Laboratorio 2: Conociendo Pepper

Laura Cruz, Andres Saladaña, Jonny Vargas

Agosto 2025

1 Investigación de las librerías

1.1 Librería `qi`

La librería `qi` es el núcleo de comunicación entre el programador y el sistema operativo de los robots *Pepper* y *NAO*. Permite conectarse al robot, crear módulos propios y acceder a servicios internos como voz, visión, movimiento y sensores.

Funciones principales:

- **Conexión:** Comunicación con el robot vía IP y puerto 9559
- **Servicios:** Acceso a módulos como `ALMotion` o `ALTextToSpeech`
- **Comportamientos:** Ejecución de movimientos o diálogos desde *Choregraphe*
- **Eventos:** Sincronización de voz, gestos y sensores

```python
import qi
import sys
session = qi.Session()
session.connect("tcp://192.168.0.100")
tts = session.service("ALTextToSpeech")
tts.say("Hola, soy Pepper")
```

Listing 1: Ejemplo de conexión con Pepper

1.2 Librería `argparse`

La librería `argparse` es un módulo estándar de Python que permite gestionar argumentos desde la línea de comandos. Con ella, un programa puede recibir parámetros al momento de ejecutarse en la terminal (como IP, puerto o archivos), lo que resulta muy útil en el trabajo con robots como *Pepper*, ya que evita modificar el código cada vez que cambian las opciones.

Características principales:

- **Automatizar configuraciones:** Posibilita pasar la IP o el puerto del robot directamente al ejecutar
- **Flexibilidad:** Permite variar parámetros sin editar el programa
- **Ayuda automática:** Con `-help` muestra instrucciones de uso

```
import argparse
parser = argparse.ArgumentParser(description="Conectar a Pepper")
parser.add_argument("--ip", default="192.168.0.100", help="IP del robot Pepper")
parser.add_argument("--port", type=int, default=9559, help="Puerto de conexi n")
args = parser.parse_args()
print(f"Conectando a {args.ip}:{args.port}")
```

Listing 2: Ejemplo de uso de argparse

1.3 Librería `sys`

La librería `sys` es un módulo fundamental de Python que proporciona acceso directo a variables y funciones del intérprete. Gracias a ella, los programas pueden interactuar con el entorno de ejecución y controlar aspectos internos del sistema. En proyectos con Pepper, resulta útil para gestionar rutas, entradas o salidas, y terminar procesos de manera controlada.

Funcionalidades clave:

- **Acceso a argumentos:** Permite leer los parámetros pasados al script mediante `sys.argv`
- **Gestión de la salida:** Facilita escribir mensajes directamente en la consola o finalizar el programa con `sys.exit()`
- **Control del entorno:** Posibilita manipular el path de búsqueda de módulos con `sys.path`

```python
import sys
if len(sys.argv) < 2:
print("Uso: python script.py <ip>")
sys.exit(1)</ip>
ip = sys.argv[1]
print(f"Conectando a Pepper en {ip}")
```

Listing 3: Ejemplo de uso de sys

1.4 Librería `os`

La librería so forma parte del conjunto estándar de Python y su propósito principal es facilitar la interacción con el sistema operativo de Pepper. A través de ella se pueden gestionar rutas de archivos, ejecutar procesos y manipular directorios dentro del robot. Su uso resulta fundamental, ya que permite resolver de manera adecuada las rutas de los recursos, verificar la existencia de carpetas, listar archivos disponibles, identificar el entorno de ejecución y comprender los permisos necesarios para el correcto funcionamiento de Pepper.

1.5 Librería `json`

La librería `json` es un módulo estándar de Python que permite trabajar con datos en formato JSON. Es útil para guardar configuraciones, leer información proveniente de servicios web y enviar o recibir datos estructurados. En el caso de Pepper, se emplea para almacenar parámetros de funcionamiento o serializar secuencias de movimiento.

Características principales: - Lectura y escritura de archivos JSON - Intercambio de datos con APIs y servicios web - Serialización y deserialización de estructuras de Python

```
1 import json
2 data = { "accion": "caminar", "velocidad": 0.2 }
3 json_string = json.dumps(data)      # Serializar a JSON
4 print(json_string)
5 data_recuperada = json.loads(json_string)  # Deserializar
6 print(data_recuperada["accion"])
```

1.6 Librería `motion (ALMotion)`

La librería `motion`, también conocida como `ALMotion`, proporciona la API para controlar los movimientos del robot Pepper. Permite mover articulaciones, ejecutar posturas predefinidas y coordinar movimientos complejos de manera sincronizada con la comunicación verbal.

Características principales: - Control de articulaciones - Ejecución de posturas predefinidas - Coordinación de gestos y movimientos autónomos

```
1 from naoqi import ALProxy
2 motion = ALProxy("ALMotion", "192.168.0.100", 9559)
3 motion.wakeUp()
4 motion.moveTo(0.5, 0, 0)   # Avanzar 0.5 metros
5 motion.rest()
```

1.7 Librería `httplib`

La librería `httplib` (o `http.client` en Python 3) permite establecer conexiones HTTP, enviar peticiones y recibir respuestas desde servidores web. En Pepper, se utiliza para integrar servicios externos, consumir APIs y ampliar sus capacidades de interacción.

Características principales: - Enviar solicitudes HTTP (GET, POST) - Conectarse a servicios web externos - Integración con aplicaciones en la nube

```
1 import http.client
2 conn = http.client.HTTPConnection("example.com")
3 conn.request("GET", "/")
4 response = conn.getresponse()
5 print(response.status, response.reason)
6 conn.close()
```

A continuacion se explicara la descarga e instalación de **Choregraphe**, y como se realizo una *coreografía sencilla* para el robot Pepper, tambien el acceso por consola al robot mediante `ssh` para crear y ejecutar una coreografia sencilla mediante codigo.

2 Instalación de Choregraphe

2.1 Descarga

- Descargar el instalador de Choregraphe compatible con Pepper (NAOqi) desde un repositorio "https://github.com/UoA-CARES/pepper-demo".
- Guardar el archivo en el PC e iniciar la instalación.

2.2 Instalación y primer arranque

- Ejecutar el instalador y seguir el asistente.

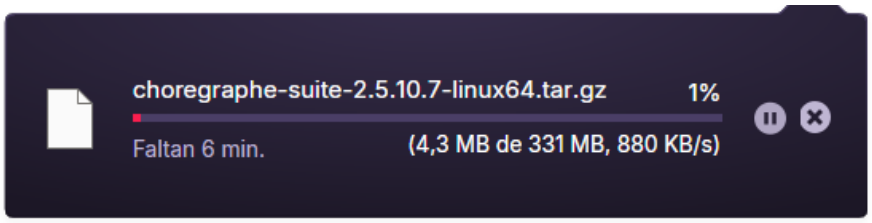

Figura 1: Proceso de descarga de Choregraphe

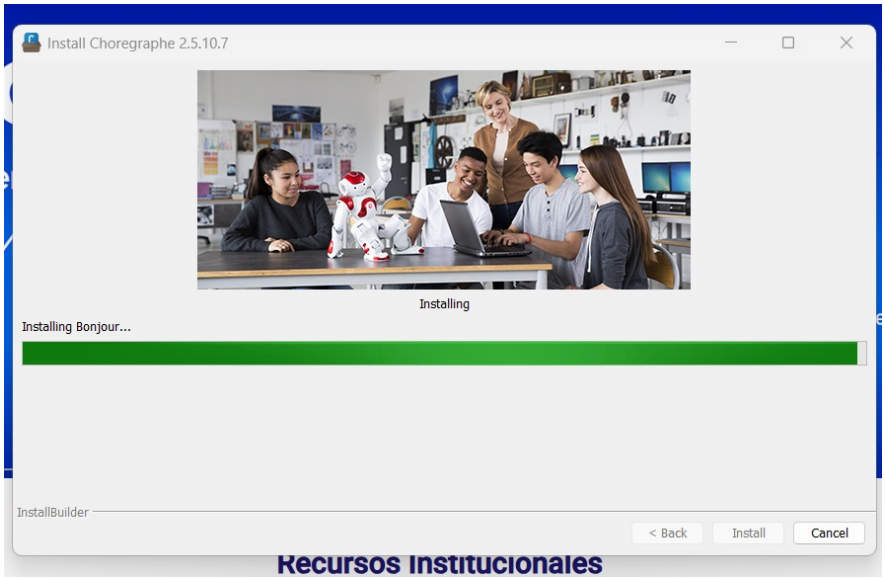

Figura 2: Instalacion de Choregraphe

- Abrir Choregraphe y verificar que inicia correctamente.

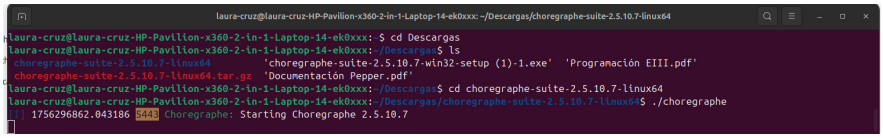

Figura 3: Abrir programa por medio del terminal

3 Coreografía sencilla en Choregraphe

3.1 Conexión al robot

1. Con el robot y el PC en la misma red, obtener la IP del robot (por voz, ajustes o visor de red).
2. En Choregraphe, conectar al robot ingresando la IP.

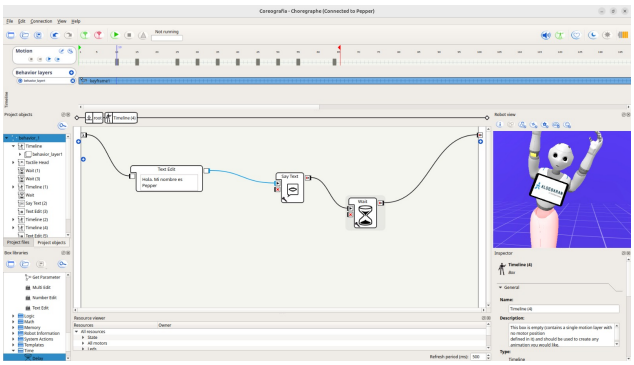

Figura 4: Bloque coreographe

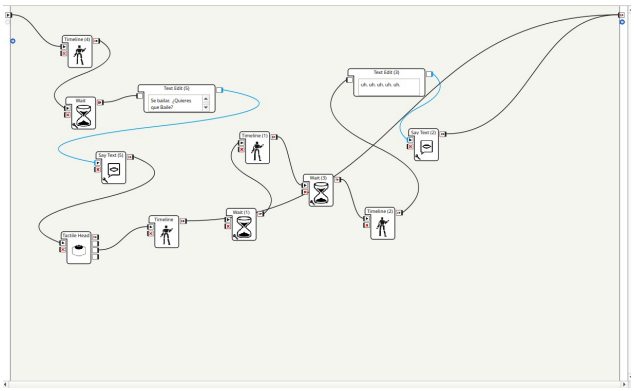

Figura 5: Bloques Coregraphe

3.2 Creación del proyecto y bloques

- Crear un proyecto nuevo (*File* → *New Project*).
- Arrastrar bloques básicos: por ejemplo, **Say** (hablar), **Timeline** para hacer una secuencia.
- Encadenar los bloques para formar la coreografía (Inicio → Say → Animation → Fin). Ajustar parámetros (texto, velocidad, etc.).
- Ejecutar la coreografía en el simulador o directamente en el robot.

4 Acceso por consola (SSH) y script en Python

4.1 Ingreso por SSH

Asegurar que el PC y Pepper estén en la misma red. Luego, desde la terminal del PC, ejecutar:

```
ssh nao@IP_PEPPER
```

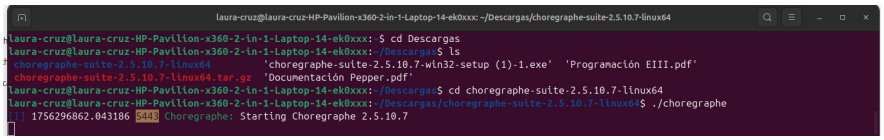

Figura 6: Coreografia en choregraphe

```

ssh nao@192.168.0.100
La contrasena es nao
```

Listing 4: Conexion SSH al robot Pepper

4.2 Creacion de archivo nano

Crear un archivo llamado PepperRobot.py con el edito de nano despues guardar y con Ctrl + O, Enter y salir con Ctrl+ X y ejecutar:

```
nano baile.py
python Pepper_robot.py
```

Listing 5: Crear/editar archivo Python en el robot

4.3 Librerías utilizadas (explicación)

- **naoqi.ALProxy**: crea un proxy para comunicarte con módulos del robot.
- **ALTextToSpeech**: síntesis de voz (`tts.say("texto")`).
- **ALMotion**: control de motores, posturas y movimientos (`wakeUp()`, `moveTo()`, `angleInterpolationWithSpeed()`, etc.).
- **ALRobotPosture**: cambios rápidos a posturas conocidas (`StandInit`, `Crouch`, etc.).

```
 GNU nano 7.2                                        peper_robot.py
    # --- Final ---
    speech.say("¡Dale a tu cuerpo alegría, Macarena!")
    ArmR = [-0.3, -0.8, 0.0, 0.0]
    ArmL = [-0.3, 0.8, 0.0, 0.0]
    motion.angleInterpolationWithSpeed(["RShoulderPitch", "RShoulderRoll", "RElbowYaw", "RElbowRoll"], ArmR, 0.4)
    motion.angleInterpolationWithSpeed(["LShoulderPitch", "LShoulderRoll", "LElbowYaw", "LElbowRoll"], ArmL, 0.4)
    time.sleep(2)

    speech.say("¡Muchas gracias! Ha sido increíble bailar con ustedes.")
    posture.goToPosture("StandInit", 0.6)
    motion.rest()


def add_macarena_to_menu(session):
    """
    Integrar la Macarena al menú principal
    """
    print("\n=== COREOGRAFÍA ESPECIAL ===")
    respuesta = raw_input("¿Quieres que Pepper baile la Macarena? (s/n): ")

    if respuesta.lower() == "s":
        try:
            macarena_choreography(session)
        except Exception as e:
            print("Error durante la coreografía:", e)
    else:
        print("¡Tal vez la próxima vez!")
```

Figura 7: Coreografia en terminal

```
 GNU nano 7.2                                        peper_robot.py
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

import qi
import time
import argparse
import sys


def macarena_one_round(motion_service, speech_service, speed):
    """
    Una ronda de la Macarena (se repite dentro de la coreografía principal)
    """

    # Definir articulaciones
    JointNamesL = ["LShoulderPitch", "LShoulderRoll", "LElbowYaw", "LElbowRoll"]
    JointNamesR = ["RShoulderPitch", "RShoulderRoll", "RElbowYaw", "RElbowRoll"]
    JointNamesH = ["HeadPitch", "HeadYaw"]

    # ========== PASO 1 ==========
    speech_service.say("Mano derecha al frente")
    ArmR = [-0.8, -0.3, 0.0, -0.1]
    ArmL = [1.0, 0.2, -1.2, -0.8]
    motion_service.angleInterpolationWithSpeed(JointNamesR, ArmR, speed)
    motion_service.angleInterpolationWithSpeed(JointNamesL, ArmL, speed)
    motion_service.angleInterpolationWithSpeed(JointNamesH, [0.0, 0.3], speed)
    time.sleep(1)
```

Figura 8: Coreografia en terminal

4.4 Codigo completo de la coreografia

- Python Software Foundation, argparse — Parser for command-line options, arguments and sub-commands, Python 3 Documentation, [Online]. Available: https://docs.python.org/3/library/argparse.html
- Python Software Foundation, sys — System-specific parameters and functions, Python 3 Documentation, [Online]. Available: https://docs. python.org/3/library/sys.html.
- Python Software Foundation, json — JSON encoder and decoder, Python 3 Documentation, [Online]. Available: https://docs.python.org/3/library/json.html
- Aldebaran Robotics, ALMotion API, Documentation, [Online]. Available: http://doc.aldebaran.com/2-1/naoqi/motion/almoti
- Python Software Foundation, http.client — HTTP protocol client, Python 3 Documentation, [Online]. Available: https://docs.python.org/3/library/http.client.html