UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
FACULTAD DE INGENIERÍA ELECTRÓNICA

Tarea #7

Interacción programática con el Robot Pepper: Configuración, Coreografías y Control Remoto

Presentado a: Ing. Diego Alejandro Barragan Vargas
Juan Diego Báez Guerrero, Cód.: 2336781.

Resumen*— *Este documento detalla el proceso de interacción programática con el robot Pepper utilizando herramientas de software especializadas. Se abordan las librerías esenciales (qi, argparse, sys, os, almath, math, motion, httplib y json) para el control del robot, la creación de coreografías mediante Choregraphe, y la ejecución remota de scripts vía SSH. El trabajo incluye una metodología paso a paso para la configuración del entorno, ejemplos prácticos de código Python, y la documentación de los resultados obtenidos en las pruebas con el robot humanoide.

Abstract*— *This document details the programmatic interaction process with the Pepper robot using specialized software tools. It covers essential libraries (qi, argparse, sys, os, almath, math, motion, httplib and json) for robot control, choreography creation through Choregraphe, and remote script execution via SSH. The work includes a step-by-step methodology for environment setup, practical Python code examples, and documentation of results obtained in tests with the humanoid robot.*

I. INTRODUCCIÓN

La robótica moderna ha evolucionado como una disciplina central en la integración de tecnologías avanzadas, abarcando desde el diseño de hardware hasta el desarrollo de software especializado. Entre las diversas plataformas disponibles, Pepper, un robot humanoide diseñado por SoftBank Robotics, se ha consolidado como una herramienta versátil en el ámbito educativo y de investigación. Su capacidad para interactuar con los seres humanos mediante gestos, voz y movimientos coordinados lo posiciona como un recurso clave para explorar las posibilidades de la programación y la inteligencia artificial [1].

Pepper es un robot humanoide cuya principal característica es la capacidad de interpretar emociones humanas y responder a ellas de manera interactiva. Diseñado para mejorar la comunicación entre humanos y máquinas, es ampliamente utilizado en instituciones académicas y comerciales para explorar avances en tecnología e inteligencia artificial [9]. Ubuntu, por otro lado, es un sistema operativo basado en Linux conocido por su estabilidad y flexibilidad. En el presente trabajo, se utiliza para acceder a la consola de Pepper y ejecutar scripts personalizados mediante comandos específicos como SSH [11].

En este contexto, el presente informe tiene como objetivo explorar las herramientas y librerías esenciales para el desarrollo en la plataforma de Pepper. Las librerías seleccionadas, tales como qi, argparse, sys, os, almath, entre otras, proporcionan funcionalidades fundamentales para habilitar la interacción entre el desarrollador y el robot, así como para la ejecución de tareas complejas de movimiento, comunicación y análisis [2]. Adicionalmente, se describirá el concepto de coreografía aplicada a Pepper, entendido como la sincronización de movimientos programados con fines educativos o interactivos [10].

Se abordarán las etapas necesarias para el desarrollo de proyectos, desde la instalación y configuración del software Choregraphe hasta la implementación de coreografías sencillas mediante códigos Python personalizados. Por otra parte, se analizará el acceso a la consola de Pepper utilizando la interfaz SSH, lo que permite una interacción directa con el sistema operativo del robot y la creación de scripts desde el editor nano. Esta metodología proporciona una comprensión más profunda de los procesos de programación, promoviendo el desarrollo de soluciones creativas y funcionales [3].

Finalmente, el informe se documentará de manera integral en Overleaf, incluyendo cada paso detallado, imágenes de apoyo y las referencias según el formato IEEE, con el objetivo de garantizar una estructura profesional y acorde a los estándares académicos. Este enfoque busca fomentar la aplicación práctica de conocimientos en robótica, programación y documentación técnica, pilares fundamentales para el desarrollo de habilidades en el ámbito tecnológico.

II. MARCO TEÓRICO

A. Librería qi

Esta librería es fundamental para la comunicación entre módulos, eventos y servicios de Pepper. Permite establecer conexiones con las interfaces de programación del robot, facilitando la ejecución de comandos desde código Python. Es la base del desarrollo interactivo en Pepper, ya que sin ella no sería posible manipular de manera eficiente sus capacidades sensoriales y motoras [1].

```
import qi
session = qi.Session()
session.connect("tcp://pepper.local:
9559")
motion = session.service("ALMotion")
motion.wakeUp()  # Despierta al robot
```

B. Librería argparse

Argparse es utilizada para interpretar argumentos desde la línea de comandos, permitiendo que los programas sean más dinámicos y versátiles. Esta herramienta es útil en proyectos con Pepper porque permite ajustar parámetros como velocidad, duración o tipo de movimiento directamente desde la terminal, lo que agiliza las pruebas y mejora la flexibilidad del código [2].

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--angulo",
type=float,
    default=30.0,
    help="Ángulo de rotación")
args = parser.parse_args()
```

C. Librerías sys y os

Las librerías sys y os son esenciales para interactuar con el entorno del sistema operativo. Con sys se gestionan argumentos de ejecución, mientras que os permite manipular archivos, rutas y procesos del sistema. Su uso es fundamental para acceder a recursos del sistema del robot Pepper, automatizar tareas y gestionar archivos creados por los scripts [3].

```
import sys, os
if not os.path.exists("config"):
    os.mkdir("config")
sys.path.append("/opt/aldebaran/lib
/python2.7/site-packages")
```

D. Librería almath

Almath es una librería matemática avanzada diseñada especialmente para cálculos geométricos y transformaciones espaciales. Es clave para que Pepper realice movimientos complejos y coordinados, como rotaciones precisas o desplazamientos calculados en el espacio tridimensional. Su integración garantiza mayor precisión en los movimientos programados [4].

```
import almath
target = almath.Pose2D(0.5, 0.0,
math.pi/2)
# x, y, theta
current = almath.Pose2D
(0.0, 0.0, 0.0)
path = almath.computePath
(current, target)
```

E. Librería math

La librería math provee funciones matemáticas básicas como raíces cuadradas, senos, cosenos, logaritmos y constantes como pi. Aunque es una librería general de Python, su utilidad en robótica es evidente al utilizarla para calcular trayectorias, ángulos o distancias. Se emplea en el control de movimientos simples y cálculos de lógica de navegación [5].

```
import math
def rotacion_robot(grados):
    radianes = math.radians(grados)
    return math.sin(radianes), math.cos
    (radianes)
```

F. Librería motion

Esta librería permite el control de los motores y articulaciones de Pepper. A través de motion, los desarrolladores pueden definir trayectorias, velocidades, y acciones específicas como caminar, saludar o inclinar la cabeza. Es una de las más utilizadas cuando se diseñan coreografías o comportamientos expresivos en robots humanoides [6].

```
import motion
# Inicializa el movimiento
motion.moveInit()
# Avanza 20 cm
motion.moveTo(0.2, 0, 0)
# Gira la cabeza
motion.setAngles("HeadYaw", 0.5, 0.1)
```

G. Librería httplib

Httplib permite que los scripts de Python se comuniquen con servicios web a través de protocolos HTTP. En proyectos con Pepper puede utilizarse para enviar o recibir información desde plataformas externas, lo que es útil para tareas como consultas en línea, interacciones con APIs o integración con sistemas domóticos [7].

```
import httplib
conn = httplib.HTTPSConnection("api.
robot.com")
conn.request("GET", "/status")
resp = conn.getresponse()
print(resp.read())
```

H. Librería json

Json es una librería para manejar objetos codificados en formato JSON, que es muy común en la transmisión de datos entre sistemas. En el contexto de Pepper, facilita la

estructuración y lectura de datos entre módulos o incluso para guardar configuraciones de usuarios, comandos predefinidos o información recibida de sensores [8].

```
import json
config = {
    "velocidad": 0.7,
    "modo": "interactivo",
    "sensores": ["sonar", "tacto"]
}
with open("config_pepper.json", "w")
as f:
    json.dump(config, f, indent=4)
```

I. Pepper

Pepper es un robot humanoide desarrollado por SoftBank Robotics. Su diseño está enfocado en la interacción social y en la capacidad de interpretar y responder a emociones humanas. Utilizado en ámbitos como la educación, la investigación y el comercio, Pepper se ha convertido en una herramienta innovadora para explorar las fronteras de la tecnología robótica y la inteligencia artificial [9].

Figura 1: Foto de Pepper. Fuente: Instagram, Representación de Facultad de Ingeniería Electrónica Santoto. Disponible en: [14]

J. Choregraphe

La coreografía en el contexto de la robótica se refiere a la programación de movimientos sincronizados, que pueden incluir gestos y desplazamientos diseñados para simular interacciones naturales o dinámicas animadas. En el caso de Pepper, estas coreografías pueden desarrollarse mediante el software Choregraphe, que permite integrar creatividad con funcionalidad técnica [10].

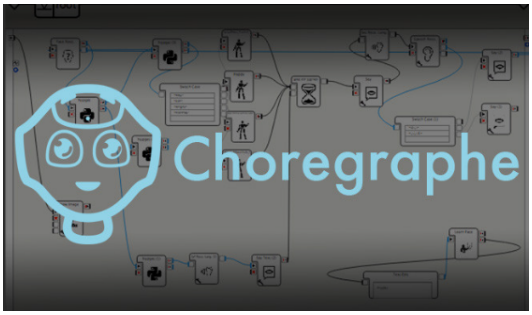

Figura 2: Choregraphe. Fuente: [12].

K. Ubuntu

Ubuntu es un sistema operativo basado en Linux que combina estabilidad, flexibilidad y un entorno amigable para desarrolladores. Este sistema es ampliamente utilizado para interactuar con robots como Pepper mediante herramientas como SSH, que facilitan la conexión remota, la personalización y la ejecución de scripts en tiempo real [11].

Figura 3: Ubuntu. Fuente: [13].

III. PROCEDIMIENTO

A. Configuración de Choregraphe

Primero se realizará la instalación del programa Choregraphe desde la página oficial de Aldebaran, buscando la versión adecuada para el robot social Pepper, la cual corresponde a Choregraphe 2.5.10.7. Esta versión es compatible con las capacidades del sistema operativo NAOqi instalado en el robot y garantiza una integración fluida entre el entorno de desarrollo y la plataforma robótica.

1) Descarga e Instalación en Windows

a) Descarga del Programa

Una vez localizado el archivo de instalación en el sitio oficial de SoftBank Robotics, se procede a su descarga. Es importante asegurarse de elegir la versión correcta y el sistema operativo correspondiente (en este caso, Windows).

Luego de haber descargado el instalador, se ejecuta el archivo para comenzar el proceso como se observa en la figura 4.

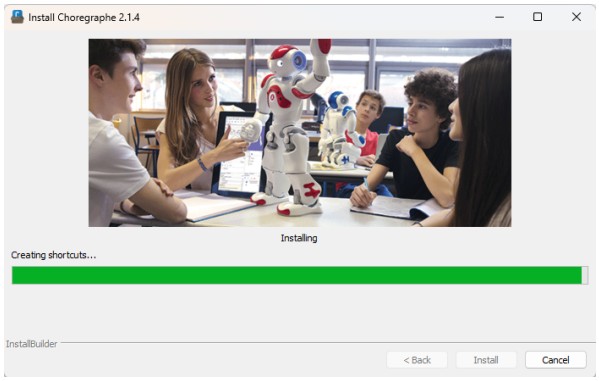

Figura 4: Descarga gráfica de Choregraphe. Fuente: Creación propia

Una vez completada la instalación, el asistente mostrará una ventana de finalización indicando que el programa ha sido correctamente instalado. Este paso se ilustra en la figura 5.

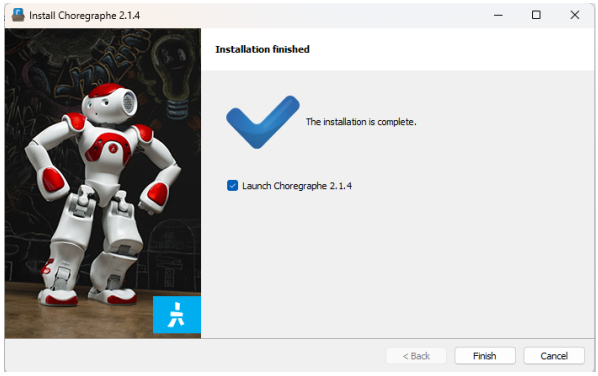

Figura 5: Finalización de la instalación de Choregraphe. Fuente: Creación propia

2) *Configuración de la Conexión con Pepper*

a) *Establecimiento de Red*

Para establecer una conexión adecuada entre Choregraphe y el robot social Pepper, es necesario conectarse previamente a la misma red Wi-Fi en la que se encuentra el robot. En este caso, la red se identifica con el nombre Ciego. Una vez conectado, se abre Choregraphe para iniciar la comunicación, como se muestra en la figura 6.

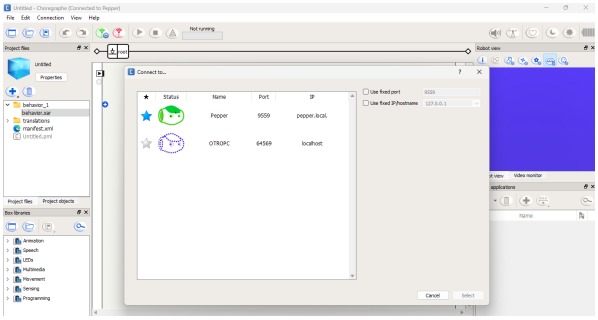

Figura 6: Conexión del robot Pepper con Choregraphe. Fuente: Creación propia

b) *Verificación de Conexión*

Después de haber realizado la conexión, es fundamental verificar que Pepper haya sido correctamente detectado por el software. En la interfaz principal de Choregraphe debe aparecer el modelo de Pepper junto con su dirección IP. Este proceso de verificación se representa gráficamente en la figura 7.

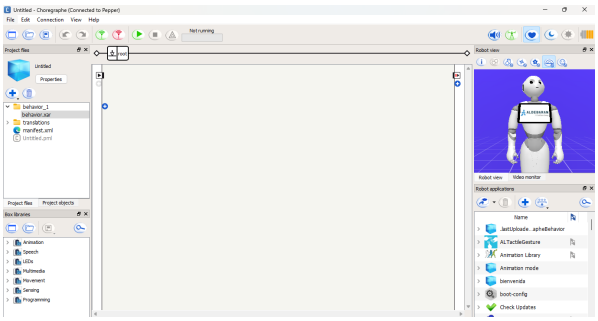

Figura 7: Verificación de la conexión con el robot en Choregraphe. Fuente: Creación propia

c) *Pruebas Iniciales del Programa*

Luego de establecer la conexión entre Choregraphe y el robot Pepper, se puede interactuar directamente con la interfaz del entorno para probar funciones básicas y verificar la operatividad del sistema. En la figura 8 se presenta una vista general de la interfaz gráfica de Choregraphe, la cual permite programar comportamientos mediante el uso de bloques visuales llamados cajas, que se organizan en el área central del entorno. Estas cajas representan acciones como movimiento, habla o decisiones lógicas, y se integran dentro de un contenedor principal llamado *root*, que estructura el flujo del comportamiento.

A la izquierda de la interfaz se encuentra el panel del proyecto, desde donde se gestionan los archivos asociados al comportamiento, como el archivo principal `behavior.xar` y otros recursos complementarios. Justo debajo de este panel

se sitúa la biblioteca de cajas, dividida en secciones estándar y avanzadas, que agrupan acciones básicas y funciones complejas, respectivamente. Estas cajas pueden ser arrastradas al área central para conformar el programa del robot de forma visual e intuitiva.

En el lateral derecho, se encuentran las bibliotecas de posturas predefinidas del robot, como ”StandInit.º ”StandZero”, útiles para definir posiciones iniciales o transiciones entre acciones. También es posible visualizar el listado de aplicaciones instaladas en el robot. En la parte superior de la ventana se ubican los controles de ejecución, que permiten iniciar, pausar o reiniciar la simulación del comportamiento diseñado. Esta interfaz también refleja el estado de conexión actual, ya sea con un robot físico o con uno simulado en el entorno virtual.

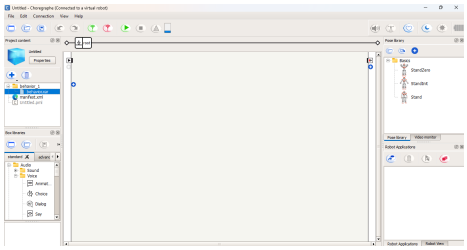

Figura 8: Interfaz gráfica del entorno Choregraphe. Fuente: Creación propia

B. Creación de Coreografía en Choregraphe

1) Diseño de Movimientos

Para el desarrollo del comportamiento del robot Pepper se empleó el entorno gráfico Choregraphe, una herramienta diseñada para facilitar la programación basada en bloques mediante cajas modulares. En la figura 9, se observa una coreografía compuesta por cinco animaciones secuenciales: Happy, Kisses, Chill, Mystical y Golf. Estas acciones fueron seleccionadas de la biblioteca predeterminada del software y conectadas en orden para formar una rutina expresiva y coherente.

La estructura fue organizada dentro del contenedor root, que actúa como nodo principal del flujo. Cada caja representa una animación con una transición directa hacia la siguiente, permitiendo que el robot ejecute los movimientos sin interrupciones. Esta metodología es ideal para diseñar secuencias complejas sin necesidad de programación textual, optimizando así el tiempo de desarrollo.

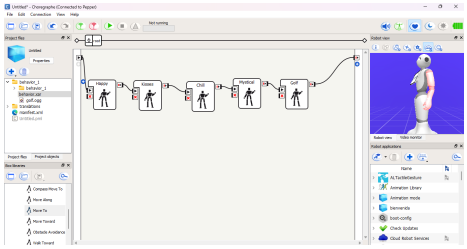

Figura 9: Coreografía diseñada en Choregraphe con animaciones encadenadas. Fuente: Captura propia

a) Interfaz del Programa

Choregraphe presenta una interfaz gráfica amigable que integra todas las herramientas necesarias para crear y probar comportamientos robóticos. A la izquierda se encuentra el panel de archivos del proyecto, donde se gestionan los elementos del comportamiento como el archivo `behavior.xar`, clips de audio y recursos de texto.

La parte inferior izquierda contiene la biblioteca de cajas, categorizadas según su funcionalidad (movimiento, sonido, visión, etc.). Estas cajas pueden ser arrastradas al espacio de trabajo central, donde se construye visualmente el comportamiento mediante conexiones lógicas entre ellas. A la derecha, la vista 3D permite visualizar en tiempo real las posturas y gestos del modelo virtual de Pepper durante la ejecución.

b) Transferencia y Ejecución en Pepper

Comprobada la correcta ejecución en el entorno virtual, se procedió a la transferencia del comportamiento al robot Pepper mediante una conexión directa desde Choregraphe. La coreografía fue reproducida de forma satisfactoria, con transiciones suaves y sincronización adecuada entre movimientos.

La ejecución física confirmó la fidelidad entre el entorno simulado y el comportamiento real, validando tanto la secuencia como la expresividad alcanzada, aspecto fundamental en entornos de interacción humano-robot.

2) Pruebas de la Coreografía

a) Simulación

Durante las pruebas en el simulador se prestó especial atención a la naturalidad de los movimientos y a la integración entre las animaciones. Se evaluaron diferentes combinaciones y secuencias para determinar cuáles transmitían mejor una sensación de fluidez y dinamismo. Las pruebas también sirvieron para asegurar que el comportamiento no presentara bloqueos o fallos de ejecución, elementos críticos en sistemas interactivos.

Además, se midió el tiempo total de la coreografía para verificar su adecuación a posibles escenarios de presentación

o interacción. El simulador permitió realizar iteraciones rápidas sin desgaste físico del hardware.

b) *Ejecución en el Robot*

Con el comportamiento final ya probado en simulación, se ejecutaron múltiples ensayos en el robot Pepper. Se evaluó la precisión de las posturas, la sincronización entre gestos y la estabilidad durante la ejecución. Adicionalmente, se observó la respuesta del entorno, comprobando que los movimientos no generaran conflictos en espacios reducidos ni interfirieran con elementos cercanos.

La ejecución en el robot validó tanto la viabilidad técnica del diseño como su impacto visual, permitiendo concluir que la coreografía es adecuada para aplicaciones de interacción social, presentaciones o actividades lúdicas en entornos educativos.

C. *Programación Vía SSH*

1) *Acceso al Robot desde Ubuntu*

a) *Comando SSH*

Para establecer una conexión remota con el robot Pepper, se utilizó el protocolo `SSH` desde la terminal de Ubuntu. Esta conexión permite acceder al sistema de archivos del robot y ejecutar scripts directamente en su sistema operativo NAOqi. En la Figura 10 se muestra el ingreso exitoso mediante el comando `ssh nao@<IP_Pepper>`.

Figura 10: Conexión al robot Pepper mediante SSH desde terminal de Ubuntu. Fuente: Creación propia

b) *Resolución de Problemas*

Durante la conexión, es posible que se presenten advertencias relacionadas con autenticación, certificados previos o permisos. Estas pueden resolverse eliminando entradas previas del archivo `known_hosts` o reiniciando la conexión con privilegios adecuados. Se recomienda también verificar que el robot esté conectado a la misma red que el computador.

2) *Creación y Ejecución de Scripts en Python*

a) *Creación del Archivo con Nano*

Una vez dentro del sistema del robot, se creó un directorio específico para los scripts de prueba. En él, se utilizó el editor de texto `nano` para escribir un archivo en Python, el cual contiene instrucciones básicas de movimiento. En la Figura 11 se ilustra el proceso de creación del directorio y el archivo de script.

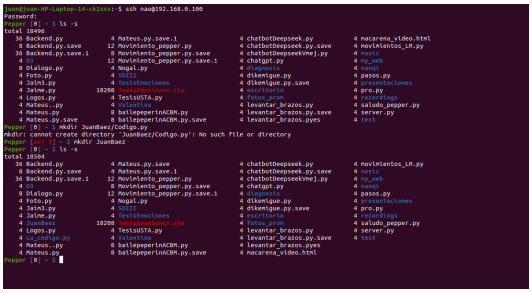

Figura 11: Proceso de creación de directorio y archivo Python en Pepper. Fuente: Creación propia

b) *Uso de Librerías Clave*

El código implementado emplea las librerías esenciales como `qi` para la comunicación con servicios de NAOqi, así como `time` para gestionar las pausas entre movimientos. En la Figura 12 se presenta un ejemplo de script que permite a Pepper ejecutar un saludo básico.

Figura 12: Ejemplo de script Python para movimiento de saludo en Pepper. Fuente: Creación propia

c) *Ejecución y Verificación*

Finalmente, para ejecutar el script creado, se utilizó el comando `python nombre_archivo.py` dentro del directorio correspondiente. Si el script es válido y el servicio de movimiento está disponible, el robot ejecuta el movimiento programado. Se recomienda verificar que el robot esté en posición estable y sin obstáculos antes de iniciar cualquier ejecución.

IV. CONCLUSIONES

La interacción con el robot Pepper permitió comprender la importancia de las librerías especializadas en la programación de robots humanoides. Herramientas como qi, almath y motion demostraron ser fundamentales para el control preciso de movimientos y la comunicación entre los módulos del robot, evidenciando cómo estas bibliotecas permiten desarrollar comportamientos complejos con una estructura de código clara y eficiente.

El entorno gráfico Choregraphe resultó ser una herramienta clave para diseñar coreografías de forma visual. Su interfaz amigable facilitó la creación de movimientos expresivos sin necesidad de escribir código, lo que permitió experimentar con secuencias dinámicas de manera más intuitiva. Esta metodología no solo ahorra tiempo en la programación, sino que también abre posibilidades a usuarios sin experiencia previa en desarrollo de software.

Por otro lado, la conexión remota a través de SSH desde Ubuntu reforzó la capacidad de controlar el robot de forma directa, permitiendo ejecutar scripts escritos en Python con mayor flexibilidad. Esta práctica evidenció la versatilidad del sistema operativo de Pepper y la posibilidad de personalizar sus acciones sin depender exclusivamente de interfaces gráficas.

Cada etapa del proyecto, desde la instalación de software hasta la creación y prueba de coreografías, fue documentada cuidadosamente. Este enfoque detallado ofrece una guía replicable para futuros desarrollos con Pepper, útil tanto en contextos educativos como de investigación. La claridad en los pasos seguidos y la solución de problemas comunes son aportes significativos para quienes se inicien en este tipo de proyectos.

Las pruebas realizadas tanto en simulador como en el robot físico validaron el correcto funcionamiento de los movimientos diseñados. Este proceso de validación práctica fue esencial para ajustar parámetros de velocidad, ángulos y tiempos, garantizando una ejecución fluida, segura y coherente con las intenciones del diseño original.

Finalmente, la elaboración del informe técnico en Overleaf bajo el formato IEEE permitió organizar el trabajo de manera profesional. La inclusión de imágenes, código comentado y referencias normativas favorece la claridad en la presentación del contenido y respalda el cumplimiento de los estándares académicos exigidos. Esta experiencia fortalece las competencias en documentación técnica, esenciales en el ámbito de la ingeniería electrónica y la robótica.

REFERENCIAS

[1] J. Lozano, Ïntroducción a la Robótica Humanoide", Ediciones USTA, 2016.
[2] P. García, "Programación avanzada con Python para robots", Alfaomega, 2019.
[3] M. Torres, Çonectividad en sistemas embebidos", Universidad Nacional, 2018.
[4] S. Rosales, Çálculo geométrico aplicado a robots", Editorial Robótica, 2025.
[5] R. Navarro, "Matemáticas computacionales en robótica", Pearson, 2015.
[6] F. Cuofano, Robot Motion Programming: Theory and Practice", Robotics Press, 2024.
[7] Observatorio de Software Libre, Çonectividad HTTP en plataformas robóticas", 2025.
[8] MindOnMap, "Data Handling in Robot Systems", Online Documentation, 2023.
[9] SoftBank Robotics, "Pepper: The humanoid robot for your business,"SoftBank Robotics, 2018.
[10] SoftBank Robotics, Çhoregraphe Suite: Documentation and Tutorials,"SoftBank Robotics, 2017.
[11] Canonical, Übuntu: The open-source operating system,Çanonical, 2023.
[12] LIG Inc., Çhoregraphe Example,"Disponible en: https://liginc.co.jp/wp-content/uploads/2015/12/ch.jpg.
[13] Vidabytes, Übuntu Example,"Disponible en: https://vidabytes.com/wp-content/uploads/2022/02/ubuntu.jpg.
[14] Instagram, Representación de Facultad de Ingeniería Electrónica Santoto,"Disponible en: https://www.instagram.com/p/CzB7VNXLmc7/?img_index=1.