

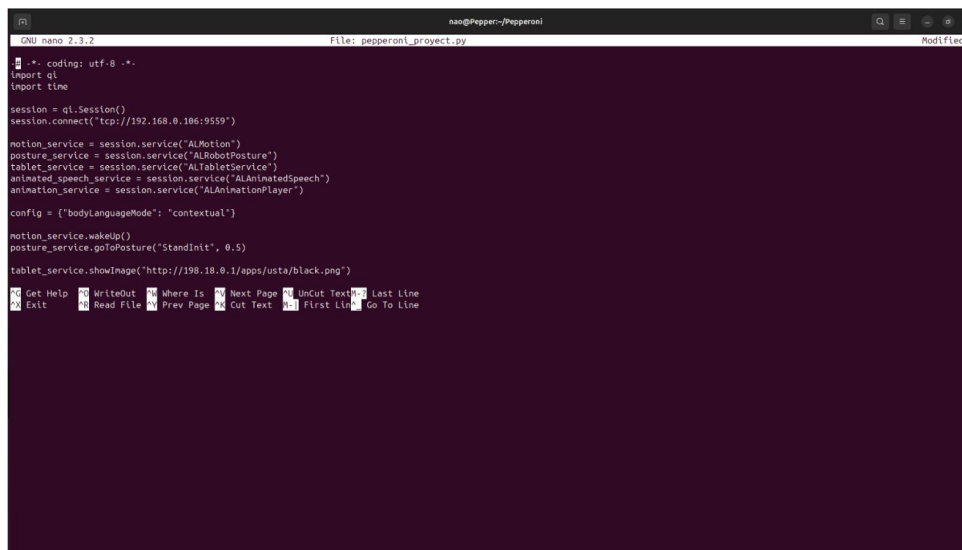
Explicación del Código en Pepper

Oscar Grande, Didier Posse

Introducción

En este documento presento la explicación del código que se desarrolló para el robot Pepper. La idea fue programar una pequeña exposición en la que el robot pudiera hablar, mostrar imágenes en su tablet y al mismo tiempo realizar gestos que acompañaran lo que decía. A lo largo del informe voy a ir mostrando las imágenes del código, explicando qué se hace en cada parte y cómo eso se refleja en la presentación que el robot realiza.

Configuración inicial



```
GNU nano 2.3.2 File: pepperoni_project.py
-*. coding: utf-8 -*-
import qi
import time

session = qi.Session()
session.connect('tcp://192.168.0.106:9559')

motion_service = session.service("ALMotion")
posture_service = session.service("ALRobotPosture")
tablet_service = session.service("ALTabletService")
animated_speech_service = session.service("ALAnimatedSpeech")
animation_service = session.service("ALAnimationPlayer")

config = {'bodyLanguageMode': 'contextual'}

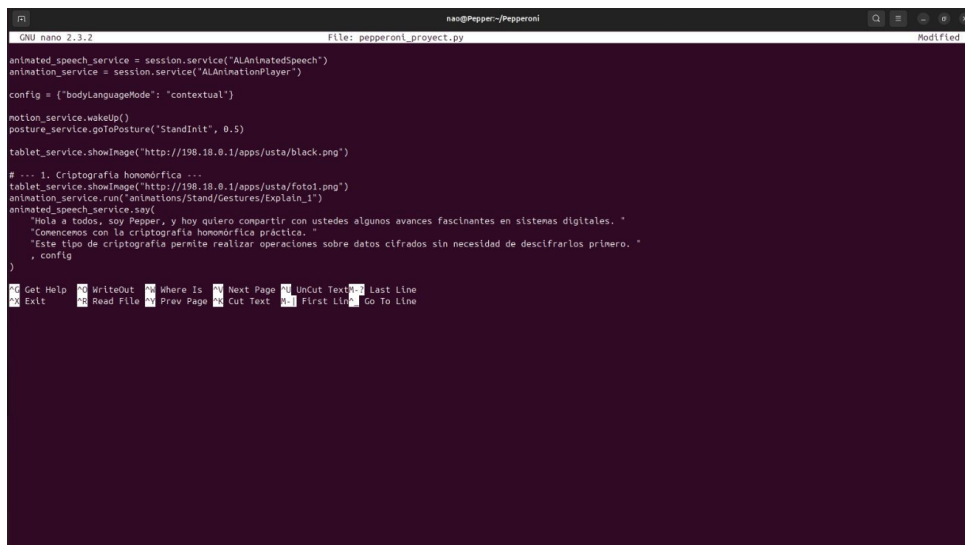
motion_service.wakeup()
posture_service.goToPosture('StandInit', 0.5)
tablet_service.showImage('http://198.18.0.1/apps/usta/black.png')

? Get Help  ? WriteOut  ? Where Is  ? Next Page  ? UnCut Text  ? Last Line
? Exit      ? Read File  ? Prev Page  ? Cut Text   ? First Lin  ? Go To Line
```

Figure 1: Configuración de librerías y servicios iniciales en Pepper

En esta primera parte del código se importan las librerías principales como `qi` y `time`. Luego, se establece la conexión con el robot a través de su dirección IP y el puerto. También se inicializan los servicios que se van a usar: movimiento, posturas, pantalla táctil (tablet), voz animada y animaciones de gestos. Para dejar al robot listo, se ejecuta la instrucción `wakeUp()` y se lo coloca en postura inicial con `StandInit`. Es decir, aquí se prepara todo el entorno para que el robot pueda empezar la exposición sin problemas.

Introducción y criptografía homomórfica



```
nao@Pepper:~/Pepperoni
GNU nano 2.3.2 File: pepperoni_project.py Modified
animated_speech_service = session.service("ALAnimatedSpeech")
animation_service = session.service("ALAnimationPlayer")
config = {"bodyLanguageMode": "contextual"}

motion_service.wakeup()
posture_service.goToPosture("StandInit", 0.5)
tablet_service.showImage("http://198.18.0.1/apps/usta/black.png")

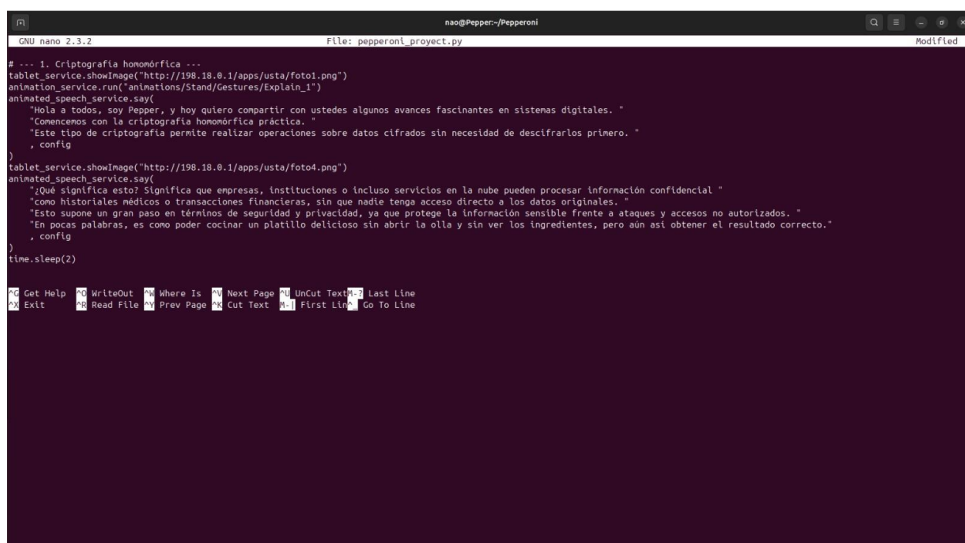
# --- 1. Criptografia homomorfica ---
tablet_service.showImage("http://198.18.0.1/apps/usta/foto1.png")
animation_service.run("animations/Stand/Gestures/Explain_1")
animated_speech_service.say(
    "Hola a todos, soy Pepper, y hoy quiero compartir con ustedes algunos avances fascinantes en sistemas digitales. "
    "Comencemos con la criptografia homomorfica practica. "
    "Este tipo de criptografia permite realizar operaciones sobre datos cifrados sin necesidad de descifrarlos primero. "
    , config
)

Get Help WriteOut Where Is Next Page UnCut Text Last Line
Exit Read File Prev Page Cut Text First Line Go To Line
```

Figure 2: Enter Caption

En esta parte del código, lo primero que hace Pepper es mostrar una imagen en su tablet, en este caso **ALUSIVA AL TEXTO Y TEMATICA**. Después, se lanza una animación de explicación, lo que hace que el robot acompañe sus palabras con movimientos de brazos. A nivel de discurso, aquí Pepper se presenta y empieza hablando de la **criptografía homomórfica práctica**. El código deja claro que la idea es que la presentación sea dinámica, no solo con voz, sino también con gestos y apoyo visual.

Ejemplo de criptografía



```
nao@Pepper:~/Pepperoni
GNU nano 2.3.2 File: pepperoni_project.py Modified
# --- 1. Criptografia homomorfica ---
tablet_service.showImage("http://198.18.0.1/apps/usta/foto1.png")
animation_service.run("animations/Stand/Gestures/Explain_1")
animated_speech_service.say(
    "Hola a todos, soy Pepper, y hoy quiero compartir con ustedes algunos avances fascinantes en sistemas digitales. "
    "Comencemos con la criptografia homomorfica practica. "
    "Este tipo de criptografia permite realizar operaciones sobre datos cifrados sin necesidad de descifrarlos primero. "
    , config
)
tablet_service.showImage("http://198.18.0.1/apps/usta/foto4.png")
animated_speech_service.say(
    "¿Qué significa esto? Significa que empresas, instituciones o incluso servicios en la nube pueden procesar información confidencial "
    "como historiales médicos o transacciones financieras, sin que nadie tenga acceso directo a los datos originales. "
    "Esto supone un gran paso en términos de seguridad y privacidad, ya que protege la información sensible frente a ataques y accesos no autorizados. "
    "En pocas palabras, es como poder cocinar un platillo delicioso sin abrir la olla y sin ver los ingredientes, pero aun así obtener el resultado correcto."
    , config
)
time.sleep(2)

Get Help WriteOut Where Is Next Page UnCut Text Last Line
Exit Read File Prev Page Cut Text First Line Go To Line
```

Figure 3: Explicación práctica de la criptografía homomórfica

Aquí el robot muestra una nueva imagen en la tablet (**Imagen explicativa acerca del tema**) y sigue con otra parte del guion programado. Lo interesante es que se explican ejemplos concretos como el manejo de historiales médicos y transacciones financieras de forma segura. Incluso se utilizan metáforas dentro del texto programado en `say()`, como comparar el proceso con cocinar sin abrir la olla. Esto demuestra que el guion no solo fue técnico, sino también pensado para que la audiencia entienda mejor.

Almacenamiento en ADN

```

GNU nano 2.3.2                                File: pepperoni_project.py                                Modified
animated_speech_service.say(
    "¿Qué significa esto? Significa que empresas, instituciones o incluso servicios en la nube pueden procesar información confidencial."
    "Como historiales médicos o transacciones financieras, sin que nadie tenga acceso directo a los datos originales."
    "Esto supone un gran paso en términos de seguridad y privacidad, ya que protege la información sensible frente a ataques y accesos no autorizados."
    "En pocas palabras, es como poder cocinar un plato delicioso sin abrir la olla y sin ver los ingredientes, pero aún así obtener el resultado correcto."
    , config
)
time.sleep(2)

# --- 2. DNA Data Storage ---
tablet_service.showImage("http://198.18.8.1/apps/usta/foto2.png")
animation_service.run("animations/Stand/Gestures/Explain_2")
animated_speech_service.say(
    "Ahora pasemos a otro tema sorprendente: el almacenamiento de datos en ADN."
    "El ADN no solo contiene la información genética de los seres vivos, también puede ser utilizado como un medio para guardar información digital."
    "Un solo gramo de ADN puede almacenar casi un exabyte de datos, lo que equivale a millones de discos duros convencionales."
    "Además, es increíblemente duradero y estable, lo que lo convierte en una alternativa al almacenamiento tradicional que con el tiempo puede deteriorarse."
    , config
)

Get Help      WriteOut  Where Is     Next Page    UnCut Text   Last Line
Exit          Read File  Prev Page   Cut Text     First Line   Go To Line

```

Figure 4: Tema de almacenamiento en ADN

Luego, se pasa al segundo tema: el almacenamiento de datos en ADN. Aquí Pepper muestra la imagen **acerca del tema** y utiliza otra animación de explicación. En el guion se habla sobre la enorme capacidad del ADN para guardar información, explicando que un gramo puede almacenar un exabyte. Esta parte me pareció muy llamativa porque combina ciencia con un tema futurista, y al mismo tiempo el código hace que la explicación no sea aburrida gracias a los gestos y la tablet.

```

nao@Pepper:~/Pepperoni
GNU nano 2.3.2 File: pepperoni.project.py Modified
tablet_service.showImage("http://198.18.0.1/apps/usta/foto2.png")
animation_service.run("animations/Stand/Gestures/Explain_2")
animated_speech_service.say(
    "Ahora pasemos a otro tema sorprendente: el almacenamiento de datos en ADN. "
    "El ADN no solo contiene la información genética de los seres vivos, también puede ser utilizado como un medio para guardar información digital. "
    "Un solo gramo de ADN puede almacenar casi un exabyte de datos, lo que equivale a millones de discos duros convencionales. "
    "Además, es increíblemente duradero y estable, lo que lo convierte en una alternativa al almacenamiento tradicional que con el tiempo puede deteriorarse. "
    , config
)
tablet_service.showImage("http://198.18.0.1/apps/usta/foto5.png")
animated_speech_service.say(
    "Imaginen un futuro en el que bibliotecas enteras, archivos históricos o incluso las bases de datos más grandes del mundo "
    "puedan conservarse en algo tan pequeño y estable como una cadena de ADN. "
    "Esto revolucionaría la manera en que manejamos y preservamos la información. "
    "Vamos a ver un video corte de como funciona el almacenamiento de datos en ADN."
    , config
)
time.sleep(2)
tablet_service.playVideo("http://198.18.0.1/apps/usta/video2.mp4")
Get Help WriteOut Where Is Next Page UnCut Text Last Line
Exit Read File Prev Page Cut Text First Line Go To Line

```

Figure 5: Explicación de interfaces hápticas por ultrasonido

En esta sección se carga la imagen **foto** en la tablet, mientras Pepper usa otro gesto de la librería. El tema aquí es cómo las ondas ultrasónicas pueden generar sensaciones táctiles en el aire. Me parece que este bloque de código está diseñado para que el público entienda una tecnología bastante compleja de una forma más cercana. El robot incluso menciona ejemplos de aplicación, lo cual se ve escrito directamente en la parte del guion del `say()`.

Conclusión y despedida

```

nao@Pepper:~/Pepperoni
GNU nano 2.3.2 File: pepperoni.project.py Modified
# --- 3. Interfaces hápticas ---
tablet_service.showImage("http://198.18.0.1/apps/usta/foto3.png")
animation_service.run("animations/Stand/Gestures/Explain_3")
animated_speech_service.say(
    "Finalmente, exploremos las interfaces hápticas de ultrasonido. "
    "Estas tecnologías permiten generar sensaciones táctiles en el aire utilizando ondas ultrasónicas. "
    "De modo que podemos sentir texturas, botones o superficies sin necesidad de tocarlas físicamente. "
    "Esto abre la puerta a nuevas formas de interacción en la realidad virtual, ofreciendo experiencias más inmersivas y realistas. "
    , config
)
tablet_service.showImage("http://198.18.0.1/apps/usta/foto6.png")
animated_speech_service.say(
    "También tiene aplicaciones en medicina, por ejemplo, en simuladores de cirugía donde los médicos pueden entrenar y sentir órganos o tejidos de forma virtual. "
    "Y no olvidemos la accesibilidad: personas con discapacidad visual podrían interactuar con interfaces digitales mediante el tacto en el aire. "
    "En conclusión, estas tecnologías representan un futuro donde la seguridad, el almacenamiento y la interacción humano-máquina alcanzan niveles extraordinarios. "
    "Aquí dejo un ejemplo de como se ve un tipo de interfaz háptica"
    , config
)
Get Help WriteOut Where Is Next Page UnCut Text Last Line
Exit Read File Prev Page Cut Text First Line Go To Line

```

Figure 6: Argumentos interfaces hapticas

[H]

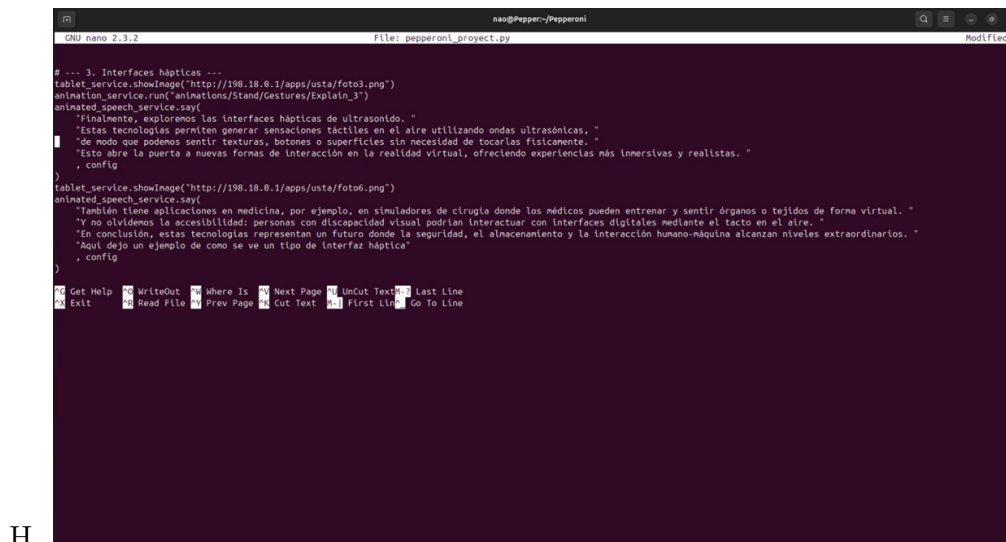


Figure 7:

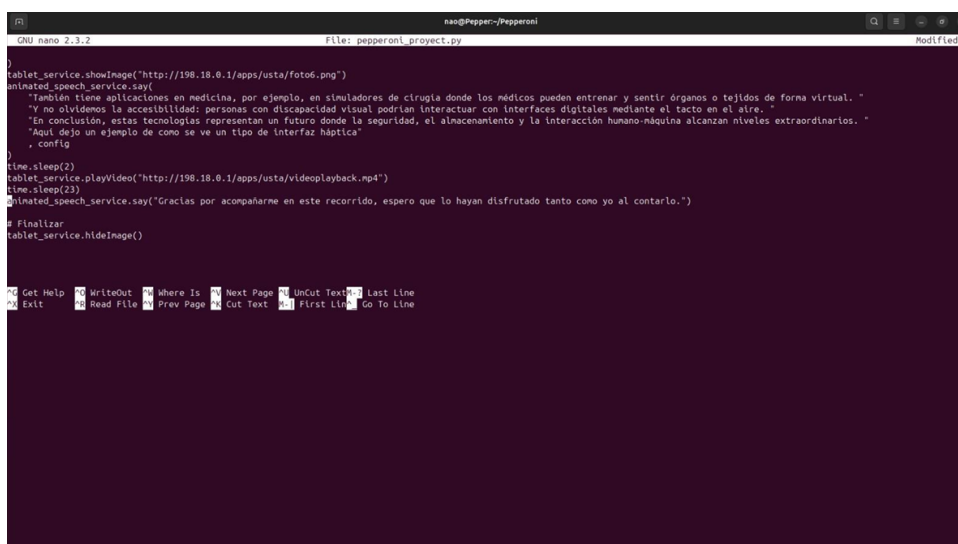


Figure 8: Conclusiones

En las últimas partes del código no se usan tantas imágenes, sino que Pepper se centra en dar un mensaje de cierre. Primero resume los tres temas tratados y luego se despidе con un gesto programado en la animación Hey_1. Finalmente, se agrega una pequeña pausa con `time.sleep()` para que la presentación termine de manera natural. Este detalle muestra que se pensó no solo en el contenido, sino también en la experiencia del espectador.

final

En general, el código está bien estructurado porque combina elementos principales como las imágenes en la tablet, los gestos del robot y el discurso animado. La fundamentación en temas como criptografía, ADN y ultrasonido, combinado con integrar distintas herramientas de Pepper en un mismo programa. +