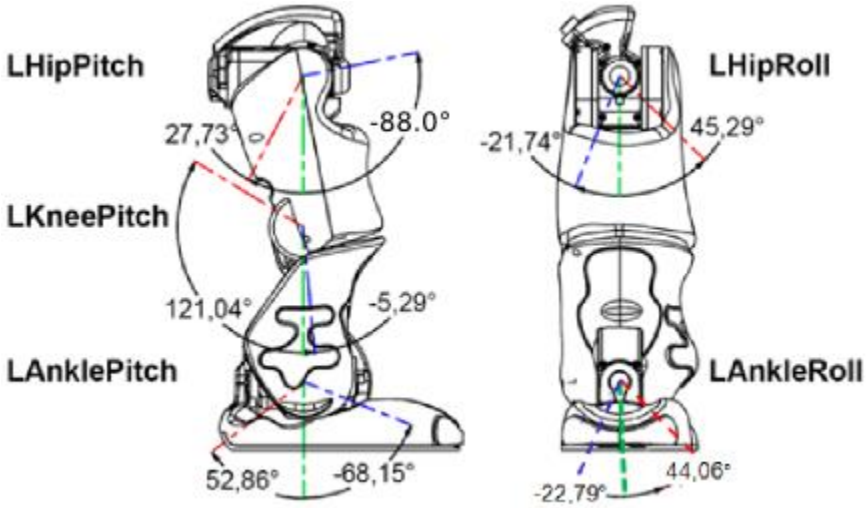

Figura 1.3 Articulaciones de la pierna izquierda [11]

En la **Figura 1.3** se puede observar las articulaciones de la pierna izquierda, en la cual se detallan los límites de movimiento de cada uno de ellos. De la misma forma en la **Tabla 1.1** se cuenta con un resumen detallado de los rangos de cada una de las articulaciones correspondientes a la pierna izquierda del robot NAO.

Tabla 1.1 Rango de movimiento para las Articulaciones de la pierna izquierda [11]

Nombre	Movimiento	Rango (grados)	Rango (radianes)
LHipRoll	Articulación de la cadera izquierda derecha e izquierda (X)	-21.74 a 45.29	-0.38 a 0.79
LHipPitch	Articulación de la cadera izquierda delante y detrás (Y)	-88.00 a 27.73	-1.54 a 0.48
LKneePitch	Articulación de la rodilla izquierda (Y)	-5.29 a 121.04	-0.09 a 2.11
LAnklePitch	Articulación del tobillo izquierdo delante y detrás (Y)	-68.15 a 52.86	-1.19 a 0.92
LAnkleRoll	Articulación del tobillo izquierdo derecho e izquierdo (X)	-22.79 a 44.06	-0.40 a 0.77

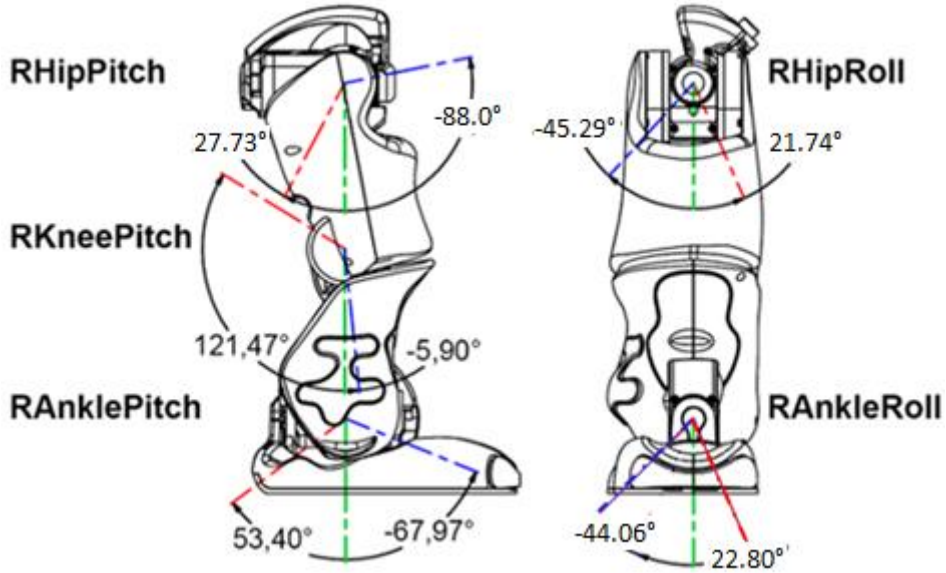

Figura 1.4 Articulaciones de la pierna derecha [11]

En la **Figura 1.4** se pueden observar las articulaciones de la pierna derecha, en la misma se detallan los límites de movimiento de cada una de las articulaciones. En la **Tabla 1.2** se tiene un resumen detallado de los rangos de cada una de las articulaciones correspondiente a la pierna derecha del robot NAO.

Tabla 1.2 Rango de movimiento de las articulaciones de la pierna derecha [11]

Nombre	Movimiento	Rango (grados)	Rango (radianes)
RHipRoll	Articulación de la cadera derecha derecha e izquierda (X)	-45.29 a 21.74	-0.79 a 0.38
RHipPitch	Articulación de la cadera derecha anterior y posterior (Y)	-88.00 a 27.73	-1.54 a 0.48
RKneePitch	Articulación de la rodilla derecha (Y)	-5.90 a 121.47	-0.10 a 2.12
RAnklePitch	Articulación del tobillo derecho delante y detrás (Y)	-67.97 a 53.40	-1.18 a 0.93
RAnkleRoll	Articulación del tobillo derecho derecho e izquierdo (X)	-44.06 a 22.80	-0.77 a 0.40

1.4.2 PYTHON

Python es un software libre que fue creado a inicios de los 90 por el programador Guido Van Rossum, es el lenguaje de programación de más alto nivel. La sintaxis simple, clara,

sencilla, gestor de memoria, cantidad de librerías disponibles, hace que trabajar con Python sea sencillo, rápido y comprensible para el usuario.[12]

Las características de la sintaxis de este lenguaje de programación respecto a su equivalente como es el lenguaje C, son más compactos, legibles, etc. A continuación, se puede observar la sintaxis utilizada en ambos lenguajes para el tradicional 'Hola Mundo'.

```
print 'Hello, wolrd!'   |       #include<stdio.h>
                        |       int main(void){
                        |               printf ("Hello, world! \n");
                        |               return 0;
                        |       }
```

Figura 1.5 Sintaxis de "Hello, world!" en Python y C [Autoría propia]

En la **Figura 1.5** se puede observar las diferentes sintaxis utilizadas para imprimir en consola "Hello, world!", en el lado izquierdo para Python y en el lado derecho para el lenguaje de programación en C. En el caso de Python es suficiente con una línea de instrucción a comparación con C, en el cual es necesario citar librerías y abrir un lazo principal para que se imprima la instrucción requerida por el usuario.[12]

1.4.3 COPELLIASIM EDU

CoppeliaSim Edu es un simulador de robótica desarrollado por Coppelia Robotics con base en Zúrich. El entorno de desarrollo de este software de simulación se basa en una arquitectura de control distribuido, cada objeto o modelo se puede controlar de forma individual con la ayuda de un script integrado. CoppeliaSim Edu es versátil e ideal para controladores que se puede escribir en C/C++, Java, MATLAB, Python, etc.

Además, CoppeliaSim Edu es utilizado en desarrollar algoritmos de simulaciones de automatización de fábricas, desarrollo de prototipos y con fines educativos relacionados con la robótica.[13]

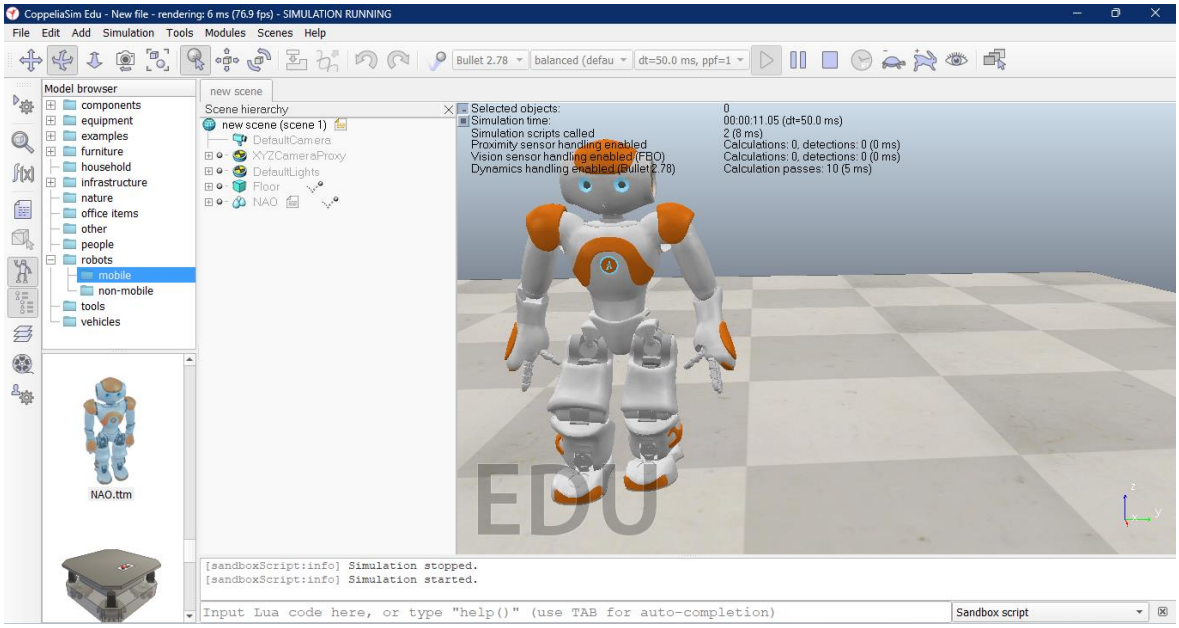

Figura 1.6 CoppeliaSim Edu [Autoría propia]

El entorno y bibliotecas de los diferentes prototipos de robots se muestran en la **Figura 1.6**, junto con las librerías de funciones API, mismos que permiten implementar escenarios de simulación para aplicaciones industriales, investigación y desarrollos académicos. Para el desarrollo de este trabajo se utilizará CoppeliaSim Edu en la versión 4.3.0 revisión 10, disponible en la página web oficial del desarrollador.[13]

Las características para tomar en cuenta son las siguientes:

Multiplataforma: disponible para diferentes sistemas operativos como: Windows, MacOS y Linux.

Motores de simulación gráfica: ODE, Bullet, Vortex y Newton.

Navegador: drag and drop.

Enfoques de programación: la integración se da mediante scripts incrustados, plugins, add-ons, nodos ROS o APIs para clientes remotos.

Lenguajes de programación: se acopla a scripts desarrollados en C/C++, Python, Java, Lua, MATLAB, Octave.

Registro de datos y visualización: representación de gráficas, gráfico X/Y o curvas 3D.

[14]

1.4.4 CHOREAGRAPHE

Choreagraphe es un software que permite la programación de robots humanoide NAO desarrollado por Aldebaran Robotics, herramienta que permite la conexión macroscópica de comportamientos de alto nivel, es decir a los 25 grados de libertad con los que cuenta NAO. Además, permite realizar ajustes finos de diferentes movimientos cartesianos y permite que se realice una programación en Python.[11]

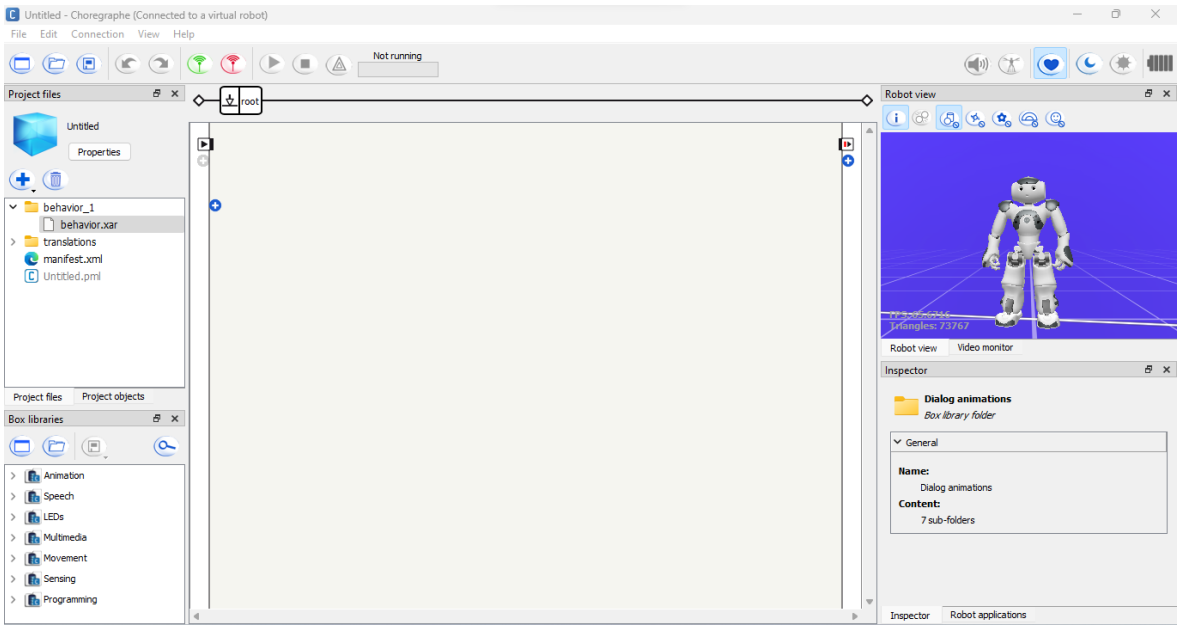

Figura 1.7 Choreagraphe [Autoría propia]

En la **Figura 1.7** se puede observar las diferentes librerías, acciones y la ventana de visualización para observar los comportamientos y aminaciones configuradas en un robot simulado o de forma directa en el robot real. Choreagraphe permite crear comportamientos complejos como bailar, interacción con personas, enviar correos electrónicos, etc.

1.4.5 MATLAB – SIMULINK

MATLAB es un software de programación utilizado por millones de ingenieros para analizar datos, desarrollar algoritmos, programas, modelos, etc. El método utilizado para desarrollar estas aplicaciones es con un lenguaje de programación expresado en arreglos y matrices.

Una de las herramientas otorgadas por MATLAB es Simulink que se trata de un entorno de programas a partir de diagramas de bloque que son utilizados para diseñar diferentes modelos multidominio. Simulink permite simular antes de funcionamiento, en las primeras etapas de desarrollo con la finalidad de conocer el comportamiento de los diferentes

sistemas o modelos que se están implementando en la industria, educación e investigación.[15]

Figura 1.8 MATLAB – Simulink

Simulink (Véase **Figura 1.8**) se usa ampliamente en sistemas de control automático y procesamiento digital de señales, así como el diseño basado en modelos, este software entrega varias ventajas como la interacción con MATLAB y la generación de scripts en diferentes lenguajes de programación, la comunicación con otros entornos de simulación, equipos, softwares, etc. [16]

UDP Send y **UDP Receive**, mismos que se pueden apreciar en la **Figura 1.9**, son bloques que permiten enviar y recibir datos entre las aplicaciones de Simulink y Python, esto mediante la configuración de una dirección IP y un puerto remoto.[16]

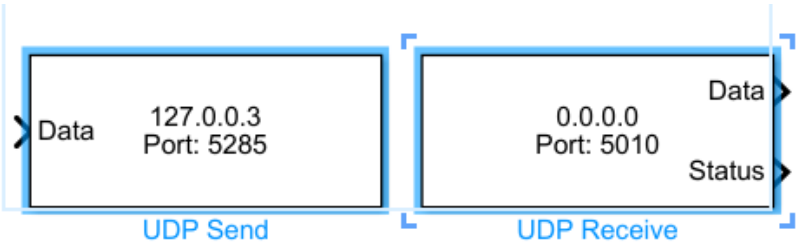

Figura 1.9 Bloques UDP Send y Receive [Autoría Propia]

1.4.6 SISTEMA DE COMUNICACIÓN

El sistema de comunicación utilizado es el protocolo UDP, el mismo esta implementado en simulink que con la ayuda de Python se traslada la información requerida al robot NAO.

1.4.6.1 COMUNICACIÓN UDP

User Datagram Protocol (UDP) es un protocolo de comunicación diseñado en el establecimiento de conexiones de baja latencia y tolerancia de pérdidas en sitios web. Una transmisión que no requiere funciones de recuperación y comprobación de errores. Este protocolo envía los datos al destino de forma continua, independientemente de la recepción de la información, si el datagrama se pierde en tránsito no se volverá enviar.[17]

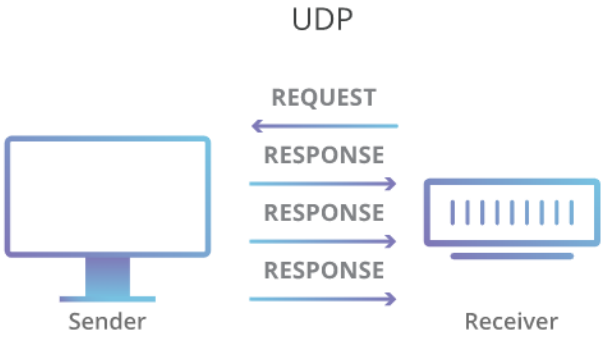

Figura 1.10 Comunicación por protocolo UDP [18]

El protocolo de comunicación UDP es ideal para aplicaciones que requieren comunicaciones en tiempo real, como transmisiones de redes multitarea, transmisiones en tiempo real, etc. En la **Figura 1.10** se puede observar que el protocolo UDP realiza un proceso de forma sencilla, envía datagramas directamente a un ordenador desde el destino, sin la necesidad de tener el mensaje de confirmación de llegada del datagrama.[18]

El datagrama del protocolo de comunicación UDP se detalla en la **Figura 1.11**.

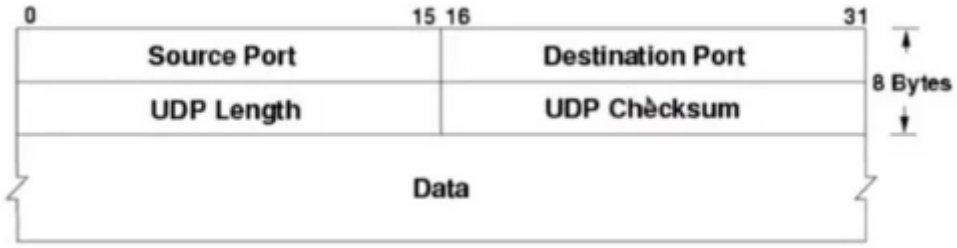

Figura 1.11 Datagrama del protocolo UDP [19]

Source Port o puerto fuente: hace referencia al punto de origen del datagrama, campo opcional que puede ser rellenado por ceros. Esta información es útil para el receptor con la finalidad de responder al paquete enviado.[20]

Destination Port o puerto de destino: receptor del datagrama en la cual se indica el servicio solicitado y la confirmación del proceso de recibir el mensaje.[20]

UDP Leght: es la longitud del datagrama UDP en Bytes, en el mismo se encuentran los datos del mensaje, el tamaño máximo del datagrama puede ser de 64 Kb.[20]

UDP Checksum: es la suma de comprobación o verificación y se utiliza para posibles errores durante el proceso de transmisión. Se tienen en cuenta la información de cabecera,

datos de usuario y pseudocabecera, la verificación de estos datos no es obligatoria y pueden ser llenados con ceros.[20]

1.4.7 SISTEMA DE TELEOPERACIÓN

La teleoperación comprende un conjunto de tecnologías que permite la operación a distancia de un dispositivo por el usuario. Por tanto, teleoperar es la acción de operar o gobernar a distancia determinado dispositivo por parte del usuario. [21]

El sistema de teleoperación está formado por diferentes elementos, como se puede observar en la **Figura 1.12.**

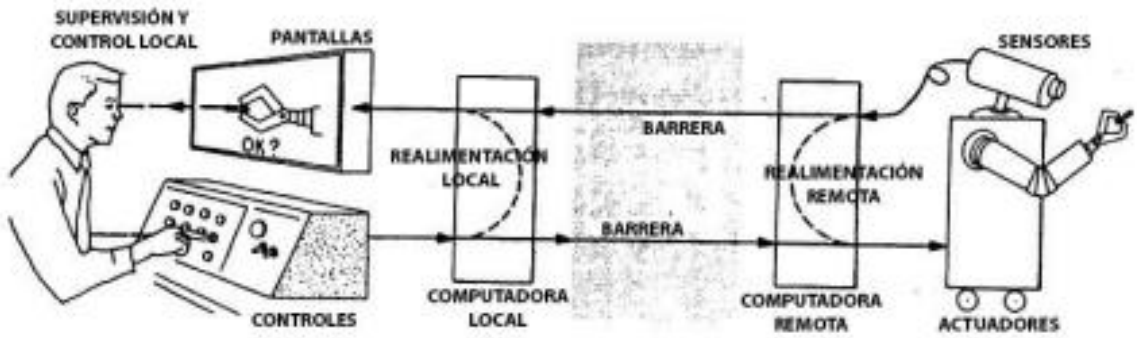

Figura 1.12 Elementos básicos de un sistema de teleoperación[22]

1.4.7.1 ELEMENTOS DE UN SISTEMA DE TELEOPERACIÓN

Los elementos de un sistema de teleoperación se describen a continuación.

Operador: es el usuario o ser humano que a distancia realiza el control o la operación, en este caso el manipulador, robot, etc. La acción que realiza puede ir desde el control continuo hasta la intervención intermitente. [23]

Dispositivo teleoperado: es la máquina que opera de forma remota, misma que es controlada por el operador. En este caso el dispositivo teleoperado es un Robot NAO V6.[23]

Interfaz: son dispositivos que permiten la interacción del usuario con el sistema de teleoperación. Puede ser monitores de vídeo o dispositivos que permiten enviar o recibir información. [23]

Control y canales comunicación: es el conjunto de dispositivos que modulan, trasmiten y adaptan las señales que transmiten entre la zona remota y local. Por lo general se contará con uno o varias unidades de procesamiento. [23]

Sensores: son dispositivos que recopilan información, en la zona local y remota, con la finalidad de ser utilizada por el interfaz y el control. [23]

1.4.7.2 APLICACIONES DE TELEOPERACIÓN

Las primeras aplicaciones de sistemas de teleoperación se dan en la industria nuclear, este ha sido el principal consumidor. Con el transcurso de los años se fue viendo la aplicabilidad en otros sectores, es especial en las industrias de servicio. A continuación, se enumeran los diferentes campos de aplicación más significantes. [24]

Aplicaciones en el espacio

Las principales actividades que realizan son la experimentación y exploración planetaria, mantenimiento y operación de satélites, construcción y mantenimiento de estaciones espaciales. Las razones de usar sistemas de manipulación remota son las siguientes:

Seguridad; las operaciones espaciales son de alto riesgo.

Costo; los equipos para seres humanos son más caros en relación con un sistema teleoperado.

Tiempo; este tipo de misiones son sin tripulación, con la finalidad de reducir tiempo para cumplir con los objetivos establecidos.

[24]

Aplicaciones en la industria nuclear

Las aplicaciones dentro de esta industria son mucho más numerosas, la utilidad de sistemas teleoperados radica en tratar y manipular sustancias radiactivas. Además, la movilización dentro de ambientes contaminados, sin la necesidad de exponer a peligros a los seres humanos. Las aplicaciones que se dan dentro de esta industria son en la operación y mantenimiento dentro de reactores, tuberías, instalaciones de combustible nuclear, descontaminación de instalaciones y la actuación en lugares donde se presenta algún desastre nuclear. [24]

Aplicaciones submarinas

En este caso los manipuladores están colocados en un vehículo submarino denominado Remote Operated Vehicle mismo que es teleoperado. La utilidad de este tipo de sistemas radica en la posibilidad de acceder a zonas y profundidades donde el acceso para una persona implica poner en peligro al submarinista. Las aplicaciones que se pueden dar son las siguientes: mantenimiento, inspección y construcción de instalaciones submarinas. Uno

de los ejemplos más relevantes es el caso de VICTOR un sistema francés de exploración submarina. [24]

Aplicaciones militares

En aplicaciones militares se usa tecnología de sistemas teleoperados, estos sistemas de monitorización remota son conocidos como Unmanned Air Vehicles. Uno de los ejemplos que se pueden mencionar es US Air Force Predator. Estos vehículos tienen un campo de aplicación extenso como la vigilancia, adquisición de objetivos militares, identificación de enemigos, reconocimiento entre otros. [24]

Aplicaciones médicas

En los últimos años se ha fortalecido de forma importante la aplicación de las tecnologías de teleoperación en el sector de la medicina. Desde el desarrollo de prótesis o dispositivos de asistencia a discapacitados hasta realizar telecirugía o telediagnóstico, se debe mencionar que estrictamente no pertenece al sector de la teleoperación. [24]

Se tiene un caso particular que llama la atención de la primera cirugía asistida por teleoperación. El sistema usado es ZEUS la parte de maestro en este caso del cirujano estaba situada de New York en Manhattan mientras que el paciente estaba en Satrasburgo Francia, la cirugía realizada fue una Colecistomía Laparoscópica fue todo un éxito. Se realizó a través de un canal privado de Internet, se usó el protocolo UDP/IP con un retardo promedio de 224 mseg. [24]

1.4.7.3 MÉTODOS DE CONTROL DE TELEOPERACIÓN

Los objetivos de un sistema de teleoperación es realizar el control de un equipo de forma renta y este sea robusto ante retardos, saturación de actuadores y errores que puede tener el operador.

Control Bilateral

En el control bilateral existe realimentación, en este caso el esclavo debe seguir los movimientos realizados por el maestro, de la misma forma la realimentación cinestésica de las fuerzas del esclavo al maestro en medio de comunicación con retardos. [24]

Control supervisado

El control supervisado permite tener autonomía sobre los manipuladores, mientras el operador monitorea y da comandos de alto nivel al manipulador. Este tipo de sistemas son más utilizados en la actualidad, este permite utilizar modelos virtuales y estimar parámetros. [24]

Control supervisado y coordinado

En el control coordinado el operador controla los actuadores, existe un lazo de control incluido en el sitio remoto, de cualquier manera, no hay autonomía en el elemento final, los lazos de control en el esclavo son usados cuando el operador puede controlar directamente el esclavo debido a los retardos de comunicación.

El control supervisado el manipular al esclavo puede hacer parte de las tareas autónomas, mientras el operador monitorea y da comandos de alto nivel para que el manipulador los ejecute, este tipo de sistemas son uno de los más usados y estudiados. [24]

1.4.7.4 INTERFACES

Los interfaces son importantes en el campo de la teleoperación, puesto que la interfase es el contacto indirecto del hombre con las máquinas. Una de las más tradicionales es las directas, el operador controla el manipulador o vehículo desde controladores de mano, como son joysticks o applets de java en el ordenador. Además, el operador tiene una retroalimentación visual por medio de cámaras en el sitio remoto. En la **Figura *1.13*** se puede observar un ejemplo de interfase directa. [22]

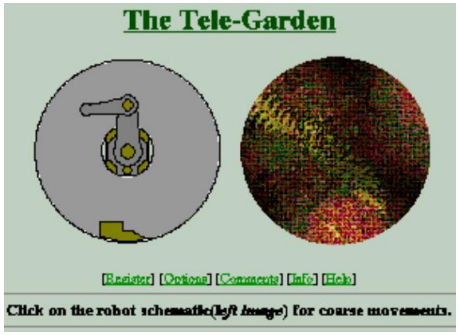

Figura 1.13 Telegarden, ejemplo de Interfase Directa [24]

En los interfaces que se tienen podemos encontrar dispositivos hápticos, estos dispositivos intentan replicar o mejorar la experiencia táctil de manipular y percibir un entorno real a través de dispositivos mecatrónicos y control por computadora, con la pantalla de información se genera un lazo de retroalimentación en el sistema de teleoperación. La funcionalidad de estos dispositivos depende de los grados de libertad que brinde el dispositivo, en el caso de un Joystick genérico para videojuegos controla hasta tres grados de libertad. En la **Figura 1.14** se puede observar el Joystick, equipo que tiene la capacidad de controlar hasta tres grados de libertad de un robot.

Figura 1.14 Joystick, controla hasta tres grados de libertad [4]

Los comandos para un robot también pueden ser dictados por voz del usuario, la interfaz de teleoperación es llamativa puesto que permite al usuario libertad de movimiento, sin embargo, al no contar un alto grado de control en sistemas complejos, esta función es utilizada en la activación de tareas menores. [4]

2. METODOLOGÍA

A continuación, se expone la metodología empleada en el Trabajo de Integración Curricular, en la misma se muestra de manera explícita todo el trabajo realizado y el procedimiento que se elaboró en la investigación.

La información recopilada de fuentes primarias y secundarias, las fuentes primarias están constituidas por: libros, artículos científicos y revistas técnicas, mientras que las fuentes secundarias están conformadas por: manuales o guías de usuario de Robot NAO mismos que se encuentran en sitios web, trabajos previos e informes técnicos.

2.1 ARQUITECTURA DEL SISTEMA DE TELEOPERACIÓN

El sistema de teleoperación implementado en el presente trabajo presenta una arquitectura de una estación local, un servidor y una estación remota. Como se puede observar en la **Figura 2.1**. en la misma se detalla cada uno de los elementos que componen la arquitectura implementada. Este sistema se puede conectar a través de 2 formas: conexión inalámbrica y conexión por cable.

El sistema de comunicación puede ser por conexión inalámbrica (Wifi) o conexión por cable (ethernet), si se utiliza el protocolo por conexión Wifi puede llegar a ser una desventaja por la pérdida de información y la velocidad de la comunicación, puestos estos parámetros estarían sujetos a la disponibilidad y velocidad de la red Wifi. En el trabajo desarrollado se implementa la comunicación entre las dos estaciones mediante un medio físico.

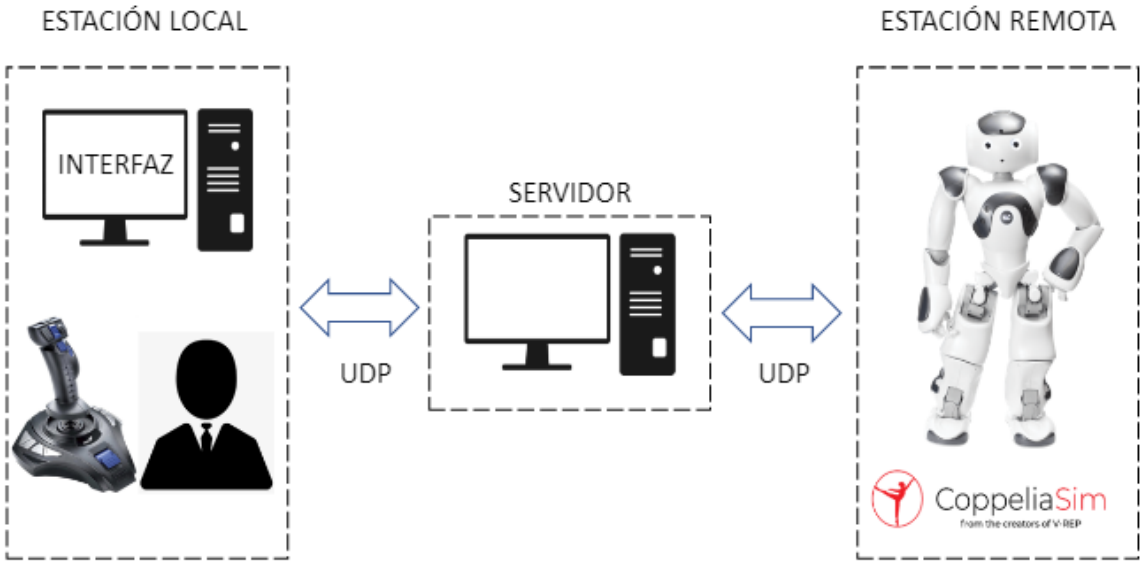

Figura 2.1 Arquitectura del sistema de Teleoperación implementado [Autoría Propia]

2.1.1 ESTACIÓN LOCAL

En la estación local el usuario es el encargado de ingresar las diferentes órdenes para el robot NAO, a través del joystick Metalstrit 3D. Sin embargo, el robot opera en un entorno bidimensional, es decir en los ejes X y Y, con la finalidad de controlar la posición y el ángulo de operación del robot. Estos datos ingresados a través del joystick son recopilados por la interfaz desarrollada y a través del sistema de comunicación implementado en simulink con el protocolo UDP, está información se envía a la PC de la estación remota. [25]

Para establecer la comunicación entre las dos computadoras se debe realizar configuraciones, con la finalidad que la dirección IP de la PC local y PC remota se encuentren dentro de la misma puerta de enlace, es decir, las dos direcciones IP deben estar dentro de una misma red. En la **Figura 2.2** se puede observar la dirección IP configurada y la puerta de enlace para la estación local.

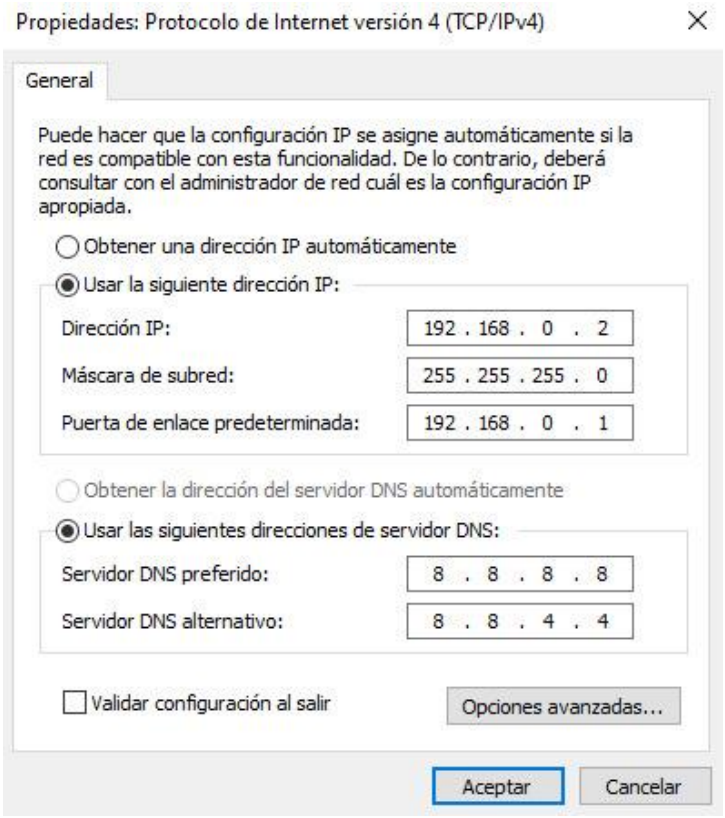

Figura 2.2 Configuración de la dirección IP para la estación local [Autoría Propia]

2.1.1.1 JOYSTICK METALSTRIKE 3D

Con la finalidad de obtener la información requerida por el usuario, se hace uso del joystick Metalstrike 3D y la implementación de bloques en Simulink para obtener la información ingresada por el usuario.[25]

Pilot Joystick All

Esta librería proporciona una interfaz de joystick piloto para la plataforma de Windows, cuenta con canales analógicos X, Y, Z, R, U y V. Además, tiene valores fijos a través de sus botones. Los valores de las salidas en cada uno de los canales son [-1,1]. Este bloque tiene la capacidad de entregar un Point of view, es decir un valor en grados de este canal. En la **Figura *2.3*** se puede observar el bloque Pilot Joystick All, librería utilizada para recopilar la información requerida por el operador, este es el encargado de entregar los datos analógicos, valores de los botones y el ángulo. [26]

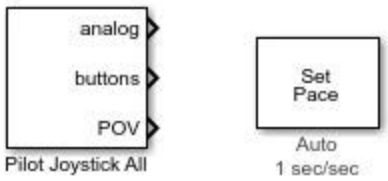

Figura 2.3 Bloques utilizados en Simulink para obtener la información ingresada por el usuario a través del Joystick [Autoría Propia]

Set Pace

El bloque 2 de la **Figura 2.3** permite ejecutar la simulación a un ritmo especifico, para que las animaciones conectadas sean estéticamente agradables. El ritmo es la relación entre segundos de simulación y segundos de reloj. [27]

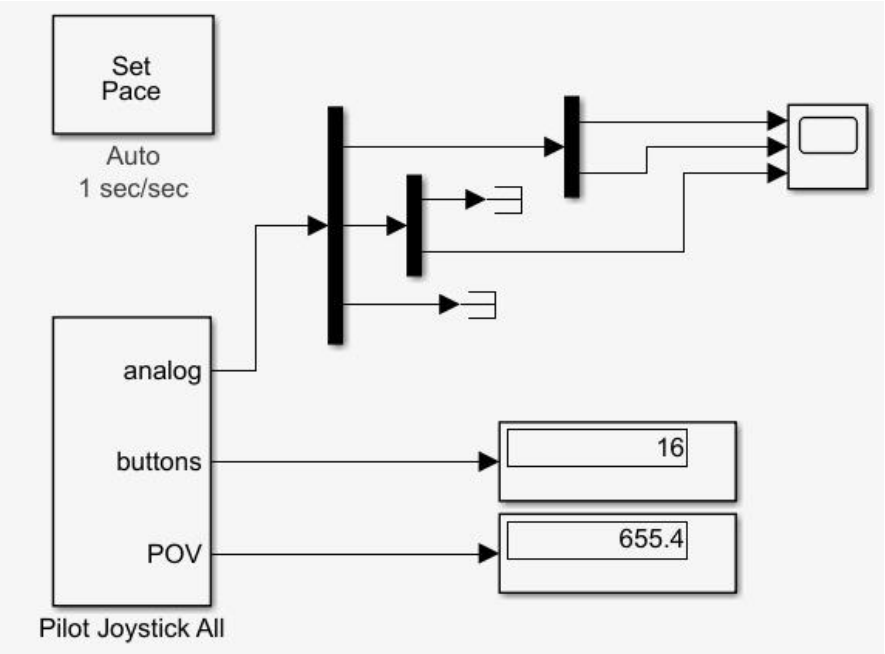

Figura 2.4 Diagrama de bloques implementado para verificar los datos ingresados a través del joystick [Autoría Propia]

En la **Figura 2.4** se puede observar el diagrama de bloques implementado en Simulink, para visualizar el comportamiento del joystick al momento de ingresar información por parte del usuario en la estación local.

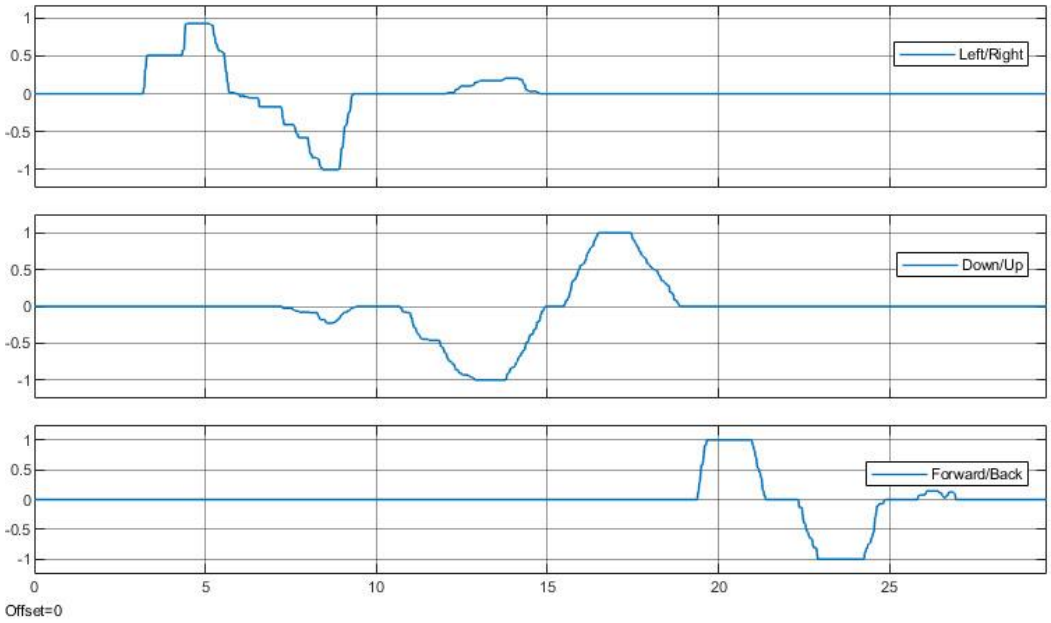

Figura 2.5 Lectura de datos ingresados por el usuario mediante el joystick en Simulink [Autoría Propia]

Los datos analógicos ingresados por el usuario son graficados en Simulink, en este escenario se tiene tres diferentes datos, cada uno de los comportamientos requeridos se visualizan entre los valores -1 y 1 como valores máximos y mínimos, mismos que se puede observar en la **Figura 2.5**. en este caso los datos ingresados no son simultáneos.

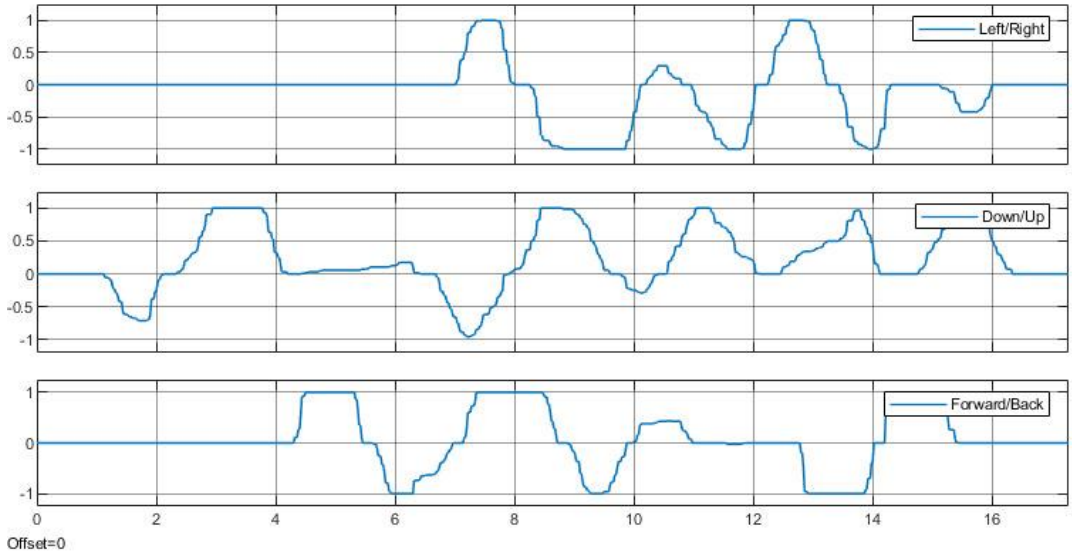

Figura 2.6 Lectura de datos simultáneos ingresados por el usuario mediante el joystick en Simulink [Autoría Propia]

En la **Figura 2.6** se puede observar que los datos ingresados por el usuario pueden ser registrados de forma simultánea. Los valores visualizados varían entre -1 y 1 que son los valores mínimos y máximos respectivamente. Estos datos son ingresados en la estación local mediante el uso del joystick.

Los datos recibidos en los dos casos son:

- Derecha – Izquierda
- Arriba – Abajo
- Adelante – Atrás

2.1.1.2 SISTEMA DE COMUNICACIÓN ESTACIÓN LOCAL

La estación local posee bloques de comunicación de envío y recepción de información (véase **Figura 2.7**), en cada uno de ellos se realiza las configuraciones de las direcciones IP y puertos tanto para la estación local y la estación remota.

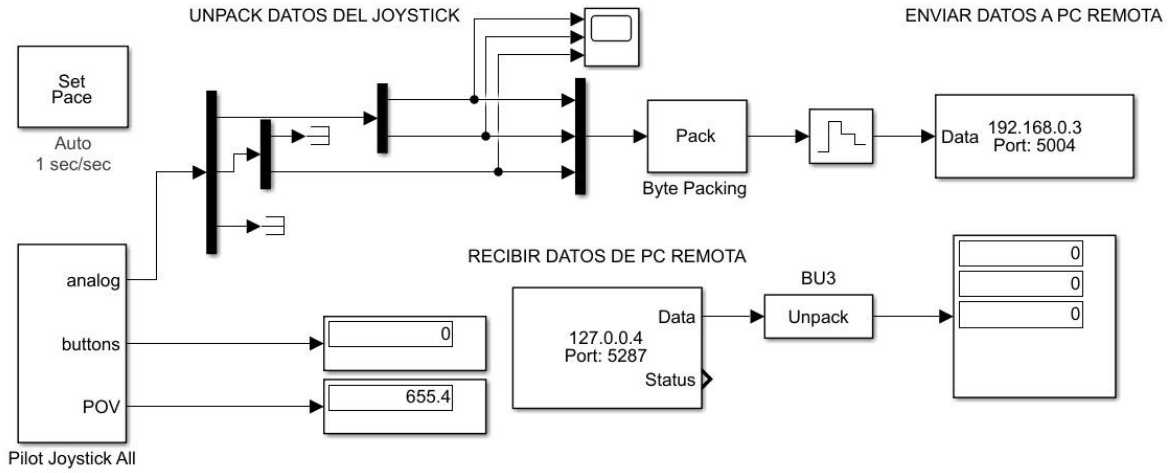

Figura 2.7 Diagrama de Bloques de la estación Local [Autoría Propia]

En la estación local se cuenta con el bloque UDP send y bloque UDP receive, la configuración de cada uno de bloques se puede observar en la **Figura 2.8** y la **Figura 2.9** respectivamente. En el caso del bloque send para enviar la información ingresada por el usuario mediante el joystick MetalStrike y en el caso del bloque UDP receive para la retroalimentación del usuario con información proveniente de la estación remota.

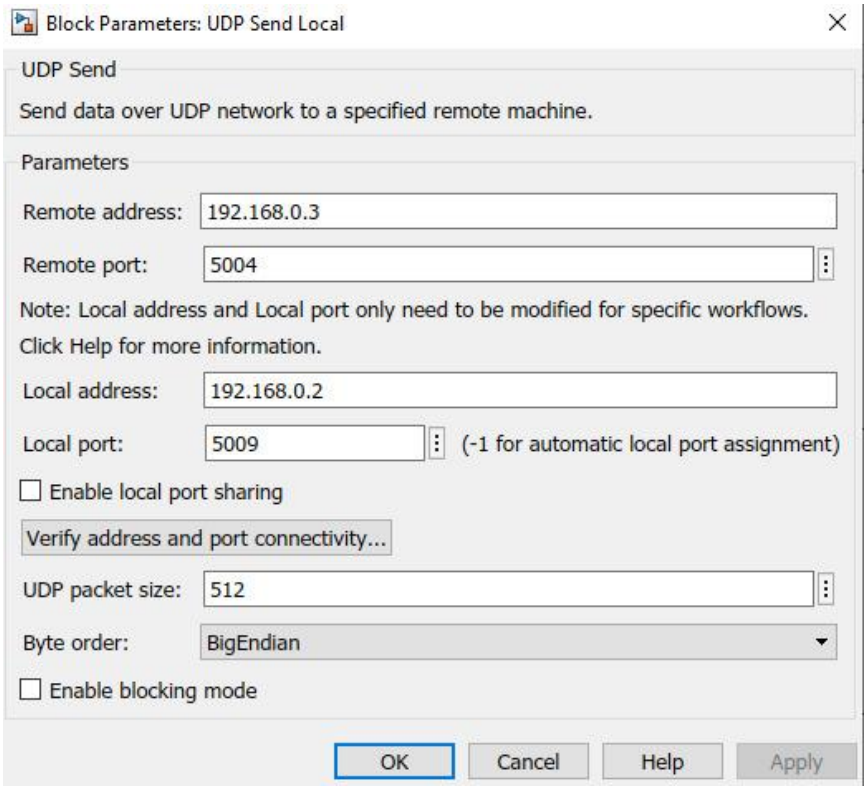

Figura 2.8 Configuración del Bloque UDP Send de la Estación Local [Autoría Propia]

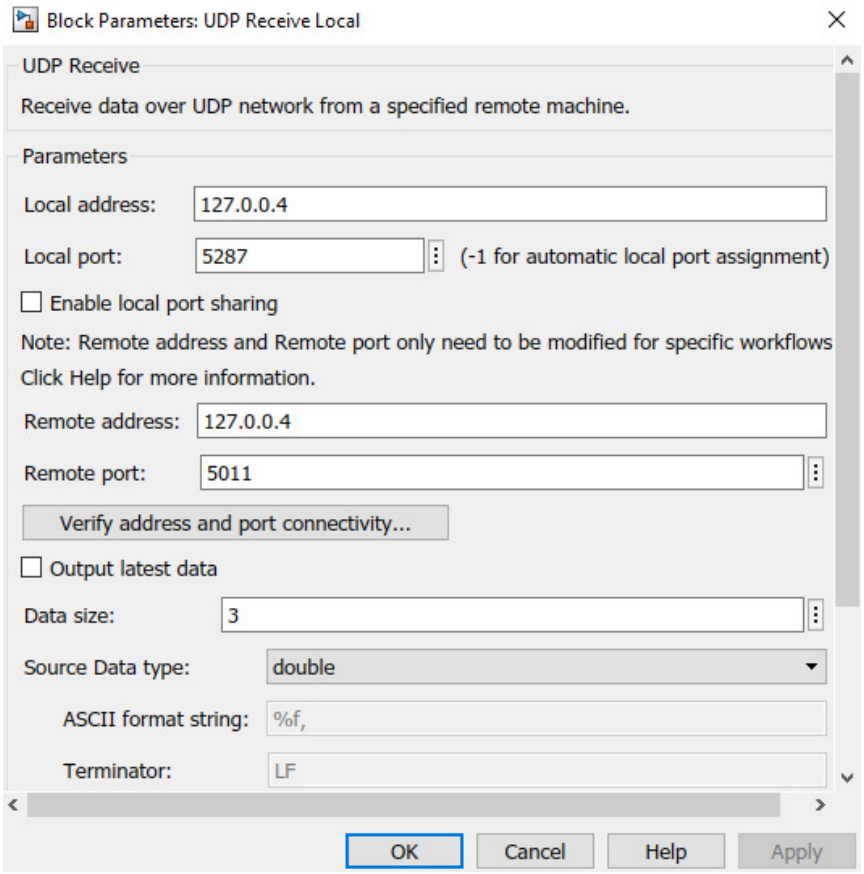

Figura 2.9 Configuración del Bloque UDP Receive de la Estación Local [Autoría Propia]

2.1.2 SERVIDOR - ESTACIÓN REMOTA

En la estación remota se debe realizar las configuraciones respectivas para la comunicación entre computadoras, para la misma se debe establecer una dirección IP y una puerta de enlace, misma que debe ser la misma red configurada para la computadora de la estación local como se puede observar en la **Figura 2.10**.

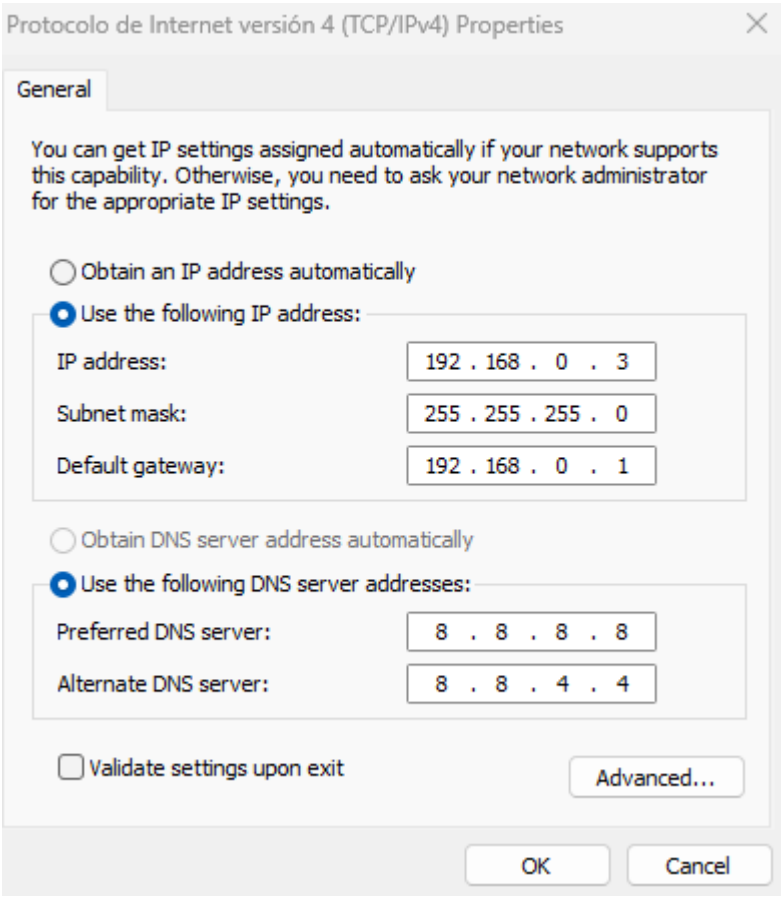

Figura 2.10 Configuración de la dirección IP para la estación remota [Autoría Propia]

2.1.2.1 SISSTEMA DE COMUNICACIÓN ESTACIÓN REMOTA

En la estación remota se tiene una arquitectura similar al sistema implementado en la estación local, con la diferencia que se debe tener dos sistemas, el uno para la recepción de información proveniente de la estación local y el segundo para enviar la información recibida al simulador en el cual se encuentra el robot NAO, es decir a CoppeliaSim Edu y la información que este debe enviar a la estación local. En la **Figura 2.11** se puede observar el sistema implementado, en el cual se tiene bloques UDP Send, bloques UDP Receive, bloques de empaquetados y desempaquetados de información.

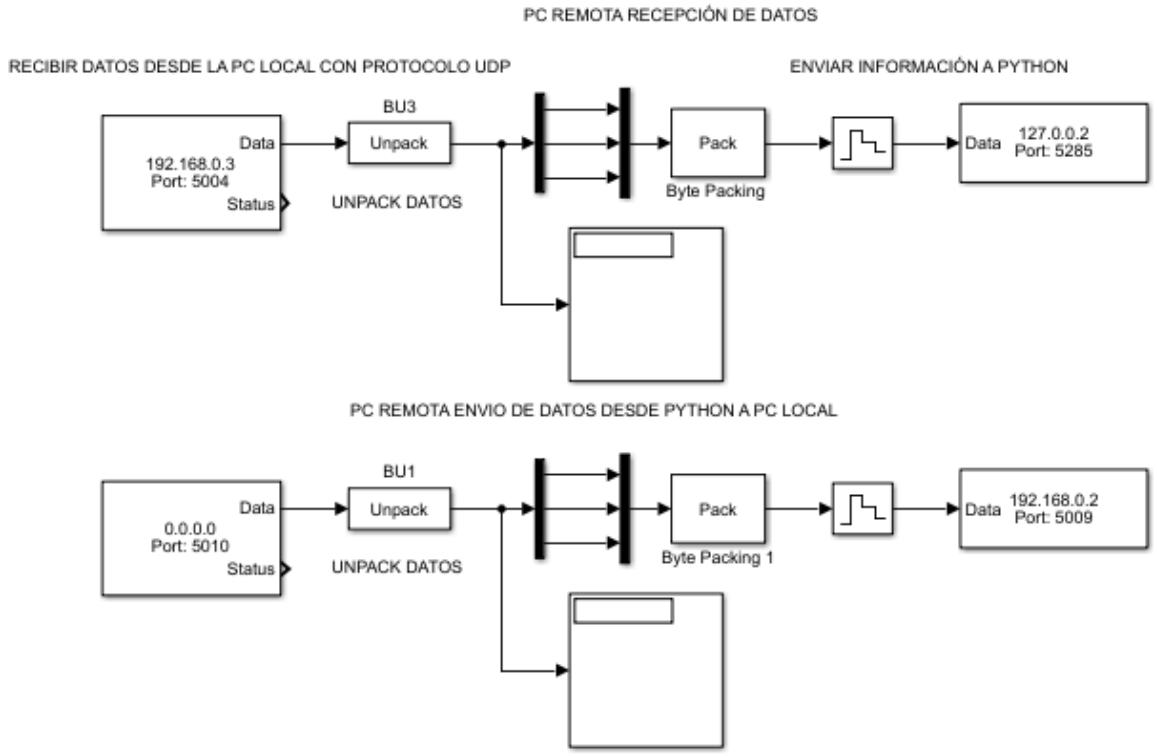

Figura 2.11 Sistema de Comunicación entre la PC Local, PC Remota y el Simulador CoppeliaSim Edu [Autoría Propia]

Tomando como referencia las configuraciones realizadas para la estación local en las **Figura 2.8** y **Figura 2.9** se debe realizar las configuraciones de direcciones IP y puertos para la estación local y estación remota. Además, de la configuración de estos bloques para la recepción y envió de información entre la PC Remota y Python que se comunica con el programa de simulación, es decir con CoppeliaSim Edu.

En la **Figura 2.11** se encuentra el bloque Zero Order Hold, este me permite realizar barridos para obtener la información cada determinado tiempo, con la finalidad de contar con un tiempo de muestreo bajo. El tiempo de muestreo adecuado o recomendado es de 20 milisegundos. Sin embrago, al realizar pruebas este tiempo se modificó y se establece en 50 milisegundos, en donde se tiene un sistema de comunicación más confiable y se evita que los retardos sean mayores.

2.1.3 CONFIABILIDAD DE LOS DATOS

Utilizar protocolo UDP para la comunicación entre la estación local y la estación remota, limita en conocer si la información que es receptada por la PC remota desde la PC local es la correcta, no existe una retroalimentación de la información receptada. Esto puede ocasionar la perdida de datagramas al momento de establecer la comunicación. Es una de

las principales desventajas de la comunicación UDP. Sin embargo, la velocidad es una de las ventajas de este tipo de comunicación por lo cual se hace común utilizar en operaciones en tiempo real. Con la finalidad de observar la fiabilidad de la información se plantea escenarios donde se pueda apreciar los datos enviados y receptados en la estación local y estación remota respectivamente.

2.1.3.1 ESCENARIO 1

2.1.3.1.1 Velocidad X

En la **Figura 2.12** se puede apreciar la velocidad para el eje X, los datos son ingresados por el joystick en la estación local y se recibe en la estación remota, en la cual se puede observar que el valor de 0.190497 representado en el eje Y tiene un tiempo de 18.81 segundos para la señal en la PC de maestro y 19.23 segundos para la señal en la PC de esclavo, generando una diferencia de 0.42 segundos, a este valor se le considera el tiempo de retardo en la comunicación entre la estación local y estación remota.

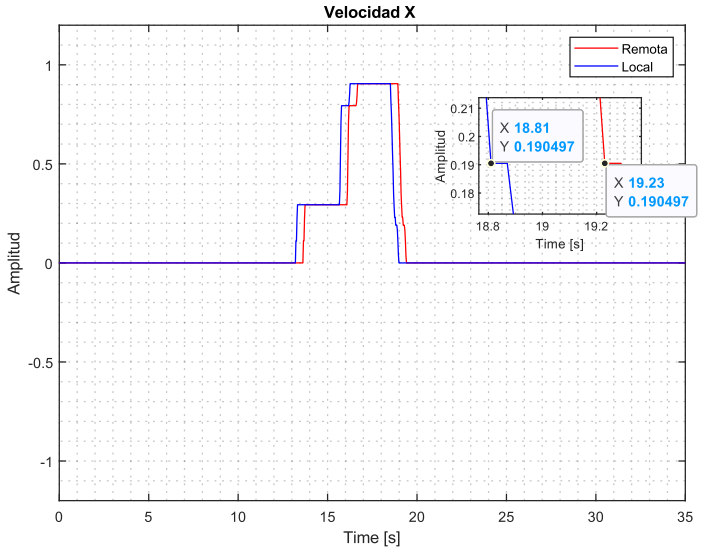

Figura 2.12 Velocidad en X, dato ingresado por el Joystick [Autoría Propia]

2.1.3.1.2 Velocidad Y

En la **Figura 2.13** se puede apreciar la velocidad para el eje Y, los datos recopilados y graficados son ingresados por el joystick en la estación local y se recibe en la estación remota, en donde se puede hacer una comparación entre estos datos, en la estación local para el punto 0.174627 se tiene un tiempo de 37.59 segundos en la PC maestro, el valor de este dato en la estación remota se tiene un valor de 38.01 segundos, generando una diferencia de 0.42 segundos, valor considerado como el tiempo de retardo entre la comunicación entre las dos estaciones.

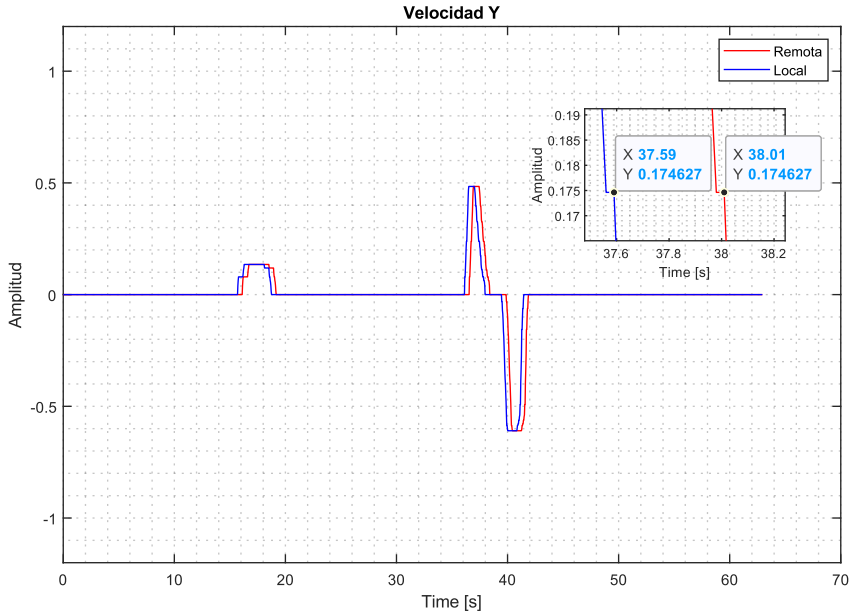

Figura 2.13 Velocidad en Y, dato ingresado por el Joystick [Autoría Propia]

2.1.3.1.3 Velocidad Theta

En la **Figura 2.14** se puede apreciar los datos ingresados para el joystick para Theta, mismos que son graficados y se puede evidenciar un determinado retraso entre la señal obtenida en la PC remota y la PC local. En este caso para un valor 0.595264 en la PC de maestro se tiene un tiempo de 47.97 segundos, en la PC esclavo se tiene un tiempo de 38.01 segundos, al momento de obtener la diferencia se tiene un valor de 0.42 segundos que será el retardo de la comunicación entre las dos estaciones.

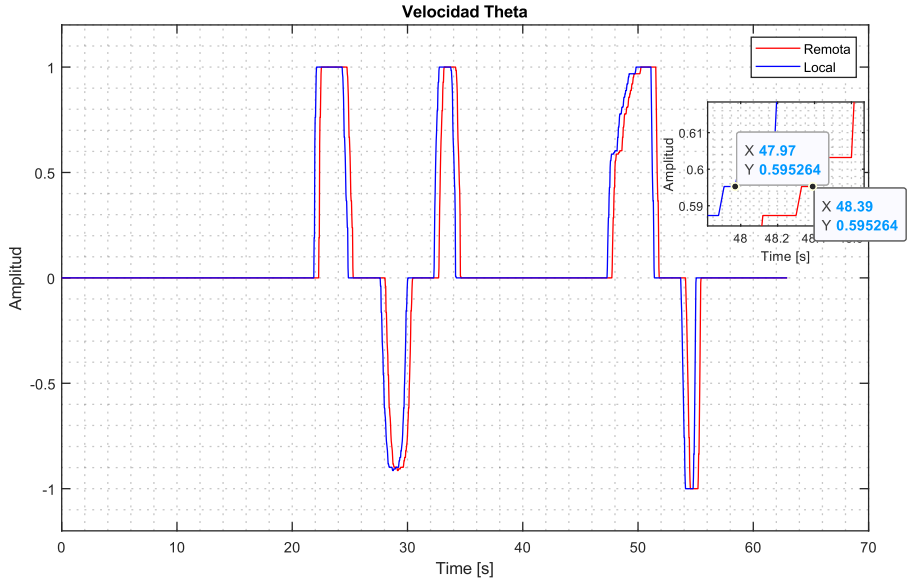

Figura 2.14 Velocidad de Theta, dato ingresado por el Joystick [Autoría Propia]

En el primer escenario planteado se puede observar que los valores ingresados a través del joystick, para llegar desde la PC de la estación local a la PC de la estación remota se tiene un retado en la comunicación de0.42 segundos, mismos que se pueden apreciar para cada una de las ordenes en las **Figura *2.12***, **Figura *2.13*** y **Figura *2.14*** con las ampliaciones realizadas para determinados valores que pueden llegar a tomar, es decir valores entre -1 y 1.

2.1.3.2 ESCENARIO 2

2.1.3.2.1 Velocidad X

En la **Figura *2.15*** se puede observar la velocidad para el eje X, estos datos son ingresados a través del joystick, en este caso a diferencia del primer escenario las ordenes en un tramo del tiempo son inmediatos, y no se puede establecer una diferencia a simple vista, después de un tiempo se puede observar una diferencia, en este caso la diferencia entre la señal obtenida en la estación local y la estación remota para un mismo valor se tiene un retardo de 0.39 segundos. Con la finalidad de observar la diferencia se realiza una ampliación misma que se puede observar en la figura.

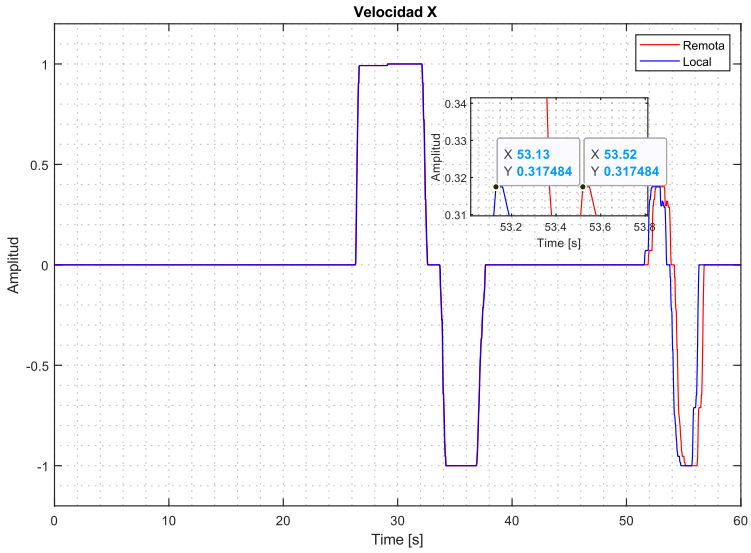

Figura 2.15 Velocidad en X, dato ingresado a través del Joystick [Autoría Propia]

2.1.3.2.2 Velocidad Y

La velocidad en Y en la estación local y en la estación remota se puede observar en la **Figura 2.16**, se puede observar un retardo en la comunicación para un dato de 0.39 segundos entre las dos estaciones. Se puede observar que el dato en la estación remota tiene un retardo en relación con el dato de la estación local, esta diferencia de tiempo se visualiza de forma fácil.

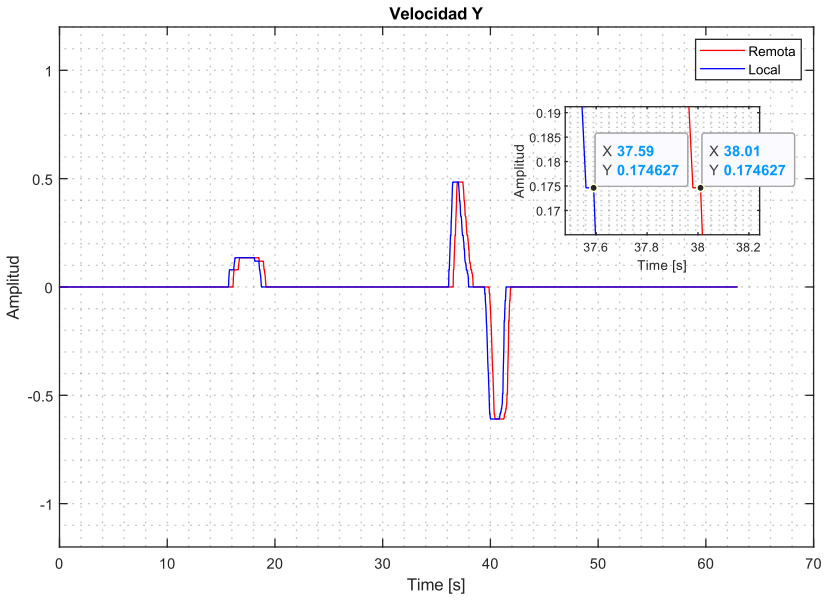

Figura 2.16 Velocidad en Y, dato ingresado a través del Joystick [Autoría Propia]

2.1.3.2.3 Velocidad Theta

En la **Figura *2.17*** se puede observar la velocidad ingresada para Theta, se realiza una ampliación de un punto especifico, en este caso para un valor negativo, se tiene una diferencia de 0.39 segundos entre la estación local y la estación remota, mismo que se puede verificar en la ampliación realizada.

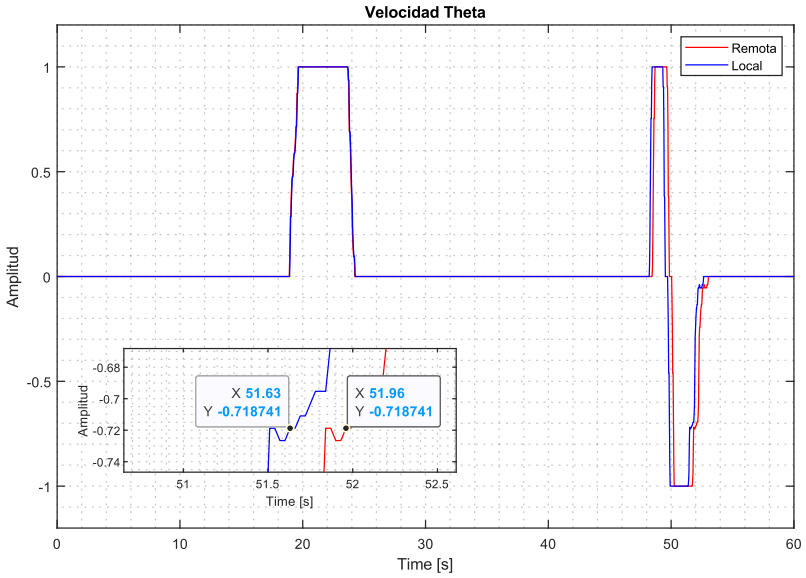

Figura 2.17 Velocidad Theta, dato ingresado por el Joystick [Autoría Propia]

En el segundo escenario presentado se puede visualizar los datos ingresados para los ejes X, Y y Theta, en donde en un tramo del tiempo se tiene que la información es inmediata y

en otro tramo se cuenta con un retardo, en este caso de 0.39 segundos entre estaciones. Se puede observar que se tiene un determinado retraso en llegar la información de la PC maestro a la PC local, pero no existe perdida de información, esto se puede observar en las **Figura *2.15***, **Figura *2.16*** y **Figura *2.17***. Estos datos se obtienen al momento de establecer un tiempo de muestreo de 20 milisegundos en Simulink y trabajar con el solver discreto.

2.1.3.3 ESCENARIO 3

2.1.3.3.1 Velocidad X

La velocidad en X en la estación local y la estación remota se pueden observar en la **Figura *2.18***, en la misma se puede observar que los datos recopilados en las dos estaciones no se tiene una diferencia a simple vista, por lo cual se realiza una ampliación en un punto especifico, de la misma se toma un valor para las dos señales y se observar el tiempo. Para la velocidad en X se tiene el valor de 0.531785, el tiempo en la estación local es de 10.05 segundos, para la estación remota se tiene un tiempo de 10.08 segundos, dando un retardo en la comunicación de 0.03 segundos.

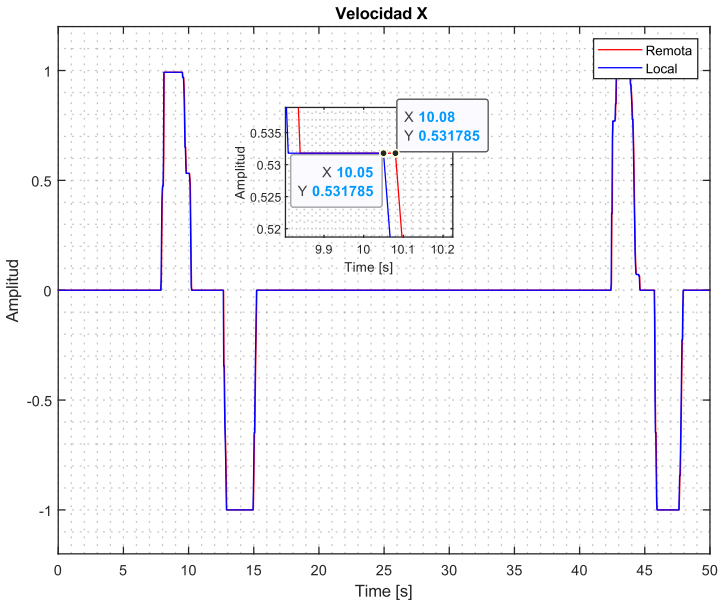

Figura 2.18 Velocidad en X, dato ingresado a través del Joystick [Autoría Propia]

2.1.3.3.2 Velocidad Y

En la **Figura *2.19*** se puede observar los datos de la velocidad en Y para la estación local y la estación remota, no se puede observar el retardo entre las dos señales a simple vista, por lo cual se realiza una ampliación para un determinado valor, en la misma se puede observar que el retardo para la comunicación es de 0.03 segundos.

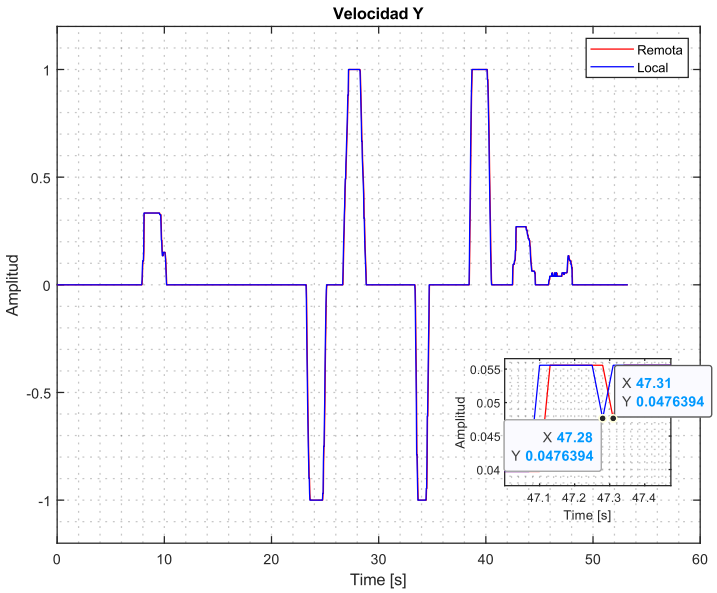

Figura 2.19 Velocidad en Y, dato obtenido a través del Joystick [Autoría Propia]

2.1.3.3.3 Velocidad Theta

En la **Figura *2.20*** se puede observar los datos para la velocidad en Theta, el retardo de la comunicación entre las dos estaciones no se puede apreciar, para el mismo se realiza una ampliación en un determinado valor, en la misma se puede observar que el retardo se tiene de 0.03 segundos.

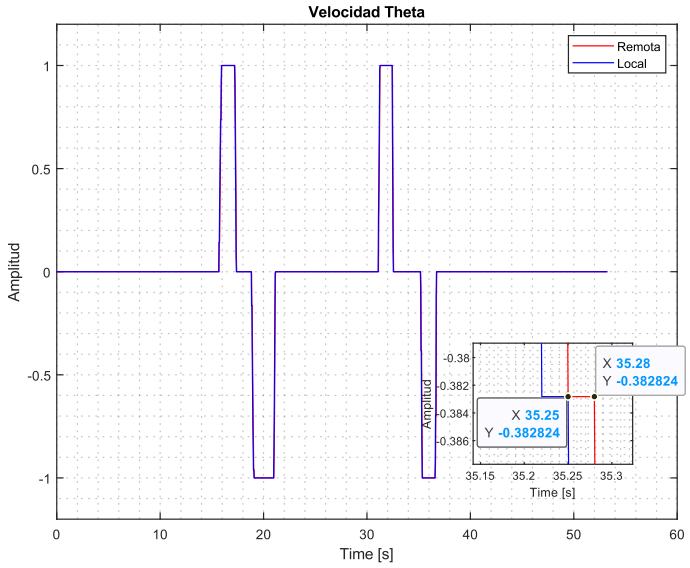

Figura 2.20 Velocidad en Theta, dato obtenido a través del Joystick [Autoría Propia]

En las **Figura *2.18***, **Figura *2.19*** y **Figura *2.20*** se puede observar que el retardo de la comunicación entre las dos estaciones es más pequeño a diferencia de los dos escenarios

planteados anteriormente. En este caso a simple vista no se puede observar el retardo para lo cual se realiza una ampliación en determinados valores y se observa que se tiene un retardo de 0.03 segundos, mismos que se obtienen con la configuración de un tiempo de muestreo de 30 milisegundos que se encuentran configurados en MATLAB/Simulink y con un solver de Fixed Step.

2.2 ENTORNO DE SIMULACIÓN COPPELIASIM EDU PARA EL ROBOT HUMANOIDE NAO

2.2.1 MARCO DE REFERENCIA DE COPPELIASIM CON RESPECTO AL ROBOT NAO V6

El robot NAO realizará diferentes acciones de movimiento, para lo cual se debe tener un entorno de simulación de CoppeliaSim Edu, el mismo es importante conocer el ambiente que brinda el programa para que el robot NAO interactúe, en este caso tiene un marco de referencia característico, el cuál es una cuadrícula subdividida por cuadros más pequeños, el entorno tiene ejes de referencia con un plano cartesiano en donde se tiene el eje X que se encuentra a lo largo de una línea horizontal y en el sentido vertical se tiene el eje Y, en el caso del eje Z se está proyectando hacia fuera de la pantalla o perpendicular a los ejes (X, Y). Con los detalles brindados es fácil ubicar el origen de coordenadas (0,0) en el centro del plano cartesiano. [28]

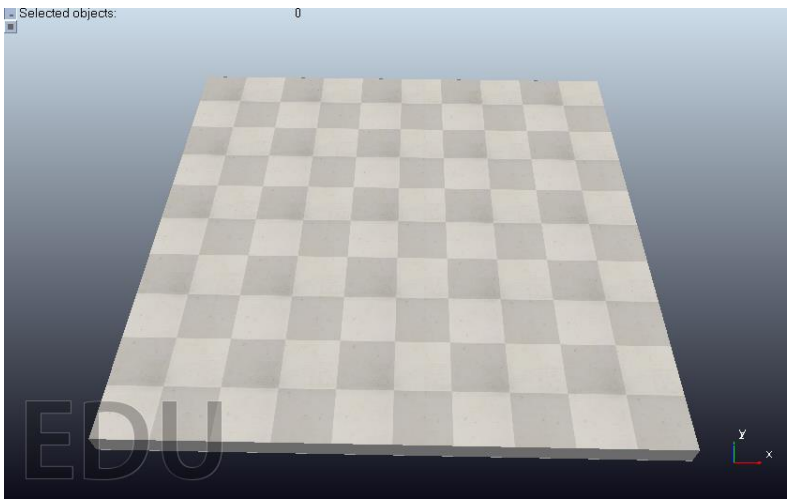

Figura 2.21 Entorno de simulación de CoppeliaSim Edu [Autoría Propia]

En la **Figura 2.21** se puede observar el entorno de simulación de CoppeliaSim Edu, en donde se puede observar los diferentes ejes, que permiten tener una referencia al usuario del lugar de donde se va a ubicar el robot NAO. Cada una de las cuadriculas de este

ambiente tiene las siguientes dimensiones de 50x50 cm dando resultado de 2.5 metros de largo y 2.5 metros de ancho. Con estos valores se tiene una idea de la cantidad de espacio para las diferentes acciones que puede realizar el Robot NAO. [28]

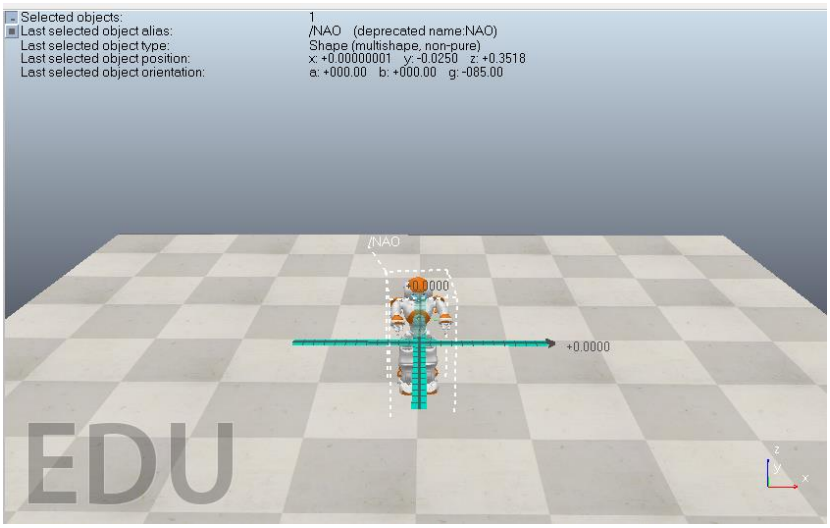

Figura 2.22 NAO V6 en el ambiente de simulación CoppeliaSim Edu [Autoría Propia]

En la **Figura 2.22** se puede observar el ambiente de simulación CoppeliaSim Edu en el cual se tiene la ubicación del robot NAO, en este caso este se encuentra ubicado en el origen en las coordenadas (0,0), es decir se encuentra en el centro de los cuatro cuadrantes. Este será el punto de inicio del robot, el cual se va a mover en diferentes direcciones o realizar las actividades solicitadas por el usuario.

Para que el robot inicie con el movimiento se debe tener en consideración que este se va a mover a través de ordenes ingresadas por parte del usuario, en este caso a través de un joystick. Además, se va a contar con una retroalimentación. El robot se va a mover en función de las ordenes requeridas o el seguimiento de trayectos establecidos, los mismos que son monitoreados por el usuario a través de la interfaz desarrollada.

El software de simulación CoppeliaSim Edu, cuenta con un modelo estándar del robot NAO y con diferentes accesorios para acondicionar diferentes escenarios en donde el robot va a realizar las actividades solicitadas por el usuario.

2.2.2 ODOMETRÍA DEL ROBOT NAO EN EL ENTORNO DE SIMULACIÓN COPPELIASIM EDU

El software de simulación CoppeliaSim Edu cuenta con un marco de referencia general (World Frame) mediante el cual se puede conocer la posición de los diferentes objetos dentro del espacio de trabajo. El marco de referencia se encuentra divido en 4 diferentes

cuadrantes, se relaciona con el plano cartesiano con los ejes (X, Y) tanto para valores positivos y valores negativos. Los detalles se pueden encontrar en la **Tabla 2.1**. [29]

Tabla 2.1 División de Cuadrantes del Marco de Referencia de CoppeliaSim Edu

Cuadrante	Eje X	Eje Y
I	Positivo (+)	Positivo (+)
II	Negativo (-)	Positivo (+)
III	Negativo (-)	Negativo (-)
IV	Positivo (+)	Negativo (-)

En la **Figura 2.23** se puede observar la distribución de los diferentes cuadrantes en el espacio de trabajo, con la finalidad de obtener la posición y orientación real del robot humanoide NAO V6 en CoppeliaSim Edu, para este se hace uso de funciones Sim.GetObjectPosition() y Sim.GetObjectAngle(). Con la información obtenida es posible orientar el robot y que este cumpla las funciones requeridas por el usuario.

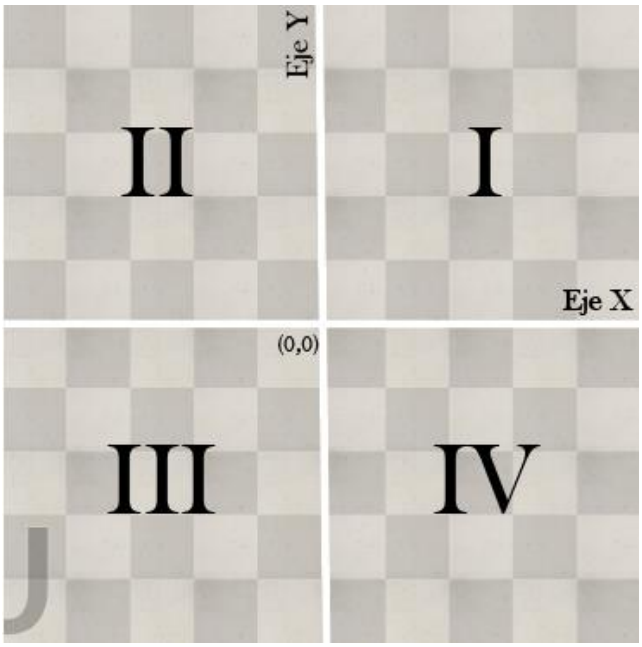

Figura 2.23 Cuadrantes del espacio de trabajo en CoppeliaSim [Autoría Propia]

Para conocer el desplazamiento del NAO dentro de los cuadrantes es necesario tomar en cuenta ciertos parámetros, entre ellos el valor de la posición real y la posición deseada, los valores de la posición deseada son ingresados por el usuario a través del joystick. Con la

finalidad de conocer el ángulo se toma como punto de referencia el eje X positivo y con ello se define el ángulo Theta para cada uno de los cuadrantes, como se detallan en las ecuaciones (*2.1*), (*2.2*), (*2.3*) y (*2.4*).

Cuadrante I

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) \quad (2.1)$$

Cuadrante II

$$\theta = \pi + \tan^{-1}\left(\frac{y}{-x}\right) \quad (2.2)$$

Cuadrante III

$$\theta = -\pi + \tan^{-1}\left(\frac{-y}{-x}\right) \quad (2.3)$$

Cuadrante IV

$$\theta = \tan^{-1}\left(\frac{-y}{x}\right) \quad (2.4)$$

Para obtener la distancia se debe considerar los valores de la posición deseada y posición real, la diferencia de estos dos valores permite conocer la distancia que falta para que el robot llegue al destino requerido por el usuario, se debe considerar que el error o el módulo de la distancia se aproxime a cero el robot va a llegar a la posición deseada, mismo que se detalla en la ecuación (*2.5*).

Módulo

$$D = \sqrt{(Xdeseada - Xreal)^2 + (Ydeseada - Yreal)^2} \quad (2.5)$$

2.2.3 COMUNICACIÓN DE DATOS ENTRE COPPELIASIM – PYTHON EL ROBOT NAO

El robot NAO tendrá que interactuar con el software de CoppeliaSim Edu, los comandos y algoritmos de control son implementados a través de otros programas entre ellos Python, por lo tanto, es importante enviar y recibir datos necesarios para efectuar la acción de control de Python – CoppeliaSim.

En este caso se hace uso de un cubo con la función de implementar en este, un código que permite realizar el enlace de datos entre los dos programas.

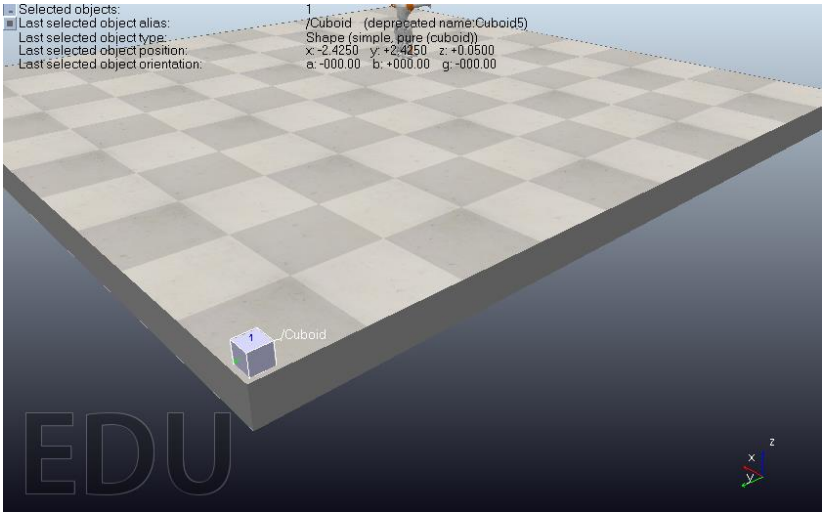

Figura 2.24 Cubo en el entorno de simulación CoppeliaSim Edu, mismo que alberga el código para la comunicación entre Python – CoppeliaSim Edu [Autoría Propia]

Dentro del ambiente de simulación de CoppeliaSim Edu se debe colocar un cubo, de ser posible este debe ser poco visible dentro de los cuadrantes donde el robot va a cumplir las diferentes actividades requeridas por el usuario, como se puede observar en la **Figura 2.24**. Este cubo alberga el código que permite realizar la comunicación entre Python – CoppeliaSim Edu. Las líneas de código que se muestran en la **Figura 2.25** están implementadas dentro de la estructura del cuboide de V-rep y con esto se tendrá creado una API remota.

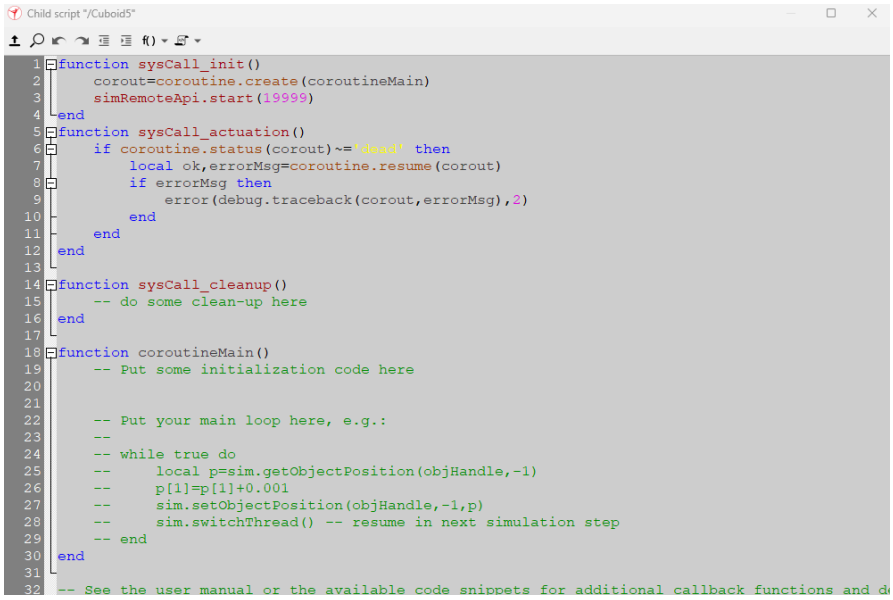

Figura 2.25 Líneas de código implementadas dentro del cuboide [Autoría Propia]

En Python se debe agregar las líneas de código que se muestran en la **Figura *2.26***, este inicia la comunicación entre los programas de Python y Coppelia mediante la dirección IP y el número de puerto. Esto permitirá enviar y recibir datos, mismos que son necesarios para cumplir las funciones requeridas por el usuario.

```
#COMUNICACION PYTHON - COPPELIASIM EDU -- CONEXIÓN V-REP

    clientID = sim.simxStart('127.0.0.1', 19997, True, True, 5000, 5)
```

Figura 2.26 Código de conexión entre Python y CoppeliaSim Edu [Autoría Propia]

2.2.4 COMANDOS DE EJECUCIÓN DE COPPELIASIM - PYTHON PARA EL ROBOT NAO

Después de realizar la comunicación entre los programas se obtiene una gran lista de comandos útiles que ofrece CoppeliaSim Edu en Python, para verificar que los programas están conectados es necesario ejecutar el comando más conocido como es Hola Mundo como se detalla en la **Figura 2.27**, realizar este paso es importante, puesto que con ello se puede confirmar que los dos programas están conectados y por lo tanto se puede seguir con otras tareas más complejas.

```
sim.simxAddStatusbarMessage(clientID,'Hello World!',sim.simx_opmode_oneshot)
```

Figura 2.27 Hello World! de CoppeliaSim Edu [Autoría Propia]

Dentro de CoppeliaSim Edu posee parámetros de posición del Robot, mismos que deben ser conocidos con la finalidad de cerrar el lazo de control, para la cual se hace uso de los comandos de la **Figura 2.28**. El comando simGetObjectOrientation, la información recopilada por este comando está en radianes, es decir la orientación del robot. El comando simGetObjectPosition, recupera la posición del objeto en este caso del robot NAO.

```
#Comunicación Coreographe - CoppeliaSim Edu - Python - Naoqi
NAO_pos=sim.simxGetObjectPosition(clientID, NAO_Handle[1],-1,sim.simx_opmpde_streaming)[1]
pos_x=NAO_pos[0]
pos_y=NAO_pos[1]
NAO_orient=sim.simxGetObjectOrientation(clientID,Nao_Handle[1],-1,sim.simx_opmode_streaming)[1]
gamma_real=NAO_orient[2]
```

Figura 2.28 Comandos para el lazo de control con Python - CoppeliaSim Edu [Autoría Propia]

2.3 ENTORNO DE SIMULACIÓN PYTHON PARA EL ROBOT NAO

2.3.1 COMANDOS DE EJECUCIÓN SDK NAOQI PARA EL ROBOT

El software de Python sirve de intermediario entre CoppeliaSim Edu y MATLAB/Simulink el mismo que tomará los datos de los dos programas y enviará de una plataforma a otra, aparte de ejecutar una acción de intermediario de datos Python tiene una función importante y este permite conectar las librerías de Choregraphe a Python mediante el SDK de NAOqi, es útil ya que el software Choregraphe puede ejecutar funciones sea de caminar, voz, audio, diferentes posturas del robot, contar con todos estos comandos en Python permite enviar datos y que estos ejecuten mediante NAOqi al Robot NAO.

Con la finalidad de contar con todas las funcionalidades es importante descargar la librería de NAOqi y se debe crear un Proxy mediante la dirección IP y número del puerto del robot real o el robot que se tiene en Choregraphe y asignar un atributo en proxy creado en Python, el comando se puede observar en la **Figura 2.29**.

```
movProxy=ALProxy("ALMotion",NaoIP,NaoPort)
postureProxy=ALProxy("ALRobotPosture", NaoIP,NaoPort)
```

Figura 2.29 Líneas de comando del Proxy de movimiento y posición de NAOqi [Autoría Propia]

Para el movimiento del robot en el software de simulación de CoppeliaSim Edu, se debe enviar datos de velocidad en m/s. Se tiene dos opciones para enviar esta información con coordenadas (X, Y) y el ángulo respectivo de la velocidad angular. La segunda opción es enviar el módulo de la velocidad lineal con ucont y velocidad angular con wcont. En la **Figura 2.30** se puede observar las líneas de código implementadas para enviar órdenes de velocidad en m/s.

```
#Enviar datos de Velocidad en m/s
movProxy.move(ucont,0,wcont)
```

Figura 2.30 Comandos de NAOqi para efectuar las órdenes de velocidad m/s [Autoría Propia]

Otra de las opciones que se maneja para el movimiento del robot es el comando presentado en la **Figura 2.31**, este comando toma los valores ingresados por el joystick en la estación local, el mismo hace que se mueva a una velocidad normalizada tanto en el eje X, Y y theta.

```
#movimiento del robot a través de datos ingresados por
#un Joystick
movProxy.moveToward(Recibir[0], Recibir[1], Recibir[2])
```

Figura 2.31 Comando para la obtención de datos desde un Joystick

Este comando maneja tres parámetros mismos que son adimensionales, detallados a continuación:

X: velocidad normalizada en el eje X, con valores de +1 y -1, que corresponden a la velocidad máxima en las direcciones de avance y retroceso, respectivamente.

Y: velocidad normalizada en el eje Y, con valores de +1 y -1, que corresponden a la velocidad máxima en las direcciones izquierda y derecha respectivamente.

Theta: velocidad normalizada en el eje Z de +1 y -1, que corresponden a la velocidad máxima en sentido horario y antihorario respectivamente.

[30]

2.4 ENTORNO DE SIMULACIÓN DE MATLAB - SIMULINK

2.4.1 COMUNICACIÓN DE DATOS ENTRE SIMULINK - PYTHON PARA EL ROBOT NAO

Las diferentes actividades y los controles para el mismo son implementados en MATLAB/Simulink, de este entorno se deben enviar los datos a Python, para lo cual se implementa un sistema de comunicación con los bloques UDP, los datos a enviar son datos de posición u ordenes de cada una de las actividades requeridas por el usuario. Esta información es empaquetada en un vector (array), los valores de este se discretizan con un tiempo de muestreo de 50 milisegundos, este valor es considerado como el tiempo de respuesta para el robot NAO, estos datos se envía a la dirección IP y el puerto especifico configurado en los bloques de send UDP (Véase **Figura 2.32**).

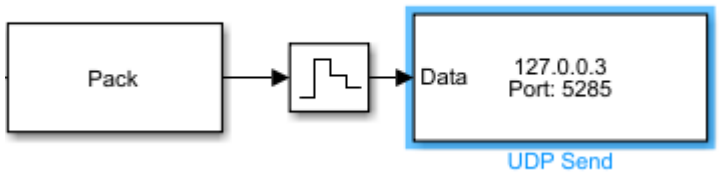

Figura 2.32 Bloques que permite realizar la transmisión de datos de Simulink a Python [Autoría Propia]

El entorno de desarrollo creado en Python enviará los datos del robot, esta información es recibida por el bloque de comunicación receive UDP, después de este bloque se desempaqueta en un vector (array) (Véase **Figura 2.33**).

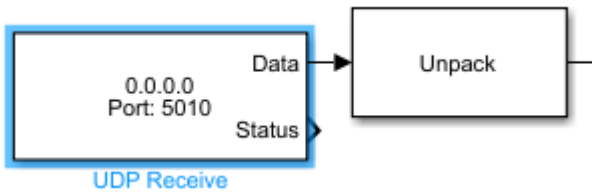

Figura 2.33 Bloques que permite realizar la recepción de datos de Python a Simulink [Autoría Propia]

En el caso de la programación implementada en Python, se declara la comunicación UDP con la dirección IP y el puerto, estos deben ser los mismos datos que fueron especificados en la comunicación desarrollada en Simulink (Véase **Figura 2.34**).

```
#Recepción de datos Matlab - Simulink a Python
UDP_IP="127.0.0.2"
UDP_PORT=5285
sock=socket.socket(socket.AF_INET,
                   socket.SOCK_DGRAM)#UDP Create Datagram Socket
sock.bind((UDP_IP, UDP_PORT))
Controlador,addr=sock.recvfrom(1024) #buffer size 1024 bytes
Control=unpack('>ddd',Controlador)
```

Figura 2.34 Líneas de datos para la recepción de datos de Simulink a Python [Autoría Propia]

Para enviar la información de Python a Simulink se debe realizar un trabajo similar a la recepción y se declara la dirección IP y el número de puerto especificado en el bloque UDP dentro del programa de Simulink (Véase **Figura 2.35**).

```
#Comunicación - Transmisión de datos Matlab - Simulink Python
Realimentacion=[pos_x,pos_y_gamma_real]
mensajebody=pack('>ddd',*Realimentacion)
sock.sendto(mensajebody,("127.0.0.2",5010))
```

Figura 2.35 Transmisión de datos de Python a Simulink [Autoría Propia]

2.5 IMPLEMENTACIÓN DE ESCENARIOS

Para observar el comportamiento del robot NAO V6 dentro del programa de simulación CoppeliaSim Edu, se implementa tres escenarios, con diferentes niveles de dificultad, en la misma a través de la estación local con la ayuda del joystick se direccionará desde un punto inicial hasta un punto final al NAO. Como punto de partida se encuentra la posición

inicial del robot NAO, y como punto de llegada se tiene una computadora (Véase **Figura 2.36**). En el primer caso, se tiene un escenario con dificultad baja, es decir un escenario sin obstáculos, en la segunda imagen se presenta un escenario con dificultad media con la presencia de un obstáculo, en este caso una maceta con una planta, el mismo que debe ser rebasado por el robot. Por último, se tiene un escenario con dificultad alta, con varios obstáculos, sillas que asemejan a un aula de clase. En este caso el robot tiene varias opciones para llegar desde el punto inicial al punto final, evadiendo los obstáculos.

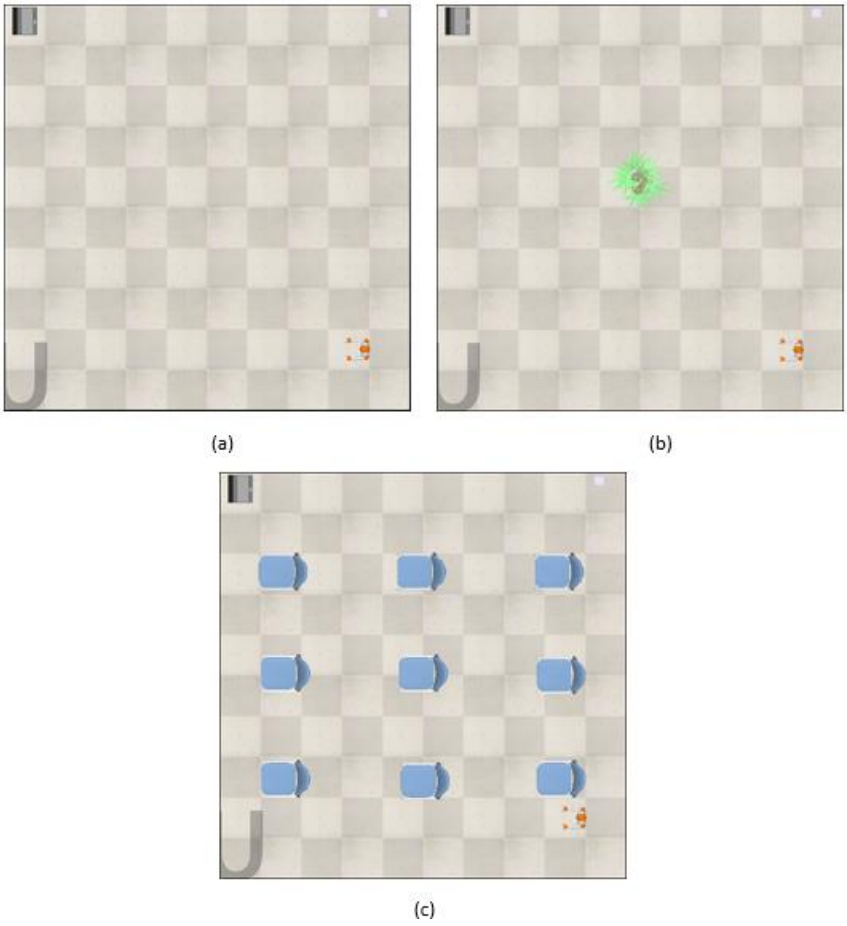

Figura 2.36 Escenarios implementados (a) Escenario con dificultad baja (b) Escenario con dificultad media (c) Escenario con dificultad alta

2.6 INTERFAZ GRÁFICA

La interfaz de usuario se desarrolla con la finalidad que el operador pueda interactuar con el sistema de manera sencilla e intuitiva. [31]

Como parte del Trabajo de Integración Curricular se desarrolló la interfaz gráfica en App Designer, herramienta proporcionada por MATLAB. El sistema cuenta con 6 ventanas, 3 para la estación local y 3 para la estación remota.

En la **Figura 2.37** se puede observar la pantalla de la estación local, en la misma se puede observar la información correspondiente al trabajo realizado, y este cuenta con botones que me permite visualizar las interfaces para la información del usuario y la PC Local.

Figura 2.37 Interfaz desarrolla en App Designer para la operación del robot NAO V6 PC Local [Autoría Propia]

En la **Figura *2.38*** se puede observar la ventana correspondiente a la información para el usuario en la estación local, en la misma se debe seleccionar los apartados requeridos por el usuario para visualizar la información.

Figura 2.38 Ventana de Información para el usuario [Autoría Propia]

En la **Figura 2.39** se puede observar la ventana correspondiente a la PC Local, en la misma se puede seleccionar el escenario deseado en el cual el NAO va a cumplir con la tarea. Se tiene botones que permiten interactuar con las ventanas previas mencionadas. Con el switch se puede iniciar la simulación, el mismo que me dará la oportunidad de leer todos los datos provenientes de la estación remota y estos serán dibujados en el plano XY.

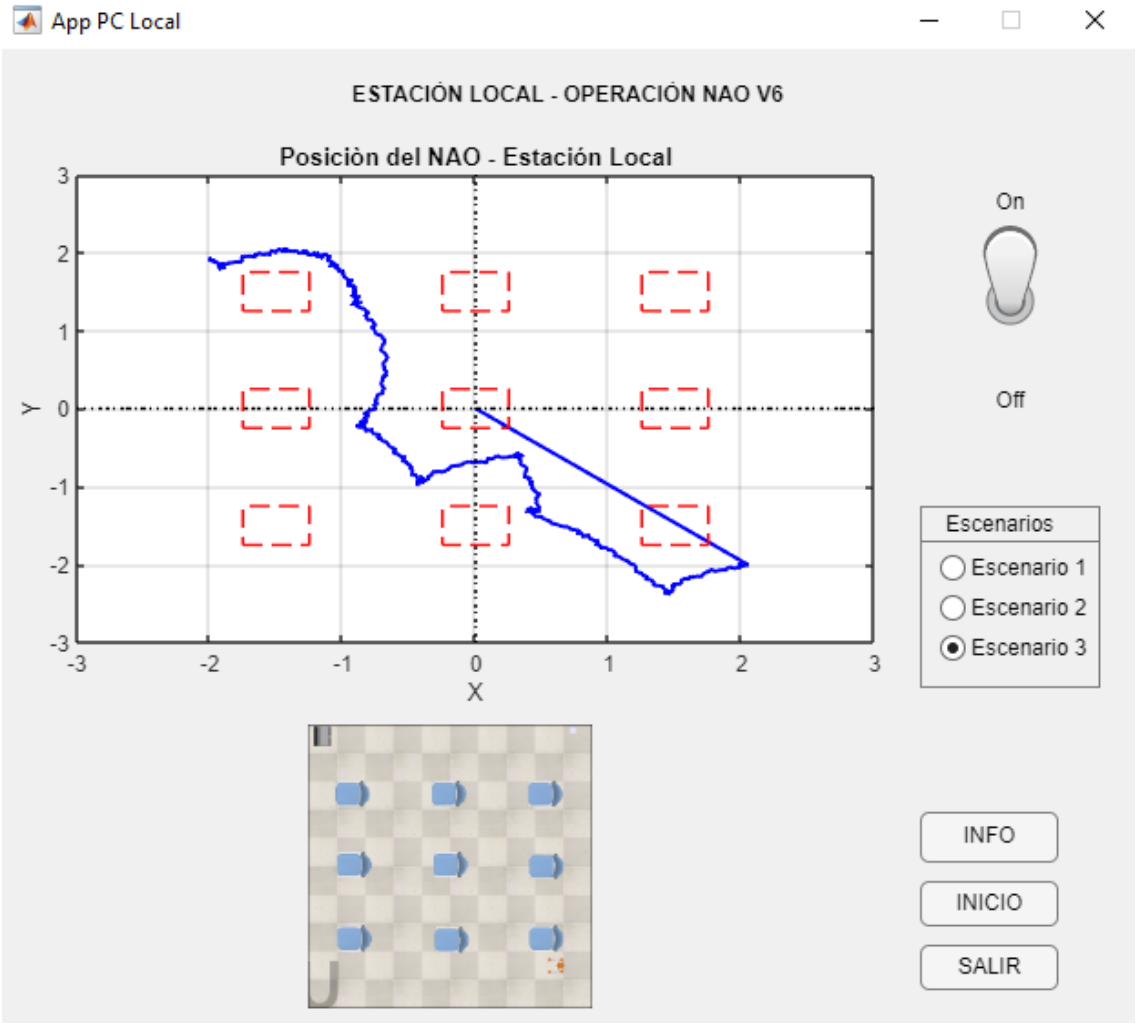

Figura 2.39 Ventana de la PC Local [Autoría Propia]

En la ***Figura 2.40*** se puede observar la portada para el interfaz desarrollado para la PC Remota, en la misma se tiene botones que permite interactuar con otras ventanas como la de información para el usuario y la ventana donde se visualiza el comportamiento del robot.

Figura 2.40 Interfaz desarrollada en App Designer para la PC Remota [Autoría Propia]

En la ***Figura 2.41*** se puede observar la interfaz desarrollada para la PC Remota con la finalidad de brindar la información necesaria al usuario, en la misma se debe seleccionar el tema requerido. Además, se cuenta con botones que permiten interactuar con las otras ventanas desarrolladas para la estación remota.

Figura 2.41 Interfaz desarrollada para la información del usuario en la PC Remota [Autoría Propia]

En la ***Figura 2.42*** se puede observar la ventana de la PC Remota, en la misma se tiene un switch que permitirá observar el escenario seleccionado desde la PC Local, botones que permiten interactuar con las ventanas de las ***Figura 2.40*** y ***Figura 2.41***. finalmente se tiene un switch que permite obtener toda la información proveniente de CoppeliaSim Edu y dibujar estos datos en el plano XY.

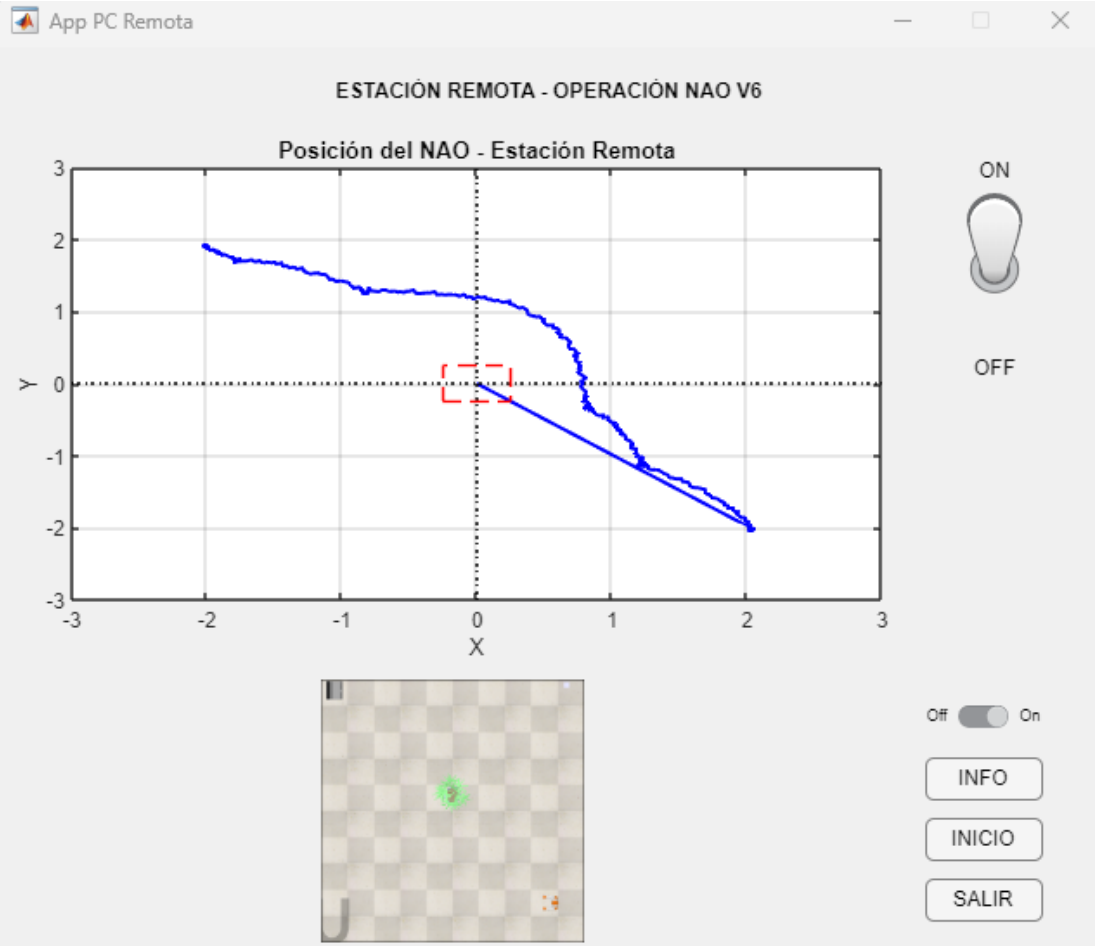

Figura 2.42 Ventana de la PC Remota [Autoría Propia]

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

Para que el robot NAO pueda movilizarse desde un punto inicial a un punto final en el simulador CoppeliaSim Edu, se sigue un proceso ordenado en la cual se ejecuta una serie de programas y librerías tanto en la estación local como la estación remota. Mismos que se encuentran detallados en las interfaces de usuario de la estación Local y estación Remota. Además, se cuenta con el Manual de Usuario en el cual se detallan todos los pasos que se deben seguir para poner en marcha el sistema desarrollado como parte del Trabajo de Integración Curricular.

3.1 Resultados

3.1.1 Resultados Escenario 1

La operación del robot NAO a través de joystick en el escenario 1 de dificultad baja se obtiene la respuesta presentada en la **Figura *3.1***. la imagen de la izquierda corresponde a la obtenida en la PC local y la imagen de la derecha es la obtenida en la PC remota, en la misma se observa el recorrido realizado por el NAO en el plano XY desde el punto inicial (2, -2) en donde se encuentra el NAO hasta el punto final (-2, 2) en donde se encuentra la computadora como referencia del objetivo para el robot. Este recorrido realizado es a partir de las ordenes ingresadas por el usuario en la estación local a través de un joystick, que me permite ingresar órdenes para que el robot llegué a su objetivo.

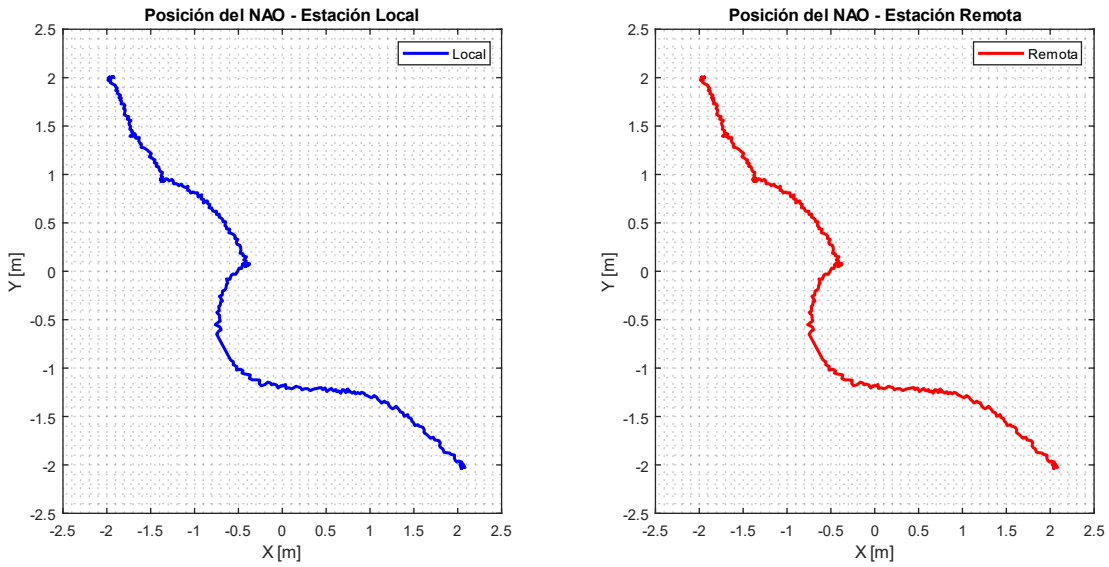

Figura 3.1 Recorrido realizado por el NAO para el escenario 1, gráfica obtenida para la PC Local (azul) y para la PC Remota (rojo) [Autoría Propia]

En la ***Figura 3.2*** se puede observar la posición que tiene el robot, para la estación local en la columna de la izquierda y estación remota en la columna de la derecha. Estos datos son generados en CoppeliaSim Edu, mismos que son recopilados en las estaciones local y remota. En este caso se tiene datos para cada uno de los ejes y el ángulo theta, en la misma se puede observar que para X y Y el dato de la posición varía entre -2 y 2. En el caso del eje X, de un valor positivo se pasa a un valor negativo, por otro lado, para el eje Y de un valor negativo se pasa a un valor positivo.

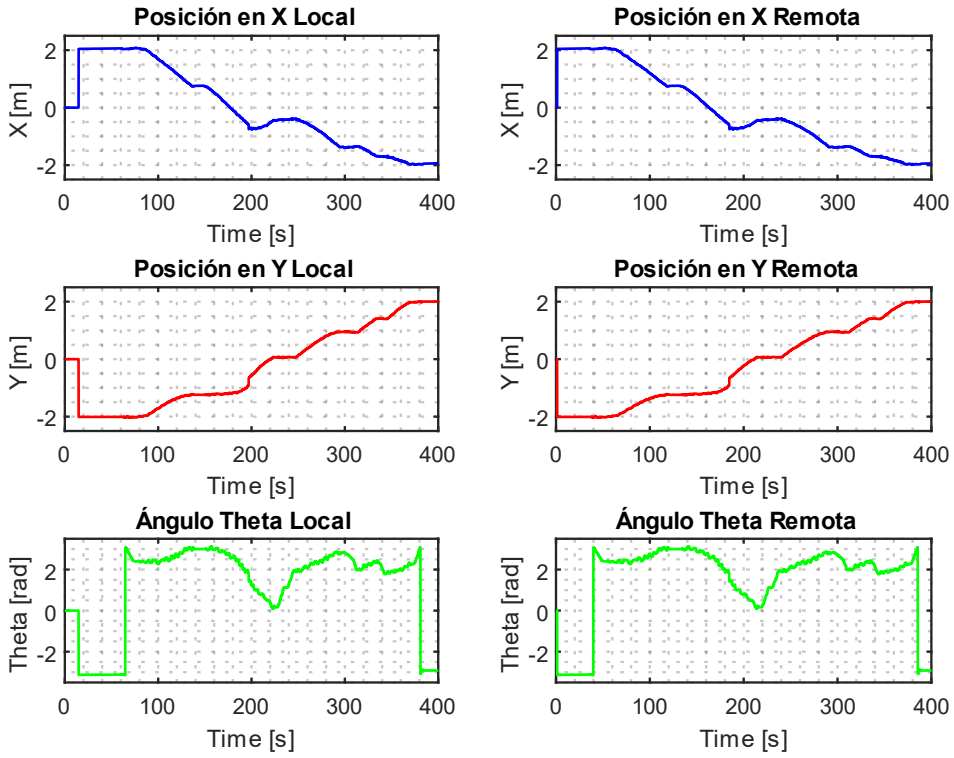

Figura 3.2 Posiciones y ángulo theta del NAO para el escenario 1, gráfica obtenida para la PC Local (columna izquierda) y para la PC Remota (columna derecha) [Autoría Propia]

En la ***Figura 3.3*** se puede observar las órdenes ingresadas con el joystick en la estación local las mismas que son enviadas a la estación remota mediante el protocolo de comunicación UDP. Los datos recibidos por el robot en la estación remota son los mismos datos ingresados en la estación local, se puede observar la similitud de estas entre las gráficas de la columna izquierda y derecha, datos para la estación local y estación remota respectivamente. Para la velocidad en X se tiene valores positivos, es decir el robot cumple con un movimiento hacia adelante, en el caso de la velocidad para el ángulo se tiene valores positivos y negativos, es decir que el robot giró en sentido antihorario para valores positivos y giró en sentido horario para valores negativos. Se tiene valores pequeños para el dato de la velocidad en y, es decir no existe movimiento derecha o izquierda.

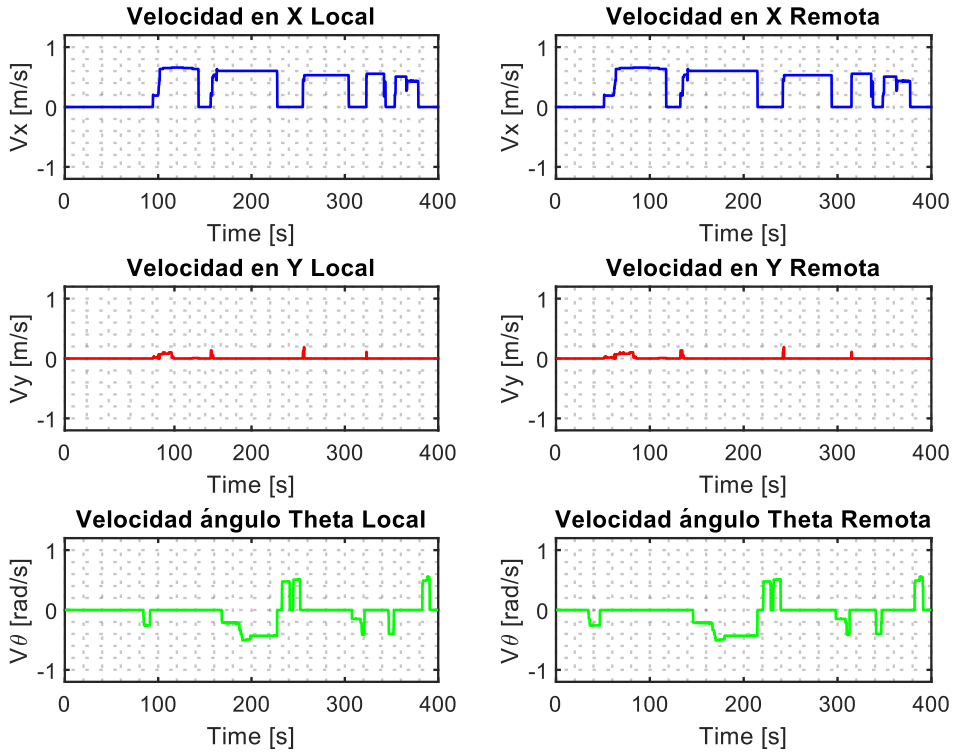

Figura 3.3 Órdenes ingresadas para la operación del robot NAO en la estación local (columna izquierda) y recibidas en la estación remota (columna derecha) [Autoría Propia]

3.1.2 Resultados Escenario 2

La operación del robot NAO a través del joystick en el escenario 2, con dificultad media, es decir que este ambiente de simulación tiene un obstáculo en la parte central del escenario, los resultados de este recorrido se pueden observar en la **Figura *3.4***. El inicio del recorrido está marcado por la posición inicial del robot en las coordenadas (2, -2), hasta el lugar de destino que esta dado por las coordenadas (-2, 2) lugar en el cual se encuentra una computadora, para llegar a su destino el robot debe superar una maceta que se encuentra como obstáculo en las coordenadas (0, 0). La imagen de la derecha corresponde al recorrido del robot en el plano XY que se puede observar en la PC de la estación remota, la información utilizada en esta estación se envía a la PC de la estación local, el recorrido del robot en el plano XY se presenta en la imagen de la derecha. La información entre las dos estaciones se envía mediante el protocolo de comunicación UDP implementado en el sistema de teleoperación.

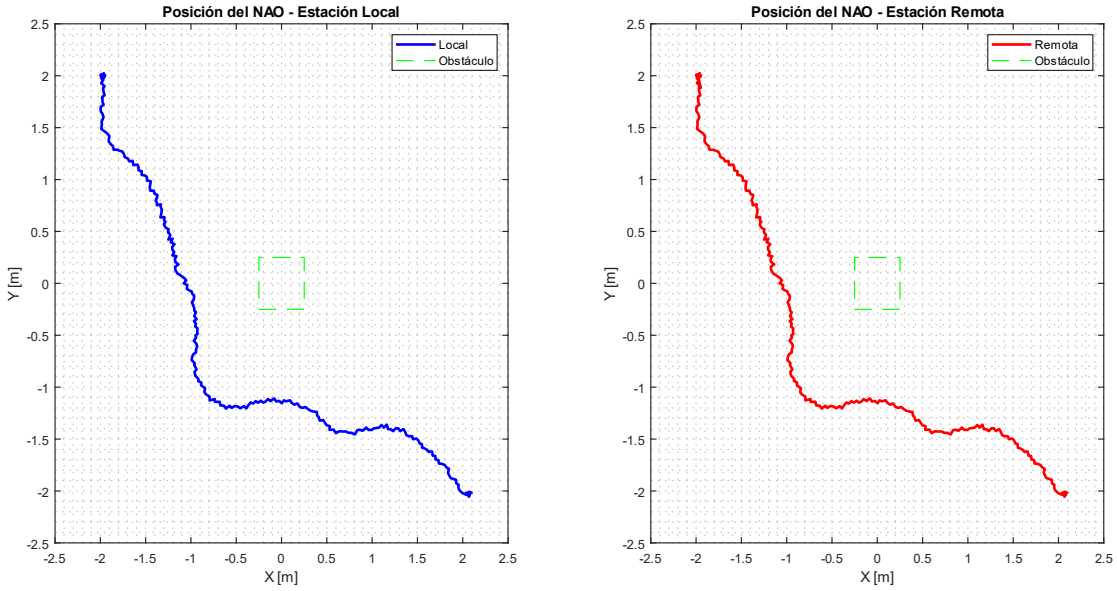

Figura 3.4 Recorrido realizado por el NAO para el escenario 2, gráfica obtenida para la PC Local (azul) y para la PC Remota (rojo) [Autoría Propia]

En la ***Figura 3.5*** se puede observar las posiciones y el ángulo theta, para la estación local en la columna de la izquierda y para la estación remota en la columna de la derecha. La información que se presenta en las imágenes es obtenida de CoppeliaSim Edu.

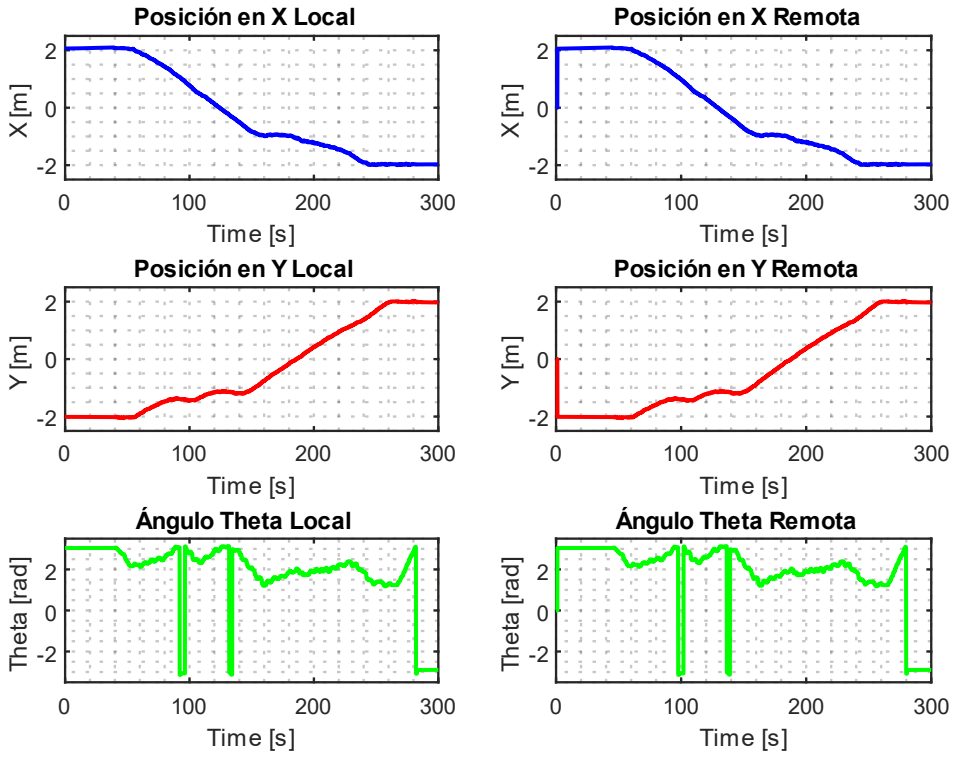

Figura 3.5 Posiciones y ángulo theta del NAO para el escenario 2, gráfica obtenida para la PC Local (columna izquierda) y para la PC Remota (columna derecha) [Autoría Propia]

Las órdenes ingresadas mediante el joystick en la estación local son presentadas en la columna de la izquierda y las órdenes recibidas en la estación remota presentadas en la columna de la derecha de la ***Figura 3.6*** son utilizados para la operación del robot NAO, para llevar a este desde la posición inicial a la posición final. Las velocidades de las dos columnas presentan una similitud para los datos de las dos estaciones. Para la velocidad en X se presenta valores positivos, es decir el robot tiene un movimiento hacía adelante. La velocidad en Y, no presenta mayor variación puesto que no existe un recorrido a la izquierda o derecha. El dato final de la velocidad del ángulo theta toma valores negativos es decir el robot tuvo un giro en sentido horario en la mayor parte del recorrido, y en la parte final toma valores positivos es decir el robot giro en sentido antihorario.

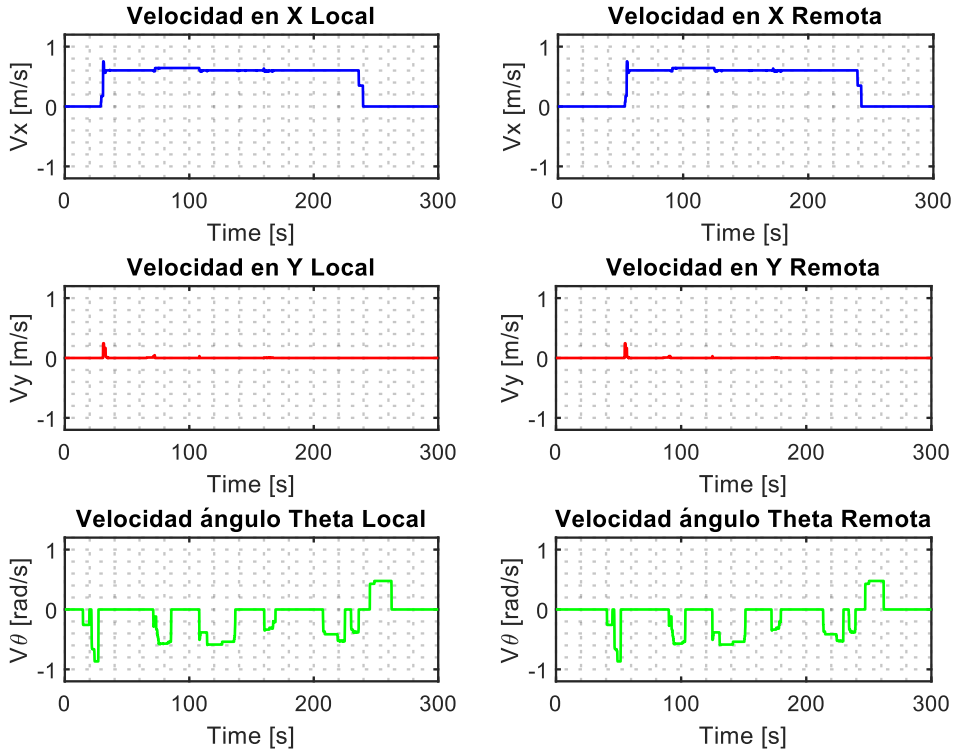

Figura 3.6 Órdenes ingresadas para la operación del robot NAO en la estación local (columna izquierda) y recibidas en la estación remota (columna derecha) [Autoría Propia]

3.1.3 Resultados Escenario 3

Para el escenario 3 de dificultad alta se tiene la presencia de varios obstáculos, que el robot debe evitar desde el punto inicial hasta el punto final. El recorrido realizado por el NAO se puede observar en la **Figura *3.7***, en la imagen de la derecha se tiene el recorrido realizado en la estación remota y estos datos se envían por protocolo UDP a la estación local y se presentan en la imagen de la derecha.

En el plano XY se puede observar el recorrido realizado por el robot desde el punto inicial en las coordenadas (2, -2) donde se encuentra el robot hasta el punto final en las coordenadas (-2, 2). Además, se tiene varios obstáculos con sillas que asemejan un aula de clase y la evasión de los obstáculos que tiene que realizar con la finalidad de llegar hasta el punto final donde se encuentra la computadora. La operación del robot NAO se realiza en la estación local a través del joystick y la información ingresada se envía a través de comunicación UDP para la estación remota en donde se encuentra el robot. En este escenario se puede observar un desplazamiento de derecha e izquierda, es decir se ingresa valores de velocidad para el eje Y, esto con la finalidad de evadir el obstáculo con el cual se encontró el robot al momento de cumplir con la tarea asignada por el usuario.

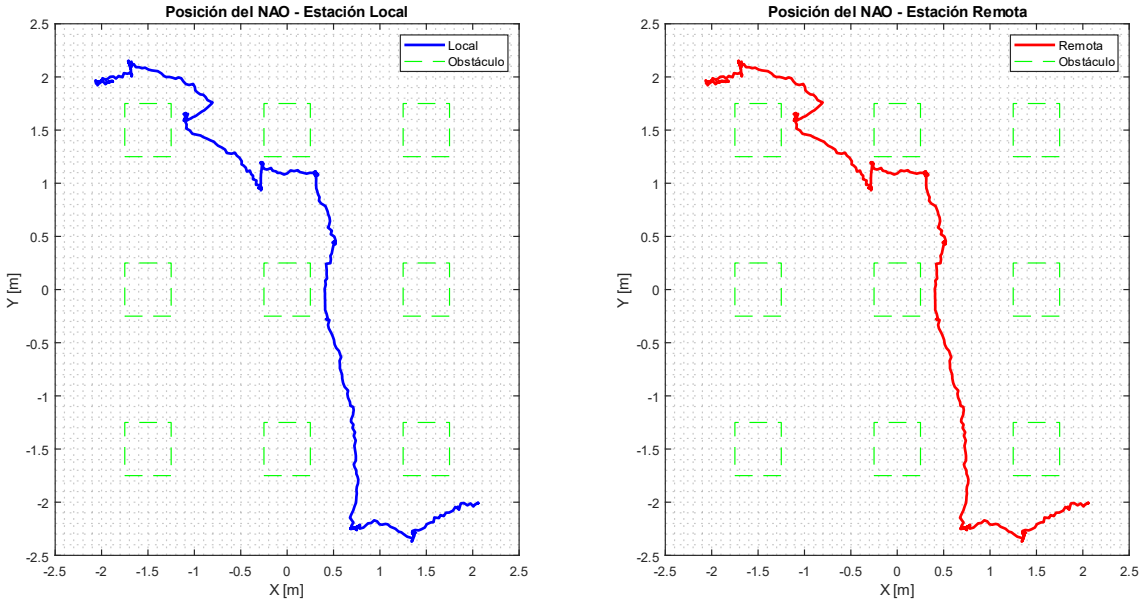

Figura 3.7 Recorrido realizado por el NAO para el escenario 3, gráfica obtenida para la PC Local (azul) y para la PC Remota (rojo) [Autoría Propia]

En la ***Figura 3.8*** se puede observar las posiciones y ángulos en la estación local en la columna de la izquierda y la estación remota en la columna de la derecha. Se puede observar el cambio de posición que tiene el robot entre valores de 2 a -2 para el eje X y para el eje Y se tiene un cambio de -2 a 2. Para el ángulo theta se puede observar cambios que se ingresaron, es decir realizar giros en sentido horario y sentido antihorario sobre el mismo eje.

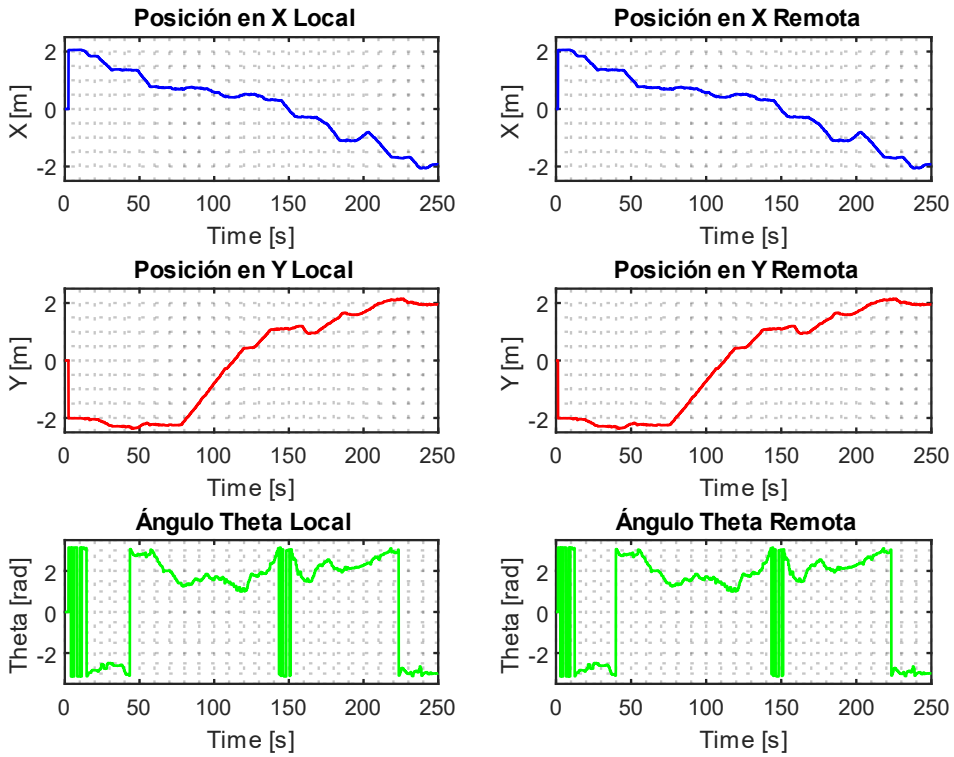

Figura 3.8 Posiciones y ángulo theta del NAO para el escenario 3, gráfica obtenida para la PC Local (columna izquierda) y para la PC Remota (columna derecha) [Autoría Propia]

Las órdenes ingresadas por el joystick en la estación local se presentan en la imagen de la columna de la izquierda, datos que son enviados a través del protocolo UDP a la estación remota, datos que se presentan en la imagen de la columna de la derecha de la **Figura *3.9***. Se ingresa valores positivos para la velocidad en X para que el robot se mueva hacia adelante, y se tiene valores negativos que hacen que el robot retroceda. Los valores ingresados para la velocidad en Y se tiene valores positivos y negativos, es decir tiene movimientos a la izquierda y derecha respectivamente. Finalmente se tiene los valores ingresados para Theta, se cuenta con valores positivos y negativos los mismos que representan el giro del robot sobre el mismo eje. Para el sentido antihorario valores positivos y para el sentido antihorario valores negativos.

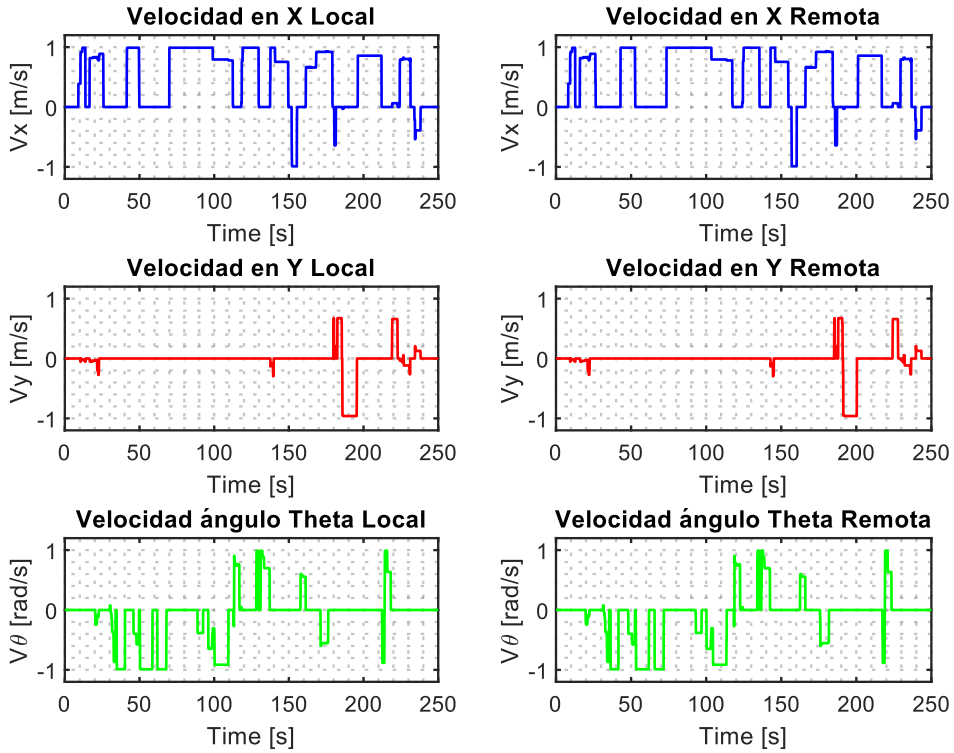

Figura 3.9 Órdenes ingresadas para la operación del robot NAO en la estación local (columna izquierda) y recibidas en la estación remota (columna derecha) [Autoría Propia]

3.2 Conclusiones

En base a la revisión bibliográfica de este documento se presenta un algoritmo de comunicación basado en el protocolo UDP, el sistema implementado permite tener una operación a distancia del robot NAO. El usuario tiene la facilidad de operar el robot desde la estación local mediante un joystick y un equipo que se encuentra en la estación remota (Robot NAO).

Al realizar diferentes pruebas con la finalidad de verificar la comunicación establecida entre las dos estaciones, se observan retardos al momento de enviar la información desde la estación local; para lo cual, se realizó modificaciones en el tiempo del barrido establecido en el algoritmo de comunicación para la lectura de los comandos ingresados por el usuario, estableciéndose en la estación local un tiempo de 50 milisegundos y en la estación remota de 100 milisegundos.

El envío de ordenes desde la estación local hacía la estación remota a través del protocolo de comunicación UDP, se implementó por la importancia de la rapidez en la transmisión de datos; sin embargo, no se conoce si el dato enviado por el emisor llegó de la forma correcta

al receptor, puesto que este protocolo no entrega una retroalimentación de la información recibida.

El control del robot NAO se realiza a través de un joystick, mismo que se encuentra en la estación local, este permitió que el usuario ingrese datos de velocidad que varían entre valores de -1 y 1, mismos que para el robot se transforman en órdenes para movimientos o giros en función de las necesidades requeridas por el usuario. Con la finalidad de que el robot pueda movilizarse sin inconvenientes se hace uso de saturadores para los valores mínimos y máximos ingresados mediante el joystick.

La interfaz desarrollada en MATLAB/App Designer permitió operar a distancia el robot humanoide, ya que esta interfaz es amigable al usuario y permite conocer en la estación local, el recorrido que se encuentra realizando el robot en la estación remota.

3.3 Recomendaciones

Se recomienda realizar un análisis de las características mínimas que debe tener las computadoras de las dos estaciones, puesto que el desarrollo del sistema implica la ejecución de varios programas de forma simultánea, entre ellos tenemos MATLAB/Simulink, Python, Choregraphe y CoppeliaSim Edu, caso contrario puede presentarse dificultades al momento de ejecutar el sistema realizado.

Se recomienda usar los datos de velocidad ingresados a través del joystick y leer de forma directa en el script de Python con el comando API, específico para valores ingresados por el joystick.

Para establecer y poner en marcha el algoritmo de comunicación desarrollado en el Trabajo de Integración Curricular se recomienda seguir todos los pasos detallados en el Manual de Usuario para la teleoperación del robot NAO, mismo que se puede encontrar en el apartado de anexos.

Como trabajos futuros de investigación se podría realizar un algoritmo de comunicación a través de WiFi y de esta forma evitar la utilización de cables entre la estación local y la estación remota.

Se recomienda investigar la utilización de los sensores de proximidad y hacer uso de estos para enviar mensajes de presencia de obstáculos desde la estación remota hacia la estación local y de esta forma entregar más herramientas al usuario para la facilidad de operación del robot humanoide.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] I. Caparroso, O. Avilés, and J. Hernández, “Una introducción a la Robótica Industrial,” *Ciencia e Ingeniería Neogranadina*, vol. 8, pp. 53–67, 1999.

[2] G. Niemeyer, C. Preusche, and G. Hirzinger, “Telerobotics,” *Springer Handbook of Robotics*, vol. 25, pp. 741–757, 2008.

[3] F. Tanaka, A. Cicourel, and J. R. Movellan, “Socialization between toddlers and robots at an early childhood education center,” *Proceedings of the National Academy of Sciences*, vol. 104, pp. 17954–17958, 2007.

[4] S. Ballesteros, “Sistema de teleoperación mediante una interfaz natural de usuario,” Trabajo Fin de Grado, Universidad Carlos III de Madrid, Madrid, 2012.

[5] J. Gálvez, “Teleoperación del robot NAO mediante dispositivos móviles Android,” Trabajo Fin de Grado, Universidad Calos III de Madrid, Madrid, 2012.

[6] D. Paredes, “Teleoperación del robot humanoide Pepper,” Trabajo Fin de Grado, Universidad de Alicante, Alicante, 2022.

[7] D. Gouaillier *et al.*, “The NAO humanoid: a combination of performance and affordability,” *ArXiv*, Jul. 2008.

[8] R. Poddighe, “Playing Tic-tac-toe with the NAO humanoid robot,” 2013.

[9] Aldeberan, “NAO el robot humanoide programable.” https://www.aldebaran.com/es/nao (accessed May 24, 2023).

[10] N. Kofinas, E. Orfanoudakis, and M. G. Lagoudakis, “Complete Analytical Forward and Inverse Kinematics for the NAO Humanoid Robot,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 77, no. 2, pp. 251–264, Feb. 2015.

[11] Aldebaran Robotics, “NAO Documentation” (v1.14.5) [Online] Available: http://doc.aldebaran.com/2-8/home_nao.html [Accessed: May 30, 2023].

[12] A. Marzal and I. Gracia, *Introducción a la programación con Python*. Universitat Jaume I. Servei de Comunicació i Publicacions, 2009.

[13] CoppeliaSim V-REP, “CoppeliaSim User Manual.” (Version 4.5) [Online] Available: https://www.coppeliarobotics.com/helpFiles/ [Accessed: May 30, 2023].

[14] D. Díaz, “Simulación de un entorno industrial mediante la herramienta de trabajo CopelliaSim (V-REP),” Trabajo Fin de Grado, Universidad Politécnica de Valencia, Valencia, 2020.

[15] MathWorks, “MATLAB - El lenguaje del cálculo técnico.” [Online] Available: https://la.mathworks.com/products/MATLAB.html [Accessed: May 31, 2023].

[16] MathWorks, “Simulación y diseño basado en modelos con Simulink - MATLAB.” [Online] Available: https://la.mathworks.com/products/simulink.html [Accessed May 31, 2023].

[17] C. Partridge and S. Pink, “A Faster UDP,” *IEEWACM TRANSACTIONS ON NETWORKING*, vol. I, no. 4, 1993.

[18] H. Zheng and J. Boyce, “An Improved UDP Protocol for Video Transmission Over Internet-to-Wireless Networks,” *IEEE Trans Multimedia*, vol. 3, no. 3, 2001.

[19] F. Taha AL-Dhief *et al.*, “Performance Comparison between TCP and UDP Protocols in Different Simulation Scenarios,” *International Journal of Engineering & Technology*, pp. 172–176, 2018.

[20] A. Malinowski and B. Wilamowski, “Industrial Communication Systems - User Datagram Protocol - UDP,” 2010.

[21] P. Cazorla, “Teleoperación de un brazo robot mediante el sensor Kinect,” Sep. 2014.

[22] J. Guerrero, “Diseño de un sistema de Teleoperacion implementando redes WLAN,” Trabajo Fin de Grado, Universidad Militar Nueva Granada, Bogotá, 2014.

[23] A. Camacho and D. Pérez, “Diseño de un sistema robótico teleoperado para fines didácticos,” Trabajo Fin de Grado, Instituto Politécnico Nacional, México, 2008.

[24] E. Nuño and L. Basañez, “Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente,” Tesis de Doctorado, Universidad Politécnica de Cataluña, Catalunya, 2004.

[25] Genius, “MetalStrike FF ForceFeedback Joystick for PC.”

[26] MathWorks, “Pilot Joystick All - Simulink.” [Online] Available: https://la.mathworks.com/help/aeroblks/pilotjoystickall.html [Accessed: May 31, 2023].

[27] MathWorks, “Simulation Pace - Simulink.” [Online] Available: https://la.mathworks.com/help/aeroblks/simulationpace.html?searchHighlight=set%20pace&s_tid=srchtitle_set%20pace_2 [Accessed: May 31, 2023].

[28] E. Nuñez and J. Ortega, “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,” Trabajo Fin de Grado, Escuela Politécnica Nacional, Quito, 2021.

[29] A. Enríquez, “Diseño y simulación del control de posicionamiento de un robot humanoide NAO,” Trabajo de Fin de Grado, Escuela Politécnica Nacional, Quito, 2022.

[30] Aldebaran Robotics, “Locomotion control API - NAO Software 1.14.5 documentation.” [Online] Available: http://doc.aldebaran.com/1-14/naoqi/motion/control-walk-api.html#almotionproxy-movetoward1 [Accessed jun. 04, 2023].

[31] F. Toapanta, “Diseño y simulación del control en cuatro cuadrantes de una máquina DC con un conversor DC/DC tipo Cuk Bidireccional,” Trabajo de Fin de Grado, Escuela Politécnica Nacional, Quito, 2022.

5. ANEXOS