Proyecto Pepper

Miguel Ángel Jiménez Morales,

Dikersson Alexis Cañon VAnegas

25 de abril de 2025

1)Definición de librerías:

Librería qi

- **Descripción**: Es la interfaz principal para interactuar con el framework NAOqi de SoftBank Robotics.
- **Uso**: Permite la comunicación con el robot Pepper, gestionando conexiones, servicios y módulos.

2. Librería `argparse`

- **Descripción**: Módulo estándar de Python para analizar argumentos de línea de comandos.
- **Uso**: Facilita la creación de interfaces de línea de comandos, permitiendo definir qué argumentos requiere el programa y gestionando la entrada del usuario.

3. Librería `sys`

- **Descripción**: Proporciona acceso a algunas variables utilizadas o mantenidas por el intérprete de Python y a funciones que interactúan fuertemente con el intérprete.
- **Uso**: Permite manipular diferentes partes del entorno de ejecución de Python, como la salida estándar, argumentos de línea de comandos y más.

4. Librería `os`

- **Descripción**: Proporciona una forma portátil de usar funcionalidades dependientes del sistema operativo.

- **Uso**: Permite interactuar con el sistema operativo para tareas como manipulación de archivos y directorios, gestión de procesos y más.

5. Librería `almath`

- **Descripción**: Librería matemática optimizada para la robótica, utilizada en el entorno NAOqi.

- **Uso**: Facilita cálculos relacionados con movimientos, como la corrección de pasos para evitar colisiones.

6. Librería `math`

- **Descripción**: Módulo estándar de Python que proporciona acceso a funciones matemáticas definidas por la biblioteca estándar de C.

- **Uso**: Ofrece funciones como raíz cuadrada, logaritmos, trigonometría y más.

7. Librería `motion`

- **Descripción**: Módulo de NAOqi que permite controlar los movimientos del robot.

- **Uso**: Se utiliza para gestionar poses, rigidez, movimientos y caminar del robot.

8. Librería `httplib`

- **Descripción**: Módulo de Python 2 para implementar clientes HTTP.

- **Uso**: Permite realizar solicitudes HTTP y HTTPS, gestionar conexiones y más.

9. Librería `json`

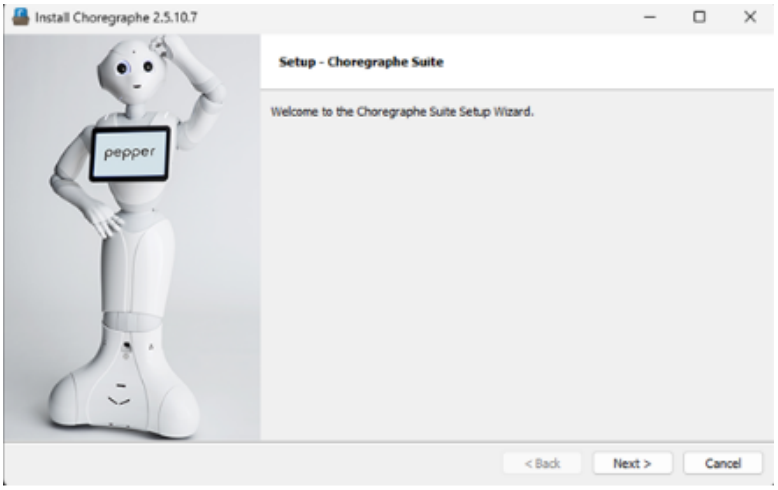
- **Descripción**: Módulo estándar de Python para trabajar con datos en formato JSON (JavaScript Object Notation).

- **Uso**: Facilita la serialización y deserialización de objetos Python a JSON y viceversa.

- **Detalles adicionales**: Es ampliamente utilizado para el intercambio de datos entre aplicaciones web y servidores.

  1.2. Se describir de forma sencilla el paso a paso para abrir coreographe

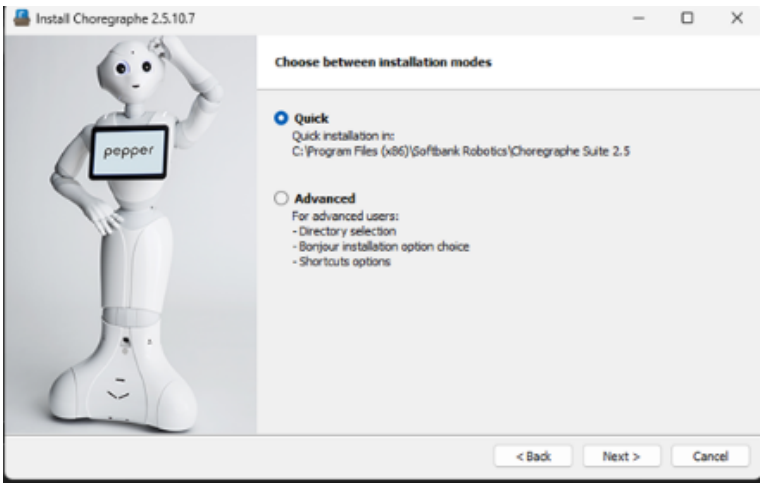
  Paso 1:



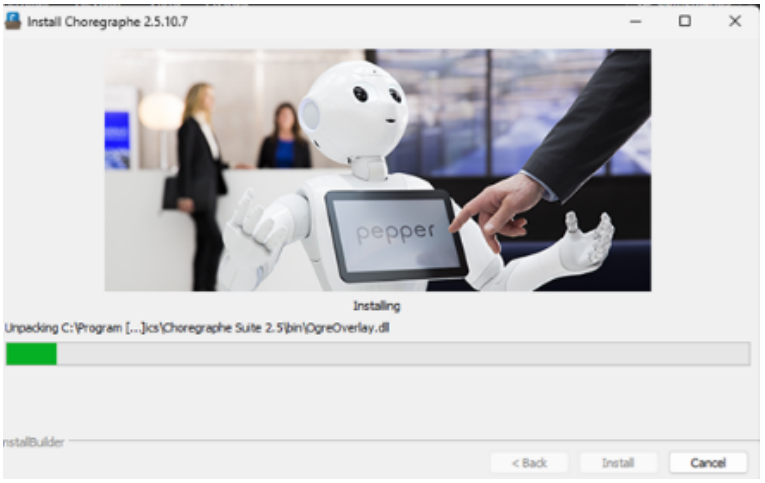
  Paso 2:

  Se da NEXT y aceptamos los acuerdos

  Paso 3:

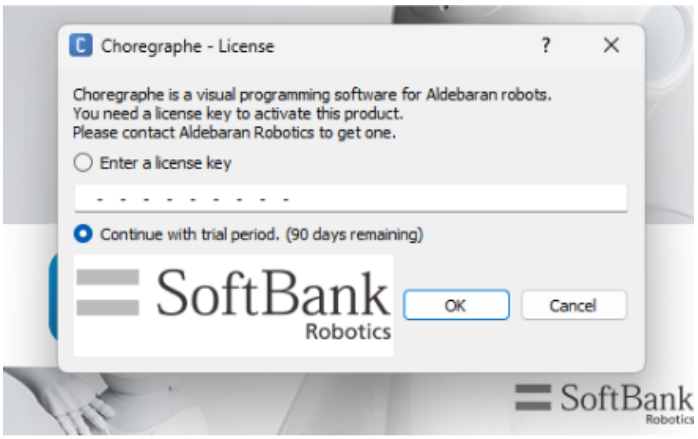


Paso 4:



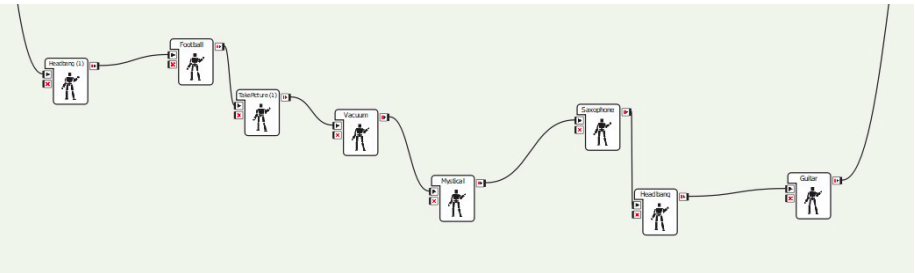
- Esperamos a que instale

Paso 5:

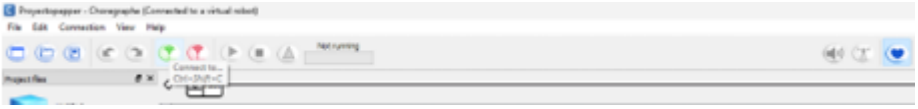


- Le damos OK y activamos el “Trial Period”

- Se obtienen los bloques de movimientos para montar la coreografía

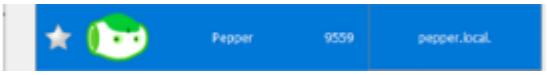


Paso 6:



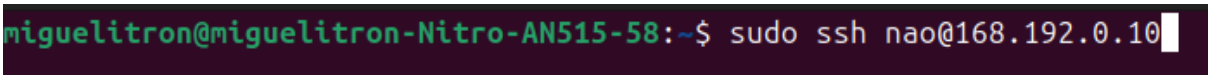
- Le damos click para, seleccionamos al robot pepper y estaremos conectados

Paso 7:

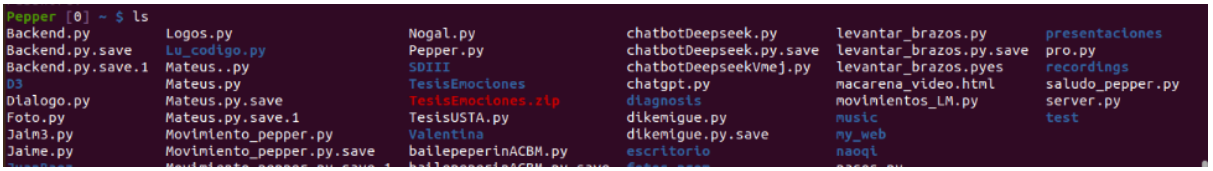


2. Terminal de Pepper:

1. Paso 1: Primero debes abrir la terminal desde Ubuntu colocar los datos de conectividad al robot (ssh, @, dirección IP) estando conectado en la red de ciego".



2. Paso 2: Encontrarías librerías, documentaciones, pasos y más



3.Paso 3: Se crea un archivo nano y se copia y pega el código de la coreografía

3. Explicacion de codigo:

Se explicará linea por linea como abarque la creación del código en Python:

1)Librerías:

from naoqi import ALProxy
import almath
import time

`from naoqi import ALProxy`: Importa el módulo `ALProxy` de la librería `naoqi`, que permite controlar distintos módulos del robot Pepper, como el movimiento, la postura, y la síntesis de voz.

`import almath`: Se utiliza para realizar conversiones matemáticas necesarias en los movimientos, como convertir grados a radianes.

`import time`:Ayuda a introducir pausas en el código, lo que es esencial para controlar los tiempos entre los movimientos del robot.

2) Dirección Ip

robotIP = "198.162.0.10"
port = 9559

Explicación: Define la dirección IP y el puerto de conexión del robot Pepper. En este caso, se conecta a un robot local (`127.0.0.1`), con el puerto `9559`, que es el puerto estándar de comunicación con Pepper.

3)Comandos para controlar los movimientos del robot:

```
motionProxy = ALProxy("ALMotion", robotIP, port)
postureProxy = ALProxy("ALRobotPosture", robotIP, port)
ttsProxy = ALProxy("ALTextToSpeech", robotIP, port)
```

Explicación:

- motionProxy: Crea un proxy para controlar los movimientos del robot a través del módulo ALMotion.
- postureProxy: Crea un proxy para controlar las posturas del robot mediante el módulo ALRobotPosture.
- ttsProxy: Crea un proxy para la síntesis de voz del robot a través del módulo ALTextToSpeech.

4) Despierta el robot y mas.

```
motionProxy.wakeUp()

postureProxy.goToPosture("StandInit", 0.5)

ttsProxy.say("Hola, estoy listo para bailar")
```

- motionProxy.wakeUp(): Despierta el robot para que esté listo para moverse.
- postureProxy.goToPosture("StandInit", 0.5): Hace que el robot asuma la postura inicial (de pie).
- ttsProxy.say("Hola, mis bellakosr"): El robot dice "Hola, estoy listo para bailar", usando la síntesis de voz.

5)Movimientos

```
ttsProxy.say("Observa esto")

motionProxy.moveTo(0.3, 0, 0)

motionProxy.angleInterpolationWithSpeed("RShoulderPitch", -1.0, 0.2)  # Brazo derecho arriba

motionProxy.angleInterpolationWithSpeed("RElbowYaw", 1.5, 0.2)

time.sleep(0.5)
```

6)Parámetros para los movimientos de robots y ajustes del robot

```
motionProxy.moveTo(-0.3, 0, 0)

motionProxy.angleInterpolationWithSpeed("RShoulderPitch", 1.5, 0.2)  # Baja brazo derecho
```

motionProxy.angleInterpolationWithSpeed("LShoulderPitch", -1.0, 0.2) # Brazo izquierdo arriba

motionProxy.angleInterpolationWithSpeed("LElbowYaw", -1.5, 0.2)

time.sleep(0.5)

Explicación:

- motionProxy.moveTo(-0.3, 0, 0): El robot retrocede 30 cm.
- motionProxy.angleInterpolationWithSpeed("RShoulderPitch", 1.5, 0.2): Baja el brazo derecho.
- motionProxy.angleInterpolationWithSpeed("LShoulderPitch", -1.0, 0.2): Levanta el brazo izquierdo.
- motionProxy.angleInterpolationWithSpeed("LElbowYaw", -1.5, 0.2): Ajusta el codo izquierdo para completar el movimiento.
- time.sleep(0.5): Pausa de medio segundo.

  ttsProxy.say("Y ahora hacia la derecha")

  motionProxy.moveTo(0, -0.2, 0)

  time.sleep(0.5)

- ttsProxy.say("Y ahora hacia la derecha")

  motionProxy.moveTo(0, -0.2, 0)

  time.sleep(0.5)

7)Movimientos de Cabeza y Reverencia

ttsProxy.say("Girando mi cabeza")

names = ["HeadYaw"]

angles = [almath.TO_RAD*45, almath.TO_RAD*-45, 0]

times = [1.0, 2.0, 3.0]

isAbsolute = True

motionProxy.angleInterpolation(names, angles, times, isAbsolute)

- ttsProxy.say("Girando mi cabeza"): El robot dice que va a girar su cabeza.

- names = ["HeadYaw"]: Define que el movimiento es en el eje de rotación de la cabeza.

- angles = [almath.TO_RAD*45, almath.TO_RAD*-45, 0]: Define los ángulos de la cabeza (giro de 45 grados a la derecha, -45 grados a la izquierda, y vuelta a la posición inicial).

- times = [1.0, 2.0, 3.0]: Define la duración de cada uno de los movimientos de la cabeza.

- isAbsolute = True: Define que los movimientos de la cabeza son absolutos (en lugar de relativos).

- motionProxy.angleInterpolation(names, angles, times, isAbsolute): Realiza el movimiento de la cabeza.

8) Giro final y descanso:

```
ttsProxy.say("Hasta la próxima")

motionProxy.moveTo(0, 0, almath.TO_RAD*360)  # Giro completo
```

Explicación:

- ttsProxy.say("Hasta la próxima"): El robot dice "Hasta la próxima".

- motionProxy.moveTo(0, 0, almath.TO_RAD*360): Realiza un giro de 360 grados.