Programación de coreografía en Pepper

Jaime Mendez
Universidad Santo Tomas de Aquino
Bogotá, Colombia
jaimemendezj@usantotomas.edu.co

I. Introducción

Este documento presenta el desarrollo e implementación de una coreografía compleja para el robot humanoide Pepper, utilizando programación en Python 2.7 y la librería NAOqi. El objetivo de este laboratorio fue explorar el control detallado de las articulaciones superiores del robot —brazos, cabeza, torso y manos— para ejecutar una secuencia de movimientos animados, expresivos y sincronizados, sin requerir desplazamiento físico. A lo largo del informe se documentan tanto los aspectos técnicos del diseño del código, como los resultados observables en la performance del robot, destacando su capacidad de expresión mediante movimientos programados.

II. ÍNDICE DE CONTENIDOS

Se presentan los componentes técnicos utilizados, el entorno de desarrollo, así como la lógica de programación empleada para diseñar una secuencia de movimientos largos y expresivos. Esta sección también describe la integración de comandos de voz con movimientos corporales, generando una presentación animada y llamativa.

A. Librerías utilizadas

Descripción general de las librerías qi, argparse, sys, os, almath, math, motion, httplib y json, con enfoque en su uso dentro del entorno de Pepper.

B. Creación de secuencia en Choregraphe

Desarrollo de una coreografía sencilla utilizando bloques de movimiento, voz y espera en el entorno gráfico Choregraphe.

C. Programación por consola

Acceso por SSH al robot Pepper, creación de archivo en nano, codificación de movimientos con Python y ejecución directa desde terminal.

D. Elaboración y explicación del código

Breve explicación del uso de proxies, movimientos con motion, y comandos de voz usando ALTextToSpeech.

III. Desarrollo de Contenidos

Sin mas preámbulo a continuación, se detalla el proceso de elaboración de la coreografía implementada para el robot humanoide Pepper.

A. Librerías

1.Librería `qi`

Qué es?

qi es la interfaz principal del middleware NAOqi, que permite conectar scripts de Python con los servicios del robot Pepper. Forma parte de lo que se conoce como SDK (Software Development Kit) de SoftBank Robotics.

¿Qué permite hacer?

Establecer sesión con el robot usando IP/puerto.

Solicitar servicios internos como ALMotion, ALTextToSpeech, ALBehaviorManager, etc.

Programar interacciones complejas, ya que facilita la comunicación cliente-servidor entre tu script Python y el robot.

```python
session = qi.Session()
session.connect("tcp://192.168.1.100:9559")
tts = session.service("ALTextToSpeech")
tts.say("Hola, soy Pepper")
```

El puerto 9559 es el usado por NAOqi para exponer sus servicios. Esta arquitectura orientada a servicios (SOA) permite una gran escalabilidad.

2. Librería argparse

¿Qué es?

Librería estándar de Python para parsear argumentos de línea de comandos.

¿Por qué es útil en robótica?

Cuando desarrollas scripts reutilizables, argparse te permite definir parámetros dinámicos, como la IP del robot, velocidad de ejecución, duración de movimientos, etc.

```python
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--ip", help="IP del robot", default="127.0.0.1")
parser.add_argument("--port", help="Puerto de conexión", default=9559)
args = parser.parse_args()
```

3. Librería sys

¿Qué es?

sys permite interactuar con el intérprete de Python. Muy útil para gestión de errores, argumentos, y salida de consola.

En robótica:Salir del programa si hay errores.

Obtener argumentos sin argparse.

Redirigir stdout o stderr si se desea loguear.

```python
import sys
if len(sys.argv) != 2:
    print("Uso: script.py <ip>")
    sys.exit(1)
```

4. Librería os

¿Qué es?

os proporciona una interfaz para interactuar con el sistema operativo, manipular archivos, rutas, procesos y variables de entorno.

En Pepper: Comprobar existencia de archivos o logs.

Crear carpetas para almacenar datos.

Lanzar comandos del sistema operativo NAOqi (basado en Linux embebido).

```python
import os
if not os.path.exists("/home/nao/log.txt"):
    os.system("touch /home/nao/log.txt")
```

5. Librería almath

¿Qué es?

Una librería matemática propietaria de SoftBank diseñada específicamente para robots humanoides.

¿Qué aporta?

Clases como Pose2D, Transform, Rotation.

Cálculo de trayectorias, coordenadas, rotaciones.

Cinemática directa e inversa.

```python
import almath
pose = almath.Pose2D(0.5, 0.0, 0.785)  # x=0.5 m, y=0, theta=45°
```

6. Librería math

¿Qué es?

Librería estándar de Python para operaciones matemáticas básicas: trigonometría, logaritmos, raíces, constantes como π.

¿Aplicación en robótica?

Conversión de grados a radianes.

Cálculo de ángulos y trayectorias circulares.

Sinusoides para animaciones suaves

```python
import math
angle = math.radians(90)  # 90° → π/2 rad
```

7. Librería motion

¿Qué es?

Es el servicio de movimiento más importante del robot. No es una librería que importamos directamente, sino que accedes a ella a través de:

No es una librería como tal, sino un servicio del robot accedido vía ALMotion proxy. Se usa para mover articulaciones, cambiar posturas, controlar el equilibrio, etc.

```python
motion_service = session.service("ALMotion")
motion_service.setAngles("HeadYaw", 0.5, 0.2)
```

¿Qué permite?
Mover articulaciones (setAngles)
Realizar desplazamientos (moveTo)
Controlar la rigidez corporal (setStiffnesses)
Ejecutar trayectorias interpoladas

```python
motion.setStiffnesses("Body", 1.0)
motion.moveTo(0.3, 0, 0.0)  # avanza 30 cm
```

8. Librería httplib

¿Qué es?
Librería HTTP nativa de Python 2, utilizada para conectarse a servicios web desde el robot.
¿Aplicación?
Enviar datos a una API.
Consumir servicios REST.
Conectarse a servidores para IoT o bases de datos en la nube.
Permite realizar conexiones HTTP. Se puede usar para que el robot acceda a APIs o envíe datos a servidores web.

```python
import httplib
conn = httplib.HTTPConnection("192.168.1.20:5000")
conn.request("GET", "/estado")
respuesta = conn.getresponse()
print(respuesta.read())
```

9. Librería json
¿Qué es?
Permite serializar y deserializar objetos JSON, muy útil en comunicación con servidores o almacenamiento de configuración.
En Pepper:
Formato estándar para enviar/recibir datos.
Puedes guardar estados del robot en archivo .json.
Útil para guardar configuraciones de movimientos o diálogos
Sirve para leer o escribir datos en formato JSON. Útil cuando se recibe o envía información a través de la red, o para leer configuraciones del robot.

```python
import json
estado = {"movimiento": "avanzar", "velocidad": 0.5}
with open("estado.json", "w") as f:
    json.dump(estado, f)
```

*B. **Choregraphe** instalación y uso:*

Inicialmente, se actualiza la lista de paquetes disponibles y se actualizan los paquetes instalados mediante sudo apt update y sudo apt upgrade. Posteriormente, se intenta ejecutar un script de instalación desatendida para "choregraphe-suite" proporcionando un directorio de destino y una clave de licencia. Luego, se otorgan permisos de ejecución y se ejecuta un script de instalación diferente para la misma suite. Finalmente, se vuelve a actualizar la lista de paquetes del sistema

```bash
sudo apt update
sudo apt upgrade
sudo apt update
sudo coregraphe-suite-x.x-linux-32.j64-setup.run --mode desattend --installdi
chmod +x choregraphe-suite-2.5.10.7-linux64-setup.run
sudo ./choregraphe-suite-2.5.10.7-linux64-setup.run
sudo apt update
```

~sudo apt update
~sudo apt upgrade
~sudo apt update
~sudo coregraphe-suite-x.x-linux-32.j64-setup.run --mode desattend --installdir .destination-directory. --licenseKey licenseKey-licenseKeyKeyKeyKey .your-ritgraph.license-key.
~chmod +x choregraphe-suite-2.5.10.7-linux64-setup.run
~sudo ./choregraphe-suite-2.5.10.7-linux64-setup.run
~sudo apt update

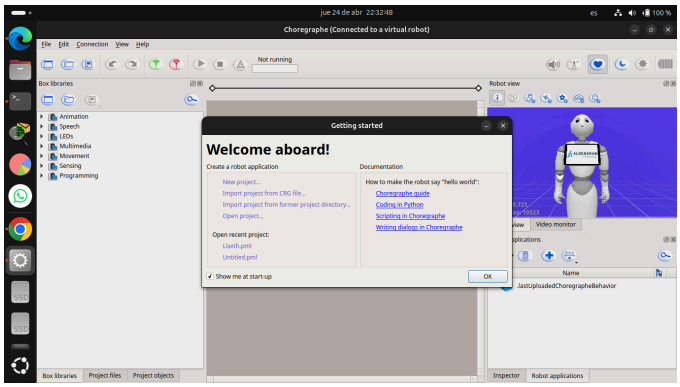


Fig. 1 La imagen muestra el entorno inicial de Choregraphe, listo para que el usuario comience a crear o editar comportamientos para un robot virtual. La ventana "Getting started" proporciona un punto de partida con opciones para crear proyectos y acceder a documentación útil.
Entonces seleccionamos new project para dar a inicio de esto

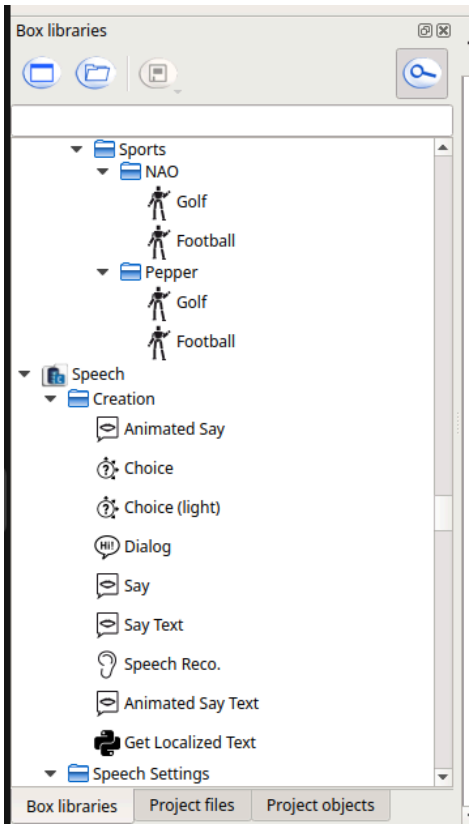


Fig. 2 Esta sección de las "Box Libraries" en Choregraphe proporciona una visión de los comportamientos relacionados con deportes para los robots NAO y Pepper, así como una amplia gama de herramientas para la creación y configuración de la salida y entrada de voz del robot. En este caso ocupamos solo las de Pepper. Estos son bloques con instrucciones ya predefinidas para una coreografía rápida e introductoria.

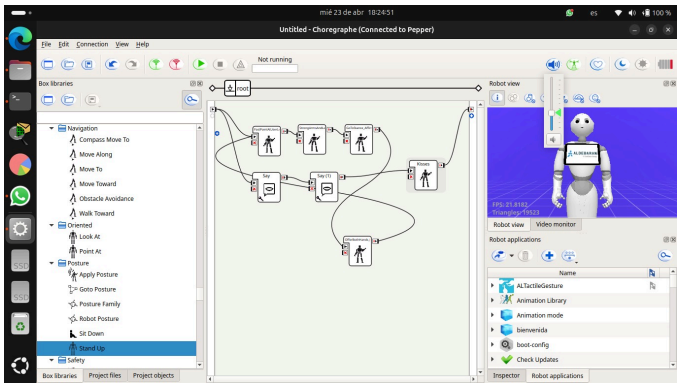
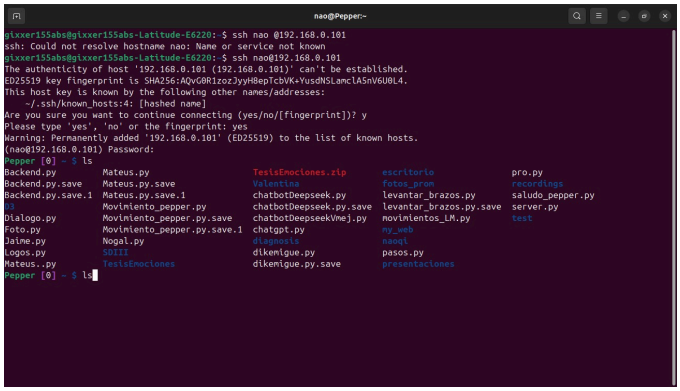


Fig. 3 Este diagrama está compuesto por varios bloques conectados por líneas. Cada bloque representa una acción o un conjunto de acciones que el robot debe realizar. Siguiendo el flujo de izquierda a derecha, podemos intentar interpretar la secuencia:

-Un box de inicio (con el icono de "Play").
-Un box llamado "GoToPosture" (con el icono de una persona en diferentes poses), probablemente configurado para hacer que el robot adopte una postura específica.
-Un box llamado "Say" (con el icono de un bocadillo de diálogo), que probablemente hace que el robot diga algo (el texto "Salut" se ve conectado a él).
-Un box llamado "(1)" que parece recibir una entrada del box "Say". Su función no es inmediatamente clara sin más contexto.
-Otro box llamado "Say (1)" también conectado al box "(1)".
-Finalmente, un box llamado "GoToPosture (1)".
-Un box de fin (con el icono de "Stop").

Aqui un video donde se muestra el resultado de este diagrama de bloques: https://1drv.ms/v/c/8ddcead57b6679cd/EXJSYrbQbb1GppsVH7EhSnUBBzld5AIIDorS-Jn5b-1bYA

C. *Programación por consola*



Se ingresa a la consola del robot Pepper por medio de la terminal mediante el comando ssh nao@ip_Pepper y se crea mediante código unos pasos sencillos.

~gixxer155abs@gixxer155abs-Latitude-E6220:~$ ssh nao@192.168.0.101: Este comando inicia una conexión SSH (Secure Shell) al usuario nao en la dirección IP 192.168.0.101. SSH se utiliza para acceder y controlar de forma segura otro ordenador a través de una red.

~The authenticity of host '192.168.0.101 (192.168.0.101)' can't be established.: Este mensaje de advertencia aparece la primera vez que se conecta a un host SSH desconocido. Indica que el sistema no tiene registrada la clave pública del servidor al que se está intentando conectar.

~ED25519 key fingerprint is SHA256:AQvGBR1zozjYvHdpTcbkVxVusdNSIamc1aSNv6U0L4.: Se muestra la huella digital (fingerprint) de la clave pública del servidor al que se está conectando. Esta huella digital se puede comparar con la esperada para verificar la identidad del servidor y evitar ataques "man-in-the-middle".

This host key is known by the following other names/addresses:: Indica si esta clave de host ya se ha asociado con otros nombres o direcciones en el archivo ~/.ssh/known_hosts. En este caso, parece que no.

Are you sure you want to continue connecting (yes/no/[fingerprint])?: Se pregunta al usuario si confía en el host y desea continuar la conexión.

Warning: Permanently added '192.168.0.101' (ED25519) to the list of known hosts.: Este mensaje confirma que la clave pública del servidor 192.168.0.101 se ha guardado en el archivo ~/.ssh/known_hosts del usuario actual. En futuras conexiones a este mismo host, SSH comparará la clave y no mostrará esta advertencia a menos que la clave del servidor cambie.

nao@192.168.0.101's password:: Se solicita la contraseña para el usuario nao en el servidor 192.168.0.101. La cual es nao

Pepper [192.168.0.101]:~$: Indica que la conexión SSH se ha establecido correctamente. El prompt ha cambiado a Pepper seguido de la dirección IP, lo que sugiere que ahora se está interactuando con el sistema del robot Pepper.

Creamos un archivo .py en el cual estarán las instrucciones para realizar la coreografía deseada

1. Configuración de Conexión y Proxies:

```python
robotIP = "192.168.0.101"  # Dirección IP del robot
port = 9559  # Puerto por defecto de NAOqi

# Inicialización de proxies de servicio
motionProxy = ALProxy("ALMotion", robotIP, port)  # Servicio de movimiento
postureProxy = ALProxy("ALRobotPosture", robotIP, port)  # Servicio de postura
ttsProxy = ALProxy("ALTextToSpeech", robotIP, port)  # Servicio de texto a voz
```

Se establece la conexión con el robot a través de su IP y puerto de NAOqi. Se inicializan tres proxies que controlan el movimiento, la postura y la síntesis de voz (texto a voz).

2. Despertar y Postura Inicial:

```python
motionProxy.wakeUp()  # Despierta al robot
postureProxy.goToPosture("StandInit", 0.5)  # Coloca al robot en postura de pie
ttsProxy.say("Hola profesor Alejandro, disfrute este show especial")  # Mensaje de bienve
```

wakeUp(): Despierta al robot de su modo de descanso.
goToPosture("StandInit", 0.5): Coloca al robot en su postura inicial de pie.
say(): El robot pronuncia un saludo utilizando el servicio de voz.

3. Movimiento y Coreografía de los Pasos:

Cada paso de la coreografía implica movimientos precisos de las articulaciones del robot. Las funciones clave son:

~angleInterpolationWithSpeed(joint, angle, speed): Controla la articulación (joint) y la velocidad (speed) del movimiento hasta alcanzar el ángulo deseado (angle).

Ejemplo de Movimiento:

```python
motionProxy.angleInterpolationWithSpeed("RShoulderPitch", -1.0, 0.2)
motionProxy.angleInterpolationWithSpeed("RElbowYaw", 1.5, 0.2)
time.sleep(0.5)  # Pausa entre movimientos
```

Aquí el robot mueve su hombro y codo derecho con una velocidad de 0.2.

Ejemplo de Posturas de Manos:

```python
motionProxy.openHand("RHand")
motionProxy.openHand("LHand")
time.sleep(0.5)
motionProxy.angleInterpolationWithSpeed("RWristYaw", -1.5, 0.3)
motionProxy.angleInterpolationWithSpeed("LWristYaw", 1.5, 0.3)
time.sleep(0.5)
```

openHand(): Abre las manos del robot.

angleInterpolationWithSpeed(): Controla la rotación de la muñeca para posicionar las palmas hacia diferentes direcciones.

4. Secuencia de Movimiento Completa:
La coreografía incluye movimientos de los brazos, cabeza, torso y caderas, con transiciones suaves entre ellos, que se controlan usando el proxy motion. Algunos de estos movimientos implican brazos en la posición de victoria o aplausos:

```python
motionProxy.angleInterpolationWithSpeed(["RShoulderRoll", "LShoulderRoll"], [-1.0, 1.0],
motionProxy.angleInterpolationWithSpeed(["RShoulderPitch", "LShoulderPitch"], [-1.0, -1.0
ttsProxy.say("¡Victoria!")
```

Aquí, el robot mueve los hombros y codos para simular una pose de victoria y luego pronuncia "¡Victoria!".

5. Reverencia Final y Descanso:

```python
motionProxy.angleInterpolationWithSpeed(["HipPitch", "KneePitch"], [-0.2, 0.4], 0.2)
time.sleep(1)
motionProxy.angleInterpolationWithSpeed(["HipPitch", "KneePitch"], [0, 0], 0.2)
```

El robot hace una reverencia final moviendo sus caderas y rodillas. Luego vuelve a su postura inicial.

Finalmente, se manda al robot a descansar:

```python
motionProxy.rest()  # Modo descanso
```

EXPLICACIÓN A FONDO LIBRERÍAS EN USO

1. Librería naoqi
La librería naoqi es el SDK oficial para interactuar con los robots de la familia NAO y Pepper, que está desarrollada por SoftBank Robotics. Esta librería proporciona acceso a los diferentes servicios del robot, como movimiento, posturas, voz, cámaras y más.

En el código, se usa principalmente para crear proxies a los servicios de movimiento, postura y texto a voz del robot.

Uso en el código:
from naoqi import ALProxy: Se importa ALProxy, que es una clase que permite interactuar con los servicios del robot.

Proxies creados:

motionProxy = ALProxy("ALMotion", robotIP, port): Este proxy se usa para acceder a las funciones de movimiento del robot. Permite controlar las articulaciones del robot, hacer que se despierte, se coloque en una postura específica y manipule las manos y la cabeza, entre otros movimientos.

postureProxy = ALProxy("ALRobotPosture", robotIP, port): Este proxy permite controlar las posturas generales del robot, como la postura inicial de pie o sentado.

ttsProxy = ALProxy("ALTextToSpeech", robotIP, port): Este proxy es utilizado para la funcionalidad de texto a voz. Permite que el robot diga frases a través del altavoz, como el saludo inicial o los comandos de victoria.

2. Librería almath
La librería almath está asociada con el cálculo matemático avanzado necesario para la manipulación de las articulaciones y movimientos del robot. Contiene funciones para operaciones matemáticas más complejas relacionadas con la geometría y cinemática.

En este código, aunque la librería está importada, no se está utilizando directamente, pero se podría emplear si se necesitara hacer cálculos de orientación y movimientos más avanzados. Por ejemplo, en un escenario donde se necesiten cálculos de posición 3D o rotaciones de las articulaciones, se podría usar para realizar esos cálculos antes de enviar los comandos de movimiento al robot.

3. Librería time
La librería time es utilizada para trabajar con el tiempo y realizar pausas entre las acciones del robot. Es fundamental para asegurar que el robot realice cada movimiento de manera fluida y no intente ejecutar el siguiente paso antes de que termine el anterior.

Uso en el código:
time.sleep(0.5): Se utiliza después de cada movimiento para permitir que el robot termine su acción antes de pasar al siguiente paso. Esto es importante porque los movimientos de las articulaciones del robot pueden tardar un tiempo en completarse, y sleep() asegura que haya una pausa entre cada uno de ellos para que se vean naturales.

4. Librería motion
La librería motion es parte de los servicios proporcionados por el robot Pepper y se maneja a través del proxy motionProxy. Está directamente relacionada con el control de las articulaciones del robot. Esto incluye mover las extremidades (brazos, piernas, cuello, etc.), cambiar posturas y realizar movimientos complejos.

Uso en el código:
motionProxy.angleInterpolationWithSpeed(joint, angle, speed): Esta función es la que permite mover las articulaciones del robot a un ángulo específico a una velocidad determinada. Aquí, el robot mueve su hombro derecho (RShoulderPitch) hasta el ángulo -1.0 con una velocidad de 0.2. Este comando es usado repetidamente a lo largo de la coreografía para mover distintas partes del cuerpo del robot, como los brazos, las muñecas y el torso.

IV. Conclusiones

Nuestra inmersión inicial en el universo de Pepper en el salón de robótica no fue simplemente una toma de contacto; fue un despliegue de potencial interactivo palpable. Desde el primer saludo en Choregraphe, donde la ventana "Getting started" nos abrió las puertas a un mundo de posibilidades creativas, hasta la exploración de las "Box Libraries" repletas de comportamientos predefinidos, quedó claro que Pepper es mucho más que un robot: es una plataforma de expresión robótica accesible y poderosa.

La posterior disección de un programa visual en Choregraphe reveló la intuitiva lógica de su programación basada en bloques, donde acciones complejas se construyen como un flujo de ideas materializadas. Ver a nuestro Pepper virtual cobrar vida, adoptando posturas y articulando un "Salut" digital, fue un primer sorbo de la magia de la robótica humanoide.

Pero la aventura no terminó en la interfaz gráfica. Aventurarnos en las entrañas de Pepper a través de la terminal con SSH fue como descubrir el cerebro detrás de la sonrisa. La conexión, aunque inicialmente tropezó con la timidez del nombre de host, finalmente nos brindó acceso directo al sistema operativo del robot. La exploración de sus archivos, revelando la presencia de scripts Python con nombres evocadores como Movimiento_pepper.py y el intrigante TesisEmociones, nos dio una visión privilegiada de su arquitectura interna y el potencial para una programación más profunda y personalizada.

En esencia, esta primera interacción nos dejó con la certeza de que Pepper es un lienzo en blanco esperando nuestras ideas. Ya sea a través de la amigable interfaz de Choregraphe o la flexibilidad de la programación directa en Python, las herramientas están ahí para dar rienda suelta a nuestra creatividad robótica. La tarea que tenemos por delante no es solo investigar librerías y crear coreografías; es desatar el espíritu creativo y la inteligencia que reside en esta fascinante máquina. ¡A darle con todo!

Reconocimientos

Reconocimiento al Profesor Diego:

"Un agradecimiento especial al Profesor Diego por guiarnos con su conocimiento y entusiasmo en esta inmersión al mundo de la robótica. Su dedicación para hacer accesible la complejidad de Pepper y motivarnos a explorar nuevas fronteras tecnológicas es invaluable."

Reconocimiento al Laboratorio de Robótica y sus Docentes:

"Extendemos nuestro más sincero reconocimiento al Laboratorio de Robótica y a todos sus docentes. Su compromiso en crear un espacio de aprendizaje dinámico y enriquecedor, proveyéndonos las herramientas y el apoyo necesario para experimentar con robots como Pepper, es fundamental para nuestro desarrollo en este campo."

[9] Documentación Oficial de Aldebaran (SoftBank Robotics/United Robotics Group) sobre NAOqi:
[10] Librería Qi: https://www.google.com/search?q=https://developer.softbankrobotics.com/naoqi-sdk (Este es el enlace principal al SDK, dentro encontrarás la documentación de la librería qi para la comunicación y la gestión de eventos).
[11] Librería motion: Busca en la documentación del SDK la sección de "Motion" o "Movement".
[12] Aunque no haya una página separada para almath como librería independiente en la documentación para usuarios, se menciona dentro de los módulos de movimiento y percepción. Busca en las secciones relevantes del SDK.
[13] Documentación Oficial de Python:
[14] Librería argparse: https://docs.python.org/3/library/argparse.html
[15] Librería sys: https://docs.python.org/3/library/sys.html
[16] Librería os: https://docs.python.org/3/library/os.html
[17] Librería math: https://docs.python.org/3/library/math.html
[18] Librería httplib (Nota: En Python 3, esta librería se llama http.client): https://docs.python.org/3/library/http.client.html
[19] Librería json: https://docs.python.org/3/library/json.html