Informe: Librerías y Coreografía Sencilla para Pepper (NAOqi 2.5)

Marco Gómez - Juan P. Pedraza

4 de septiembre de 2025

Índice

1. Introducción

Este informe resume las librerías más usadas para programar el robot Pepper, los pasos de instalación y uso de Choregraphe, y dos formas de crear una coreografía sencilla: (1) con la interfaz de Choregraphe y (2) por terminal mediante SSH y un script de Python (NAOqi 2.5 / Python 2.7).

2. Librerías y servicios usados en Pepper

Librería/Servicio	Tipo	Para qué sirve / Ejemplo
`qi`	SDK NAOqi (pynaoqi)	Crea sesiones y obtiene servicios: `qi.Session` → `ALMotion`, `ALRobotPosture`, `ALTextToSpeech`.
`argparse`	Estándar Python	Manejo de argumentos CLI (`--ip`, `--port`).

`sys`	Estándar Python	Salir con códigos de error, imprimir excepciones, modificar `sys.path` si es necesario.
`os`	Estándar Python	Rutas y archivos (ej. guardar un log en `/home/nao`).
`almath`	SDK NAOqi	Utilidades matemáticas y de cinemática: `TO_RAD`, transformaciones.
`math`	Estándar Python	Funciones matemáticas (`math.radians`, etc.).
`ALMotion` (*servicio*)	NAOqi	Control de articulaciones y base móvil. Se accede vía `session.service("ALMotion")`.
`httplib`	Estándar Python 2	Conexiones HTTP simples (en Python 3 es `http.client`).
`json`	Estándar Python	Serializar/leer datos en JSON (configuraciones, logs).

3. Instalación y preparación del entorno

3.1. Choregraphe Suite 2.5

1. Descargue e instale Choregraphe Suite 2.5 para su sistema operativo (Windows/Linux/macOS).
2. Asegúrese de que el PC y el robot Pepper estén en la **misma red**. Obtenga la IP del robot desde la tableta o con el botón del pecho (el robot la dice en voz alta).
3. Abra Choregraphe y conecte el robot: `Connection` → `Connect to...` → `IP del robot`.

3.2. SDK Python (pynaoqi)

1. Descargue el **pynaoqi** correspondiente a su OS (versión 2.5.x para Python 2.7).
2. Añada las rutas del SDK a `PYTHONPATH` (en Windows, variables de entorno; en Linux, export en su shell) para disponer del módulo `qi` y `almath`.
3. En el robot, ya vienen incluidos NAOqi y Python 2.7; no es necesario instalar nada.

3.3. Comprobaciones rápidas

- **Ping/latencia**: pruebe con `ping <ip_robot>`.
- **Puerto NAOqi**: por defecto `9559` debe estar accesible.

4. Coreografía con Choregraphe (paso a paso)

1. Cree un proyecto nuevo: `File` → `New Project`.
2. En el *Workspace*, añada los bloques: `Set Stiffness`, `Go to Posture (StandInit)`, `Say`, `Timeline` (o un `Animation`).
3. Conecte los bloques en secuencia: `Stiffness` → `Posture` → `Say` → `Timeline`.
4. En **Say**, escriba: “Hola, vamos a bailar”.

5. Abra el **Timeline**, pulse **Record** y mueva suavemente: asentir cabeza, saludar con brazo derecho y un pequeño balanceo lateral. Detenga la grabación.

6. Guarde el proyecto y ejecútelo con el botón **Play**. Si desea, cárguelo al robot: `Project` → `Upload to robot`.

4.1. Evidencia de choregragphe

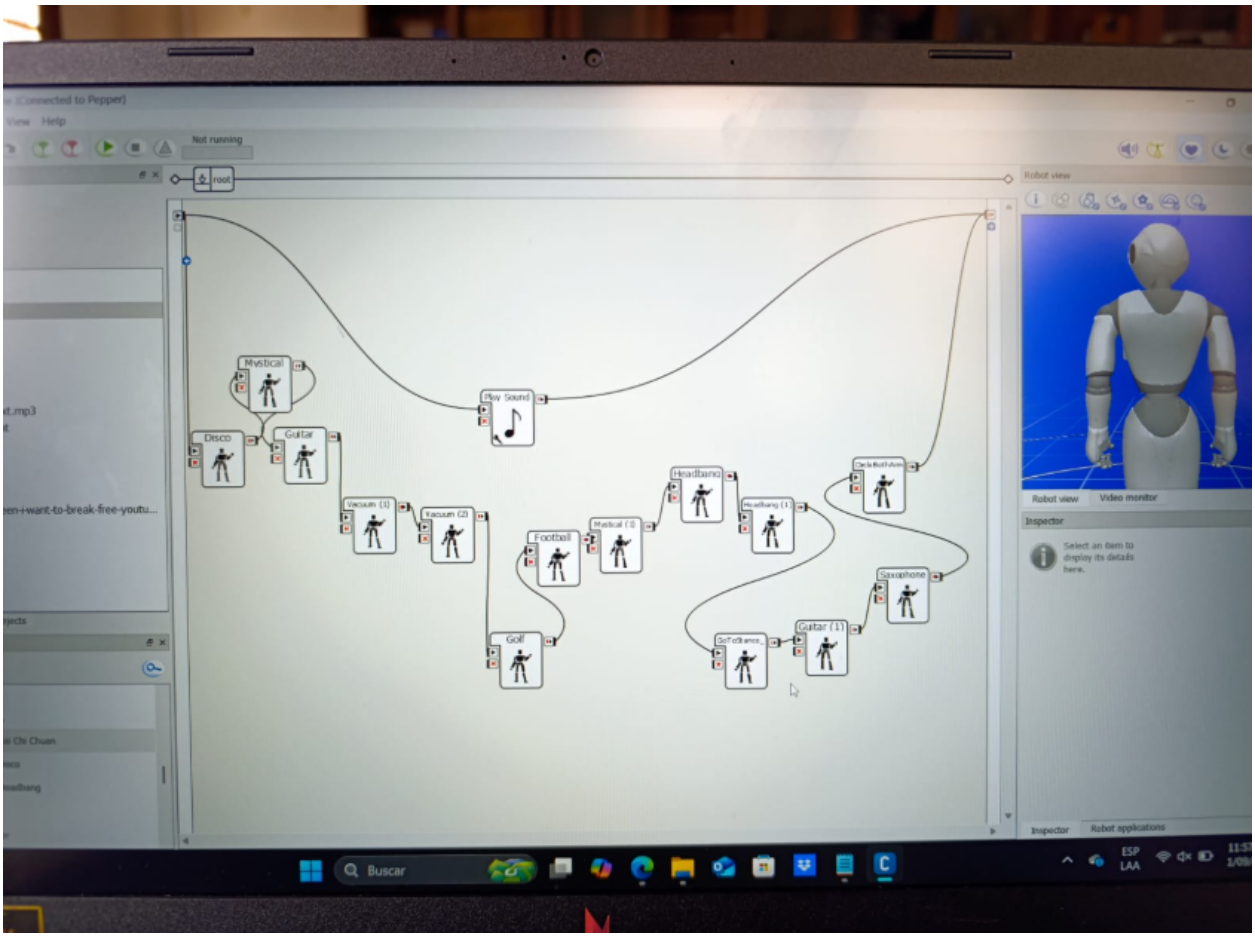

Figura 1: Enter Caption

5. SSH al robot y script por terminal

5.1. Ingreso por SSH

Desde su terminal (Windows 10+ o Linux/macOS):

```
ssh nao@<ip_del_robot>
# contraseña inicial por defecto: nao  (cámbiela con: passwd)
```

5.2. Crear archivo con nano y ejecutar

```
cd /home/nao
nano pepper_choreography.py   # pegue el código de la Sección 7
python pepper_choreography.py --ip 127.0.0.1 --port 9559
```

Nota: Dentro del robot use `127.0.0.1`. Desde el PC, ejecute el mismo archivo con la IP del robot.

6. Código Python completo (NAOqi 2.5)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Coreograf a sencilla para Pepper usando NAOqi 2.5 (Python 2.7).
Demuestra el uso de: qi, argparse, sys, os, math, almath, json y httplib.
Se puede ejecutar DENTRO del robot por SSH (127.0.0.1) o desde el PC con
    la IP del Pepper.
"""
from __future__ import print_function

import qi
import argparse
import sys
import os
import math
import almath  # del SDK NAOqi (transformaciones y constantes)
import time

# json es est ndar
try:
    import json
except Exception:
    import simplejson as json  # fallback si estuviera disponible

# Pepper/NAO corren Python 2.7: el nombre del m dulo HTTP es "httplib"
try:
    import httplib  # Python 2
except Exception:
    import http.client as httplib  # por si alguien lo ejecuta con Python
        3 en el PC


def http_ping(ip):
    """
    Prueba muy simple de HTTP para comprobar conectividad con el robot.
    Devuelve el status code (e.g., 200/404) o None si falla.
    """
    try:
        conn = httplib.HTTPConnection(ip, timeout=2)
        conn.request("GET", "/")
        resp = conn.getresponse()
        status = resp.status
        conn.close()
```

```
        return status
    except Exception:
        return None


def coreografia(session):
    """
    Coreograf a corta y segura: saludo con cabeza, movimiento de brazo
        derecho,
    desplazamiento lateral suave, giro peque o y cierre.
    """
    motion = session.service("ALMotion")            # "Librer a" motion (
        servicio ALMotion)
    posture = session.service("ALRobotPosture")
    tts = session.service("ALTextToSpeech")

    # Encender motores
    motion.wakeUp()
    posture.goToPosture("StandInit", 0.8)

    tts.say(u"Hola, soy Pepper. Vamos a bailar.")

    # 1) Asentir con la cabeza (HeadPitch) usando almath para pasar a
        radianes
    names = ["HeadPitch"]
    angles = [10 * almath.TO_RAD]
    times = [1.0]
    motion.angleInterpolation(names, angles, times, True)
    motion.angleInterpolation(names, [-10 * almath.TO_RAD], times, True)
    motion.angleInterpolation(names, [0.0], times, True)

    # 2) Saludo con el brazo derecho
    names = ["RShoulderPitch", "RShoulderRoll", "RElbowYaw", "RElbowRoll",
        "RWristYaw", "RHand"]
    keyframes = [
        [-1.0, -1.3, -1.0],  # RShoulderPitch
        [-0.1, -0.3, -0.1],  # RShoulderRoll
        [ 1.5,  1.0,  1.5],  # RElbowYaw
        [ 1.0,  0.8,  1.0],  # RElbowRoll
        [ 1.0,  0.7,  1.0],  # RWristYaw
        [ 1.0,  1.0,  1.0],  # Mano abierta
    ]
    times = [[1.0, 1.8, 2.6] for _ in names]
    motion.angleInterpolation(names, keyframes, times, True)
    time.sleep(0.3)
    motion.setAngles("RHand", 0.0, 0.2)  # cerrar mano suave

    # 3) Balanceo lateral con la base
    motion.moveInit()
    motion.moveTo(0.0,  0.05, 0.0)    # 5 cm izquierda
    motion.moveTo(0.0, -0.10, 0.0)    # 10 cm derecha
    motion.moveTo(0.0,  0.05, 0.0)    # volver al centro

    # 4) Peque o giro a un lado y al otro
```

```
    motion.moveTo(0.0, 0.0, math.radians(20))
    motion.moveTo(0.0, 0.0, math.radians(-20))

    tts.say(u" Listo ! Gracias.")

    # Volver a postura neutra y relajar
    posture.goToPosture("StandInit", 0.8)
    motion.rest()


def main():
    parser = argparse.ArgumentParser(description=u"Coreograf a sencilla
       para Pepper (NAOqi 2.5).")
    parser.add_argument("--ip", type=str, default="127.0.0.1",
                        help="IP del robot (por defecto 127.0.0.1 si
                           ejecutas dentro del robot por SSH).")
    parser.add_argument("--port", type=int, default=9559, help="Puerto de
       NAOqi (por defecto 9559).")
    parser.add_argument("--ping", action="store_true", help="Realiza un
       ping HTTP de prueba al robot.")
    args = parser.parse_args()

    # Ejemplo de uso de json y os: registrar metadatos simples de la
       ejecuci n
    meta = {"autor": "demo", "pasos": ["cabeza", "saludo", "balanceo", "
       giro"]}
    log_path = os.path.join(os.path.expanduser("~"), "pepper_choreo_log.
       json")
    try:
        with open(log_path, "w") as f:
            json.dump(meta, f)
    except Exception:
        pass

    if args.ping:
        status = http_ping(args.ip)
        print("HTTP status:", status)

    try:
        app = qi.Application(["pepper_choreo", "--qi-url=tcp://{}:{}".
           format(args.ip, args.port)])
        app.start()
        session = app.session
        coreografia(session)
        app.stop()
    except RuntimeError:
        print("No se pudo conectar a NAOqi en {}:{}".format(args.ip, args.
           port))
        sys.exit(1)


if __name__ == "__main__":
    main()
```

7. Problemas frecuentes y soluciones

- **No conecta a 9559**: confirme IP, red y que el robot está despierto. Pruebe `ssh nao@<ip>`.
- **Error de módulo `qi`/`almath` en el PC**: falta añadir `pynaoqi` a `PYTHONPATH`. En el robot no ocurre.
- **Python 3**: Pepper usa Python 2.7. Ejecute con `python` en el robot (no `python3`).
- **Seguridad**: cambie la contraseña por defecto con `passwd` al primer acceso por SSH.

8. Conclusiones

Se mostraron las librerías clave del ecosistema NAOqi, un flujo sencillo para crear una coreografía en Choregraphe, y un ejemplo equivalente vía SSH con Python. El enfoque híbrido (visual + código) permite iterar rápido y luego refinar movimientos con precisión.