Conociendo a Pepper

Oscar Grande, Didier Posse.

August 2025

1 Introduccion a Pepper

Pepper es un robot humanoide el cual cuenta que puede interactuar con las personas, teniendo conversaciones con estas y con la capacidad de tener un reconocimiento emocional de ellas, ademas es capaz de exponer informacion de importancia segun sea el caso, fuera de los increibles bailes que puede hacer. Este es un gran acercamiento para los seres humanos con la inteligencia artificial acompañado de mas tematicas de robotica, pero es vital que se conozca acerca de su funcionamiento e informacion de importancias por ello se empezara por estudiar y conocer las siguientes librerias que hacen parte de su funcionamiento.

2 Libreria qi

el robot funciona con ayuda de lenguajes de programacion como python, C++ o Java, en la cual se aplica la libreria qi, la cual es la libreria oficial de framework NAOqi. Esta funciona como la interfaz de comunicaciones entre los programas del desarrollador y los servicios internos de Pepper (funciona como un middleware). Este se conecta a traves del puerto 9559 y permite descubrir las funcionalidades como en forma de servicios. **lo cual permite llamar a funciones de esos servicios, Descubrir servicios disponibles o crear tus propios servicios**. Pepper puede ejecutar muchas funciones en paralelo y emitir y escuchar eventos y expone como servicios a cada modulo del robot (voz, movimiento, sensores, memoria, tablet, etc). Para explorar algunos servicios disponibles en Pepper se pueden consultar usando qicli info y para acceder a cualquier servicio que se necesite "session.service("NombreDelServicio")" para aplicaciones mas complejas usar ALBasicAwareness y ALTextToSpeech para que Pepper salude cuando detecta una presencia, o ALLandMarkDetection con navegación para que se mueva hacia un marcador específico. **Los servicios de a continuación son algunos de los mas importantes en Pepper:**

Servicio	Comando	Para que sirve
ALMotion	session.service("ALMotion")	Controla movimientos, articulaciones, locomoción, rigidez, posturas, trayectorias interpoladas.
ALTextoToSpeech	session.service("ALTextToSpeech")	Síntesis de voz: hablar texto, configurar idioma, entonación, velocidad, SSML, acompañado de gestos.
ALMemory	session.service("ALMemory")	Almacena y recupera datos, gestiona eventos (publisher/subscriber) como detección de personas, palabras, toques.
ALNavigation	session.service("ALNavigation")	Módulo de mapeo y navegación (SLAM); explora entornos y navega de forma autónoma
ALDialog	session.service("ALDialog")	Manejo de diálogo basado en reglas, sistemas de conversación predefinidos
AlRobotPosture	session.service("ALRobotPosture")	Permite cambiar la postura del robot (sentarse, pararse, posición de inicio, etc.)
AlBasicAwareness	session.service("ALBasicAwareness")	Permite percepción básica del entorno: atención a estímulos visuales, auditivos, personas
ALSoundLocalization	session.service("ALSoundLocalization")	Localiza la fuente de sonido en el entorno
ALLandMarkDetection	session.service("ALLandMarkDetection")	Reconoce marcadores visuales específicos llamados NAOMarks para tareas de localización u acción
LearnFace(choregraphe)	...	Permite almacenar y reconocer rostros dentro del entorno del software Choregraphe
ALUserInfo	session.service("ALUserInfo")	Maneja datos persistentes relacionados con los usuarios (perfiles, preferencias)
ALUserSession	session.service("ALUserSession")	Gestiona el estado de usuarios activos y sus interacciones
ALSystem	session.service("ALSystem")	Controla aspectos del sistema operativo del robot (reinicio, apagar, información del sistema)
ALTabletService	session.service("ALTabletService")	Controla la tablet de Pepper: carga de webs, aplicaciones, imágenes, vídeo, control de Wi-Fi, diálogo, navegadores

Table 1: Comandos y servicios mas importantes

3 Libreria Argparse:

Librería estándar de python para el manejo de argumentos de linea de comandos.permitiendo que el script sea dinámico y configurable sin necesidad de modificar el codigo. Aqui se podra definir la IP y el puerto de conexión al robot, también podrá cambiar modos de prueba (simulando en pc o en robot real) seleccionar servicios específicos (solo movimiento, solo voz, etc), esta librería también ayuda a evitar modificar el código cada vez que quieras correrlo con parámetros distintos.

Argumentos Posicionales: obligatorios!, dependen del orden.

Argumentos opcionales: se pasan con prefijo -, pueden tener valores por defecto.

Flags o Banderas: Son Booleanos, Solo activan/desactivan algo.

Help Automatico: Genera una ayuda con -h o –help.

ArgParse es una herramienta fundamental para crear scripts flexibles. Se aplica sobre Pepper para pasar Ip, Puerto y Modo de ejecución.

Servicio	Comando	Para que sirve
argparse.ArgumentParser()	parser = argparse.ArgumentParser()	Crea el objeto parser que gestionará los argumentos.
add$_argument$()	parser.add$_argument$("$--ip$", $type = str$)	Define un argumento (nombre, tipo, valor por defecto, ayuda).
parse$_args$()	args = parser.parse$_args$()	Procesa los argumentos introducidos en la terminal.
description	ArgumentParser(description="Conectar a Pepper")	Texto que describe lo que hace tu script. Aparece en el –help.
help	parser.add$_argument$("$--ip$", $help =$ "$DirecciónIP$")	Explica cada argumento cuando el usuario usa -h o –help.
default	parser.add$_argument$("$--port$", $default = 9559$)	Valor por defecto si el usuario no lo pasa.
type	parser.add$_argument$("$--port$", $type = int$)	Define el tipo de dato (int, float, str, etc.).
required=True	parser.add$_argument$("$--ip$", $required = True$)	Hace que el argumento sea obligatorio.
choices	parser.add$_argument$("$--mode$", $choices =$ ["voz","mov"])	Restringe los valores posibles.
action="store$_true$"	parser.add$_argument$("$--debug$", $action =$ "$store_true$")	Activa una bandera booleana si se pasa el argumento.
nargs	parser.add$_argument$("$--coords$", $nargs = 3$)	Permite que un argumento acepte varios valores (ej: X, Y, Z).
metavar	parser.add$_argument$("$--ip$", $metavar =$ "$DIRECCION$")	Personaliza el nombre mostrado en la ayuda.
-h / –help	python script.py -h	Muestra automáticamente la ayuda de todos los parámetros.

Table 2: Comandos y servicios mas importantes argparse.

4 Libreria Sys

La libreria Sys dará y apoyara el control del programador sobre el interprete de python y su interacción con el sistema operativo del robot. En el caso de Pepper la libreria qi y argparse trabajan en conjunto con la libreria sys, ayudando administrar los argumentos al script, cuando se ejecuta el script se necesita

dirigir la informacion desde la terminal, como la IP del robot o el puerto o a los parametros de comportamiento, lo que hace sys es capturar esa información como por ejemplo la Ip de Pepper y la contiene como lista. Con sys también se puede añadir manualmente la carpeta donde están instaladas las librerías de Naoqi (qi, almotion o altexttospeech, etc). En el robot o en el ordenador ya que no siempre estan en el path de python por defecto. Al momento que pueda ocurrir un fallo sys lo que hace es terminar el código con un código de error lo cual es útil si se controla desde otro sistema. Entonces es decir:

- sys esta acoplado al interprete de python, es decir sus objetos (argv, path,stdout,etc)son referencias directas a estructuras internas del interprete.

- es una librería de bajo nivel "expone las entrañas del entorno de ejecución.

-es vital porque permite correr scripts portables(cambiar ip/puerto sin reescribir).

- argparse se apoya en sys.argv para traducir argumentos, mientras que qi depende de esos argumentos (ip/port) para conectarse al robot, lo que hace sys en este caso es controlar las salidas, errores, y rutas de librerías para que todo funcione bien.

Comando	Para que sirve
sys.argv	Lista con los argumentos que recibe el script desde la terminal
sys.exit([code])	Termina la ejecución del programa de forma inmediata. Puedes pasar un código (0 = correcto, distinto de 0 = error).
sys.path	Lista de rutas donde Python busca módulos y librerías. Puedes añadir la ruta de qi o de librerías propias.
sys.version	Devuelve la versión exacta de Python que se está usando. Útil para compatibilidad con qi en Pepper.
sys.platform	Indica en qué sistema operativo se ejecuta Python (Linux, Windows, etc.). Pepper usa Linux (NAOqi OS).
sys.modules	Diccionario con los módulos cargados actualmente. Permite saber si qi ya está importado.
sys.stdin	Flujo de entrada estándar (teclado o canal de entrada). Puedes usarlo para que Pepper lea datos externos.
sys.stdout	Flujo de salida estándar (pantalla). Puedes redirigirlo para guardar logs o depuración de Pepper.
sys.stderr	Flujo de errores estándar. Útil para capturar y registrar errores de ejecución.
sys.getsizeof(obj)	Devuelve el tamaño en bytes de un objeto en memoria. Útil si trabajas con datos pesados en Pepper.
sys.getdefaultencoding	Indica la codificación por defecto de Python. Importante cuando Pepper procesa texto o habla.
sys.maxsize	Tamaño máximo que puede tener un entero (int) en esa instalación de python.
sys.exc$_i nfo$()	Devuelve información sobre la excepción que se está manejando (tipo, valor, traceback).Util para depuración.
sys.setrecursionlimit(n)	Establece el máximo de llamadas recursivas permitidas. Evita errores de stack overflow.

Table 3: Comandos y servicios mas importantes sys.

5 Librería Os

La librería os (operating System) en Python es el modulo estándar que permite a los programas interactuar con el sistema operativo subyacente. Este modulo es muy importante y útil porque el robot corre sobre NAOqi OS (Linux basado en Gentoo) y os ofrece las herramientas necesarias para trabajar directamente con su sistema. Este funciona para:

-Gestion de archivos y directorios(crear,borrar, mover y renombrar).

-Acceso a las variables de entorno del sistema(Útil para configurar servicios).

-Ejecuta comandos del sistema operativo desde python.

-Facilita el manejo de informacion del sistema (usuario,CPU,memoria y permisos).

En Pepper su funcionamiento se orienta en organizar los datos generados por el robot:

-Guardar fotos, audios, registros de interacción.

-Automatizar procesos internos (ejecutar scripts sin intervencion manual).

-Controlar recursos del sistema (Verificar memoria, CPUS, permisos).

-Integrarse con otras librerias (qi,sys, argparse)asegurando rutas y configuraciones correctas

-Acceder al entorno Linux de Pepper configurando directorios de logs y servicios de NAOqi.

Comando	Descripción	Para que sirve
os.getcwd()	Devuelve el directorio de trabajo actual.	Saber en qué carpeta corre el script en Pepper.
os.chdir(path)	Cambia el directorio de trabajo actual.	Moverse al directorio donde están los módulos qi.
os.listdir(path)	Lista archivos y carpetas de un directorio.	Ver fotos, audios o logs guardados por Pepper.
os.mkdir(path)	Crea un directorio nuevo.	Guardar datos de interacción con usuarios.
os.makedirs(path)	Crea directorios de manera recursiva.	Crear estructuras de carpetas para proyectos en Pepper.
os.remove(file)	Elimina un archivo.	Borrar grabaciones o imágenes antiguas.
os.rmdir(path)	Elimina un directorio vacío.	Mantener limpio el sistema de Pepper
os.rename(src, dst)	Renombra o mueve un archivo/directorio.	Organizar registros de conversaciones.
os.path.exists(path)	Verifica si una ruta existe.	Evitar errores al acceder a archivos en Pepper.
os.path.join(a, b)	Une rutas de forma segura.	Asegurar rutas correctas en Linux de Pepper.
os.environ	Accede a variables de entorno.	Configurar servicios de NAOqi (qi)
os.system(cmd)	Ejecuta un comando del sistema operativo.	Lanzar procesos adicionales en Pepper.
os.name	Devuelve el tipo de sistema operativo.	Confirmar que Pepper corre sobre "posix".
os.uname()	Devuelve información del sistema.	Ver versión de NAOqi OS instalada.
os.cpu$_c ount$()	Devuelve el número de CPUs.	Ajustar tareas sin saturar a Pepper.
os.getlogin()	Muestra el usuario actual.	Verificar usuario del sistema que corre el script.
os.stat(file)	Devuelve info de un archivo (tamaño, permisos, etc.).	Controlar accesos a datos sensibles.

Table 4: Comandos y servicios mas importantes os.

6 Librería Almath

Almath es una libreria matematica que se aplica para los robots humanoides ("Cerebro matematico de Pepper"), a diferencia de Math (libreria estandar de python). Esta libreria incorpora conceptos de gemoteria robotica, como:

-Espacios de coordenadas homogéneas (transformaciones 4x4)

-Cinematica directa e inversa

- Matrices de rotacion y traslacion.

-Representacion de pases 2D y 3D.

-Normas vectoriales y distancias.

Estos calculos son de vital importancia porque los actuadores de Pepper (motores) solo entiende posiciones angulares, es decir que almath traduce las "percepciones o acciones espaciales humanas" en calculos para motores.

Su funcionamiento es a base de algebra lineal y transformaciones homogéneas 4x4 que permiten combinar traslación o rotación en un solo objeto matemático, también usa quaterniones y ángulos de euler para representar rotaciones sin ambigüedades, ademas de la geometría en $\mathrm{R}^2 y R^3$.

La cual se aplica posiciones en 2D (Suelos) y 3d (espacio).

Las funcionalidades en Pepper serian las siguientes:

-Calcular trayectorias para caminar (2D).

-Ajustar desplazamientos de brazos, cabeza o torso.

-Mantener estabilidad (equilibrio dinámico).

-Convertir coordenadas de cámara (mundo).

-Calcular distancias entre Pepper y objetos o personas.

-Determinar orientaciones para mirar o apuntar con las extremedidades superiores del cuerpo.

- Esta hace conjunto con Almotion, es decir Almath hace los cálculos matemáticos y Almotion ejecuta la orden de los motores.

Comando	Representación Matemática	Para que sirve
almath.Pose2D(x, y,)	Representa una posición en 2D: traslación (x, y) y rotación (en radianes).	Programar movimientos en el suelo ("camina 1m adelante y gira 90°").
almath.Pose3D(x, y, z, wx, wy, wz)	Representa posición en 3D con orientación. wx, wy, wz = ángulos de rotación en ejes.	Definir dónde está la mano o la cabeza respecto al torso.
almath.Transform(position, rotation)	Matriz homogénea 4x4 que combina traslación y rotación.	Convertir coordenadas de cámara → coordenadas del robot.
almath.Transform.multiply(T1, T2)	Multiplica dos transformaciones homogéneas.	Encadenar movimientos (ej. brazo + torso).
almath.Transform.inverse(T)	Invierte una transformación (cambia de marco de referencia).	Pasar de mundo a cámara, o de torso a pie.
almath.rotation3DFromAxisAngle(axis, angle)	Calcula una matriz de rotación en 3D desde un eje + ángulo.	Rotar cabeza o torso hacia un punto exacto.
almath.vectorNorm(v)	Norma de un vector (magnitud: $(x^2+y^2+z^2)$).	Calcular distancia real hasta un objeto detectado.
almath.distance3D(v1, v2)	Distancia entre dos puntos en espacio 3D.	Medir cuán lejos está una persona.
almath.inRange(x, min, max)	Verifica si un valor está en un rango válido.	Comprobar si un ángulo de articulación es seguro.
almath.clamp(value, min, max)	Limita un valor entre dos límites.	Restringir velocidad de motores para seguridad.
almath.Tools.toDegrees(r)	Convierte radianes a grados.	Mostrar información en grados (más entendible).
almath.Tools.toRadians(d)	Convierte grados a radianes.	Enviar comandos de ángulos a motores.

Table 5: Comandos y servicios mas importantes Almath.

7 Libreria Math

La librería Math es un modulo estándar que proporciona funciones matemáticas avanzadas como redondeo, potencias, constantes matemáticas, trigonometría, logaritmos, conversiones, etc. Lo único es que no trabaja con matrices y vectores, si no con operaciones sobre números reales (enteros y decimales tipo float). Lo que permitira resolver operaciones matematicas complejas que no esten disponibles con operadores basicos(+,-,*,/). Se apoya de Almath y trigonometría para manejar los ángulos de articulaciones en la robótica, a su vez de cálculos en movimiento, orientación espacial, rotaciones y cinemática, ofreciendo constantes matematicas universales como pi o e, etc que aportan en simplificar la programacion de este mismo.

Las funcionalidades principales en Pepper son:

-Movimientos: Sus articulaciones y posiciones se definen en angulos radianes, los cuales requieren de funciones trigonometricas.

-Orientacion Espacial: Cuando realiza cualquier movimiento se necesita calcular transformaciones angulares con math.

-Procesamiento de Sensores: Datos de acelerometros, giroscopios y camaras pueden necesitar calculos matematicos (distancias o angulos).

-Cinematica Inversa y Directa: Usando funciones trignometricas para calcular trayectorias de brazos o desplazamiento.

-Integracion con Almath las cuales se combina con Math para realizar calculos base.

Comando	Descripción	Para que sirve
math.pi	Constante (3.14159...)	Convertir grados a radianes para mover articulaciones.
math.e	Constante de Euler (2.718...)	Modelos de crecimiento, cálculos exponenciales (ej: atenuación de señales).
math.radians(x)	Convierte grados a radianes.	Definir movimientos desde ángulos humanos (grados) a ángulos de Pepper (radianes).
math.degrees(x)	Convierte radianes a grados.	Mostrar resultados de movimientos de Pepper en grados comprensibles.
math.sin(x)	Seno de un ángulo en radianes.	Movimiento de brazos oscilatorios, caminata.
math.cos(x)	Coseno de un ángulo en radianes.	Orientación de cabeza, giros.
math.tan(x)	Tangente de un ángulo	Cálculos de inclinación de torso.
math.asin(x)	Arcoseno.	Cinemática inversa de articulaciones.
math.acos(x)	Arcocoseno.	Calcular posiciones posibles de brazos.
math.atan2(y, x)	Arcotangente considerando cuadrante.	Calcular dirección hacia un objeto en visión.
math.sqrt(x)	Raíz cuadrada.	Distancias euclidianas entre puntos detectados.
Amath.pow(x, y)1	Potencia (x^y).	Ajustes de intensidad o escalamiento de trayectorias.
math.log(x)	Logaritmo natural.	Procesamiento de datos de sensores.
math.log10(x)	Logaritmo base 10.	Normalización de señales.
math.exp(x)	e^x.	Modelos de atenuación o crecimiento exponencial.
math.floor(x)	Redondea hacia abajo.	Ajustar posiciones discretas.
math.ceil(x)	Redondea hacia arriba.	Evitar pérdida en cálculos de trayectorias.
math.fabs(x)	Valor absoluto.	Control de error en movimientos.
math.hypot(x, y)	($x^2 + y^2$), distancia en 2D.	Distancia a objetos detectados por cámara.
math.copysign(x, y)	Devuelve x con el signo de y.	Mantener direcciones en cálculos de giros.

Table 6: Comandos y servicios mas importantes Math.

8 Librería Motion

La librería Motion de NAOqi es un modulo de control de movimientos el cual le permitirá manejar posturas, cinemáticas, articulaciones o locomoción, donde se incluye la cabeza, brazos, manos, torso, piernas, desplazamiento y equilibrio. Este trabaja con angulos absolutos (Posicion exacta) o angulos relativos (respecto a su posicion actual), estos angulos se basan en radianes para los angulos, por ello este se combina con Math y Almath para los calculos previos de movimiento y generalmente se debe usar junto a ALmotion en python para ejecutar los movimientos, su funcionamiento se basa:

-movimiento de articulaciones individuales: como doblar brazo, girar cabeza.

-Posturas completas: como ponerse de pie o sentarse.

- locomoción: caminar hacia adelante, girar, retroceder.

-Manipulacion de manos: abrir y cerrar dedos.

-Cinematica inversa: como por ejemplo llevar la mano a un punto especifico en el espacio.

-Control de rigidez: activar o desactivar motores de articulaciones.

Servicio	Comando	Para que sirve
setStiffnesses(names, stiffnesses)	Activa/desactiva rigidez (motores encendidos o apagados).	Preparar brazo/cabeza para moverse o relajarla.
getStiffnesses(names)	Consulta rigidez actual.	Saber si una articulación está lista para moverse.
setAngles(names, angles, speed)	Mueve una articulación a un ángulo específico.	Girar cabeza 30°, levantar brazo.
changeAngles(names, changes, speed)	Cambia ángulo relativo a la posición actual.	Subir un poco más el brazo sin especificar posición exacta.
getAngles(names, useSensors)	Devuelve ángulos actuales	Saber en qué posición está la cabeza.
openHand(name)	Abre la mano.	Pepper da la mano, agarra objeto.
closeHand(name)	Cierra la mano.	Pepper sostiene algo o hace gesto de puño.
goToPosture(postureName, speed)	Cambia a una postura predefinida.	"StandInit", "Sit","StandZero".
getPosture()	Consulta la postura actual.	Saber si Pepper está de pie o sentado.
walkTo(x, y, theta)	Camina a una posición relativa.	Avanzar 1 metro, girar 90°.
moveTo(x, y, theta)	Similar a walkTo, pero con control continuo.	Navegar a un punto
moveToward(x, y, theta)	Movimiento continuo (mientras dure el comando).	Seguir un objeto o persona.
stopMove()	Detener movimiento de locomoción.	Interrumpir caminata inmediatamente.
getRobotPosition(useSensors)	Posición del robot en coordenadas.	Saber dónde está Pepper en un espacio.
setFootSteps(leftFootSteps, rightFootSteps, ...)	Caminar con pasos específicos.	Gait controlado para terrenos irregulares.
wbEnable(True/False)	Activa Whole Body Balancer	Mantener equilibrio en movimientos.
wbGoToBalance	Ajustar equilibrio al moverse.	Evitar caídas al estirarse.
wbSetEffectorControl(name, target)	Control de efectores (mano, torso).	Llevar la mano a una posición en el aire.
angleInterpolation(names, angleLists, timeLists, isAbsolute)	Animaciones suaves entre posiciones.	Secuencia de saludo.
setFootContactProtection(True/False)	Activar protección de contacto.	Evitar caídas cuando pisa algo inesperado.

Table 7: Comandos y servicios mas importantes para Motion.

9 Libreria httplib

La libreria httplib es un modulo estandar de python 2.x que implementa un cliente Http/ Https de bajo nivel, en cual proporciona clases y metodos para crear y gestionar conexiones con servidores que hablen el protocolo Http (HyperText Transfer Protocol). Este funciona como medio de comunicacion cliente-servidor. Ojo! porque a partir de python 3.0 fue reemplazado por el modulo http.client pero el SDK oficial corre sobre python 2.7 en el sistema NAOqi.

Principalmente este permite a un programa python establecer comunicacion con algun servidor remoto. Con el fin de enviar solicitudes (get, post, delete, head, options) a un servidor web y a su vez recibir respuestas en disctintos formatos como HDML, JSON, XML, etc. A su vez gestiona estados de conexion y errores, manejando cabeceras y transferencia de datos en la nube.

Funcionalidades:

-Integracion con servicios de la nube, conexion con bases de datos y servidores externos.

-Control remoto del robot permitiendo que un sistema externo envie instrucciones via HTTP.

- Enviar y rebir datos sensoriales internos de Pepper (bateria, temperatura, fallos) o tambien externos a un servidor para su analisis.

- Interoperabilidad consistemas de domotica o IoT, para activar sistemas inteligentes desde la red.

- Seguridad y cifrado (HTTPS) el cual brinda una comunicación protegida en entornos empresariales o con datos sensibles.

Comando	Descripción	Para que sirve
httplib.HTTPConnection(host, port)	Establece una conexión HTTP sin cifrar.	Conectar a servidores internos en la misma red (LAN).
httplib.HTTPSConnection(host, port)	Establece conexión cifrada SSL/TLS.	Conectar a servicios en la nube de forma segura.
conn.request(method, url, body, headers)	Envía una solicitud HTTP con método definido	Mandar datos de sensores (POST) o consultar comandos (GET).
conn.getresponse()	Recupera la respuesta de un servidor tras la solicitud	Procesar lo que el servidor ordena que haga Pepper
response.status	Código HTTP de la respuesta (200, 404, 500, etc.).	Saber si el servidor respondió correctamente.
response.reason	Texto asociado al código HTTP	Diagnóstico de errores en Pepper (ej: "Not Found").
response.read()	Devuelve el cuerpo completo de la respuesta (string, JSON, etc.)	Extraer instrucciones que Pepper debe ejecutar
conn.close()	Cierra la conexión activa.	Liberar recursos de red en Pepper.
conn.putrequest()	Construye manualmente una petición HTTP.	Usado en casos avanzados (peticiones personalizadas).
conn.putheader()	Añade cabeceras HTTP a la petición.	Autenticación con tokens o claves API
conn.endheaders()	Finaliza la configuración de la petición.	Completar configuración antes de enviar datos.
conn.send(data)	Envía datos binarios o JSON en la conexión.	Transmitir información compleja a un servidor remoto.

Table 8: Comandos y servicios mas importantes de httplib.

10 Libreria Json

Modulo estándar que permitirá trabajar con datos en formato Json ("Javascript Object Notation"), el cual es un formato que permite ser legible por el ser humano y procesable para las maquinas, este es un formato ligero y universal para el intercambio de datos entre aplicaciones. Aquí podrá serializar datos donde se convierte las estructuras de python en diccionarios, listas o duplas , etc en cadenas Json para que sean transmitidas o almacenadas. El deserializar datos es lo contrario del anterior, donde lee los datos Json y lo convertirá en estructuras Python que las permitirá manipular, para el intercambio de información el cual facilita la comunicación con Apis, servicios web y módulos internos de robots. En opciones avanzadas también funciona con el objetivo de caracterización de caracteres, precisión de números, orden y formato de salida o adaptación de objetos personalizados en python, ayudando a optimizar el uso y ahorro de memoria y tiempo de procesamiento.ALMotion y ALMemory trabajan en conjunto con Json para xportar los datos estructurados de manera ligera y muy fácil de transportar, algunas de las funcionalidades en Pepper especificas son las siguientes:

- Útil para almacenar logs y datos (como respuestas de los usuarios, tiempos de reacción o posiciones de robot).

-Almacenar configuraciones de movimientos, diálogos o interacciones en archivos Json para reutilizarlos en cualquier momento fácilmente.

-Usado para interactuar con Apis externas desde Pepper.

-Los servicios de Pepper devuelven mucha información estructurada en formato Json como estados de sensores, resultados de visión, configuración de movimientos, etc.

Comando	Descripción	Para que sirve
json.dumps(obj, **kwargs)	Convierte un objeto Python en cadena JSON. kwargs permite personalizar salida.	Serializar estados del robot (ej: ángulos de articulaciones) para enviarlos a un servidor de monitoreo.
json.loads(str)	Convierte una cadena JSON en un objeto Python.	Interpretar respuestas de APIs externas (traducción, clima, chatbots).
json.dump(obj, file, **kwargs)	Igual que dumps, pero escribe en un archivo JSON.	Guardar configuraciones de diálogos, rutas de movimiento o interacciones del robot.
json.load(file)	Lee un archivo JSON y devuelve un objeto Python.	Cargar parámetros personalizados (ej: respuestas de conversación predefinidas).
cls=JSONEncoder (en dumps/dump)	Permite usar un encoder personalizado para serializar objetos complejos.	Serializar posturas (objetos con ángulos y posiciones) en JSON.
$object_hook (en loads/load)$	Función personalizada que transforma el dict en un objeto específico.	Reconstruir objetos de Pepper desde JSON (ej: convertir coordenadas en una clase Postura).
$parse_int, parse_float$	Forzar que números en JSON se lean como enteros/decimales específicos.	Asegurar precisión en medidas de Pepper (ej: distancias en visión).
$sort_keys = True$	Ordena las claves al serializar.	Generar datos ordenados para depuración del comportamiento del robot.
indent=n	Formato legible con sangría.	Guardar logs de interacción humana-robot en JSON con buen formato.
separators=(',', ':')	Comprime salida quitando espacios.	Optimizar transmisión de datos a servidores desde Pepper.
skipkeys=True	Ignora claves no válidas en diccionarios al convertir a JSON.	Evita fallos si Pepper guarda datos no estándar.
$ensure_ascii = False$	Permite mantener caracteres Unicode (acentos, eñes).	Pepper puede guardar y leer texto en español correctamente.
$check_circular = False$	Omite la detección de referencias circulares	Útil en estructuras complejas, aunque debe usarse con cuidado.
$allow_nan = True$	Permite valores como NaN, Infinity.	Procesar datos de sensores que devuelven valores especiales.
default=function	Define cómo serializar objetos no estándar de Python.	Serializar directamente objetos propios de Pepper sin convertirlos manualmente.

Table 9: Comandos y servicios mas importantes de Json.