

Tarea 3
Sistemas operativos
Brayan Poloche- Karen Garcia

- **Investigación de las librerías.**

1. Librería qi.

a. ¿Qué es?

- Es la librería principal para controlar a Pepper.

b. Funcionalidades clave:

- Acceso a los servicios internos de Pepper (movimiento, habla, visión y sensores).
- Permite establecer comunicación entre un script externo y Pepper.
- Control remoto desde un computador o dispositivo externo.

2. Librería argparse.

a. ¿Qué es?

- Es la librería de Python para leer argumentos desde la terminal.

b. Funcionalidades clave:

- Se usa para leer parámetros desde la línea de comandos.
- Facilita la creación de scripts configurables desde la consola.
- Ideal para automatizar tareas con opciones personalizables.

3. Librería sys.

a. ¿Qué es?

- Es la librería que permite interactuar con el sistema y el intérprete de Python.

b. Funcionalidades clave:

- Acceso a argumentos de la línea de comandos (sys.argv).
- Control del flujo del programa.
- Manipulación de rutas y módulos.
- Configuración del intérprete.

4. Librerías os.

a. ¿Qué es?

- Es la librería que permite interactuar con el sistema operativo.

b. Funcionalidades clave:

- Permite realizar operaciones relacionadas con el sistema operativo, como trabajar con archivos o carpetas.
- Se utiliza para ejecutar comandos del sistema directamente desde Python.
- Ayuda a crear rutas dinámicas, acceder a variables de entorno o comprobar si un archivo existe.
- Es útil para automatizar tareas del sistema desde un script.
- Facilita el control del entorno de trabajo del script (ubicación, permisos, estructuras de carpetas).

5. Librería almath.

a. ¿Qué es?

- Es la librería de SoftBank que contiene funciones matemáticas avanzadas para robótica.

b. Funcionalidades clave:

- Se usa para realizar cálculos geométricos y espaciales.
- Ofrece funciones para trabajar con transformaciones 3D, vectores y trayectorias.
- Es clave para calcular movimientos complejos de Pepper en el espacio.
- Se utiliza para analizar posiciones relativas o generar trayectorias suaves.
- Facilita el manejo de coordenadas y orientaciones en entornos tridimensionales.

6. Librería math.

a. ¿Qué es?

- Es la librería matemática estándar de Python.

b. Funcionalidades clave:

- Proporciona herramientas matemáticas básicas como trigonometría o raíces cuadradas.
- Se usa para realizar cálculos numéricos dentro del script.
- Sirve para manejar constantes como π o realizar conversiones entre ángulos.
- Es una librería fundamental para el análisis y modelado de fenómenos físicos.

7. Librería motion.

a. ¿Qué es?

- Es la librería interna de Pepper para controlar el movimiento de Pepper.

b. Funcionalidades clave:

- Permite controlar el movimiento de Pepper, como caminar, mover brazos o cambiar de postura.
- Se usa para enviar comandos de desplazamiento o manipulación de partes del cuerpo de Pepper.
- Se puede usar para generar secuencias de movimientos coordinados.

8. Librería httpplib.

a. ¿Qué es?

- Es la librería para hacer conexiones HTTP (comunicarse por internet). En python 3 es http.client.

b. Funcionalidades clave:

- Permite que el script obtenga o envíe datos desde o hacia servicios en línea.
- Se utiliza para consumir APIs o servicios REST externos.
- Es útil cuando Pepper necesita interactuar con plataformas en la nube o bases de datos externas.
- Sirve para establecer comunicación cliente-servidor desde Python.

9. Librería json.

a. ¿Qué es?

- Es la librería para trabajar con el formato JSON (JavaScript Object Notation).

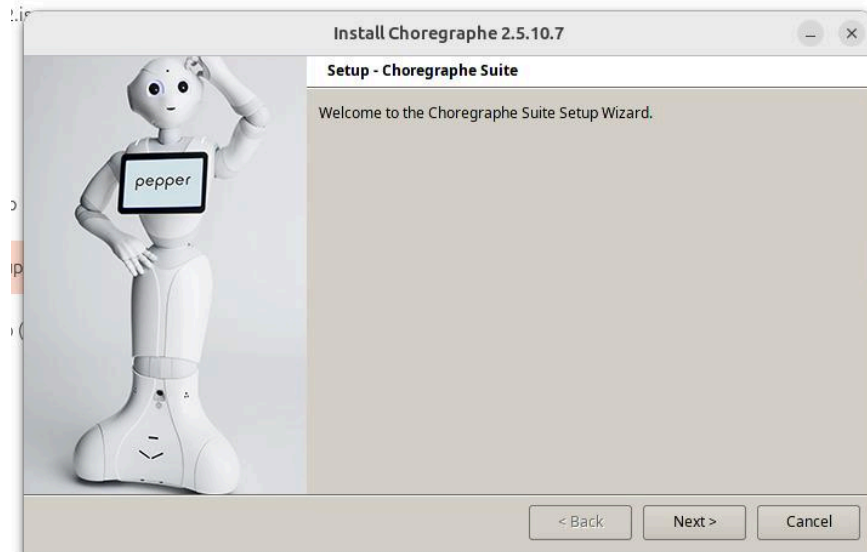
b. Funcionalidades clave:

- Permite codificar y decodificar información en formato JSON.
- Es útil para guardar configuraciones, datos o mensajes estructurados.
- Se usa para intercambiar información con otros sistemas o servicios web.
- Facilita la conversión de datos entre texto y objetos Python.
- Es común en la integración de software cuando se necesitan estructuras claras y compatibles.

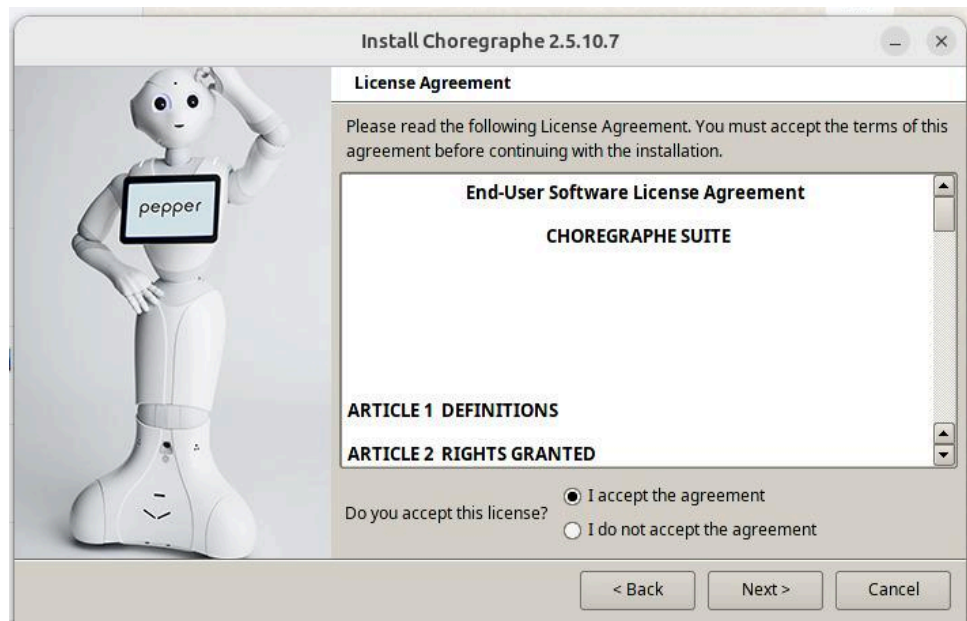
● Instalación y coreografía con Choregraphe.

- Instalación de Choregraphe.

1. Se descarga el instalador de la versión 2.5.10.7 desde la documentación oficial, disponible en: http://doc.aldebaran.com/2-5/home_pepper.html.
2. Se ejecuta el instalador descargado y aparece la siguiente pantalla de bienvenida



3. Le damos **“Next”** para seguir con su instalación y sale la siguiente ventana de aceptación de términos de las licencias.



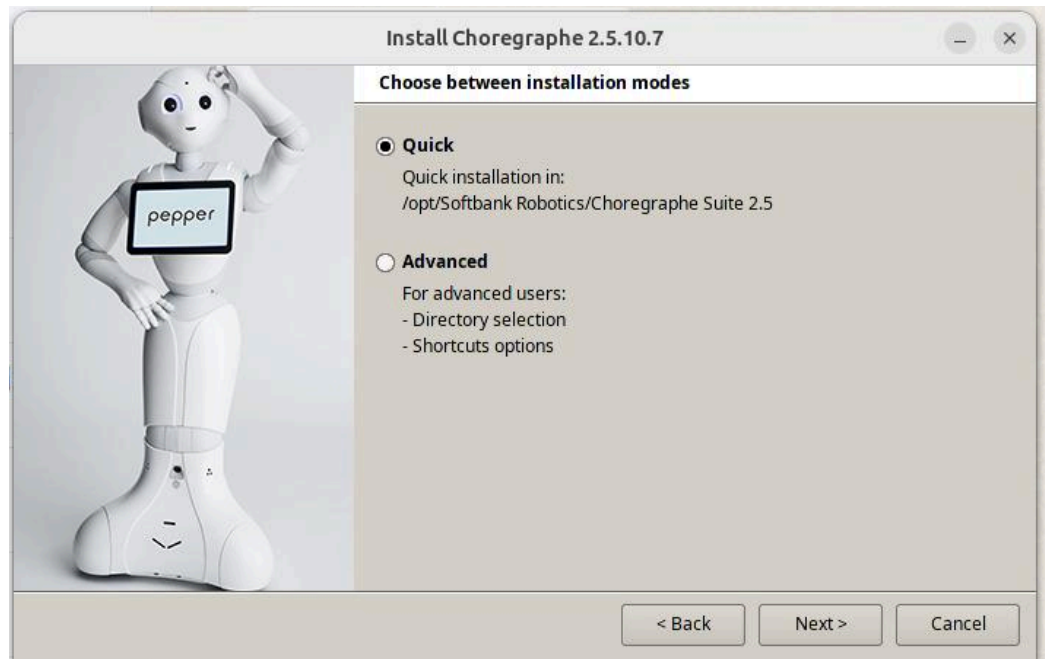
4. Aceptamos los términos de la licencia y damos **“Next”**. Luego sale la ventana donde se elige entre dos modos de instalación:

- Quick (instalación rápida):

Esta opción instala el software automáticamente en la ruta predeterminada del sistema: ****/opt/Softbank Robotics/Choregraphe Suite 2.5****

- Advanced (instalación avanzada):

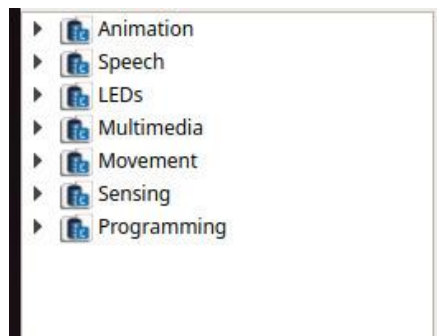
Permite configurar parámetros personalizados como: Carpeta de destino y creación de accesos directos o configuraciones específicas del sistema.



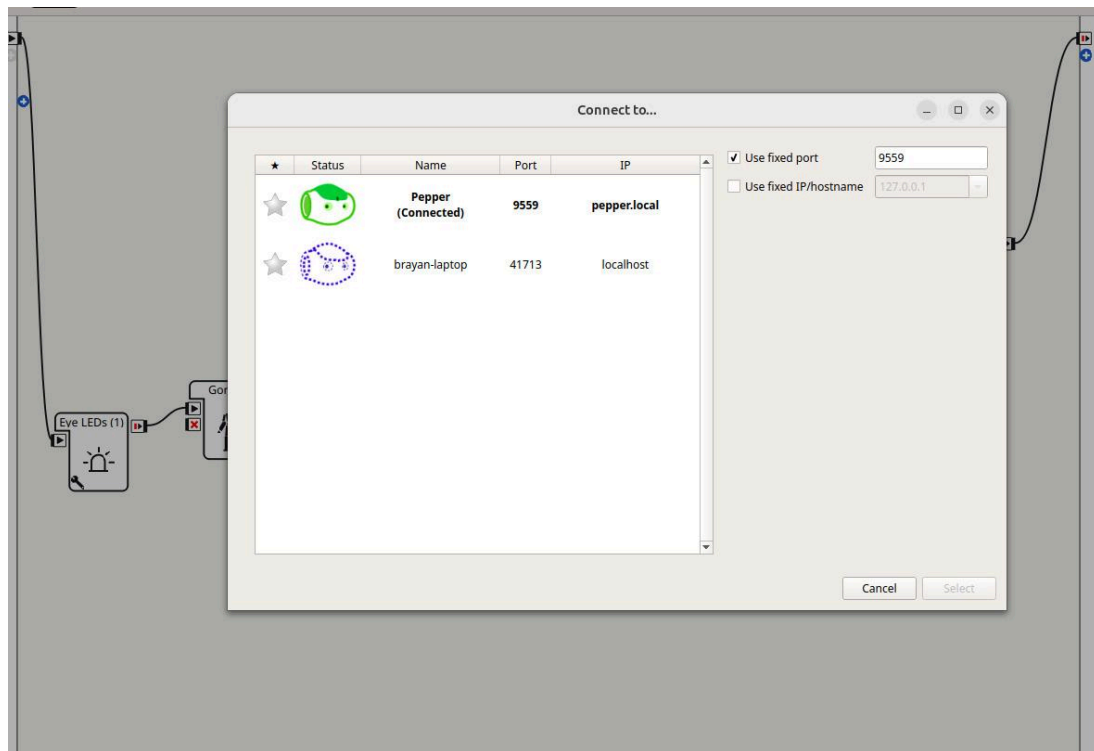
5. Se eligió la instalación rápida y se dio **“Next”**, para que finalmente se realice la instalación.

- Pasos de la coreografía.

1. Se abre Choregraphe, en la parte izquierda, se visualizan los bloques de acciones organizados de forma secuencial o paralela, con estas categorías: Animation, Speech, LEDs, Multimedia, Movement, Sensing, y Programming.



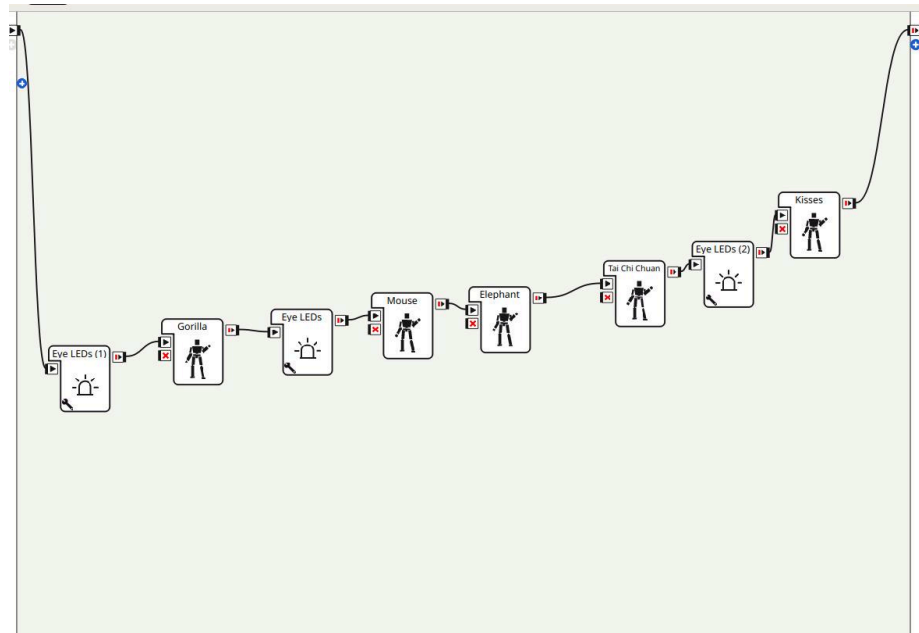
2. Desde el menú de conexión, se selecciona Pepper Pepper que aparece como conectado a través del puerto 9559. Esto permite enviar directamente las acciones programadas para su simulación.



3. Se crea una secuencia de bloques que representan las acciones específicas que ejecutará Pepper. La estructura usada en este caso es lineal, y los bloques están conectados entre sí. La coreografía incluye:

- Encendido de LEDs oculares (Eye LEDs 1): Se ilumina la cabeza de Pepper como introducción visual.
- Animación "Gorilla": Pepper realiza una animación que simula los movimientos de un gorila.
- Cambio de LEDs oculares: Se actualiza el color o el patrón de luz en los ojos de Pepper.
- Animación "Mouse": Pepper actúa como un ratón.
- Animación "Elephant": Movimiento que simula la trompa de un elefante.
- Animación "Tai Chi Chuan": Secuencia de movimientos suaves simulando el arte marcial.
- Cambio de LEDs oculares (Eye LEDs 2): Se vuelve a cambiar la luz en los ojos para acompañar la siguiente acción.

- Animación "Kisses": Pepper finaliza con un gesto de enviar besos.



4. Antes de ejecutar físicamente, se utiliza el visor 3D de Pepper (Robot View) en Choregraphe para visualizar cómo Pepper realizará la coreografía. Esto permite corregir errores y ajustar tiempos entre bloques si es necesario.



5. Una vez verificada la coreografía, se apaga el corazón de Pepper, se carga y se ejecuta la secuencia. Pepper realiza la coreografía programada, combinando movimientos corporales, gestos animados y cambios en la iluminación de sus ojos.

Video: 📺 Coreografía .mp4

- **Coreografía de Pepper por medio de la terminal.**

1. Se utiliza el comando SSH para conectarse a Pepper mediante su dirección IP local.

```
brayan@brayan-laptop:~$ ssh nao@192.168.0.100
```

- **ssh** → Establece una conexión remota y segura con el computador utilizando el protocolo SSH .
- **nao** → Es el nombre del usuario en el sistema remoto (Pepper).
- **192.168.0.100**→ Es la dirección IP asignada al robot Pepper dentro de la red local.

2. Al ser la primera vez que se conecta a esta IP, el sistema muestra una advertencia indicando que no se puede verificar la autenticidad del host. Se acepta manualmente la conexión escribiendo yes.

```
brayan@brayan-laptop:~$ ssh nao@192.168.0.100
The authenticity of host '192.168.0.100 (192.168.0.100)' can't be established.
ED25519 key fingerprint is SHA256:AQvG0R1zozJyyH8epTcbVK+YusdNSLamclA5nV6U0L4.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:2: [hashed name]
  ~/.ssh/known_hosts:3: [hashed name]
  ~/.ssh/known_hosts:4: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

3. Se solicita la contraseña del usuario para completar la conexión y se ingresa.

```
brayan@brayan-laptop:~$ ssh nao@192.168.0.100
The authenticity of host '192.168.0.100 (192.168.0.100)' can't be established.
ED25519 key fingerprint is SHA256:AQvG0R1zozJyyH8epTcbVK+YusdNSLamclA5nV6U0L4.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:2: [hashed name]
  ~/.ssh/known_hosts:3: [hashed name]
  ~/.ssh/known_hosts:4: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.100' (ED25519) to the list of known hosts.
(nao@192.168.0.100) Password:
```

4. Una vez dentro del sistema de Pepper, se ejecuta:

```
Pepper [0] ~ $ ls
Backend.py      Jain3.py      Mateus.py.save.1  TesisEmociones  chatbotDeepseekVneJ.py  fotos_gran  ny_web  saludo_pepper.py
Backend.py.save Jain4.py      Movimiento_pepper.py  TesisEmociones.zip  chatgpt.py             levantar_brazos.py  nao1    server.py
Backend.py.save.1 Logos.py     Movimiento_pepper.py.save  TesisUSTA.py       diagnosis              levantar_brazos.py.save  pasos.py  test
Dialogo.py      Mateus.py     Movimiento_pepper.py.save.1  Valentina         dickenigie.py         levantar_brazos.pyes  presentaciones
Foto.py          Mateus.py     Nogal.py             chatbotDeepseek.py  dickenigie.py.save    movimientos_LM.py    pro.py    recordings
                Mateus.py.save  S0111               chatbotDeepseek.py.save  escritorio             music
```

Esto muestra todos los archivos y carpetas disponibles, incluyendo varios scripts en Python.

- **ls** → Lista de todos los archivos y carpetas del directorio actual donde se encuentra el usuario dentro del sistema de Pepper.

5. Se abre el archivo con el editor de texto **nano** para editar el contenido del script:

```
Pepper [0] ~ $ nano levantar_brazos.py
```

- **nano levantar_brazos.py** → Abre el archivo levantar_brazos.py en el editor de texto nano para editar su contenido directamente desde la terminal.

6. Se abre el editor de texto nano y se coloca el código.

```
GNU nano 2.3.2 File: levantar_brazos.py
# -*- coding: utf-8 -*-
from naoqi import ALProxy
import time

# Crear proxies
motion = ALProxy("ALMotion", "localhost", 9559)
tts = ALProxy("ALTextToSpeech", "localhost", 9559)
animation_player = ALProxy("ALAnimationPlayer", "localhost", 9559)

# Despedida y descanso
tts.say("Hasta luego")
motion.rest()

time.sleep(2)

# Volver a levantarse
motion.wakeUp()
tts.say("Estoy listo otra vez")

# Despertar robot al inicio
motion.wakeUp()
tts.say("¡Hola mi socio!")

# Activar brazos
motion.setStiffnesses(["RArm", "LArm"], 1.0)

# Mover brazo derecho
right_arm_names = ["RShoulderPitch", "RShoulderRoll", "RElbowYaw", "RElbowRoll"]
right_arm_angles = [0.1, -0.3, 1.2, 1.0]
motion.setAngles(right_arm_names, right_arm_angles, 0.2)
time.sleep(2)

# Animación "Hey"
animation_player.run("animations/Stand/Gestures/Hey_1")
time.sleep(2)
```

```
GNU nano 2.3.2 File: levantar_brazos.py

motion.setAngles("HeadPitch", 0.0, 0.2)
time.sleep(1)

# Cabeza arriba y abajo dos veces
for i in range(2):
    motion.setAngles("HeadPitch", -0.3, 0.2)
    time.sleep(0.5)
    motion.setAngles("HeadPitch", 0.3, 0.2)
    time.sleep(0.5)

motion.setAngles("HeadPitch", 0.0, 0.2)

# Subir brazos
arms_names = ["RShoulderPitch", "LShoulderPitch", "RElbowRoll", "LElbowRoll"]
arms_up_angles = [-1.5, -1.5, 0.5, -0.5]
motion.setAngles(arms_names, arms_up_angles, 0.2)

tts.say("¡Manos al cielo mi gente!")
time.sleep(3)

# Bajar brazos lento
arms_down_angles = [1.5, 1.5, 1.0, -1.0]
motion.setAngles(arms_names, arms_down_angles, 0.05)
time.sleep(4)

for i in range(2):
    motion.setAngles("HeadPitch", -0.3, 0.2)
    time.sleep(0.5)
    motion.setAngles("HeadPitch", 0.3, 0.2)
    time.sleep(0.5)

# Relajar brazos
motion.setStiffnesses(["RArm", "LArm"], 0.0)
```

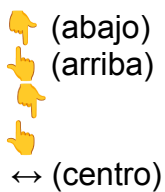
Donde:






- Encabezado e importaciones.
 - `# -*- coding: utf-8 -*-` → Especifica que el archivo usa codificación UTF-8, necesaria para que se lean correctamente los caracteres especiales (como acentos).
 - `from naoqi import ALProxy` → Importa la clase `ALProxy` de la librería `naoqi`, que permite la comunicación con los módulos de Pepper (como movimiento, habla, animaciones).
 - `import time` → Importa el módulo de tiempo para usar `sleep()`, que pausa la ejecución del programa por ciertos segundos.
- Crear proxies para controlar módulos
 - `motion` → Es el nombre de la variable que se usará para controlar el movimiento.

- `ALProxy` → Es una clase proporcionada por la librería naoqi, que se utiliza para conectarse con los módulos internos de Pepper.
 - `ALMotion` → Módulo que permite controlar todos los movimientos de Pepper, incluyendo brazos, piernas, cabeza, y postura.
 - `localhost` → Esta misma máquina.
 -
 - `9559` → Es el puerto de red en el que se está ejecutando el servidor de NAOqi en Pepper.
 - `tts` → Es la variable con la que se puede hacer que Pepper hable.
 - `ALTextToSpeech` → Módulo que permite a Pepper convertir texto en voz.
 - `animation_player` → Es la variable para acceder a esas animaciones.
 - `ALAnimationPlayer` → Módulo que permite ejecutar animaciones predefinidas del Pepper (como saludar, bailar, aplaudir, etc).
- Despedida y descanso de Pepper
 - `tts.say(...)` → Pepper habla.
 - `motion.rest()` → Pone al robot en posición de descanso (relaja todos los motores).
 - `time.sleep(2)` → Espera 2 segundos antes de continuar.
 - Despertar al robot
 - `motion.wakeUp()` → Activa los motores de Pepper (lo despierta)
 - Activar brazos
 - `motion.setStiffnesses(["RArm", "LArm"])` → Activa los motores de ambos brazos (derecho e izquierdo) para que puedan moverse.
 - Mover brazo derecho
 - `right_arm_names` → Es una lista que contiene los nombres de las articulaciones (joints) del brazo derecho de Pepper.
 - `"RShoulderPitch"` → Mueve el brazo hacia adelante y hacia atrás.

- "RShoulderRoll" → Mueve el brazo hacia los lados (abrir, cerrar, hacia afuera).
- "RElbowYaw" → Rota el codo.
- "RElbowRoll" → Flexiona o extiende el codo (doblarlo).
- right_arm_angles → Es una lista con los valores en radianes que se desean asignar a cada una de las articulaciones mencionadas antes.
 - 0.1 → rad = 5.7° (ligero movimiento hacia adelante del hombro).
 - -0.3 → rad ≈ -17.2° (ligero movimiento del hombro hacia abajo/lateral).
 - 1.2 → rad ≈ 68.8° (giro del codo).
 - 1.0 → rad ≈ 57.3° (flexión del codo).
- right_arm_names → Nombres de las articulaciones.
- right_arm_angles → Valores objetivo en radianes.
- 0.2 → Velocidad de movimiento, en un rango de 0.0 (muy lento) a 1.0 (muy rápido). Aquí, Pepper se moverá lentamente a los ángulos indicados.
- Ejecutar animación "Hey"
 - animation_player → Es un objeto que proviene del módulo ALAnimationPlayer, el cual se encarga de reproducir animaciones de Pepper.
 - run(...) → Es el método que inicia una animación. Tiene esta ruta "animations/Stand/Gestures/Hey_1" de la animación.
 - "Stand" → Pepper debe estar de pie.
 - "Gestures" → Se trata de un gesto.
 - "Hey_1" → Es un gesto que hace una especie de saludo (como levantar la mano y decir "hey").
- Cabeza arriba
 - motion → Es el objeto del proxy ALMotion, encargado de controlar los movimientos de las articulaciones de Pepper.
 - setAngles → Método que se usa para mover una o más articulaciones a posiciones específicas.

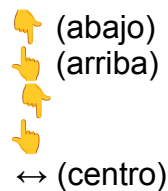
- "HeadPitch" → Es la articulación que mueve la cabeza hacia arriba o hacia abajo.
 - 0.0 → Ángulo objetivo en radianes (0 rad = cabeza recta, nivelada).
 - 0.2 → Velocidad fraccional del movimiento (de 0.0 a 1.0); en este caso, lenta y suave.
 - time.sleep(1) → Detiene el programa por 1 segundo, permitiendo que Pepper complete el movimiento de cabeza antes de continuar.
- Cabeza arriba y abajo dos veces
 - for i in range(2) → Ejecuta el bloque dentro del for dos veces (es decir, dos ciclos de movimiento).
 - motion.setAngles("HeadPitch", -0.3, 0.2) → Mueve la cabeza hacia abajo (-0.3 radianes $\approx -17^\circ$) a velocidad 0.2.
 - time.sleep(0.5) → Espera 0.5 segundos para permitir que el movimiento se vea natural y no demasiado rápido.
 - motion.setAngles("HeadPitch", 0.3, 0.2) → Mueve la cabeza hacia arriba (+0.3 radianes $\approx +17^\circ$), también a velocidad 0.2.
 - time.sleep(0.5) → Otra pausa para que Pepper mantenga ese movimiento y el gesto de "asentir" sea visible.
 - motion.setAngles("HeadPitch", 0.0, 0.2) → Después de completar los dos ciclos de asentir, devuelve la cabeza a la posición central (neutra) con suavidad (velocidad 0.2).
 - Resultado: Pepper realiza este patrón:



 (abajo)
 (arriba)


 (centro)
 - Subir ambos brazos
 - arms_names → Crea una lista con los nombres de las articulaciones de ambos brazos que se van a mover.

- "RShoulderPitch" → Mueve el hombro derecho hacia adelante/arriba.
 - "LShoulderPitch" → Igual pero en el hombro izquierdo.
 - "RElbowRoll" → Dobla el codo derecho.
 - "LElbowRoll" → Dobla el codo izquierdo.
- `arms_up_angles` → Define los ángulos objetivo en radianes para cada una de esas articulaciones.
 - `-1.5` → $\text{rad} \approx -86^\circ$ → hombros bien hacia arriba (¡brazos al cielo!).
 - `0.5` → $\text{rad} \approx 28^\circ$ → ligera flexión del codo derecho hacia adentro.
 - `-0.5` → $\text{rad} \approx -28^\circ$ → ligera flexión del codo izquierdo hacia adentro.
- `motion.setAngles` → Mueve todas esas articulaciones a sus ángulos correspondientes a una velocidad del 20%.
- Bajar brazos lentamente
 - `arms_down_angles` → Define los nuevos ángulos para bajar los brazos de Pepper de manera notoria
 - `1.5` → radianes $\approx +86^\circ$: mueve los hombros hacia abajo (brazos colgando hacia adelante).
 - `1.0` y `-1.0` → Radianes en los codos. Doblan más los codos hacia adentro, como si Pepper se estuviera encogiendo o recogiendo los brazos.
 - `motion.setAngles(arms_names, arms_down_angles, 0.05)` → Mueve lentamente las articulaciones a los nuevos ángulos, pero esta vez con una velocidad muy baja ($0.05 = 5\%$ de la velocidad máxima).
- Asentir de nuevo
 - `for i in range(2)` → Ejecuta el bloque dentro del `for` dos veces (es decir, dos ciclos de movimiento).
 - `motion.setAngles("HeadPitch", -0.3, 0.2)` → Mueve la cabeza hacia abajo (-0.3 radianes $\approx -17^\circ$) a velocidad 0.2.
 - `time.sleep(0.5)` → Espera 0.5 segundos para permitir que el movimiento se vea natural y no demasiado rápido.

- `motion.setAngles("HeadPitch", 0.3, 0.2)` → Mueve la cabeza hacia arriba (+0.3 radianes $\approx +17^\circ$), también a velocidad 0.2.
- `time.sleep(0.5)` → Otra pausa para que Pepper mantenga ese movimiento y el gesto de “asentir” sea visible.
- `motion.setAngles("HeadPitch", 0.0, 0.2)` → Después de completar los dos ciclos de asentir, devuelve la cabeza a la posición central (neutra) con suavidad (velocidad 0.2).
- Resultado: Pepper realiza este patrón:




- Relajar brazos

- `motion` → Es el proxy a `ALMotion`, encargado de controlar los movimientos de Pepper.
- `setStiffnesses(...)` → función para establecer el nivel de rigidez muscular (¡como si fueran músculos eléctricos!).
- `["RArm", "LArm"]` → Lista de grupos de articulaciones (brazo derecho e izquierdo completos).
- `0.0`: valor de rigidez → 0 = totalmente flojo, 1.0 = completamente rígido.

7. Finalmente, se ejecuta el script con el siguiente comando y Pepper realiza la coreografía:

```
Pepper [0] ~ $ python levantar_brazos.py
[I] 1745515556.157300 4295 qimessaging.session: Session listener created on tcp://0.0.0.0:0
[W] 1745515556.158132 4295 qi.path.sdklayout: No Application was created, trying to deduce paths
[I] 1745515556.158209 4295 qimessaging.transportserver: TransportServer will listen on: tcp://127.0.0.1:36541
[I] 1745515556.158264 4295 qimessaging.transportserver: TransportServer will listen on: tcp://198.18.0.1:36541
[I] 1745515556.158405 4295 qimessaging.transportserver: TransportServer will listen on: tcp://192.168.0.100:36541
```

- `python levantar_brazos.py` → Esto inicia el programa que controla el movimiento de los brazos de Pepper.

Video:  Terminal coreografía.mp4