Informe Laboratorio 2 – Robot Pepper

Nombre del estudiante

5 de septiembre de 2025

Índice

1. Introducción

En este informe se documenta la primera interacción con el robot Pepper en el salón de robótica. Se presentan los resultados de la investigación sobre las librerías utilizadas en el entorno de programación de Pepper, el desarrollo de una coreografía sencilla utilizando la herramienta Choregraphe, y la creación de una secuencia de pasos por medio de la consola de Ubuntu usando conexión `ssh`.

Para este laboratorio se trabajó con el robot Pepper conectado a la red **CIEGO**, con la dirección IP `192.168.0.106` y la contraseña de usuario `nao`.

2. Investigación de Librerías

A continuación se describen las librerías más utilizadas en el desarrollo de aplicaciones con Pepper, indicando su uso, aplicaciones prácticas y cómo activarlas.

2.1. qi

Es la librería base que permite establecer conexión con los servicios del framework NAOqi. **Uso en Pepper:** Permite controlar servicios como `ALMotion`, `ALTextToSpeech`, `ALRobotPosture`, entre otros. **Aplicaciones:** Control de movimientos, hablar, leer sensores. **Activación:**

```
import qi
app = qi.Application(["AppPepper", "--qi-url=tcp://192.168.0.106:9559"])
app.start()
```

2.2. argparse

Librería estándar de Python para manejar parámetros de línea de comandos. **Uso en Pepper:** Permite que un mismo programa se ejecute en diferentes robots cambiando solo la IP. **Aplicaciones:** Ejecutar `python baile.py --ip 192.168.0.106 --port 9559`. **Activación:**

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--ip", type=str, default="192.168.0.106")
parser.add_argument("--port", type=int, default=9559)
args = parser.parse_args()
```

2.3. sys

Librería estándar para interactuar con el intérprete de Python. **Uso en Pepper:** Añadir rutas de librerías o terminar la ejecución en caso de error. **Aplicaciones:** Manejo de excepciones críticas. **Activación:** `import sys`.

2.4. os

Permite manipular el sistema de archivos y procesos. **Uso en Pepper:** Crear carpetas o logs directamente en el robot. **Aplicaciones:** Guardar datos de sensores en archivos. **Activación:** `import os`.

2.5. almath

Librería matemática de NAOqi que maneja vectores y transformaciones espaciales. **Uso en Pepper:** Precisión en movimientos de brazos, giros de torso y equilibrio. **Aplicaciones:** Planificación de trayectorias en 3D. **Activación:** `import almath`.

2.6. math

Librería estándar de Python con funciones matemáticas comunes. **Uso en Pepper:** Movimientos periódicos o trigonométricos. **Aplicaciones:** Crear balanceo de brazos con una onda sinusoidal. **Activación:** `import math`.

2.7. motion

Es la interfaz hacia el servicio `ALMotion`. **Uso en Pepper:** Permite mover articulaciones, caminar y cambiar posturas. **Aplicaciones:** Coreografías, saludos, movimientos complejos. **Activación:**

```
motion_service = session.service("ALMotion")
```

2.8. httplib

Librería para realizar conexiones HTTP. **Uso en Pepper:** Comunicación con servidores web o APIs. **Aplicaciones:** Conectar Pepper con bases de datos externas o información en línea. **Activación:** `import httplib` (en Python 2.7) o `import http.client` en Python 3.

2.9. json

Permite trabajar con datos en formato JSON. **Uso en Pepper:** Guardar configuraciones de movimientos y leer datos estructurados. **Aplicaciones:** Integración con APIs externas. **Activación:**

```
import json
data = {"movimiento":"saludo","angulo":45}
json_str = json.dumps(data)
```

3. Desarrollo en Choregraphe

Se diseñó una coreografía sencilla usando la interfaz gráfica de Choregraphe. Los pasos consisten en movimientos básicos de brazos, giro de torso y saludo.

3.1. Proceso paso a paso

1. Conectar el PC a la red **CIEGO**.
2. Abrir Choregraphe e ingresar la IP del robot (`192.168.0.106`).
3. Crear un nuevo proyecto y agregar bloques de movimiento.
4. Configurar los bloques para generar una secuencia.
5. Ejecutar la coreografía en el robot.

4. Desarrollo por Consola SSH en Ubuntu

También se realizó una interacción mediante la consola del robot en el sistema operativo Ubuntu.

4.1. Conexión SSH

```
# Conexi\'on desde Ubuntu
ssh nao@192.168.0.106
# Contrase\~na: nao
```

4.2. Creación de archivo Python

```
nano salsota.py
```

4.3. Código del Baile

A continuación se presenta el código completo utilizado en Ubuntu para programar la coreografía en Pepper:

```
# -*- coding: utf-8 -*-
from naoqi import ALProxy
import time, math

IP = "192.168.0.106"
PORT = 9559

motion   = ALProxy("ALMotion", IP, PORT)
posture  = ALProxy("ALRobotPosture", IP, PORT)

if not motion.robotIsWakeUp():
    motion.wakeUp()
posture.goToPosture("StandInit", 0.6)

try:
    motion.setBreathEnabled("Body", False)
except:
    pass

motion.setStiffnesses("Body", 1.0)

# -------- Baile inicial (20s) --------
duration = 20.0
t0 = time.time()
dt = 0.05
f_base   = 0.35
f_hips   = 0.6
f_head   = 0.5
f_arms   = 0.45

vy_amp   = 0.20
w_amp    = 0.12
hip_amp  = 0.10
head_yaw_amp   = 0.25
head_pitch_amp = 0.08
lsp_base = 1.0
rsp_base = 1.0
sr_amp   = 0.18
el_amp   = 0.20
```

```
speed = 0.25

def safe_set(j, a):
    try:
        motion.setAngles(j, a, speed)
    except:
        pass

while (time.time() - t0) < duration:
    t = time.time() - t0
    ph_base = 2*math.pi*f_base*t
    ph_hip  = 2*math.pi*f_hips*t
    ph_head = 2*math.pi*f_head*t
    ph_arm  = 2*math.pi*f_arms*t

    vy      = vy_amp * math.sin(ph_base)
    vtheta  = w_amp  * math.sin(ph_base + math.pi/2.0)
    motion.moveToward(0.0, vy, vtheta)

    safe_set("HipRoll",  hip_amp * math.sin(ph_hip))
    safe_set("HipPitch", 0.05 + 0.03 * math.sin(ph_hip))

    safe_set("HeadYaw",   head_yaw_amp   * math.sin(ph_head))
    safe_set("HeadPitch", -0.03 + head_pitch_amp * math.sin(ph_head + math.pi/2

    lsp = lsp_base + 0.4 * math.sin(ph_arm + math.pi/2.0)
    rsp = rsp_base + 0.4 * math.sin(ph_arm - math.pi/2.0)
    safe_set("LShoulderPitch", lsp)
    safe_set("RShoulderPitch", rsp)

    safe_set("LShoulderRoll",  sr_amp * math.sin(ph_arm))
    safe_set("RShoulderRoll", -sr_amp * math.sin(ph_arm))

    safe_set("LElbowRoll", -0.4 + (-el_amp) * math.sin(ph_arm + math.pi/3.0))
    safe_set("RElbowRoll",  0.4 + ( el_amp) * math.sin(ph_arm + math.pi/3.0))

    time.sleep(dt)

motion.moveToward(0.0, 0.0, 0.0)
motion.stopMove()

# -------- Giro 360   en 4 pasos de 90   --------
for i in range(4):
    motion.moveTo(0.0, 0.0, math.pi/2, [["MaxVelTheta", 0.4]])
    time.sleep(0.5)

# -------- Movimiento lateral de derecha a izquierda --------
duration = 15.0
```

```
t0 = time.time()
vy_amp   = 0.25
w_amp    = 0.10

while (time.time() - t0) < duration:
    t = time.time() - t0
    ph = 2*math.pi*0.3*t
    vy = vy_amp * math.sin(ph)
    vtheta = w_amp * math.sin(ph + math.pi/2.0)
    motion.moveToward(0.0, vy, vtheta)
    time.sleep(0.05)

motion.moveToward(0.0, 0.0, 0.0)
motion.stopMove()
posture.goToPosture("StandInit", 0.6)
```

4.4. Ejecución

```
python salsota.py
```

5. Conclusiones

- Se comprendió el uso de las librerías fundamentales para el funcionamiento de Pepper.
- Se diseñó y ejecutó una coreografía básica mediante Choregraphe.
- Se desarrolló un baile más complejo programado directamente en Ubuntu mediante consola SSH.
- Se logró integrar librerías matemáticas y de movimiento para generar secuencias sincronizadas.

6. Anexos

- Capturas de pantalla de la conexión en Choregraphe.
- Capturas de la consola SSH y del archivo Python creado.
- Videos de las coreografías (Choregraphe y Ubuntu).