Laboratorio 2 — Conociendo a Pepper

Yojan Contreras - Cristian Losada

5 de septiembre

Resumen

Este documento describe las librerías para el correcto funcionamiento del robot Pepper y el paso a paso del procedimiento para crear una coreografía sencilla utilizando **Choregraphe** y la creación de un script en **Python** (con `nano`) que ejecute la secuencia directamente en el robot. Se incluyen comandos, ejemplos de código y la explicación de los servicios y librerías empleados.

Índice

1. Objetivo

El objetivo es desarrollar una coreografía sencilla para Pepper mediante la aplicación **Choregraphe** y, de manera complementaria, crear un script en Python editado con `nano` que ejecute la secuencia en el robot. Se documenta el acceso por consola, la creación del archivo y las librerías empleadas.

2. Requisitos previos

- Pepper y el computador anfitrión deben estar conectados a la **misma red** Wi-Fi.
- Dirección IP de Pepper (ejemplo: `192.168.0.100`).
- Usuario: `nao` ; contraseña por defecto: `nao`.
- Descargar la versión `choregraphe-suite-2.5.10.7-linux64`.
- Acceso a la terminal (SSH / scp) desde el computador.

3. Parte A — Creación de la coreografía en Choregraphe

1. Abrir **Choregraphe**.
2. Crear un nuevo proyecto.
3. Usar bloques de **Say**, **Animation**, y opcionalmente **Python Script**.
4. Probar primero en el simulador.
5. Conectar con Pepper mediante su IP y ejecutar la coreografía en el robot.

4. Parte B — Creación de un script en Python con nano

Acceso al robot

```
ssh nao@<IP_DE_PEPPER>
# ejemplo:
# ssh nao@192.168.1.100
# contraseña: nao
```

Creación del archivo

```
cd ~
nano coreografia_pepper.py
```

Código Python del script

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import qi
import argparse
import sys
import time

def main(session):
    tts = session.service("ALTextToSpeech")
    motion = session.service("ALMotion")
    leds = session.service("ALLeds")
    animated_speech = session.service("ALAnimatedSpeech")
    animation_player = session.service("ALAnimationPlayer")
    posture = session.service("ALRobotPosture")

    # ------------------ BLOQUE 1: Presentación ------------------
    leds.fadeRGB("FaceLeds", 0xFF0000, 0.8)
    animated_speech.say("^start(animations/Stand/Gestures/Hey_3) Hola, soy Pepper.^wa

    print("Coreografía completada.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--ip", type=str, default="127.0.0.1")
    parser.add_argument("--port", type=int, default=9559)
    args = parser.parse_args()
    session = qi.Session()
    try:
        session.connect("tcp://" + args.ip + ":" + str(args.port))
    except RuntimeError:
        print("No se puede conectar a Naoqi")
        sys.exit(1)
    main(session)
```

Ejecución

```
chmod +x coreografia_pepper.py
python2 coreografia_pepper.py --ip <IP_DE_PEPPER>
```

5. Parte C — Librerías y servicios empleados

- **qi:** Comunicación con servicios internos de Pepper.
- **argparse:** Manejo de argumentos desde terminal.
- **sys:** Funciones del sistema.

- **os:** Manejo de rutas y procesos.
- **almath:** Cálculos matemáticos avanzados de movimiento.
- **math:** Operaciones matemáticas básicas.
- **motion (ALMotion):** Control de motores y articulaciones.
- **httplib:** Comunicación HTTP.
- **json:** Manejo de datos en formato JSON.
- **ALRobotPosture:** Posturas predefinidas.
- **ALAnimatedSpeech:** Voz con gestos.
- **ALAnimationPlayer:** Animaciones predefinidas.
- **ALLeds:** Control de luces LED.
- **time:** Pausas y sincronización.

6. Parte D — README del repositorio

El repositorio debe contener:

- `coreografia_pepper.py`
- Carpeta `screenshots/`
- `README.md`

7. Parte E — Checklist para GitHub

```
mkdir -p CoreografiaPepper/screenshots
cp coreografia_pepper.py CoreografiaPepper/
cd CoreografiaPepper
git init
git add .
git commit -m "Coreografía Pepper - Presentación"
git remote add origin https://github.com/USUARIO/REPO.git
git push -u origin main
```

8. Consejos finales

- Ajustar tiempos para sincronizar voz y animación.
- Mantener un respaldo del script en el robot.
- Usar `ALMotion.setAngles` para movimientos más precisos.
- Probar primero en el simulador antes de ejecutar en el robot real.

—

Link video de la coreografia https://youtu.be/zr-gXhziTD4