



UNCUYO  
UNIVERSIDAD  
NACIONAL DE CUYO



FACULTAD  
DE INGENIERÍA

# Programación Orientada a Objetos

## Trabajo Práctico Integrador:

*Ingeniería Mecatrónica - Ciclo lectivo 2023*

Profesor: Mag. Ing. César Omar Aranda

Alumnos:

Borquez Juan Manuel		Legajo: 13567
Dalessandro Francisco		Legajo: 13318
Miranda Francisco		Legajo: 13250

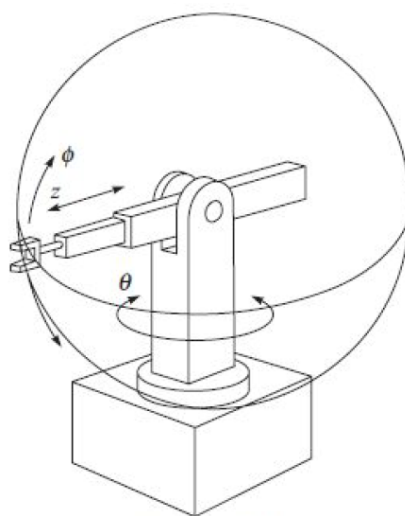
## Contenido

<u>I. Introducción y enunciado</u>	<u>3</u>
<u>II. Conceptos, fundamentos y esquemas gráficos de la solución:</u>	<u>4</u>
<u>Diagrama de clases - Servidor</u>	<u>4</u>
<u>Diagrama de clases - Cliente</u>	<u>6</u>
<u>III. Descripción de componentes software (librerías, frameworks, aplicaciones, etc.)</u>	<u>7</u>
<u>IV. Descripción paso a paso del modo de instalación y uso</u>	<u>10</u>
<u>Instalación y uso del servidor:</u>	<u>10</u>
<u>Modo de uso en la interfaz por línea de comando del servidor:</u>	<u>10</u>
<u>Instalación y uso del cliente:</u>	<u>12</u>
<u>Ejecutar cliente en la terminal:</u>	<u>12</u>
<u>Ejecutar cliente en la interfaz gráfica:</u>	<u>14</u>
<u>Modo de uso en la interfaz gráfica:</u>	<u>17</u>
<u>V. Capturas de pantalla mostrando la ejecución del aplicativo, para cada una de las interfaces de entrada/salida propuestas para interacción con el usuario.</u>	<u>18</u>
<u>VI. Conclusiones:</u>	<u>19</u>

## I. Introducción y enunciado

Este informe técnico documenta el desarrollo de un trabajo práctico universitario de carácter integrador enfocado en el paradigma de la Programación Orientada a Objetos, aplicado en los lenguajes Python, C++ y Java.

El objetivo es desarrollar una aplicación para controlar un robot "3DF" con una configuración RRR, lo que significa que tiene tres grados de libertad (RRR se refiere a los ejes de rotación alrededor de los tres ejes ortogonales). El controlador del robot opera con comandos en formato G-Code, un lenguaje de programación comúnmente utilizado para el control de máquinas CNC (Control Numérico por Computadora) y devuelve mensajes que proporcionan información sobre el estado de la operación.



El desarrollo de este proyecto se llevó a cabo con el uso de un firmware adaptado utilizando tecnología Arduino, debido a la gran dificultad económica para poder llevar a cabo unas prácticas con un robot real.

El proyecto de desarrollo se dividió en dos partes principales: el lado del servidor y el lado del cliente. Los módulos del lado del servidor están desarrollados en Python, mientras que los módulos del lado del cliente están desarrollados en C++ y Java.

El servidor tiene la responsabilidad de mantener un archivo con el registro del trabajo realizado y ofrece servicios RPC que permiten realizar diversas acciones como la conexión/desconexión al Robot, la activación/desactivación de motores del robot, y listar los comandos disponibles de control, etc. El servidor también ofrece un panel de control del robot mediante una consola tipo CLI y realiza streaming de video del funcionamiento del robot.

Por otro lado, el lado del cliente se encarga de ofrecer una GUI de control y monitoreo de parámetros con efectos de sonido, generación de alarmas auditivas. Se puede acceder al streaming de video de operación del robot a través de un buscador. Este trabajo práctico representa una excelente oportunidad para profundizar en la programación orientada a objetos y explorar la interacción entre el servidor y el cliente con la implementación de diversas funcionalidades en ambos lados.

## II. Conceptos, fundamentos y esquemas gráficos de la solución:

### Diagrama de clases - Servidor

Las clases son elementos clave en la programación orientada a objetos y representan la matriz de futuros objetos. Por lo tanto, para llevar a cabo la explicación inicial del proyecto se procederá a presentar los diagramas de clases, donde podemos obtener una idea general de la implementación de las clases, sus atributos, sus métodos y las relaciones entre ellas, sin necesidad de acceder al código.

A continuación se presentan los diagramas detallados de las clases realizadas en BoUML.

#### Primera Parte - Servidor:

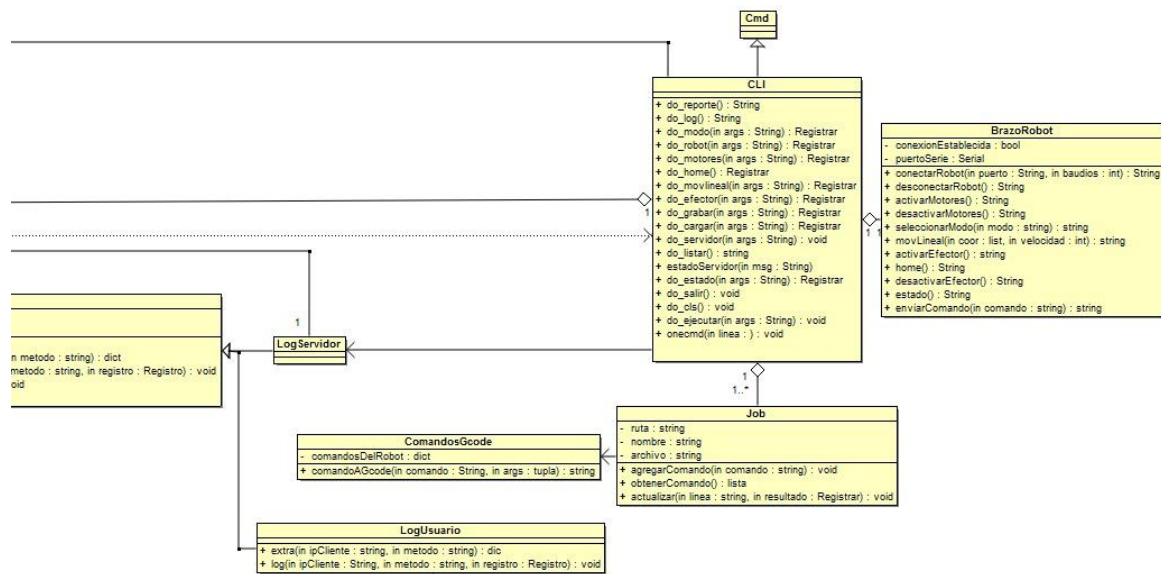


Figura 1: Diagrama de las clases utilizadas en la parte del servidor

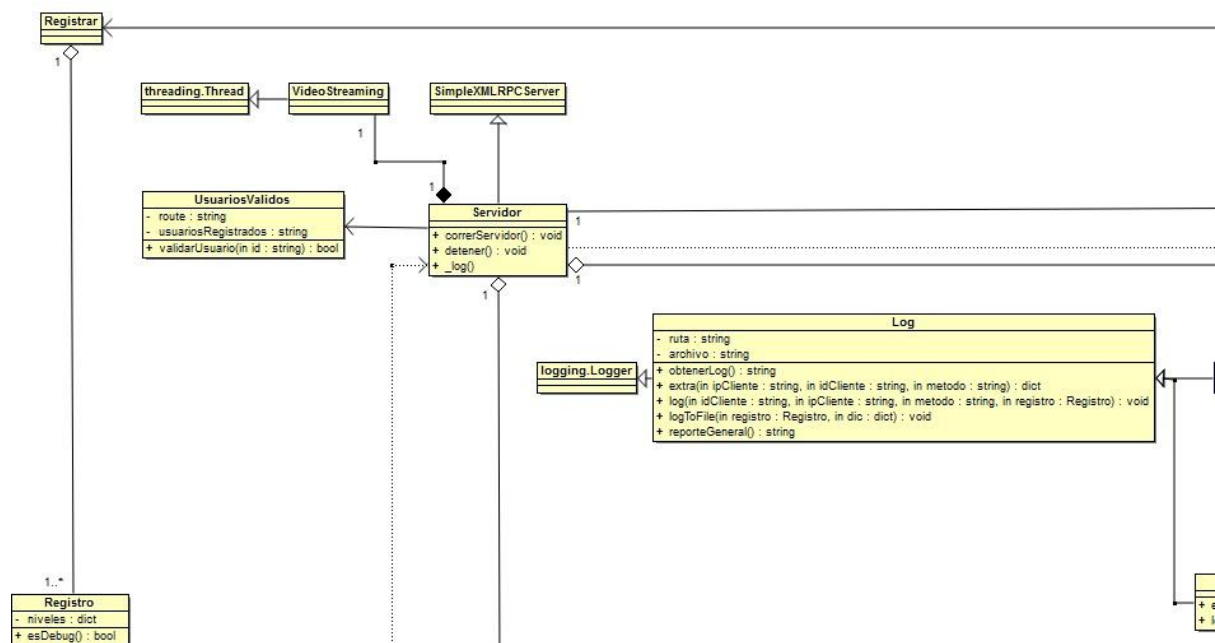


Figura 2: Diagrama de las clases utilizadas en la parte del servidor

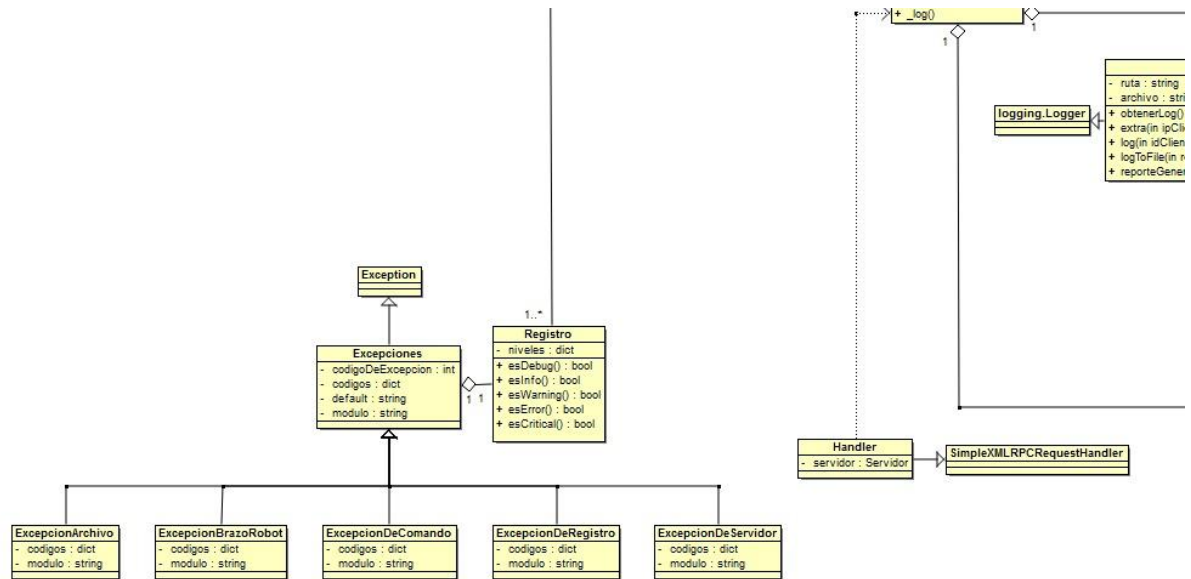


Figura 3: Diagrama de las clases utilizadas en la parte del servidor

Descripción de clases en el servidor:

**CLI (Command Line Interface):** Esta clase es una clase herencia de la clase Cmd y representa la interfaz de línea de comandos que permite a los usuarios interactuar con el brazo robot y realizar diversas operaciones de manera local. Los comandos están definidos como métodos dentro de esta clase y permiten que el administrador pueda interactuar y controlar el brazo robot y el servidor, además de poder administrar y trabajar con diversos archivos informativos.

**BrazoRobot:** Es una clase que representa el brazo robot. Se utiliza para realizar operaciones como conectar y desconectar el robot, activar y desactivar motores, mover el efector final, activar y desactivar la pinza, entre otras acciones.

**Log:** Una clase que maneja registros y logs relacionados con el funcionamiento del servidor.

**Job:** Esta clase se utiliza para cargar, almacenar y gestionar archivos de trabajo que contienen secuencias de comandos en código G para el brazo robot.

**Servidor:** Esta clase representa al servidor RPC (Remote Procedure Call). Permite iniciar y apagar el servidor para la comunicación remota con el brazo robot.

**Excepciones:** Representa un conjunto de clases de excepciones personalizadas definidas en el código. Cada excepción tiene un propósito específico para manejar errores o situaciones excepcionales.

**Registro:** Esta clase se utiliza para registrar los resultados de las operaciones en un formato sencillo de incorporar en un archivo de log de servidor y sigue el formato utilizado por el firmware de Arduino para las respuestas a los comandos.

**Comandos:** Una clase que proporciona métodos para transformar comandos a un formato GCode que el brazo robot pueda entender y ejecutar.

**Streaming:** utilizado para levantar el servicio de streaming.

UsuariosValidos: es la que se encarga de validar las peticiones de acuerdo a los usuarios válidos dentro del archivo “usuariosRegistrados.txt”.

## Diagrama de clases - Cliente

Segunda Parte - Cliente:agg

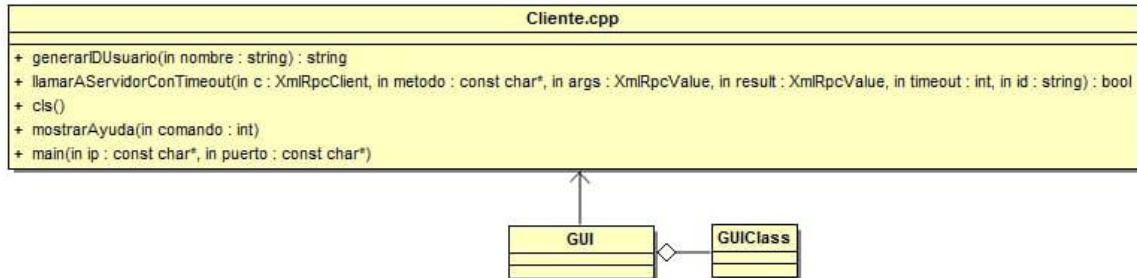


Figura 4: Diagrama de las clases utilizadas en la parte del cliente

El cliente utiliza la biblioteca XmlRpc para realizar llamadas al servidor XML-RPC y requiere un servidor XML-RPC funcional para interactuar con él.

Descripción de los métodos en el cliente:

**generarIDUsuario:** Este método toma el nombre del usuario como entrada y genera un ID de usuario a partir de él, eliminando espacios y convirtiéndolo a minúsculas.

**llamarAlServidorConTimeout:** Este método se utiliza para llamar al servidor con un límite de tiempo (timeout). Toma como entrada el cliente XmlRpc, el nombre del método, los argumentos, el resultado, el tiempo límite y el ID de usuario. Ejecuta el método en un hilo separado y lo cancela si supera el tiempo límite.

**cls():** Este método se utiliza para limpiar la consola. Utiliza funciones específicas para Windows y sistemas Unix/Linux (incluyendo macOS) para limpiar la pantalla de la consola.

**mostrarAyuda():** Muestra información de ayuda para un comando específico. Recibe un entero que corresponde al valor del comando y muestra la descripción del comando.

**signalHandler():** Este método se ejecuta cuando se recibe la señal SIGINT (Ctrl + C), siendo la interrupción por teclado, luego desconecta el cliente, imprime un mensaje y cierra el programa.

Cabe destacar que en el cliente se utiliza un mapeo (map) llamado **comandoANumero** para vincular los nombres de los comandos con valores numéricos. Este mapeo se utiliza para facilitar la identificación y ejecución de comandos específicos. La estructura de datos **std::map** es una colección que almacena pares clave-valor, similar a un diccionario en python.

Esto permite que el programa ejecute el código correspondiente al comando identificado, lo que hace que el programa sea modular y fácil de mantener, por lo que agregar y quitar métodos resulta muy sencillo.

### III. Descripción de componentes software (librerías, frameworks, aplicaciones, etc.)

Librerías utilizadas en el servidor:

- Librería cmd:

La librería cmd es un módulo de Python que proporciona una infraestructura para la creación de interfaces de línea de comandos interactivas. Permite a los desarrolladores crear aplicaciones de consola con un enfoque en comandos y respuestas interactivas. Esta librería incluye la clase Cmd, que sirve como base para crear intérpretes de comandos personalizados. Los programas que utilizan cmd pueden ofrecer un conjunto de comandos que los usuarios pueden ejecutar, y el módulo se encarga de gestionar la entrada, el análisis y la ejecución de comandos.

- Librería os:

La librería “os” es un módulo incorporado en Python que proporciona una interfaz para interactuar con el sistema operativo subyacente. Permite a los desarrolladores acceder y manipular funcionalidades relacionadas con el sistema, como el sistema de archivos, variables de entorno, rutas de directorios y otras operaciones del sistema. Con “os”, los programadores pueden crear aplicaciones que sean independientes del sistema operativo subyacente y realizar tareas como crear, eliminar, mover y administrar archivos y directorios.

- Librería subprocess:

La librería subprocess es un módulo en Python que permite a los desarrolladores iniciar y controlar procesos secundarios desde un programa Python. Proporciona una forma poderosa de interactuar con otros programas y ejecutar comandos del sistema operativo en un proceso secundario. Esto es útil para tareas como llamar a programas externos, redirigir la entrada/salida estándar y gestionar procesos de manera asíncrona. Subprocess facilita la automatización de tareas y la integración con programas externos.

- Librería platform:

La librería platform es un módulo en Python que permite a los desarrolladores acceder a información sobre la plataforma en la que se está ejecutando el código. Proporciona detalles sobre el sistema operativo, la versión de Python, la arquitectura del sistema y otra información relevante de la plataforma. Esto es útil para crear programas que se comporten de manera diferente en función de la plataforma en la que se ejecuten, o para recopilar información del sistema para fines de diagnóstico.

- Librería serial:

La librería serial es un módulo en Python que brinda soporte para la comunicación serie. Permite a los desarrolladores establecer conexiones serie con dispositivos periféricos, como microcontroladores, sensores y otros equipos electrónicos, a través de puertos seriales. Esta librería facilita la transferencia de datos bidireccionales entre la computadora y los dispositivos conectados. Ofrece funcionalidades para configurar la velocidad de transmisión, el control de flujo y la transmisión y recepción de datos a través de puertos serie, lo que es fundamental en aplicaciones que involucran la interacción con hardware externo.

- Librería time:

La librería `time` es un módulo en Python que proporciona funcionalidades relacionadas con la gestión del tiempo y la temporización. Permite a los desarrolladores medir el tiempo transcurrido, introducir retrasos, generar marcas de tiempo y manipular fechas y horas. La librería `time` es esencial para una variedad de aplicaciones que requieren sincronización, control de tiempo y programación basada en intervalos, como aplicaciones de control, registros de eventos y automatización de tareas

- Librería `threading`:

El módulo `threading` en Python proporciona un enfoque basado en subprocesos para la programación concurrente. Permite la ejecución de múltiples hilos de control simultáneamente y facilita la escritura de aplicaciones que pueden realizar múltiples tareas de manera eficiente. Los hilos son unidades de ejecución más pequeñas que los procesos y se utilizan para realizar tareas en paralelo, lo que es especialmente útil en aplicaciones que requieren la gestión de múltiples tareas concurrentes.

- Librería `flask`:

Flask es un marco de desarrollo web ligero para Python. Facilita la creación de aplicaciones web mediante la definición de rutas y vistas para manejar las solicitudes HTTP. Flask sigue el paradigma de arquitectura Modelo-Vista-Controlador (MVC) y es ampliamente utilizado en el desarrollo web. Permite crear aplicaciones web simples y complejas, proporcionando herramientas para gestionar solicitudes y respuestas HTTP, manejo de plantillas, sesiones y más.

- Librería `cv2` (OpenCV-Python):

OpenCV-Python, representada por el módulo `cv2`, es una librería de código abierto ampliamente utilizada para el procesamiento de imágenes y visión por computadora. Proporciona una amplia gama de funciones y algoritmos para el procesamiento de imágenes, incluido el procesamiento de video, la detección de objetos, la segmentación y la manipulación de imágenes. Es una herramienta esencial en aplicaciones que involucran el análisis de imágenes y video, como visión artificial, reconocimiento de patrones y aplicaciones de procesamiento de imágenes.

- Librería `logging`:

La librería `logging` en Python permite la generación de registros y seguimiento de eventos en una aplicación. Facilita la creación de registros detallados que ayudan en la depuración y el monitoreo de aplicaciones. Proporciona una forma estructurada de registrar eventos, lo que es crucial en aplicaciones de producción para identificar y solucionar problemas.

- Librería `socket`:

La librería `socket` en Python es una implementación de sockets de red que permite la comunicación a nivel de red. Permite a las aplicaciones crear conexiones de red, enviar y recibir datos a través de protocolos de red, como TCP y UDP. Esta librería es fundamental en aplicaciones que implican la comunicación de datos a través de una red, como servidores, clientes y aplicaciones de red en general.

Librerías utilizadas en el cliente:

- Librería `<iostream>`:



La librería `<iostream>` proporciona funcionalidades para entrada y salida estándar en C++. Permite leer datos desde el teclado y escribir datos en la pantalla o en un archivo. Incluye objetos como `cin` (para entrada estándar) y `cout` (para salida estándar) que son ampliamente utilizados para la interacción con el usuario.

- Librería `<stdlib.h>`:

La librería `<stdlib.h>` proporciona funciones y macros para realizar operaciones relacionadas con la gestión de memoria y control de procesos en C++. Incluye funciones para la asignación de memoria dinámica, conversión de cadenas a números, generación de números aleatorios y otras operaciones esenciales.

- Librería `<fstream>`:

La librería `<fstream>` se utiliza para operaciones de entrada y salida de archivos en C++. Permite abrir, leer y escribir archivos en disco. Proporciona clases como `ifstream` y `ofstream` que facilitan la lectura y escritura de datos en archivos.

- Librería `<algorithm>`:

La librería `<algorithm>` contiene una colección de funciones y algoritmos para manipular secuencias y colecciones de datos en C++. Incluye funciones para buscar elementos, ordenar, realizar transformaciones y más. Es una parte fundamental de la programación con contenedores en C++.

- Librería `<cctype>`:

La librería `<cctype>` ofrece funciones para el procesamiento de caracteres y cadenas de caracteres en C++. Incluye funciones como `isdigit` (para verificar si un carácter es un dígito) y `isspace` (para verificar si un carácter es un espacio en blanco).

- Librería `<iomanip>`:

La librería `<iomanip>` se utiliza para controlar el formato de salida en C++. Proporciona funciones y manipuladores que permiten establecer la precisión, el ancho y otros aspectos de la salida formateada.

- Librería `<map>`:

La librería `<map>` es parte de la biblioteca estándar de C++ y proporciona estructuras de datos como mapas (tablas de hash) que permiten almacenar datos clave-valor. Es ampliamente utilizada en C++ para implementar asociaciones de datos y búsqueda eficiente.

- Librería `<string>`:

La librería `<string>` es una parte fundamental de la biblioteca estándar de C++ que se utiliza para trabajar con cadenas de caracteres. Proporciona clases y funciones para la manipulación y gestión de cadenas, incluyendo operaciones como concatenación, búsqueda y modificación.

- Librería `<cstdlib>`:

La librería `<cstdlib>` incluye funciones que se utilizan para realizar operaciones de conversión y otras operaciones relacionadas con la gestión del sistema. También contiene la función `system` que se utiliza para ejecutar comandos del sistema.

- Librería `<future>`:

La librería <future> es parte de la biblioteca estándar de C++ y se utiliza para la programación concurrente. Proporciona clases como `async` y `future` que permiten realizar tareas asincrónicas y gestionar resultados futuros.

- Librería <chrono>:

La librería <chrono> se utiliza para medir el tiempo y realizar operaciones relacionadas con el tiempo. Es valiosa en aplicaciones que requieren medición de tiempo y control de duraciones.

- Librería "XmlRpc.h":

La librería "XmlRpc.h" es un archivo de encabezado personalizado que se utiliza para incluir funcionalidades específicas de XmlRpc en el código. XmlRpc es un protocolo de comunicación remota basado en XML y esta librería proporciona las herramientas necesarias para trabajar con él en C++.

## IV. Descripción paso a paso del modo de instalación y uso

### Instalación y uso del servidor:

Para poder utilizar la interfaz gráfica del servidor programada en python deberá seguir los siguientes pasos, debe tener en cuenta que esta interfaz le da permisos de administrador del robot y del servidor:

1. Descargar el archivo ProyectoPoo.rar
2. Descomprimir el archivo y guardar la carpeta en ProyectoPoo en el directorio que desee.
3. Luego, abrir la consola CMD.
4. El archivo de la interfaz gráfica del administrador se encuentra dentro de la carpeta ProyectoPoo > SERVIDOR > src > CLI.py. Por lo tanto, una vez dentro de la consola CMD deberá navegar hasta la ubicación del archivo CLI.py utilizando el comando `cd` (cambiar directorio). Por ejemplo:  
`cd C:\Users\Fran\Desktop\Programacion Orientada a Objetos\TP2 - Proyecto final\ProyectoPoo\SERVIDOR\src`
5. Una vez en la ubicación correcta deberá ejecutar el archivo Python usando el comando `python` seguido del nombre del archivo, en este caso es:  
`python CLI.py`

Se adjunta una imagen del procedimiento y el resultado:

```
C:\Users\Fran\Desktop\Programacion Orientada a Objetos\TP2 - Proyecto final\ProyectoPoo\SERVIDOR\src>python CLI.py
Entrada de comandos
->levantarServidor True
Servidor RPC iniciado en el puerto [('192.168.1.48', 8000)]
->levantarServidor False
Servidor Apagado
->exit
Ejecucion CLI SERVIDOR terminada
C:\Users\Fran\Desktop\Programacion Orientada a Objetos\TP2 - Proyecto final\ProyectoPoo\SERVIDOR\src>
```

El archivo CLI.py también se puede ejecutar desde Visual Studio Code.

### Modo de uso en la interfaz por línea de comando del servidor:

Se presenta una guía de uso de los comandos disponibles en la interfaz de usuario del servidor:

- `cls`: Limpia la pantalla de la consola.

Uso: `cls`



- reporte: Muestra un reporte general de la actividad del servidor.

Uso: reporte

- log: Imprime el detalle del Log del Servidor.

Uso: log

- modo: Selecciona el modo de operación en coordenadas absolutas (a) o relativas (r).

Uso: modo <modo> (Ejemplo: modo a)

- robot: Conecta o desconecta el robot.

Uso: robot <opcion> (Ejemplo: robot on o robot off)

- motores: Activa o desactiva los motores del robot.

Uso: motores <opcion> (Ejemplo: motores on o motores off)

- home: Realiza el proceso de homing del robot.

Uso: home

- movLineal: Realiza el movimiento lineal del efector final del robot.

Uso: movLineal <xx.x> <yy.y> <zz.z> [vv.v] (Ejemplo: movLineal 100.0 200.0 50.0 10.0)

- efector: Activa o desactiva el efector final del robot.

Uso: efector <opcion> (Ejemplo: efector on o efector off)

- estado: Indica el estado actual del robot (posición/modo).

Uso: estado

- grabar: Inicia/detiene el proceso de grabación de movimientos para construir un archivo de trabajo con la secuencia grabada.

Uso: grabar <opcion> (Ejemplo: grabar trabajo1 o grabar off)

- cargar: Carga un archivo de trabajo existente en el directorio.

Uso: cargar <archivo> (Ejemplo: cargar trabajo1)

- servidor: Inicia/detiene el servidor RPC según el valor dado (on/off).

Uso: servidor <opcion> (Ejemplo: servidor on o servidor off)

- listar: Muestra los archivos de trabajo en el directorio actual.

Uso: listar

- ejecutar: Ejecuta un comando dado en G-Code.

Uso: ejecutar <comando> (Ejemplo: ejecutar G01 X100 Y200 Z50 F10)

- salir: Termina la ejecución del programa.

Uso: salir

En caso de precisar más ayuda puede recurrir al comando help:

```
->help

                          Ayuda para comandos documentados
=====
cargar  efector  estado  help  listar  modo  movlineal  robot  servidor
cls     ejecutar  grabar  home  log     motores  reporte  salir

->help motores

Activa o desactiva los motores del robot.
motores on|off
```

### Instalación y uso del cliente:

Para poder utilizar el la interfaz gráfica del cliente programada en C++ existen dos métodos, se puede ejecutar el cliente desde terminal o desde la interfaz gráfica.

#### Ejecutar cliente en la terminal:

Método N°1:

Ejecutar el cliente con MSYS32 mediante compilación manual por consola, sin interfaz gráfica. MSYS2 (Minimal SYStem 2) es un entorno de desarrollo que proporciona una colección de herramientas y utilidades para programadores y desarrolladores. No debe confundirse con MSYS (Minimal SYStem), ya que MSYS2 es una versión mejorada y más activa.

MSYS2 es una plataforma que se ejecuta en sistemas Windows y proporciona un entorno similar al de sistemas Unix, lo que facilita la compilación de software de código abierto originalmente diseñado para sistemas Unix en un entorno Windows.

Podrá descargar MSYS2 desde el sitio web oficial donde explican detalladamente cómo realizar la instalación:

#### [MSYS2](#)

Una vez instalado MSYS2 puede empezar a seguir las instrucciones para utilizar esta aplicación. Debe tener en cuenta que al ejecutar el cliente obtendrá permisos de usuario del robot, pero el administrador puede llegar a intervenir en sus acciones si lo cree necesario:

1. Descargar el archivo ProyectoPoo.rar
2. Descomprimir el archivo y guardar la carpeta en ProyectoPoo en el directorio que desee.
3. Luego, abrir la consola MSYS2.
4. Una vez dentro de la consola MSYS2 deberá dirigirse a la dirección donde posea la carpeta ProyectoPoo, utilizando la función cd. Por ejemplo:

```
Fran@DESKTOP-J8TD559 MINGW64 /c/Users/Fran/Desktop/Programacion Orientada a Objetos/TP2 - Proyecto final/ProyectoPoo
$ |
```

En caso de que se presenten dificultades con el acceso a las carpetas o nombres con espacio puede recurrir al comando "dir" para observar las carpetas disponibles.

5. Se debe proceder a compilar el programa de la interfaz gráfica del usuario:  
Dentro de la ubicación de ProyectoPoo se debe colocar la siguiente línea para compilar el cliente:



```
g++ -fdiagnostics-color=always -g -Wall -I.\CLIENTE\CONSOLA\include\xmlrpc
.\CLIENTE\CONSOLA\src\*.cpp .\CLIENTE\CONSOLA\src\xmlrpc\*.cpp -lws2_32 -o
.\CLIENTE\CONSOLA\build\Cliente
```

6. Si la compilación falla obtendrá mensajes de errores, si la compilación ocurre de forma exitosa solo obtendrá mensajes informativos.
7. Luego, deberá dirigirse a la dirección ProyectoPoo > CLIENTE > BUILD

```
Fran@DESKTOP-J8TDS59 MINGW64 /c/Users/Fran/Desktop/Programacion Orientada a Objetos/TP2 - Proyecto final/ProyectoPoo
$ cd CLIENTE/BUILD
```

8. Para ejecutar la interfaz de usuario del Cliente deberá ingresar en la consola “./Cliente IP PUERTO”, donde la IP y el Puerto lo otorgará el servidor. A continuación se muestra un ejemplo:

```
Fran@DESKTOP-J8TDS59 MINGW64 /c/Users/Fran/Desktop/Programacion Orientada a Objetos/TP2 - Proyecto final/ProyectoPoo/CLIENTE/BUILD
$ ./Cliente 192.168.56.1 8000
```

9. Una vez conectado se le pedirá que ingrese su nombre para generar su ID con fin de solicitar la conexión al servidor. Una vez conectado podrá ingresar comandos para utilizar el robot.

```
Fran@DESKTOP-J8TDS59 MINGW64 /c/Users/Fran/Desktop/Programacion Orientada a Objetos/TP2 - Proyecto final/ProyectoPoo/CLIENTE/BUILD
$ ./Cliente 192.168.1.107 8000
Ingrese su nombre <Apellido Nombre>
Miranda Francisco
Su ID de usuario es: miranda_francisco
Bienvenido Miranda Francisco
Ingrese una opción: |
```

En caso requerir ayuda puede utilizar el comando “ayuda” y obtener la lista de comandos y ayuda específica para cada uno:

```
ayuda
Lista de comandos disponibles:
cargar
cls
efector
ejecutar
estado
grabar
home
listar
log
modo
motores
movlineal
reporte
robot
salir
Ingrese el comando del cual desea obtener informacion'.
motores
Comando: motores
Descripción: activa(on) o desactiva(off) los motores del robot.
```

Método N°2:

Ejecutar el cliente con MSYS2 utilizando la herramienta “make” en la consola, sin interfaz gráfica.

Make es una herramienta poderosa de MSYS2 que ofrece automatización, gestión de dependencias y muchas otras ventajas para simplificar el proceso de construcción de software y garantizar una

gestión eficiente de proyectos de desarrollo. Conscientes de estas ventajas se generó un archivo “Makefile” para permitirle a los usuarios poder utilizar construir el archivo con make.

Debe tener en cuenta que make es una herramienta que se debe instalar en MSYS32 para poder ser utilizada.

1. Descargar el archivo ProyectoPoo.rar
2. Descomprimir el archivo y guardar la carpeta en ProyectoPoo en el directorio que desee.
3. Luego, abrir la consola MSYS2.
4. Utilice el comando cd para navegar al directorio que contiene la carpeta CLIENTE dentro de ProyectoPoo. Por ejemplo:

```
Fran@DESKTOP-J8TDS59 MINGW64 /c/Users/Fran/Desktop/Programacion Orientada a Objetos/TP2 - Proyecto final/ProyectoPoo/CLIENTE
$ |
```

5. Luego, una vez dentro de la carpeta ejecute el comando make.

```
Fran@DESKTOP-J8TDS59 MINGW64 /c/Users/Fran/Desktop/Programacion Orientada a Objetos/TP2 - Proyecto final/ProyectoPoo/CLIENTE
$ make
g++ -g -Wall -O2 -I./include/xmlrpc -c src/Cliente.cpp -o build/Cliente.o
g++ -g -Wall -O2 -o build/Cliente.exe build/Cliente.o ./build/libXmlRpc.a -lws2_32
```

6. Una vez finalice la construcción sin errores podrá ejecutar la interfaz de usuario del Cliente ingresando en la consola “./Cliente IP PUERTO”, donde la IP y el Puerto lo otorgará el servidor. A continuación se muestra un ejemplo:

```
Fran@DESKTOP-J8TDS59 MINGW64 /c/Users/Fran/Desktop/Programacion Orientada a Objetos/TP2 - Proyecto final/ProyectoPoo/CLIENTE/BUILD
$ ./Cliente 192.168.56.1 8000
```

7. Una vez conectado se le pedirá que ingrese su nombre para generar su ID con fin de solicitar la conexión al servidor. Una vez conectado podrá ingresar comandos para utilizar el robot.

```
Fran@DESKTOP-J8TDS59 MINGW64 /c/Users/Fran/Desktop/Programacion Orientada a Objetos/TP2 - Proyecto final/ProyectoPoo/CLIENTE/BUILD
$ ./Cliente 192.168.1.107 8000
Ingrese su nombre <Apellido Nombre>
Miranda Francisco
Su ID de usuario es: miranda_francisco
Bienvenido Miranda Francisco
Ingrese una opción: |
```

### Ejecutar cliente en la interfaz gráfica:

Para poder realizar la ejecución de la interfaz gráfica del cliente se debe instalar JDK 21 y netbeans Apache.

El JDK es un conjunto de herramientas y bibliotecas proporcionadas por Oracle Corporation para desarrolladores de software que desean escribir aplicaciones en el lenguaje de programación Java. El JDK incluye el compilador de Java (javac), la máquina virtual de Java (JVM), las bibliotecas estándar de Java y otras utilidades que son esenciales para el desarrollo de aplicaciones Java.

Para descargar JDK deberá instalar la aplicación desde la página de descarga oficial.

#### JDK Development Kit 21.0.1 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the [Java SE OTN License \(OTN\)](#) and production use beyond the [limited free grants](#) of the OTN license will require a fee.

Linux macOS Windows

Product/file description	File size	Download
x64 Compressed Archive	185.39 MB	<a href="https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip">https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip</a> (sha256)
x64 Installer	163.82 MB	<a href="https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe">https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe</a> (sha256)
x64 MSI Installer	162.60 MB	<a href="https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi">https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi</a> (sha256)

Oracle ofrece versiones para Windows, macOS, Linux y otras plataformas. Selecciona la plataforma correspondiente haciendo clic en la opción adecuada.

[Java Downloads | Oracle](#)

Después de la instalación, verifique que Java se haya instalado correctamente abriendo una ventana de terminal CMD y escribiendo el siguiente comando:

```
java -version.
```

Luego, podrá ejecutar la interfaz gráfica de dos formas: Utilizando Apache NetBeans o utilizando Visual Studio Code.

Método N°1:

- Interfaz gráfica ejecutada en Apache NetBeans:

Por otro lado, Apache NetBeans es un entorno de desarrollo integrado (IDE) de código abierto utilizado principalmente para el desarrollo de aplicaciones Java, aunque admite varios otros lenguajes de programación como PHP, HTML5, C/C++, y más.

Para poder ejecutar la interfaz gráfica se recomienda utilizar Apache NetBeans 19, lo puede descargar de la página oficial:

[Downloading Apache NetBeans 19](#)

## Downloading Apache NetBeans 19

Apache NetBeans 19 was released on September 1, 2023.

Apache NetBeans 19 is available for download from your closest Apache mirror.

### Binaries (Platform Independent):

- [netbeans-19-bin.zip](#) (SHA-512, PGP ASC)

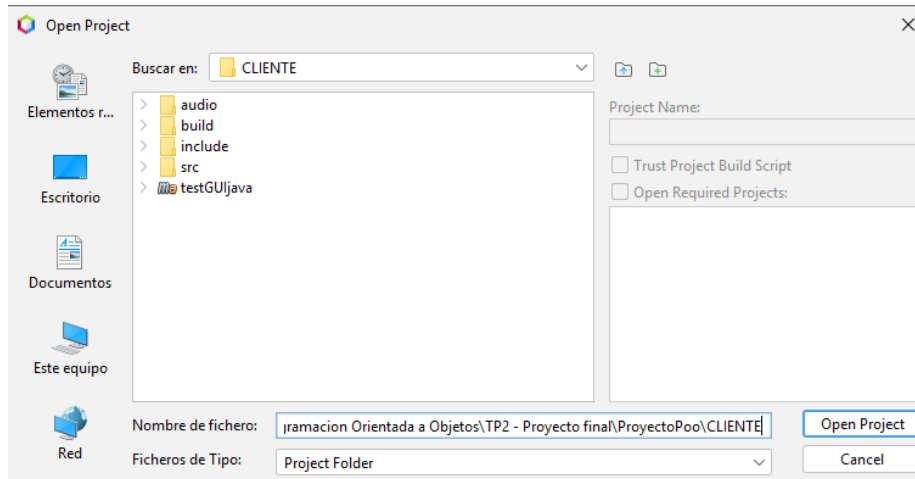
### Installers and Packages:

- [Apache-NetBeans-19-bin-windows-x64.exe](#) (SHA-512, PGP ASC)
- [Apache-NetBeans-19.pkg](#) (SHA-512, PGP ASC)
- [apache-netbeans\\_19-1\\_all.deb](#) (SHA-512, PGP ASC)
- [apache-netbeans-19-0.noarch.rpm](#) (SHA-512, PGP ASC)
- [Linux snap package](#)

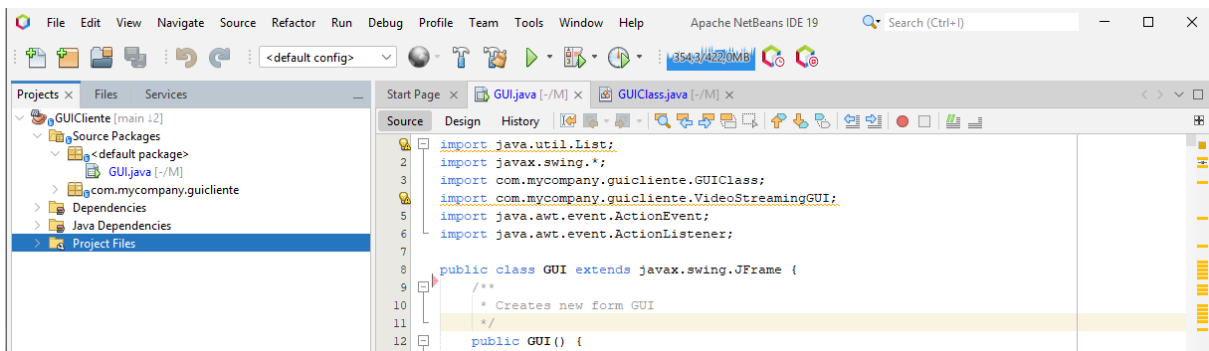
Descargue la versión correspondiente a su sistema operativo, en el proyecto se utilizó y se testeó con la versión de Windows.

Una vez que la instalación se complete podrá ejecutar Apache NetBeans desde el menú de inicio o utilizando el acceso directo que se crea en el escritorio.

Luego, ingresando al programa deberá abrir el proyecto “testGUIjava” cuya carpeta se encuentra en la carpeta ProyectoPoo > Cliente:

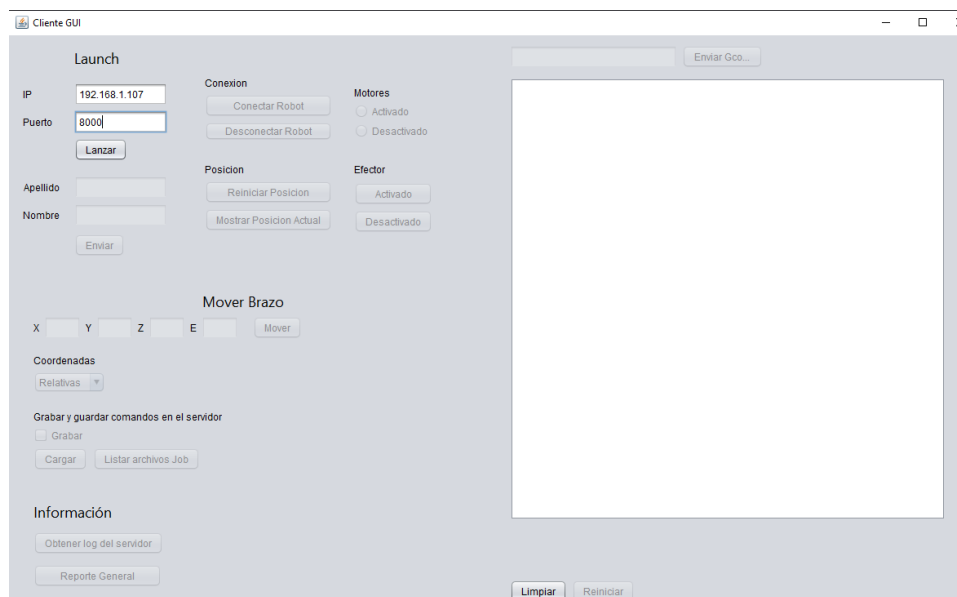


Una vez dentro del proyecto podrá observar el listado de carpetas que conforman el proyecto:



Deberá ingresar a GUICliente > Source Packages > <default package> > GUI.java. Luego puede ejecutar la interfaz con botón verde “Play” o presionando F6 en Windows.

Luego de un tiempo de carga podrá disfrutar de la interfaz gráfica:



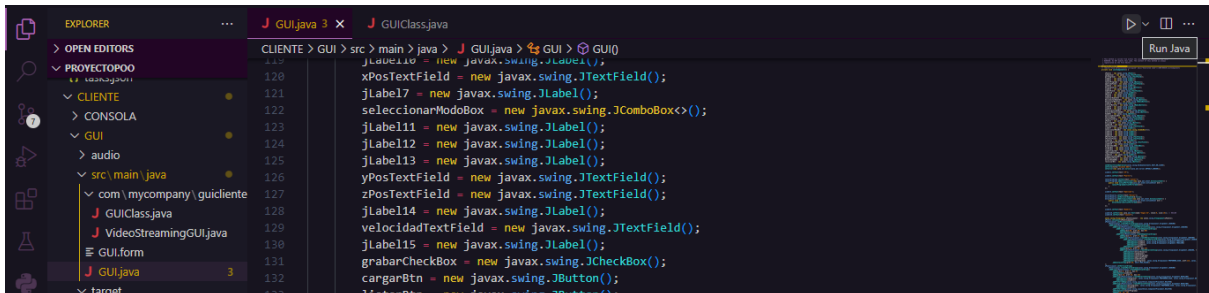
Método N°2:



- Interfaz gráfica ejecutada en Visual Studio Code:

Para ejecutar la interfaz de comando mediante Visual Studio Code deberá abrir el proyecto ProyectoPoo y luego buscar el archivo GUI.java que se encuentra en ProyectoPoo > CLIENTE > CONSOLA > src > GUI.java.

Deberá abrir el archivo GUI.java y luego presionar el botón de play que se encuentra en la parte superior del IDE.



Luego, podrá disfrutar de la interfaz gráfica del cliente.

#### Modo de uso en la interfaz gráfica:

- reporte: Genera un reporte general del robot.

Uso: reporte

- log : Obtiene el registro de actividad del servidor.

Uso: log

- modo : Selecciona el modo (a) coordenadas absolutas o (r) relativas.

Uso: modo <opcion> (Ejemplo: modo a o modo r)

- robot : Conecta (on) o desconecta (off) el robot.

Uso: robot <opcion> (Ejemplo: robot on o robot off)

- motores : Activa (on) o desactiva (off) los motores del robot.

Uso: motores <opcion> (Ejemplo: motores on o motores off)

- home : Mueve el robot a la posición home.

Uso: home

- movlineal : Realiza un movimiento lineal del efecto final del robot.

Uso: movlineal <x> <y> <z> [velocidad] (Ejemplo: movlineal 100 200 50 o movlineal 100 200 50 10)

- efector : Activa (on) o desactiva (off) el efector final del robot.

Uso: efector <opcion> (Ejemplo: efector on o efector off)

- grabar : Graba una secuencia de movimientos en un archivo de trabajo.

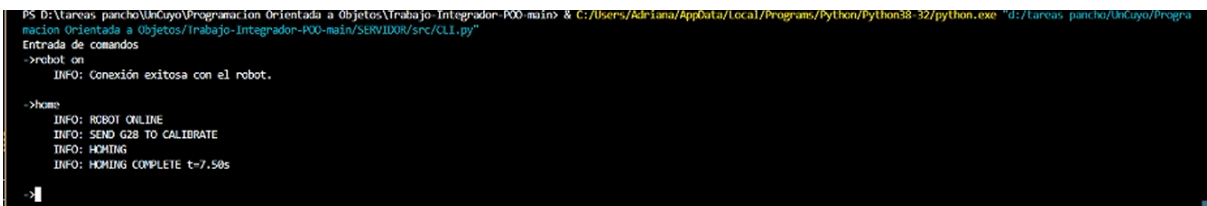
Uso: grabar [nombre\_archivo] (Ejemplo: grabar o grabar mi\_secuencia)

- cargar : Carga una secuencia de movimientos de un archivo de trabajo.  
Uso: cargar <nombre\_archivo> (Ejemplo: cargar mi\_secuencia)
- estado : Obtiene el estado actual del robot (posición y modo).  
Uso: estado
- listar : Lista los archivos de trabajo disponibles en el servidor.  
Uso: listar
- cls : Limpia la consola.  
Uso: cls
- salir : Cierra el programa.  
Uso: salir
- ejecutar : Ejecuta un comando en código G.  
Uso: ejecutar <comando\_en\_codigo\_G> (Ejemplo: ejecutar G01 X100 Y200 Z50)
- ayuda : Muestra la lista de comandos disponibles o proporciona información detallada sobre un comando específico.  
Uso: ayuda para obtener la lista de comandos o ayuda <nombre\_comando> para obtener información detallada sobre un comando (Ejemplo: ayuda motores)

## V. Capturas de pantalla mostrando la ejecución del aplicativo, para cada una de las interfaces de entrada/salida propuestas para interacción con el usuario.

La interfaz de usuario del lado del servidor está implementada como una línea de comandos (CLI) que permite a los usuarios interactuar con el brazo robot y el mismo servidor. Esta interfaz de línea de comandos proporciona una forma interactiva de controlar el brazo robot y gestionar archivos de trabajo.

El robot puede controlarse sin necesidad de un servidor. Desde la CLI, una computadora puede hacerlo.



```
PS D:\Tareas pancha\UNCuyo\programación Orientada a Objetos\trabajo-Integrador-P00-main> C:/Users/Adriana/AppData/Local/Programs/Python/Python38-32/python.exe "D:/Tareas pancha/UNCuyo/Programación Orientada a Objetos/trabajo-Integrador-P00-main/SERVIDOR/src/CLI.py"
Entrada de comandos
->robot on
INFO: Conexión exitosa con el robot.

->home
INFO: ROBOT ONLINE
INFO: SEND G28 TO CALIBRATE
INFO: HOMING
INFO: HOMING COMPLETE t=7.50s
->
```

También puede elegir levantar un servidor y utilizar un cliente remoto.

```

->home:
INFO: ROBOT ONLINE
INFO: SEND G28 TO CALIBRATE
INFO: HOMING
INFO: HOMING COMPLETE t=7.50s

->estado
INFO: ABSOLUTE MODE
INFO: CURRENT POSITION: [X:0.00 Y:170.00 Z:120.00 E:0.00]

->servidor on
[ WARN:0@229.171] global cap.cpp:344 cv::VideoCapture::open VIDEOIO(DSHOW): backend is generally available but can't be used to capture by index
Servidor RPC iniciado en el puerto [('192.168.56.1', 8800)]
->
  
```

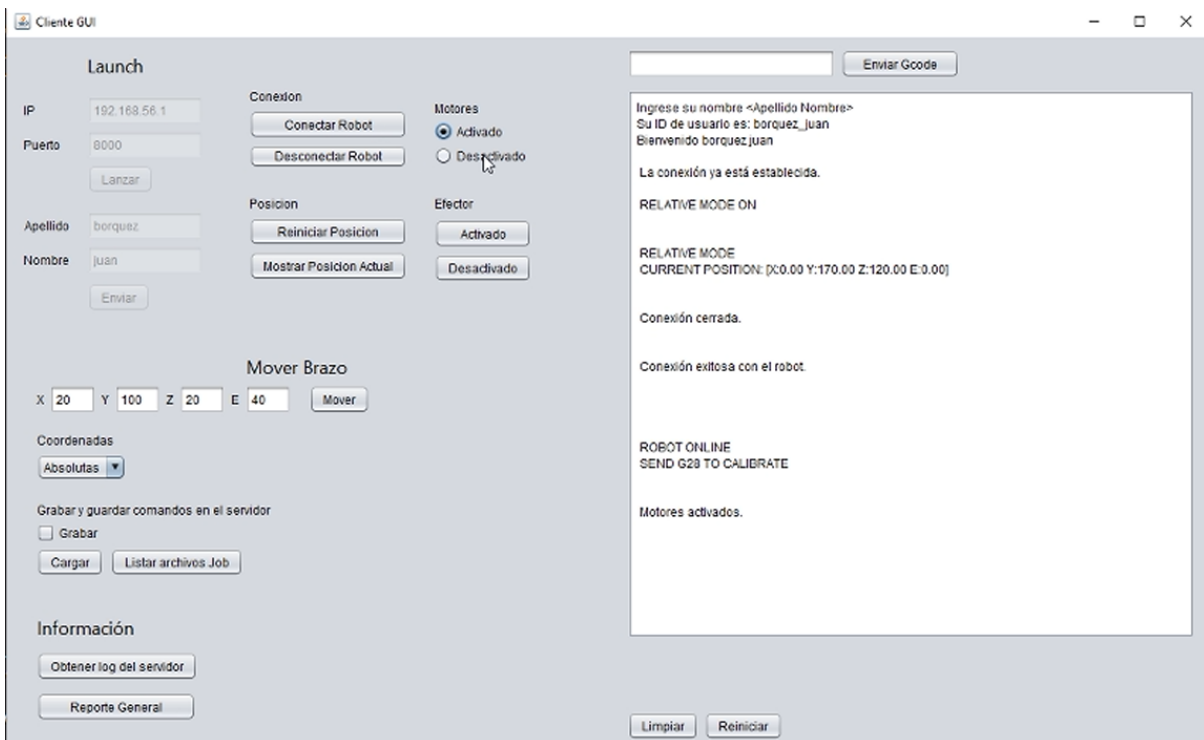
```

PS D:\tareas_panco\UnCuyo\Programacion Orientada a Objetos\Trabajo Integrador-POO-main\cliente\consola\build> ./cliente 192.168.56.1 8800
Ingrese su nombre <Apellido Nombre>
dalessandro francisco
Su ID de usuario es: dalessandro_francisco
Bienvenido dalessandro francisco

home
HOMING
HOMING COMPLETE t=4.50s
  
```

Como se puede ver, el cliente recibe la misma respuesta que si se ejecuta el servidor de manera local.

Por último, se puede hacer uso de la GUI, que facilita el uso de la aplicación del cliente.



## VI. Conclusiones:

El desarrollo de este proyecto bajo el paradigma de la programación orientada a objetos ha presentado muchas dificultades, sin embargo, todos los problemas presentados fueron una buena oportunidad para aprender acerca de la interacción entre el servidor y el cliente a través de dos lenguajes de programación diferentes, Python y C++, y el entorno de desarrollo Arduino mediante

una placa Arduino Uno. Durante el proceso, se han enfrentado varios desafíos que han enriquecido nuestra experiencia y conocimiento en el campo de la programación.

La creación de una interfaz gráfica de usuario (GUI) fue un desafío importante. Originalmente, habíamos considerado utilizar la biblioteca QT para C++; sin embargo, debido a errores y restricciones de tiempo, optamos por Java para implementar la GUI del cliente. Aunque esta elección limitó algunas funcionalidades avanzadas, nos permitió ofrecer una interfaz efectiva y funcional para la conexión, el control y monitoreo del robot.

El enfoque de programación orientada a objetos resultó ser esencial para la organización y el mantenimiento del código. La modularización de nuestro proyecto en módulos claramente definidos, tanto en el lado del servidor como en el lado del cliente, facilitó la adición de nuevas características. Además, la implementación de servicios RPC proporcionó una comunicación efectiva entre ambas partes.

En cuanto al lado del cliente, reconocemos que existen oportunidades para mejorar y ampliar sus características. Con más tiempo y recursos, podríamos haber desarrollado una GUI más sofisticada y efectos adicionales, como una representación visual más avanzada del robot y una integración más completa del streaming de video.

En resumen, este proyecto nos ha permitido abordar desafíos técnicos complejos y enriquecer nuestras habilidades en programación orientada a objetos, interacción entre lenguajes de programación y desarrollo de sistemas distribuidos. Aunque hemos enfrentado dificultades, hemos demostrado la capacidad de adaptarnos y superar obstáculos con el fin de cumplir nuestros objetivos. Este trabajo integrador representa un paso significativo en nuestro camino hacia convertirnos en profesionales capacitados.