

Mendoza - Argentina



UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO



FACULTAD
DE INGENIERÍA

MANIPULACIÓN REMOTA Y LOCAL DE UN ROBOT DE 3GDL

PROYECTO FINAL DE CÁTEDRA

Programación Orientada a Objetos
Profesor: Esp. Ing. César Omar Aranda

Cantú Tsallis, Maximiliano
maxi.cantu99@gmail.com

Abril 2023

Índice

INTRODUCCIÓN	2
1. Resumen	2
2. Descripción	2
3. Investigación previa	3
MARCO TEÓRICO	4
1. Cliente	4
2. Servidor	4
3. Tipos de control	5
DESARROLLO	6
1. Diagramas de Clases	6
1.1. Estructura Cliente	6
1.2. Estructura Servidor	8
2. Diagramas de Secuencia	10
2.1. Control Local	10
2.2. Control Remoto	11
3. Diagrama de Actividad	12
EJEMPLOS DE USO	14
1. Ejemplo Control Local	14
2. Ejemplo Control Remoto	19
CONSLUSIONES	24
BIBLIOGRAFÍA	25

INTRODUCCIÓN

1. Resumen

El siguiente informe expone el desarrollo e implementación de un sistema capaz de controlar un robot de 3 grados de libertad, mediante la implementación de la Programación Orientada a Objetos (POO). Para lograrlo, se utilizó el concepto de aplicación del tipo cliente/servidor, en el cuál el cliente se desarrolló en el lenguaje C++, mientras que el servidor se desarrolló en Python.

El objetivo es obtener un sistema en el cuál el robot se encuentre conectado directamente al servidor, para lograr el control deseado de forma manual o automática; brindando la posibilidad de un control local desde el propio servidor o un control remoto desde donde se ejecute el cliente.

2. Descripción

Para la implementación se consideró un robot planar de 3 grados de libertad, como el que se puede observar en la figura 1. El mismo cuenta con una configuración rotacional, es decir, sus 3 articulaciones son rotacionales(RRR) lo que le permite posicionarse y orientarse en el plano. Por último, el robot posee una pinza simple como efector final.

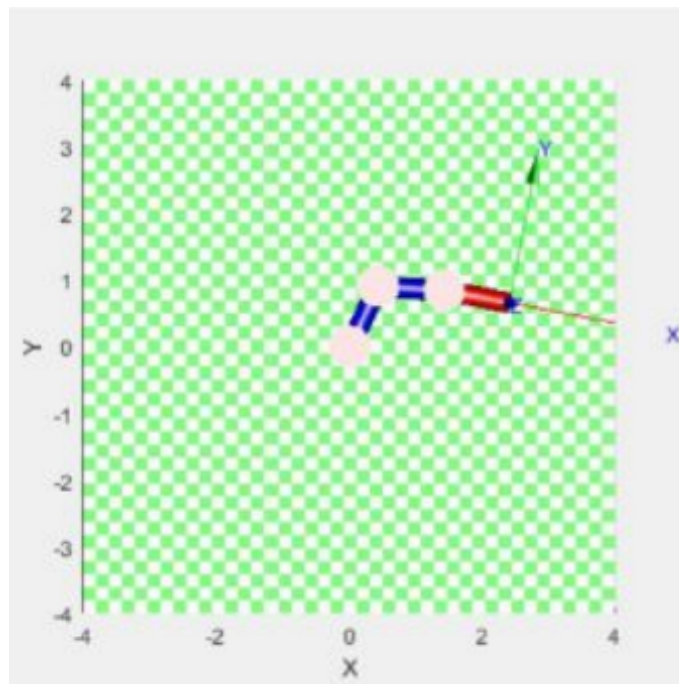


Figura 1: Representación de robot de 3gdl RRR

El robot es parametrizado por el algoritmo de Denavit-Hartenberg, el cual provee una forma sencilla y práctica de simular y calcular el movimiento de esos tipos de robots. Esta parametrización, sumada a expresiones que describen el comportamiento cinemático directo y cinemático inverso, permite realizar un control en función de las coordenadas articulares. Teniendo en cuenta la teoría de control de robots series, se resuelven 2 modos de movimiento y control del robot:

- Inverso: Ingresando posiciones articulares deseadas para cada uno de los eslabones, y obteniendo la posición y orientación del efector final.

- Directo: Ingresando la posición y orientación deseada del efector final, y obteniendo las posiciones articulares de cada eslabón.

Según el modo utilizado y los datos ingresados, es el movimiento que se obtiene del robot, y la respuesta del sistema.

Para lograr el cálculo de la cinemática del robot, el correcto funcionamiento del robot y el control satisfactorio del mismo, es necesaria una buena comunicación entre cliente/servidor y es por esto que se utiliza el protocolo XML-RPC.

Dicho protocolo es una especificación y un conjunto de implementaciones que permiten que el software se ejecute en sistemas operativos diferentes, que se ejecutan en diferentes entornos, para realizar llamadas a procedimientos remotos (RPC, por sus siglas en inglés) a través de internet. Las llamadas remotas se logran utilizando HTTP como protocolo de transmisión de mensajes, es decir, como transporte; y XML como codificación.

XML-RPC está diseñado para ser lo más simple posible y su funcionamiento se lo puede observar en la figura 2. Un mensaje es una solicitud HTTP-POST del cliente, donde el cuerpo de dicha solicitud está formateada en XML. Luego, se ejecuta el procedimiento solicitado en el servidor y se devuelve un valor también formateado en XML. Finalmente, se envía la respuesta como HTTP-POST para ser recibida por el cliente.

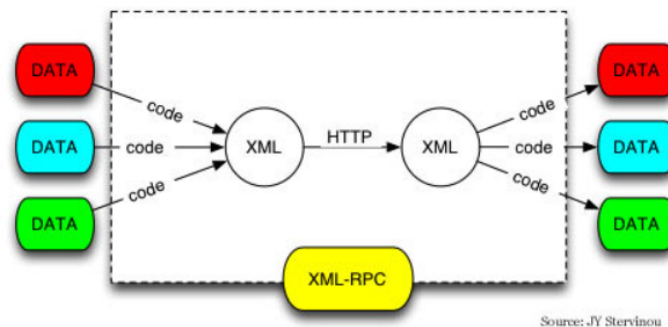


Figura 2: XML-RPC

3. Investigación previa

Para el desarrollo del cliente en C++, fue necesario investigar sobre los siguientes aspectos:

- Concepto y uso de streams de datos.
- Utilización de clases de contenedores de datos (listas, vectores).
- Manejo de los procedimientos de llamadas a procedimientos remotos, mediante protocolo XML-RPC.

Para el desarrollo del servidor en Python, fue necesario investigar sobre los siguientes aspectos:

- Manejo de los procedimientos de llamadas a procedimientos remotos, mediante protocolo XML-RPC.
- Uso de archivos.
- Librería 'Tkinter' para implementar una interfaz visual en el modo de control local.
- Lenguaje de modelado unificado (UML) para confeccionar diagramas de clases y de secuencias.

MARCO TEÓRICO

1. Cliente

El lado del cliente del sistema se desarrolla en el lenguaje C++, un lenguaje multiparadigma, interpretado, hereda su sintaxis del lenguaje C, y es de los más antiguos y difíciles de implementar. En las aplicaciones con estructura de cliente/servidor, el cliente corresponde a la parte utilizada por el usuario en cuestión, de forma remota respecto al servidor.

El usuario ingresa comandos que serán interpretados en el cliente, luego serán enviados como instrucciones al servidor para que sean ejecutadas. Al finalizar la ejecución, el cliente recibe los resultados de las tareas realizadas y las observa en la interfaz de usuario o consola.

Para lograr la comunicación del cliente con el servidor de forma exitosa, se utiliza la consola propia del sistema operativo, mediante la cual el usuario ingrese los comandos de la tarea a ejecutar, basándose en la tabla 1, donde cada comando debe ser ingresado tal como se indica, y para ingresar múltiples comandos deben ser separados por '-':

COMANDO	INSTRUCCIÓN	DESCRIPCIÓN
C	Conectar	Establecer conexión con el servidor
D	Desconectar	Desconectar del servidor
Ac	Activar	Activación del robot
De	Desactivar	Desactivación del robot
A	Ayuda	Menú de ayuda para ingresar comandos
L	Listado/Reporte	Listado o reporte de instrucciones realizadas y hora desde la conexión del servidor
MH	Modo Homing	Movimiento del robot a su posición de origen
MEx,y, α ,t	Modo Efecto	Movimiento del robot para satisfacer la posición '(x,y)', la orientación ' α ' y el tiempo 't' solicitados
MVq ₁ , q ₂ , q ₃ , v ₁ , v ₂ , v ₃	Modo Vínculo	Movimiento del robot para satisfacer las posiciones articulares 'q ₁ , q ₂ , q ₃ ', y las velocidades articulares 'v ₁ , v ₂ , v ₃ ' solicitadas
MAi	Modo Automático	Movimiento automático del robot con instrucciones pre-cargadas en el archivo 'i' almacenado

Tabla 1: Comandos a ingresar por el cliente

2. Servidor

El otro lado, el del servidor, se desarrolla en lenguaje Python, el cual también es un lenguaje interpretado y multiparadigma, pero más moderno y simple de utilizar, con una sintaxis mas intuitiva. En las aplicaciones con estructura de cliente/servidor, el servidor corresponde al sistema en el cual se ejecutan todas las instrucciones.

En este caso, las instrucciones pueden ser ingresadas de forma local por el usuario o de forma remota desde un cliente, la diferencia radica en la comunicación utilizada para enviar y recibir dichas instrucciones y solicitudes. Si se realizan de forma local, se ejecutan en el propio servidor, y se evita la transmisión y recepción de datos de forma remota.

Se desarrolla una interfaz visual para el modo de control local, utilizando la librería GUI 'Tkinter', la cual proporciona un conjunto de herramientas de ventanas robusto e independiente

que funciona como un conjunto de contenedores que implementan los widgets de 'Tk' como clases de Python. Las principales virtudes son que es rápido y que normalmente viene incluido con Python y para utilizarla simplemente hay que importarla como 'import tkinter', sin instalar demás extensiones.

3. Tipos de control

Tal como se menciona anteriormente, el control del robot puede realizarse de 2 formas diferentes y excluyentes, es decir, no se pueden ejecutar ambos controles de manera simultánea:

- **Control Local:** Se crea un objeto controlador en el cual se ejecutan los métodos respectivos a cada modo de funcionamiento del robot, donde cada uno devuelve la respuesta según lo solicitado. En dicho control, se implementa la interfaz GUI mencionada previamente para realizar el control y los mensajes de respuesta se visualizan en la consola del sistema, evitando así el ingreso de instrucciones de forma manual desde consola.
- **Control Remoto:** El usuario ingresa las instrucciones de forma remota desde el cliente y la comunicación se logra utilizando la librería XML-RPC correspondiente a dicho protocolo, haciendo uso de un objeto XmlRpcClient que permitirá el envío de peticiones al servidor. En el servidor se crea un objeto XmlRpcServer que recibe dichas peticiones o solicitudes, las ejecuta utilizando un objeto controlador y devuelve un String con la respuesta al cliente. Para ingresar las instrucciones correctamente, se deben utilizar los comandos especificados en la Tabla 1

Cabe destacar, que la ejecución de cada uno de los modos de operación del robot (modo vínculo, modo efector, etc) así también como la activación y desactivación del mismo, se ejecutan en el servidor; de forma tal que independientemente del tipo de control utilizado, se obtienen los mismos resultados en consola.

DESARROLLO

Con el fin de desarrollar una aplicación consecuente al problema planteado, se recurre a implementar un enfoque orientado a objetos, donde cada lado del sistema, es decir, cliente y servidor, se implementan con este mismo enfoque y con un protocolo de comunicación entre ellos.

1. Diagramas de Clases

Se modelan los diagramas de las figuras 3 y 4 aplicando los conceptos de UML (Lenguaje de Modelado Unificado, por sus siglas en inglés), el cual es un lenguaje visual que permite modelar procesos, sistemas y software, con el fin de obtener una representación visual de los mismos, utilizando diagramas y haciendo uso de los conceptos de clase, objeto y herencia, propios de la programación orientada a objetos.

1.1. Estructura Cliente

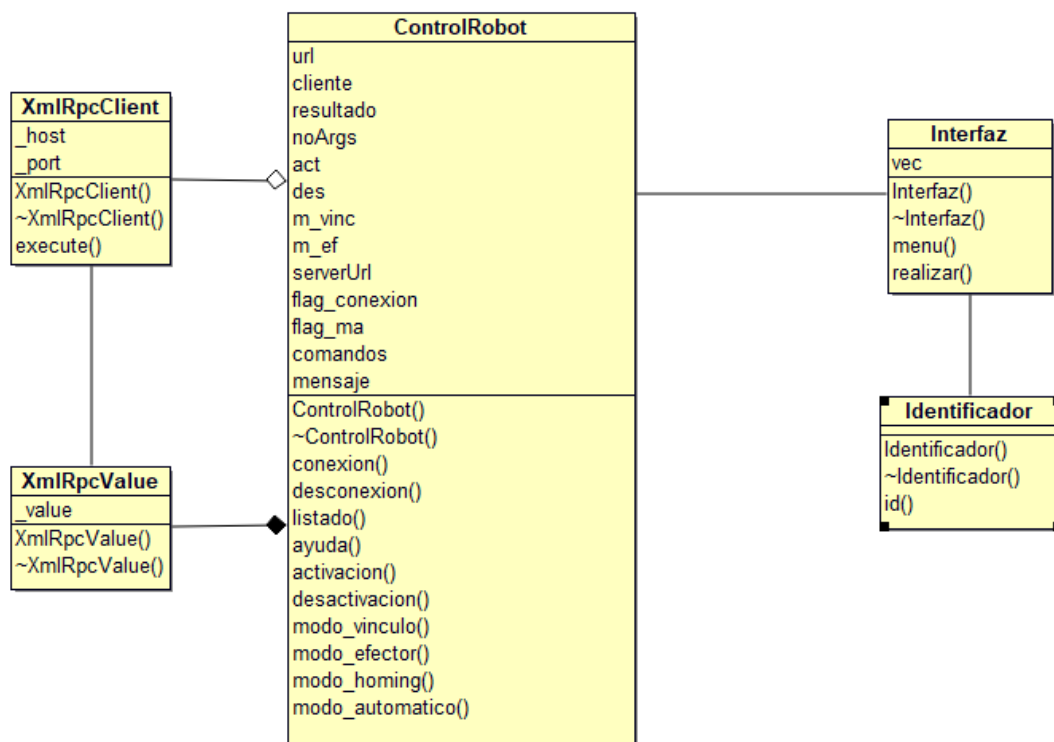


Figura 3: Diagrama de Clases del Cliente

- **ControlRobot**: objeto que se encarga de manejar las solicitudes de cada tarea al servidor.
 - `Controlrobot()`: constructor del objeto
 - `~Controlrobot()`: destructor del objeto
 - `conexion()`: establece conexión con el servidor
 - `desconexion()`: establece la desconexión con el servidor
 - `listado()`: solicita un informe con las instrucciones realizadas desde el inicio del servidor

- `ayuda()`: devuelve una guía de comandos y formas de uso del sistema
 - `activacion()`: solicita la activación del robot para iniciar el control
 - `desactivacion()`: solicita la desactivación del robot para finalizar el control
 - `modo_vinculo(q1, q2, q3, v1, v2, v3)`: solicita el movimiento de las articulaciones en función de las coordenadas articulares ingresadas
 - `modo_efector(x, y, α , t)`: solicita el movimiento del efector final en función de las coordenadas cartesianas ingresadas
 - `modo_homing()`: solicita el movimiento a la posición de inicial de cada articulación
 - `modo_automatico()`: solicita la ejecución automática de alguno o varios de los movimientos mencionados, en función de los comandos ingresados en un archivos previamente almacenado
- **Interfaz**: interfaz de tipo consola mediante la cual el usuario puede interactuar con el sistema
 - `Interfaz()`: constructor del objeto
 - `~Interfaz()`: destructor del objeto
 - `menu()`: imprime en consola un menú de inicio con instrucciones simples de uso
 - `realizar()`: realiza las llamadas a los métodos del controlador correspondientes dependiendo de los comandos ingresados por el usuario
 - **Identificador**: se encarga de identificar y acondicionar las instrucciones ingresadas por el usuario
 - `Identificador()`: constructor del objeto
 - `~Identificador()`: destructor del objeto
 - `id()`: realiza la identificación de los comandos ingresados para ejecutar las instrucciones solicitadas
 - **XmlRpcClient**: brinda el objeto necesario para establecer la comunicación como cliente del sistema, enviando las solicitudes y recibiendo las respuestas, bajo el protocolo XML-RPC
 - **XmlRpcValue**: establece los tipos de datos necesarios para que la clase `XmlRpcClient` funcione correctamente

1.2. Estructura Servidor

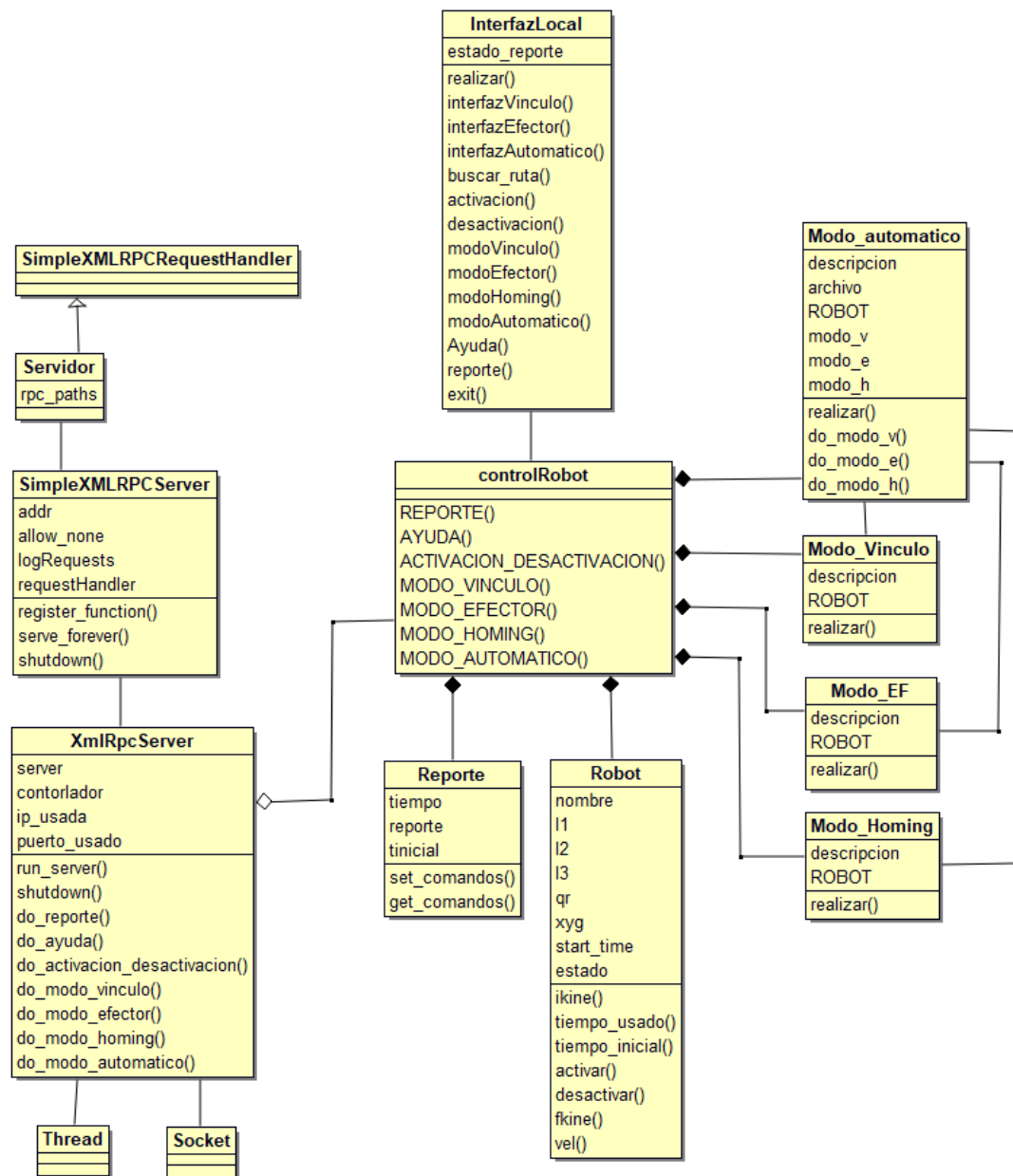


Figura 4: Diagrama de Clases del Servidor

- **controlRobot**: recibe las solicitudes y ejecuta las tareas
 - **REPORTE()**: genera informe con las instrucciones realizadas desde el inicio del servidor, para ser enviado cuando el usuario lo solicite
 - **AYUDA()**: imprime en consola una guía de uso y formatos para el en modo remoto
 - **ACTIVACION_DESACTIVACION()**: establece la activación o desactivación del robot para iniciar el control
 - **MODO_VINCULO($q_1, q_2, q_3, v_1, v_2, v_3$)**: realiza el movimiento de las articulaciones en función de las coordenadas articulares ingresadas, devuelve las coordenadas cartesianas del efector final

- `MODO_EFECTOR(x, y, α , t)`: realiza el movimiento del efector final en función de las coordenadas cartesianas ingresadas, devuelve las coordenadas articulares
- `MODO_HOMING()`: realiza el movimiento a la posición de inicial de cada articulación
- `MODO_AUTOMATICO()`: realiza automáticamente alguno o varios de los movimientos mencionados, en función de los comandos ingresados en un archivos previamente almacenado
- **InterfazLocal**: respresenta una interfaz visual gráfica en el modo local, para brindar una experiencia de uso más amigable, utilizando botones en lugar de comandos por consola
 - `realizar()`: ejecuta cada una de las ordenes solicitadas
 - `interfazVinculo()`: interfaz visual del modo vínculo para ingresar las coordenadas articulares
 - `conexion()`: establece conexión con el servidor
 - `interfazEfector()`: interfaz visual del modo efector para ingresar las coordenadas cartesianas
 - `interfazAutomatico()`: interfaz visual del modo automático para explorar la ubicación del archivo con los comandos a ejecutar
 - `ayuda()`: devuelve una guía de comandos y formas de uso del sistema
 - `activacion()`: establece la activación del robot para iniciar el control
 - `desactivacion()`: establece la desactivación del robot para finalizar el control
 - `modoVinculo(q1, q2, q3, v1, v2, v3)`: realiza el movimiento de las articulaciones en función de las coordenadas articulares ingresadas y devuelve las coordenadas cartesianas del efector final
 - `modoEfector(x, y, α , t)`: realiza el movimiento del efector final en función de las coordenadas cartesianas ingresadas y devuelve las coordenadas articulares
 - `modoHoming()`: realiza el movimiento a la posición de inicial de cada articulación
 - `modoAutomatico()`: ejecuta automáticamente alguno o varios de los movimientos mencionados, en función de los comandos ingresados en un archivos previamente almacenado
 - `Ayuda()`: inicializa una interfaz visual con la guía de uso de la interfaz gráfica principal del sistema
 - `reporte()`: entrega un informe con las instrucciones realizadas desde el inicio del servidor
 - `exit()`: realiza la desactivación del robot, cierra la interfaz visual y desconecta el sistema
- **Reporte**: almacena cada instrucción ejecutada, su respuesta, así también como el día y la hora de ejecución, en un informe para luego ser devuelto al usuario al solicitarlo.
- **Robot**: representa el modelo matemático cinemático del robot físico real de 3 grados de libertad. Cuenta con los cálculos necesarios para realizar la cinemática inversa y la directa del mismo.
- **Modo_Vinculo**: ejecuta el movimiento de las articulaciones
- **Modo_EF**: ejecuta el movimiento del efector final

- Modo_Homing: ejecuta el movimiento de las articulaciones hacia la posición de referencia o inicial
- Modo_Automatico: ejecuta automáticamente los movimientos solicitados desde un archivo
- XmlRpcServer: realiza la configuración necesaria para iniciar el servidor (haciendo uso de las clases 'Socket', 'Thread', 'Servidor', 'SimpleXMLRPCServer' y 'SimpleXMLRPCRequestHandler') y luego registrar en el mismo cada una de las funciones del robot, para establecer la comunicación satisfactoria con el cliente.

2. Diagramas de Secuencia

El diagrama de secuencia es un tipo de diagrama del lenguaje unificado de modelado(UML) que representa los eventos en orden cronológico y describe básicamente cómo los objetos intercambian mensajes en un orden determinado.

En las figuras 5 y 6 se pueden observar los diagramas de secuencia correspondientes a los dos tipos de control posibles del robot, local y remoto, respectivamente; en los cuales se ejecutan diferentes instrucciones, en cada uno de ellos, para ser lo más abarcativo y demostrativo posible de todas las instrucciones posibles.

2.1. Control Local

Se representa un caso de uso del control local, en el cual el usuario controla el robot desde el mismo servidor, utilizando la interfaz visual gráfica provisto por el mismo. En dicho caso, el usuario ejecuta en orden cronológico las siguientes instrucciones:

1. Activación del robot
2. Modo homing
3. Modo vínculo
4. Solicitud de reporte
5. Salir

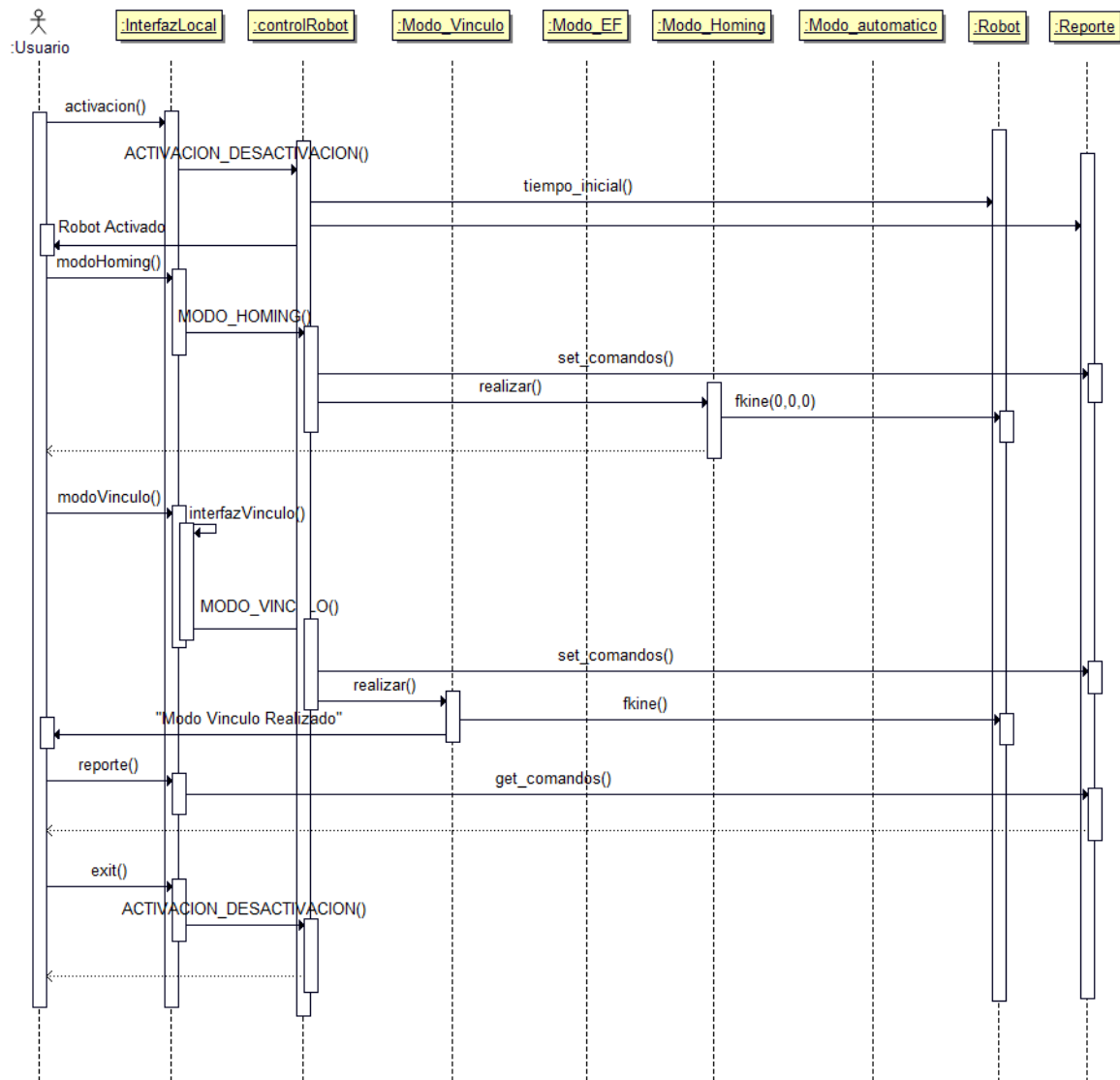


Figura 5: Diagrama de Secuencia Local

2.2. Control Remoto

En este caso, se representa un caso de uso del control remoto, en el cual el usuario controla el robot haciendo uso de un cliente conectado al servidor mediante protocolo XML-RPC, utilizando la consola del sistema en el cual se ejecuta dicho cliente e ingresando las instrucciones según el formato de comandos utilizado. El usuario ejecuta en orden cronológico las siguientes instrucciones:

1. Conexión con el servidor
2. Activación del robot
3. Modo efector final
4. Desconexión

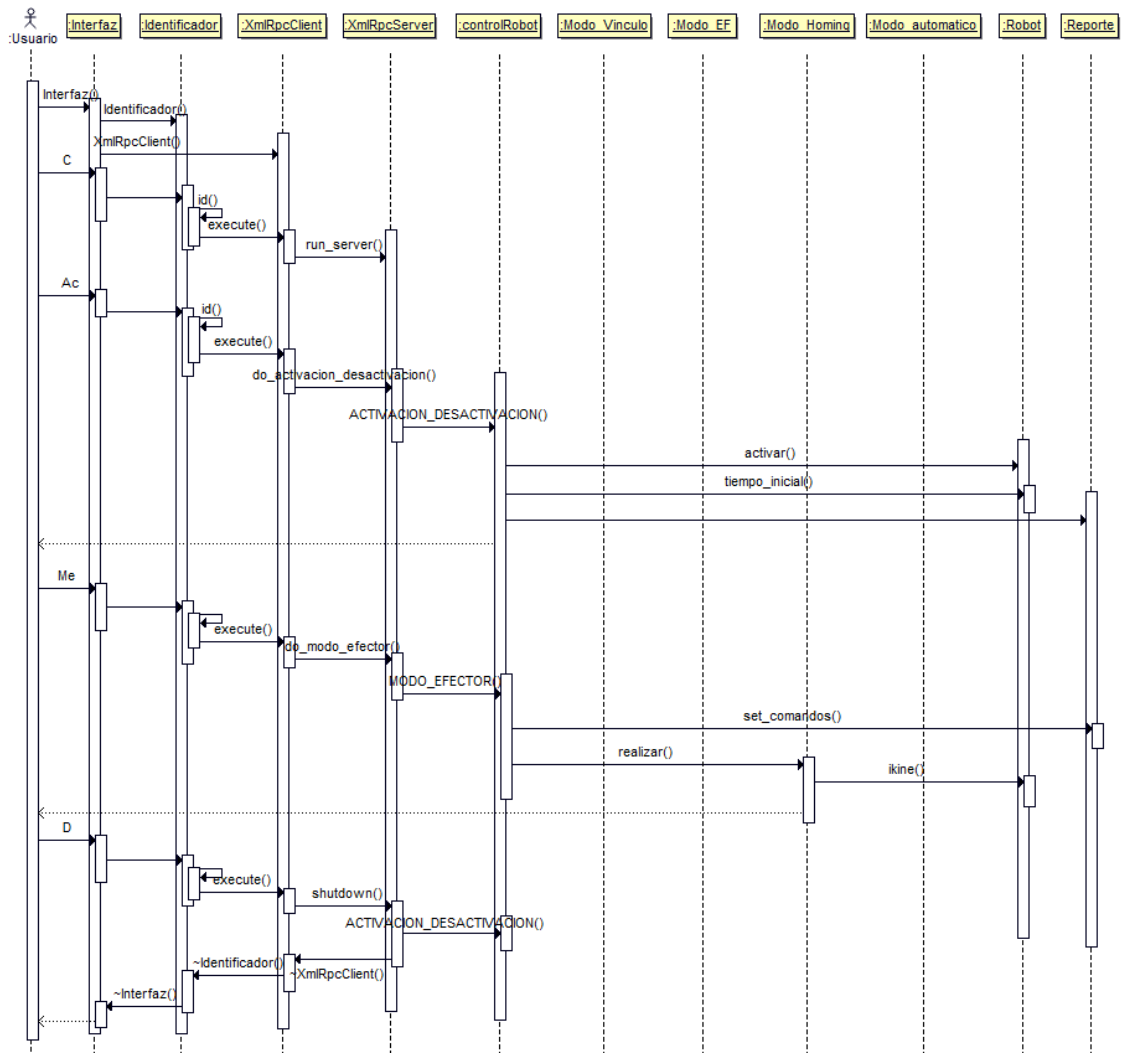


Figura 6: Diagrama de Secuencia Remoto

3. Diagrama de Actividad

Se realizó el diagrama de actividad del comportamiento general de la aplicación del cliente, ante el caso de la introducción de un comando, válido o inválido:

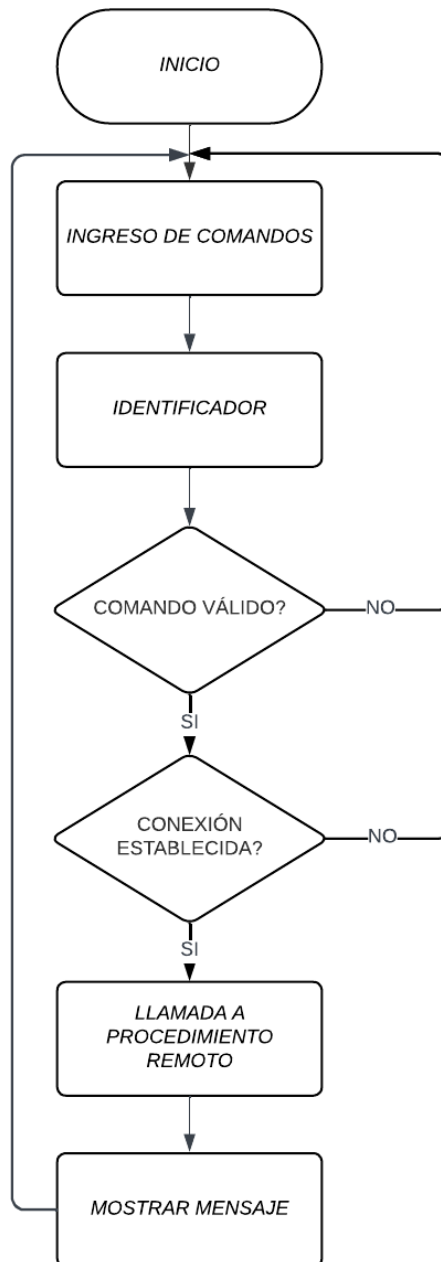


Figura 7: Diagrama General de Actividad del Sistema

EJEMPLOS DE USO

A continuación se presentan 2 casos de uso, uno para cada tipo de control, en los cuales se explica cada solicitud del usuario y se presentan las capturas de pantalla obtenidas en cada respuesta del sistema. Para mantener una consistencia y una relación en todo el presente informe, los ejemplos que se mostrarán a continuación serán los mismos que se mencionaron y diagramaron en las secciones 2.1 y 2.2, respectivamente.

Además, dentro del control local, se incluye un último caso de uso en el cual se presenta la ejecución del modo automático del sistema, para así demostrar, independientemente del tipo de control, todas las tareas posibles.

1. Ejemplo Control Local

Se lanza el servidor ejecutando el archivo 'main.py' y en el menú principal se ingresa 'local' como tipo de control:

```
maxx@Maxx-Dell:~/Documentos/P00/Proyecto Final 2023- v2/Servidor Py$ python3 main.py
Seleccionar tipo de control:
1. Local
2. Remoto
local

Control Local inicializado
```

Figura 8: Menú principal control Local

Se inicializa la ventana de la interfaz visual implementada para el control local, mediante la cual se pueden realizar todas las tareas posibles del robot:

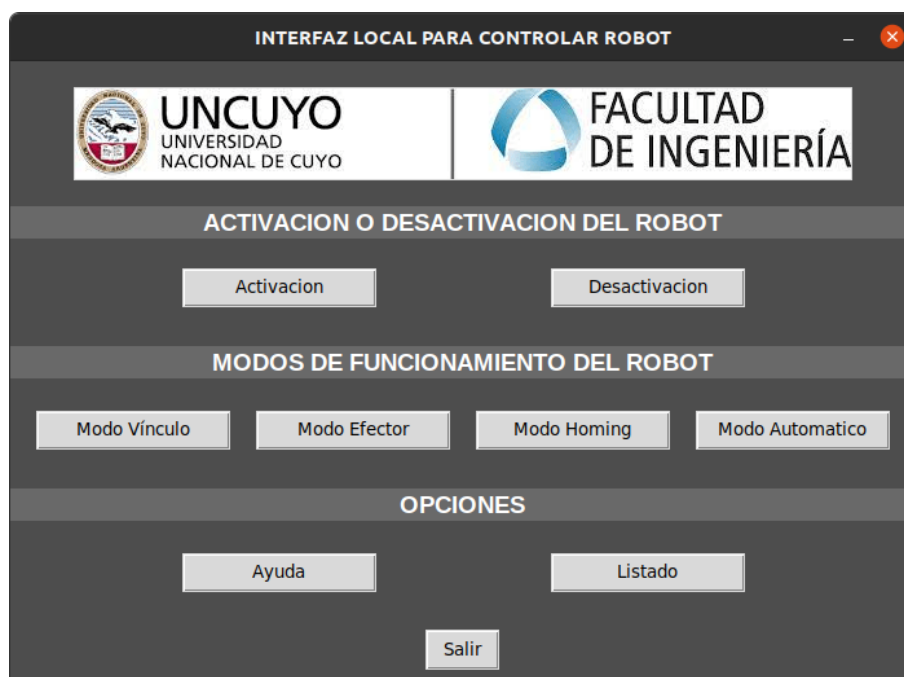


Figura 9: Interfaz GUI de control local

Como se puede observar, es una interfaz que brinda una interacción simple e intuitiva de uso. Primero se activa el robot para poder ejecutar el 'Modo Homing', obteniendo las respuestas en consola:

```
>>> Robot activado <<<

>>> Homing realizado <<<
```

Figura 10: Mensaje en consola de Robot Activado y Modo Homing

Luego, tal como se diagramó en la secuencia 2.1 del control local, se presiona en 'Modo Vínculo' y se abre la interfaz de dicho modo, en la cual se ingresa las coordenadas $q_1 = 3$, $q_2 = 1$, $q_3 = 2$, $v_1 = 2$, $v_2 = 1$ y $v_3 = 1$ para ejecutarlo correctamente:

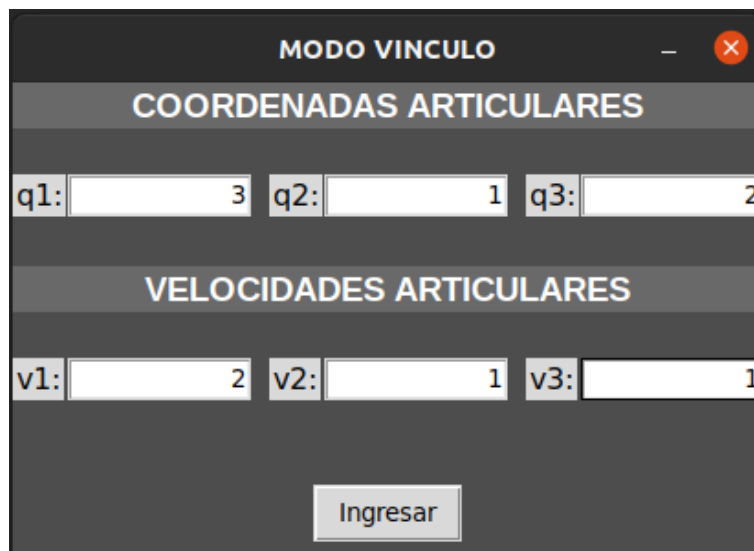


Figura 11: Interfaz Modo Vínculo

Al finalizar la ejecución del 'Modo Vínculo', se obtiene el siguiente mensaje en consola, en el cual se puede observar las coordenadas cartesianas obtenidas del efector final:

```
>>> Modo vinculo realizado:
>>> La posición del efector final es:
>>> ['-0.68', '-0.90', '6.00'] <<<
```

Figura 12: Mensaje en consola de Modo Vínculo

Continuando con la secuencia, en la interfaz principal de la figura 9, se clickea en el botón

'Listado' para solicitar un reporte de las tareas ejecutadas desde el inicio del servidor, junto con su fecha y hora de ejecución, como así también el tiempo de duración del servidor en segundos:

```
Reporte:
Fri Apr 7 11:38:16 2023 Inicialización del servidor
Fri Apr 7 11:46:23 2023 Activación
Fri Apr 7 11:47:51 2023 Modo homing
Fri Apr 7 11:48:48 2023 Modo vinculo q1=3 q2=1 q3=2 v1=2 v2=1 v3=1
Duración del servidor: 647.64 segundos
```

Figura 13: Mensaje en consola del Listado de tareas

Por último, se desactiva el robot y se cierra el sistema haciendo click en 'Salir':

```
>>> Robot desactivado <<<

Gracias por utiizar la conexion local, cerrando...
```

Figura 14: Mensaje en consola al Salir

Modo automático

Se agrega un ejemplo de uso en el control local del modo automático, para demostrar el correcto funcionamiento del mismo desde la interfaz visual implementada y así comprobar todas las tareas posibles.

Partiendo de la interfaz principal de la figura 9, se clickea en 'Modo Automático' y se abre la siguiente ventana:



Figura 15: Interfaz Modo Automático

Clickeando en 'Explorar' se abre la siguiente ventana en la cual se puede buscar y seleccionar el archivo que se desea ejecutar automáticamente. Se selecciona el archivo 'modo_auto_1.txt' y se clickea en 'Abrir':

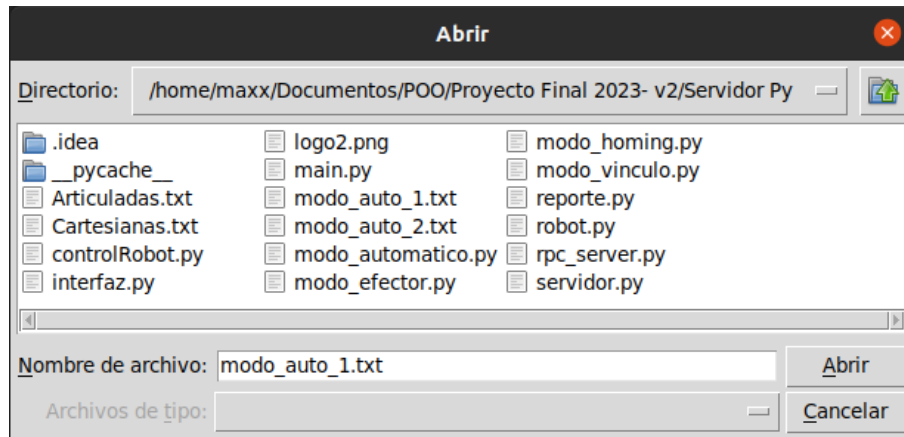


Figura 16: Interfaz explorar ruta

En la interfaz del 'Modo Automático', se observa la ruta del archivo seleccionado anteriormente para ejecutar. Al hacer click en 'Abrir' nuevamente, se ejecuta dicho modo con las instrucciones correspondientes, las cuales se pueden observar en la figura 18:



Figura 17: Interfaz Modo Automático con ruta de archivo

```
1 Mh
2 Mv101010
3 Me1111
4 Mh
```

Figura 18: Instrucciones archivo 'modo_auto_1.txt'

Al finalizar la lectura y ejecución de las instrucciones del archivo, se obtiene el siguiente mensaje en la consola, con la respuesta correspondiente a cada uno de los modos ejecutados:

```
>>> Se realizarán las instrucciones del archivo: modo_auto_1.txt

>>> Homing realizado <<<

>>> Modo vinculo realizado:
>>> La posición del efector final es:
>>> ['-0.87', '1.89', '3.00'] <<<

>>> Modo efector realizado:
>>> Las posiciones articulares son:
>>> ['-0.99', '2.65', '-0.66'] <<<

>>> Homing realizado <<<
```

Figura 19: Mensaje en consola de Modo Automático 1

2. Ejemplo Control Remoto

El control remoto permite la solicitud de las instrucciones desde el cliente, por lo tanto, primero es necesario lanzar el servidor en dicho tipo de control e ingresar IP y puerto a utilizar para la comunicación:

```
Seleccionar tipo de control:
1. Local
2. Remoto
remoto

Control Remoto inicializado

Ingresar dirección IP:
192.168.54.191

Ingresar puerto:
8890

Servidor RPC iniciado en el puerto [('192.168.54.191', 8890)]
```

Figura 20: Menú principal control remoto en servidor

Luego, se procede a compilar y ejecutar el cliente con la misma IP y puerto utilizados en el servidor:

```
maxx@Maxx-Dell:~/Documentos/P00/Proyecto Final 2023- v2/Cliene C++$ g++ -w *.cpp -o Cliente
maxx@Maxx-Dell:~/Documentos/P00/Proyecto Final 2023- v2/Cliene C++$ ./Cliente 192.168.54.191 8890
```

Figura 21: Compilación y ejecución del Cliente

Una vez inicializado el cliente, se visualiza el siguiente menú inicial con 3 instrucciones simples para continuar. Tal como se diagramó en la secuencia 2.2 del control remoto, se procede a efectuar la conexión con el servidor, ingresando nuevamente la IP, y se obtiene la respuesta de conexión satisfactoria:

```
-----
---          CONTROL REMOTO DE ROBOT          ---
-----
Introducir los comandos para cada instrucción:
Conexión al servidor: C
Menu de ayuda: A
Cerrar el programa: exit
>>> C
~~~~~
Introducir IP nuevamente:
192.168.54.191

Conectado al servidor satisfactoriamente!
```

Figura 22: Solicitud y respuesta de Conexión con servidor

Se continua ingresado el comando 'A' para solicitar el menú de ayuda, en el cual se observan los comandos y formato necesarios para ingresar las instrucciones:

```

-----
---          CONTROL REMOTO DE ROBOT          ---
-----

Introducir los comandos para cada instrucción:
Conexión al servidor: C
Menu de ayuda: A
Cerrar el programa: exit
>>> A

~~~~~
Los comandos para cada instrucción son los siguientes:

CONEXION:      C
DESCONEXION:   D
LISTADO:       L
AYUDA:         A

ACTIVACION:    Ac
DESACTIVACION: De

MODO VINCULO:  Mv
MODO EFECTOR:  Me
MODO HOMING:   Mh
MODO ARCHIVO:  Ma

Formato Mv: MVq1,q2,q3,v1,v2,v3
Formato Me: MEx,y,alfa,t
Formato Ma: MAi

Al ingresar múltiples instrucciones, separar mensajes por '-'

```

Figura 23: Solicitud y respuesta del menú de Ayuda

Se procede a realizar la activación del robot, mediante el comando 'Ac':

```

-----
---          CONTROL REMOTO DE ROBOT          ---
-----

Introducir los comandos para cada instrucción:
Conexión al servidor: C
Menu de ayuda: A
Cerrar el programa: exit
>>> Ac

~~~~~

>>> Robot activado <<<

```

Figura 24: Solicitud y respuesta de Activación del robot

Se solicita la ejecución del 'Modo Efector' según el formato correspondiente, ingresando las coordenadas $x = 1$, $y = 2$, $\alpha = 1$, $t = 2$, y se obtiene la respuesta de la ejecución de dicho modo con las coordenadas articulares obtenidas de cada eslabón:

```
-----  
---          CONTROL REMOTO DE ROBOT          ---  
-----  
Introducir los comandos para cada instrucción:  
Conexión al servidor: C  
Menu de ayuda: A  
Cerrar el programa: exit  
>>> Me1,2,1,2  
~~~~~  
>>> Modo efector realizado:  
>>> Las posiciones articulares son:  
>>> ['2.09', '-1.80', '0.70'] <<<
```

Figura 25: Solicitud y respuesta de Modo Efector

Por último, se ingresa el comando 'exit' para efectuar la desconexión del servidor y cierre del programa:

```
-----  
---          CONTROL REMOTO DE ROBOT          ---  
-----  
Introducir los comandos para cada instrucción:  
Conexión al servidor: C  
Menu de ayuda: A  
Cerrar el programa: exit  
>>> exit  
~~~~~  
El programa se cerrará...
```

Figura 26: Solicitud y respuesta de Exit

Ejemplo con instrucciones múltiples y modo automático

Con el fin de completar la demostración de uso del sistema, se demuestra su uso ingresando múltiples instrucciones de forma que las ejecuta de forma consecutiva, donde una de ellas solicita la ejecución del 'Modo Automático' del archivo 'modo_auto_2.txt':

Las instrucciones solicitadas son: 'Modo Homing', 'Modo Vínculo' y la lectura automática del archivo previamente mencionado. En la figura siguiente se pueden observar los comandos 'Mh-Mv1,3,1,3,1,3-Ma2' ingresados por consola, seguidos del mensaje de respuesta obtenido de la ejecución de cada una de las tareas solicitadas:

```
-----
---          CONTROL REMOTO DE ROBOT          ---
-----
Introducir los comandos para cada instrucción:
Conexión al servidor: C
Menu de ayuda: A
Cerrar el programa: exit
>>> Mh-Mv1,3,1,3,1,3-Ma2
~~~~~

>>> Homing realizado <<<

>>> Modo vinculo realizado:
>>> La posición del efector final es:
>>> ['0.17', '-0.87', '5.00'] <<<

>>> Se realizarán las instrucciones del archivo: modo_auto_2.txt

>>> Homing realizado <<<

>>> Modo vinculo realizado:
>>> La posición del efector final es:
>>> ['-0.87', '1.89', '3.00'] <<<

>>> Modo vinculo realizado:
>>> La posición del efector final es:
>>> ['-0.53', '0.99', '4.00'] <<<

>>> Modo efector realizado:
>>> Las posiciones articulares son:
>>> ['0.30', '0.74', '-0.04'] <<<

>>> Homing realizado <<<

>>> Modo efector realizado:
>>> Las posiciones articulares son:
>>> ['2.09', '-1.80', '0.70'] <<<
```

Figura 27: Solicitud y respuesta de múltiples instrucciones

Luego, se ingresa el comando 'L' para solicitar el reporte de las instrucciones solicitadas, fecha y hora de ejecución de cada una:

```
-----  
---          CONTROL REMOTO DE ROBOT          ---  
-----  
Introducir los comandos para cada instrucción:  
Conexión al servidor: C  
Menu de ayuda: A  
Cerrar el programa: exit  
>>> L  
~~~~~  
Reporte:  
Fri Apr 7 12:05:03 2023 Inicialización del servidor  
Fri Apr 7 12:05:32 2023 Activación  
Fri Apr 7 12:05:36 2023 Modo homing  
Fri Apr 7 12:05:36 2023 Modo vinculo q1=1.0 q2=3.0 q3=1.0 v1=3.0 v2=1.0 v3=3.0  
Fri Apr 7 12:05:36 2023 Modo automatico desde archivo: modo_auto_2.txt  
Duración del servidor: 39.16 segundos
```

Figura 28: Solicitud y respuesta de Listado

Por último, se desactiva el robot y luego se cierra la el sistema de la misma manera que en la figura 26:

```
-----  
---          CONTROL REMOTO DE ROBOT          ---  
-----  
Introducir los comandos para cada instrucción:  
Conexión al servidor: C  
Menu de ayuda: A  
Cerrar el programa: exit  
>>> de  
~~~~~  
>>> Robot desactivado <<<
```

Figura 29: Solicitud y respuesta de Desactivación del robot

CONCLUSIONES

En el presente proyecto se lograron aplicar gran cantidad de los temas estudiados y practicados en la cátedra Programación Orientada a Objetos. Se implementaron conceptos de modulación y encapsulamiento, propios del paradigma de POO, en el desarrollo de las clases tanto del lado cliente como del lado servidor; en los cuales también se integraron con el uso de las librerías necesarias para la comunicación remota entre ambos mediante el protocolo XML-RPC. Por otro lado, se logró aplicar los conocimientos estudiados sobre el lenguaje UML para realizar los diagramas de clases y de secuencia solicitados en el desarrollo del informe, y mediante los cuales se simplifica la comprensión del maquetado y modelado de la aplicación completa.

Se investigó y aprendió sobre el uso de la plataforma Linux/Ubuntu, la cual ofrece posibles soluciones a inconvenientes y un uso mas simple y amigable; así como también la consola de la misma mediante la cual se realizaron todas las pruebas y ejecuciones durante el desarrollo de cada uno de los aplicativos, utilizando los comandos correspondientes. Además, se investigó e implementó el uso de la librería 'Tkinter' con Python, para la confección de la interfaz visual utilizada en el servidor para el control Local del robot, la cual es una librería muy intuitiva, fácil de utilizar y aprender, ya que hay mucha documentación en la web y una gran comunidad dispuesta a colaborar. También se logró la implementación del 'Modo automático' mediante la manipulación de archivos previamente creados y almacenados, permitiendo la utilización de forma automática, evitando la limitación a un control solamente manual. Para finalizar, se solidificaron y afianzaron los conceptos y las reglas de programación en C++, lenguaje muy popular y utilizado pero a su vez complejo de utilizar, y que también cuenta con una gran comunidad de usuarios activos dispuestos a colaborar y brindar información sobre el mismo.

Un análisis general conlleva a concluir que se cumplieron satisfactoriamente con las solicitudes del proyecto, logrando una comunicación estable y fiable entre cliente y servidor, de forma que se obtiene el control deseado del robot en ambas modalidades, local y remoto. Se plantean como posibles mejoras la posibilidad de agregar una interfaz visual gráfica en el lado del cliente, ya que es el usuario el que debería tener la facilidad de control y manipulación del sistema, debido a que es posible que sea un operario sin grandes conocimientos respecto al uso de consolas, diferentes sistemas operativos y robots.

BIBLIOGRAFÍA

- Aula abierta cátedra Programación Orientada a Objetos
- XML-RPC for C and C++
- XML-RPC++
- Compilar C++ en G++ Linux en terminal
- Servidores XML-RPC básicos
- Curso Python. Interfaces gráficas
- Interfaces gráficas de usuario con Tkinter
- BoUML