Trabajo Integrador

Programa para el Control de un Brazo Robot

Cátedra: Programación Orientada a Objetos

Profesor: Esp. Ing. César Omar Aranda

Integrantes: Guarise, Renzo (12262)
Trubiano, Lucas (12289)

2020

Índice

Resumen

El desafío propuesto en la asignatura fue hacer un software capaz de controlar un brazo manipulador robótico de 3 grados de libertad (RRR), que es un brazo con una cadena cinemática de 3 rotaciones en el espacio. En nuestro caso elegimos controlar el robot de la marca ABB modelo IRB 460, el cual también contaba con un efector final que para nosotros es una pistola para pintar.

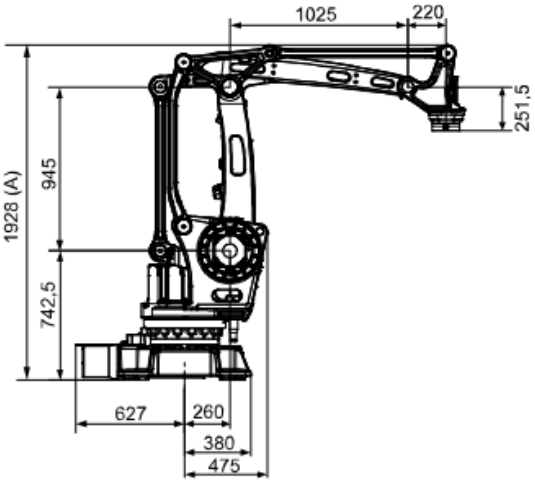
Partiendo de ese robot y de las funcionalidades pedidas en las consignas, desarrollamos una aplicación de dos capas cliente y servidor para el control remoto de nuestro robot.

La aplicación de servidor es la encargada de interactuar con el objeto robot (desde el paradigma de orientación a objetos) y también de lanzar una consola de comandos, una interfaz gráfica y un servidor que reciba los mensajes y peticiones de clientes. Por otro lado el cliente no es más que una consola que nos permite interactuar con las funcionalidades que ofrece el servidor.

El código del servidor se implementó en lenguaje Python mientras que el cliente se implementó en C++.

Sobre el Robot

Ahora vamos a entrar un poco más en detalle sobre el robot, su construcción y su funcionamiento. Consultando principalmente la documentación disponible en la página de ABB encontramos una figura o modelo 3D del robot juntos con sus especificaciones técnicas



Este robot cuenta con 4 eslabones y **3 articulaciones** (de rotación), en nuestra aplicación concretamente tratamos las **articulaciones como objetos** que formaban parte de uno mayor que era el robot, es decir, que el **objeto robot** conocía sus eslabones y dimensiones pero agregaba las distintas articulares ya que cada una tiene sus propios parámetros como límites articulares, velocidades, etc.

Luego otro punto importante de este robot fue conocer las funciones que podía hacer como objeto propiamente dicho y que información necesitaba para hacer esas funciones. Esto lo pensamos desde el punto de vista del encapsulamiento, de ocultar al resto del programa la parte de como hace el robot para moverse y dejando solamente funciones disponibles de lo que se puede hacer.

Como ejemplo de esto tenemos:

- La acción de **Activar** o **Desactivar** el Robot,
- Las funciones de **mover sus grados** articulares hacia una cierta coordenada específica (Cinemática Directa).
- O de recibir una **posición cartesiana** en el espacio y buscar las coordenadas articulares que lo llevan a esa posición (Cinemática Inversa).
- También tenemos la acción de homing para que el robot se referencie siempre a las mismas posiciones articulares.
- Y algunas otras.

Respecto del **efector final** que es un objeto separado en principio, nosotros implementamos una clase efector final pero la cual es controlada y comandada por el objeto robot. Este efector final básicamente tiene las funciones de **elegir un color** con el cual desea pintar, y puede **empezar a pintar** o **dejar de pintar**.

Para resolver los problemas de cinemática directa, cinemática inversa y trayectorias usamos el **Robotics Toolbox** de **Peter Corke** en su versión para Python que es muy similar a la que usamos en Matlab para la asignatura de Robótica 1. Y también utilizamos unas herramientas de ese toolbox que nos permiten simular en el espacio 3D un modelo simplificado de la geometría de nuestro robot.

Temas especiales investigados en Python/C++

Python:

- Servidor XML-RPC
- Librería CMD
- Threading
- Librería roboticstoolbox
- Librería Tkinter

C++:

- Cliente XML-RPC
- Manejo de excepciones
- Manejo de contenedores
- Manejo de string

Marco Teórico

Como ya dijimos el aplicativo del lado servidor se realizó en python siguiendo un paradigma orientado a objetos, para esto se dividió al robot en sí en distintos objetos, siguiendo los conceptos de encapsulamiento, herencia, abstracciones y polimorfismo.

Por el lado cliente, se utilizó el lenguaje C + + como se dijo y se siguió el mismo paradigma de programación que en el lado servidor.

Para la creación de la consola en python se creó una clase a la cual denominamos Panel de Control la cual hereda de cmd, esta es una clase contenida en el módulo cmd. La clase cmd está diseñada para usarse como clase base para shell interactivos y otros intérpretes de comandos. Del lado cliente se creó una clase denominada cmdCliente la cual establece los diferentes métodos para la implementación de la consola del lado cliente. Para el procesamiento de las entradas en el lado cliente se utilizan las librerías:vector ,sstream y string. La librería vector permite la creación de contenedores de datos del tipo array y las librerías sstream y string se utilizaron para el procesado de cadenas de caracteres.

Para la creación del servidor utilizamos el protocolo XML-RPC, que es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes. A partir del módulo en python xmlrpc.server se puede establecer un servidor de manera muy simple y del lado del cliente utilizamos la librería "XmlRpc.h" la cual permite la comunicación entre Cliente-Servidor mediante diferente metodos que hacen uso de un tipo de dato especial provisto por la librería denominado XmlRpcval lo que permite al programador liberarse de la codificación del mensaje en el lenguaje de etiquetas XML.

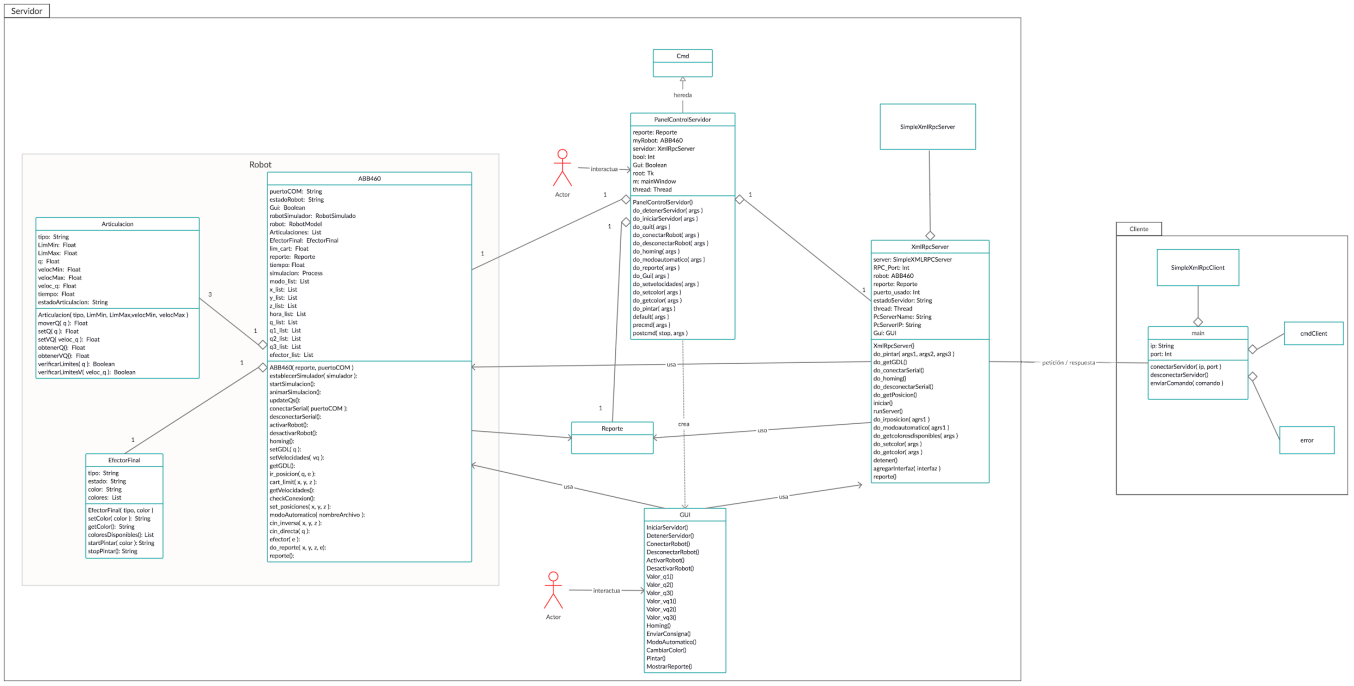
Por otra lado, para que la consola del lado servidor no se vea retenida por el loop establecido por el servidor, se lanzó a este en un hilo de manera que se pueda utilizar la consola aun cuando el servidor se encuentra escuchando al cliente. Los hilos permiten a nuestras aplicaciones ejecutar múltiples operaciones de forma concurrente en el mismo espacio de proceso. Para esto se utilizó en python el módulo Threading que nos permite la creación de hilo de manera fácil instanciando un objeto de la clase thread y pasándole el objetivo, target en inglés, el cual va a correr en el hilo.

Para resolver la cinemática directa e inversa del robot en cuestión se utilizó el módulo roboticstoolbox que a través del instanciado de la clase DHRobot y pasando como argumento la matriz DH característica del robot permite mediante distintos métodos el cálculo de estas. Por otra lado, este módulo permite realizar una representación gráfica fácilmente del robot objeto.

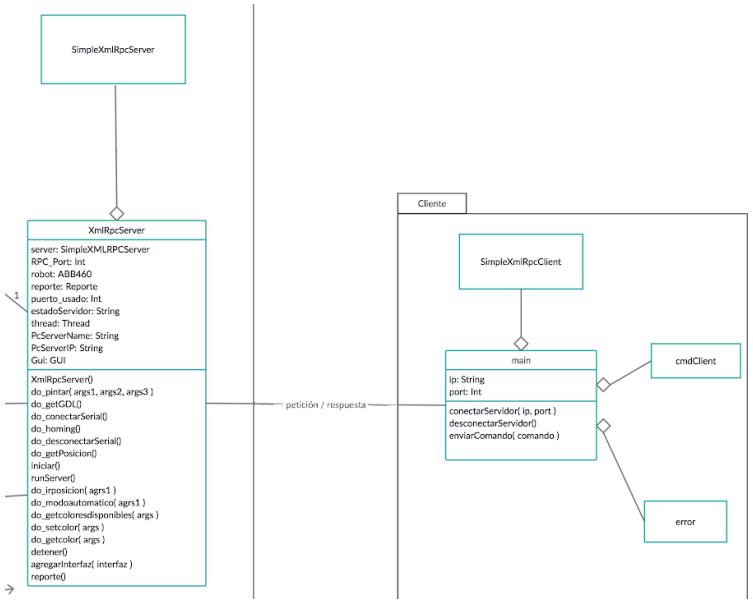
Para la realización de la interfaz gráfica se utilizó el módulo nativo de python llamado tkinter, este es un binding de la biblioteca gráfica Tcl/Tk y se considera un estándar para la interfaz gráfica de usuario (GUI) para Python.

Esquema de Orientación a Objetos con UML

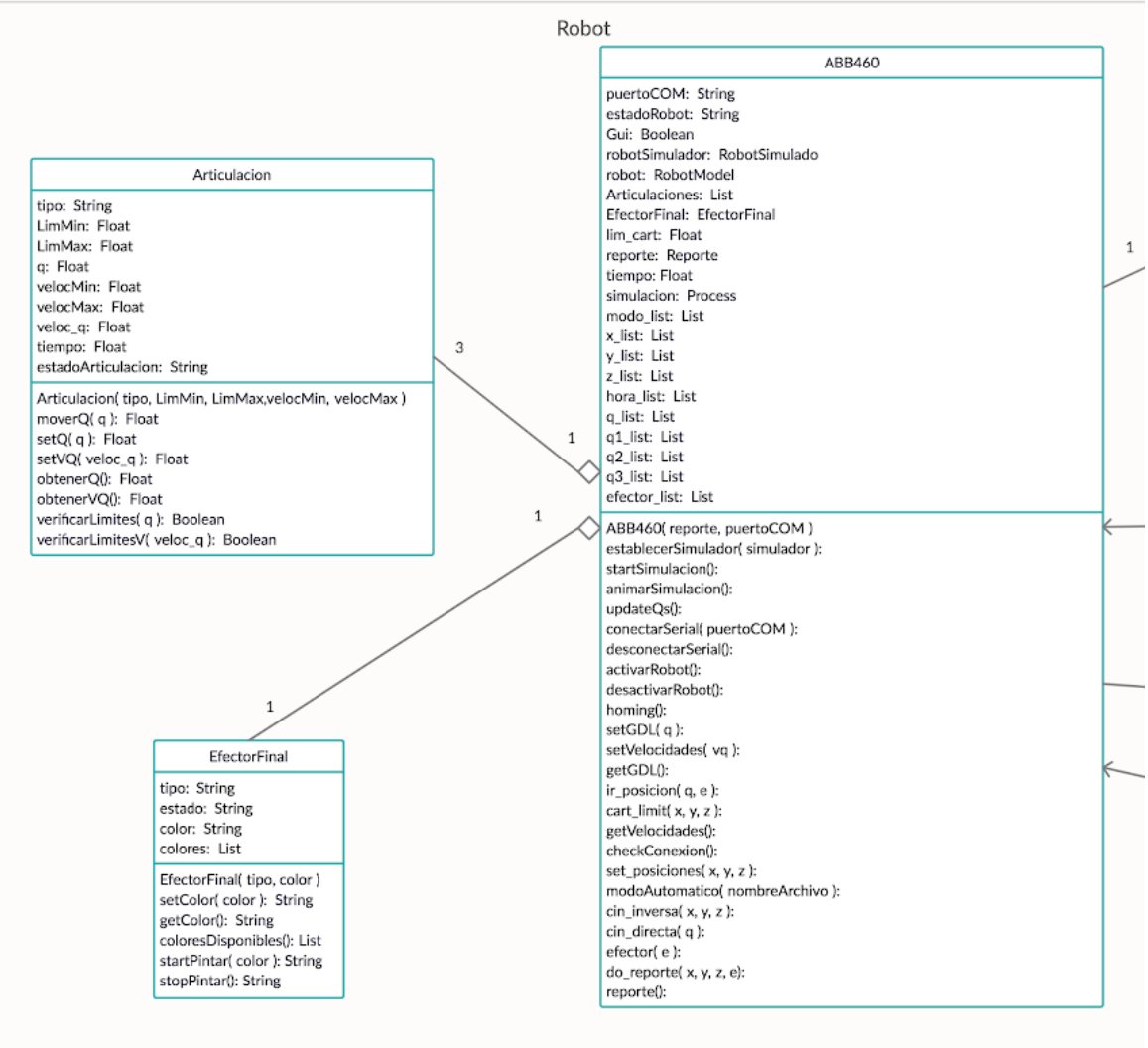
Vista General:



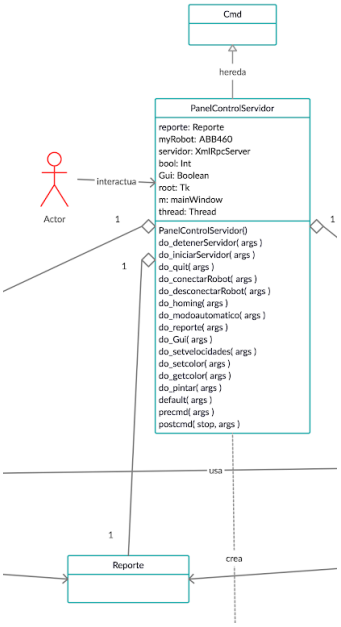
Vista Cliente-Servidor:



Vista Robot:



Vista Panel de Control Servidor:



Clases servidor:

PanelControlServidor:

Clase que hereda de cmd. Permite la interacción del usuario del lado servidor a través de una línea de comandos. A través de sus métodos se puede:

• IniciarServidor: iniciar el servidor XMLRPC,

• conectarRobot: conectar o desconectar la comunicación del robot a través del puerto serie, también activarlo o desactivarlo.

• Gui: lanzar una interfaz gráfica,

• homing: ir a una posición establecida como homing,

• setGDL: setear las posiciones articulares de las tres articulaciones

• setColor:elegir el color con el que se va a pintar.

• modoautomatico: ejecuta una serie de comandos guardados en un archivo.

• Pintar: realizar un trayecto de una posición cartesiana a otra pintando o no.

• reporte: emite un reporte completo sobre todos los comandos utilizamos y las distintas posiciones tanto cartesianas como articulares del robot.

Entre otras operaciones, es importante destacar que toda acción realizada en el panel de control de manera manual en cuanto a las acciones del robot, es decir una operación homing por ejemplo, quedarán guardadas por defecto en un archivo llamado comandosmanuales.txt, de manera que se pueda repetir una operación ya realizada, también es importante decir que este archivo se sobre escribe cuando se lanza nuevamente el panel, por lo tanto, si se lo quiere guardar es recomendable extraerlo de la carpeta donde se encuentra el programa.

ABB460:

Clase troncal de nuestra aplicación. Agrega las clases de efector final y articulación. Opera las articulaciones del robot y el efector final. Realiza los cálculos necesarios para realizar las tareas requeridas por el cliente o servidor. Algunos de sus métodos son:

- cininversa: realiza el cálculo de la cinemática inversa del robot
- cindirecta:realiza el cálculo de la cinemática directa del robot.
- reporte: formatea un reporte completo sobre todas las características del robot en una string. Método utilizado tanto por Servidor como el Cliente.
- setGDL: realiza el cambio en las posiciones articulares de los grados de libertad
- setvelocidades: permite setear las velocidades de las articulaciones.
- irposicion:realiza la logística necesaria en el robot para ir de una posición a otra.

XmlRpcServer:

Clase encarga de las características del servidor. Esta clase lanza el servidor y posee los distintos servicios proporcionados por el servidor. Es importante remarcar que esta clase realiza una instanciado de la clase SimpleXMLRPCServer mediante la cual se lanza el servidor, mediante el uso de los métodos de esta clase podemos registrar las funciones que queremos que posea nuestro servidor entre otras cosas.

EfectorFinal:

Clase que es agregada en la clase ABB460 y la cual posee todas las características del efector final.

Articulacion:

Clase que es agregada en la clase ABB460 y posee las distintas características de la articulación.

reporte:

Clase que se encarga de registrar todas las acciones realizadas tanto por el cliente como el servidor, está además registra la hora a la que se realizan las operaciones y el usuario que las realizó. Esta clase es luego pasada como argumento al método reporte de la clase ABB460 para formatear el reporte.

GUI:

La clase GUI es agregada o instanciada por PanelControlServidor el cual lanza la GUI cuando se lo ordenan por comandos

Esta clase crea todos los elementos que se necesitan en la interfaz, para que el usuario pueda interactuar con el servidor y el robot. También instancia la clase RobotSimulado que es una ventana que permite simular el robot. Además la clase GUI interactúa con varios objetos que son sus diferentes ventanas.

RobotSimulado:

Esta clase hace uso de la librería Robotics Toolbox de Peter Corke para crear una ventana emergente que tiene el robot modelado y además nos permite algunas funcionalidades como por ejemplo recibir comandos y mover al robot en tiempo real.

Clases cliente:

cmd:

Clase encargada de manejar los servicios del servidor del lado cliente. Posee métodos muy parecidos a los del Panel de Control del lado servidor. Estos son:

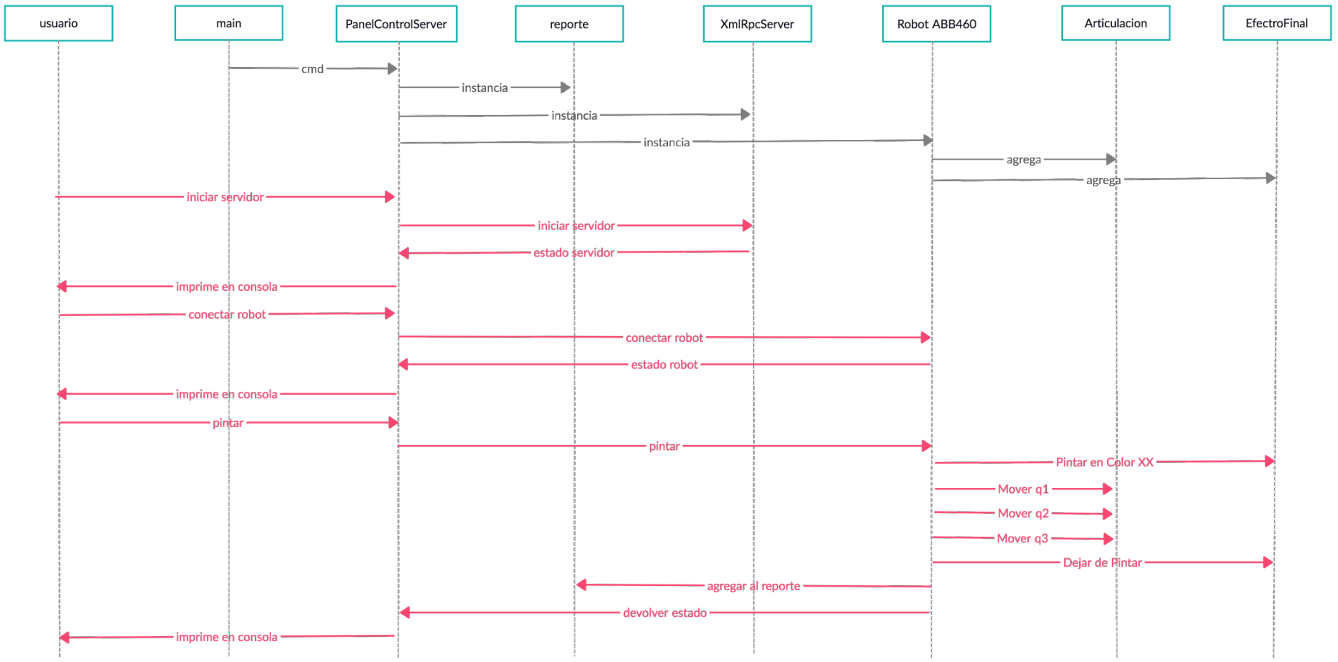
- serviciosservidor: devuelve una lista con todos los servicios disponibles
- pintar: maneja el servicio pintar del servidor.
- irposicion: maneja el servicio irposicion del servidor.
- modoautomatico: maneja este mismo servicio del servidor
- help: devuelve la lista de servicios y la descripción del comando pedido.
- enviarcomandos: enviá el servicio requerido al servido con los argumentos necesario.

Hay que destacar que si bien la consola acepta comandos del tipo ‘servicio argumento’, se realizaron métodos para el manejo de los servicios de manera que la interacción con el usuario sea más amigable.

Error:

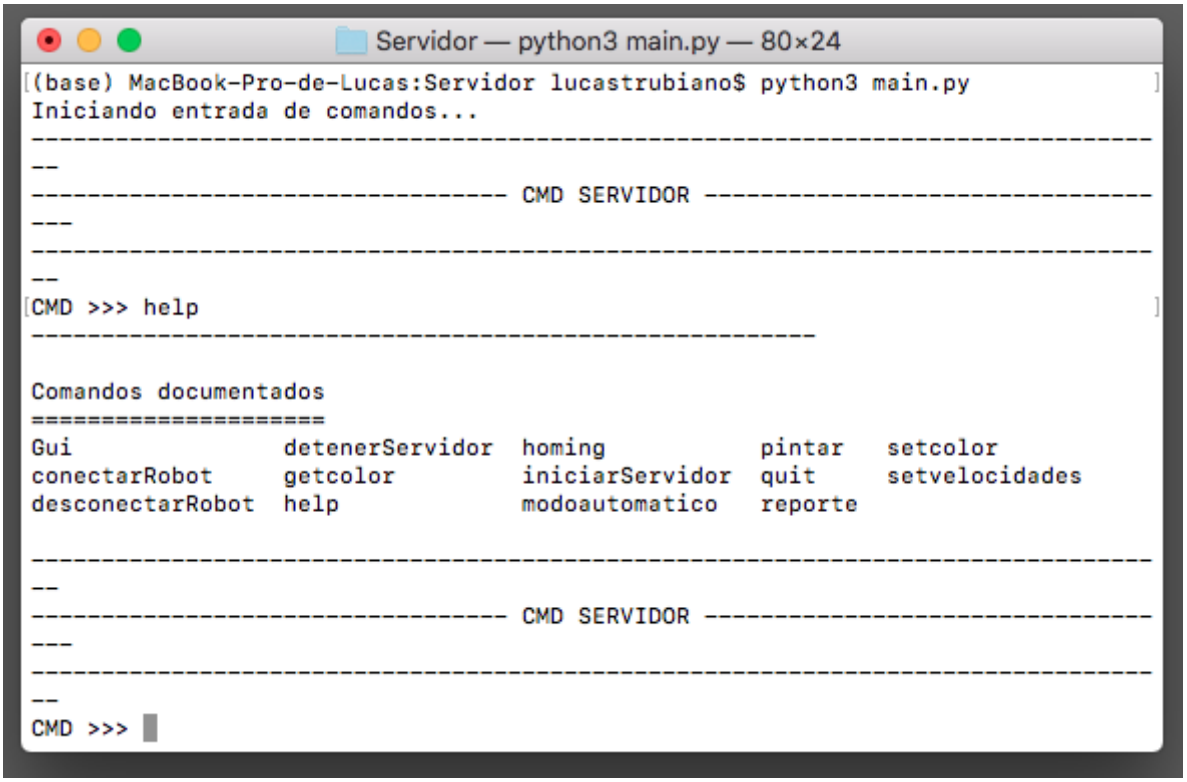
Clase destinada al manejo de errores en el lado cliente.

Esquema Temporal de Ejecución



En la figura mostrada anteriormente se ejemplifica cómo interactúan las clases en un diagrama temporal, solo están mostradas algunas acciones de todas las posibles. Pero sirve para ver como pasan los mensajes de clases en clases y son devueltas luego al usuario.

Guia sobre el Servidor

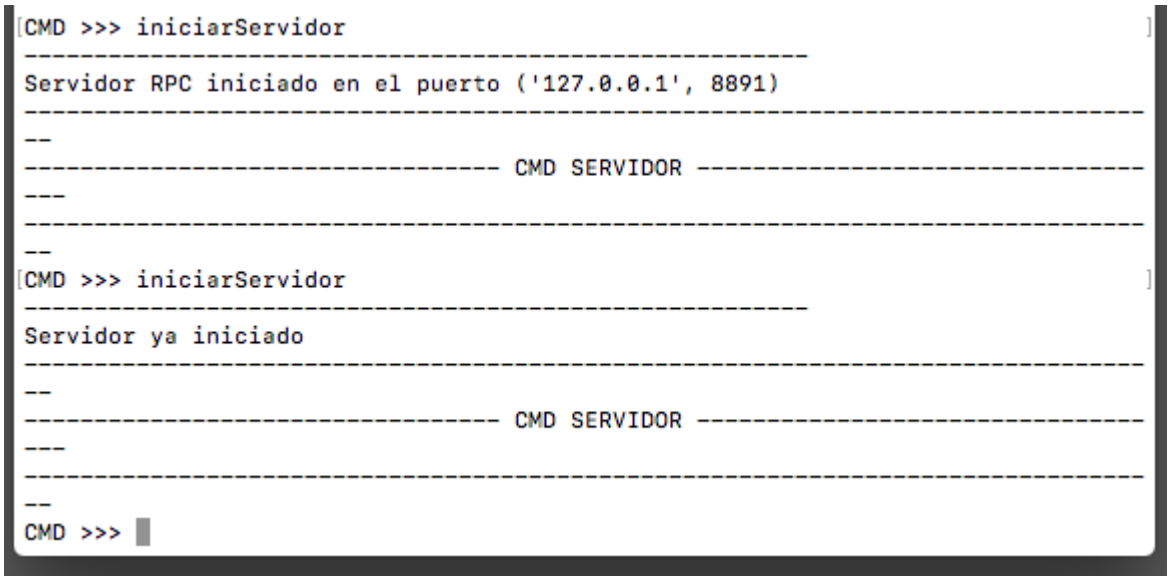


Para ejecutar el servidor lo primero que hay que hacer es:

- ejecutar en una consola el comando **python3 main.py** o simplemente **python main.py** (donde el archivo main es el que se encuentra dentro de la carpeta servidor)
- luego podemos escribir un típico comando **help** para ver los comandos disponibles.

¿Cómo usar el servidor desde consola?

- Comando **iniciarServidor**:



Si llamo a **iniciarServidor** por primera vez inicia el servidor en el puerto 8891, esta función no recibe parámetros. Si la llamo con el servidor ya iniciado muestra un mensaje de que el servidor ya se inició.

- Comando **detenerServidor**:

```
[CMD >>> detenerServidor                                                    ]
------------------------------------------------------------
Servidor RPC detenido
Servidor Detenido
[Desea salir [s/n]?  n                                                      ]
------------------------------------------------------------------------------
--
---------------------------------- CMD SERVIDOR ------------------------------
---
------------------------------------------------------------------------------
--
CMD >>>
```

La función **detenerServidor** me permite si el servidor está iniciado, detener su ejecución y además nos pregunta si deseamos salir de toda la aplicación o no. Si deseamos cerrar el servidor y salir de la aplicación pondremos S de si. Si queremos detener el servidor pero no cerrar la aplicación entonces pondremos N de no.

¿Cómo usar el robot desde consola?

- Comando **conectarRobot**:

```
[CMD >>> help conectarRobot                                                 ]
------------------------------------------------------------
Conecta por puerto serial el robot
------------------------------------------------------------------------------
--
---------------------------------- CMD SERVIDOR ------------------------------
---
------------------------------------------------------------------------------
--
[CMD >>> conectarRobot COM3                                                 ]
------------------------------------------------------------
Conectado [COM3]
Activado [COM3]
------------------------------------------------------------------------------
--
---------------------------------- CMD SERVIDOR ------------------------------
---
------------------------------------------------------------------------------
--
CMD >>>
```

El comando **conectarRobot** recibe como argumento un puerto serie que puede estar denominado por un **COMX** y a continuación se conecta a ese puerto y activa el robot.

- Comando **desconectarRobot**:

```
CMD >>> desconectarRobot
------------------------------------------------------
Desconectado [COM3]
------------------------------------------------------------------------------------
--
---------------------------------- CMD SERVIDOR ----------------------------------
---
------------------------------------------------------------------------------------
--
CMD >>>
```

La función desconectarRobot lo que hace es desactivar el robot y desconectarlo si es que este estaba conectado sino no hace nada y muestra un mensaje de error.

- Comando **getcolor**:

```
CMD >>> getcolor
------------------------------------------------------
Negro
------------------------------------------------------------------------------------
--
---------------------------------- CMD SERVIDOR ----------------------------------
---
------------------------------------------------------------------------------------
--
CMD >>>
```

Este comando devuelve el color que tiene seteado el efector final de nuestro robot.

- Comando **homing**:

```
CMD >>> homing
------------------------------------------------------
Yendo a Home [0.0,0.0,0.0]
------------------------------------------------------------------------------------
--
---------------------------------- CMD SERVIDOR ----------------------------------
---
------------------------------------------------------------------------------------
--
CMD >>>
```

El comando **homing** manda al robot a las coordenadas articulares [0,0,0], para que las mismas busquen una posición de referencia siempre igual.

- Comando **modoautomatico**:

```
CMD >>> modoautomatico comandos.txt
------------------------------------------------------
comandos.txt
Archivo leido exitosamente
------------------------------------------------------------------------------------
--
---------------------------------- CMD SERVIDOR ----------------------------------
---
------------------------------------------------------------------------------------
--
CMD >>>
```

Este comando nos permite leer un archivo de secuencias ya programadas que el robot puede ejecutar de forma automática. Se recibe como parámetro el nombre de un archivo con extensión **txt**, por ejemplo **mi_archivo.txt**
Y muestra un mensaje si lo leyó correctamente o no.

- Comando **pintar**:

```
[CMD >>> pintar
 ---------------------------------------------------------
 Ingrese posicion de inicio:
[? 0.1,0.1,0.1
 Ingrese posicion hasta la que desea ir:
[? 0.2,0.2,0.2
 Ingrese si quiere pintar en el trayecto a no [S/N]
[? S
 Yendo a p=0.1,0.1,0.1
 Yendo a p=0.2,0.2,0.2
 ------------------------------------------------------------------------------
 --
 ---------------------------------- CMD SERVIDOR ----------------------------
 ---
 ------------------------------------------------------------------------------
 --
 CMD >>>
```

El comando **pintar** no recibe argumentos, pero si:

- Como primer paso nos pide una posición cartesiana inicial (por ejemplo X = 10 cm, Y = 10 cm y Z = 10 cm) que se le debe pasar en metros, con las coordenadas ordenadas y separados por coma primero **x**, luego **y** y al final **z**. como se ve en el ejemplo.
- Luego le pasamos en el mismo formato una posición final (también cartesiana y en metros).
- Y por último un parámetro para elegir si queremos que pinte en ese trayecto o no.

- Comando **reporte**:

```
CMD >>> reporte
------------------------------------------------------
Estado del robot:Activado
Tiempo de ejecucion: 1359.7309837341309

Lista de comandos usados

Servidor  2020-11-21 20:35:59.664634   help
Servidor  2020-11-21 20:36:02.238174   iniciarServidor
Servidor  2020-11-21 20:36:14.966891   iniciarServidor
Servidor  2020-11-21 20:36:29.231964   detenerServidor
Servidor  2020-11-21 20:36:35.002542   iniciarServidor
Servidor  2020-11-21 20:36:44.999773   detenerServidor
Servidor  2020-11-21 20:46:06.472944   help
Servidor  2020-11-21 20:46:15.611490   conectarRobot
Servidor  2020-11-21 20:50:56.290855   desconectarRobot
Servidor  2020-11-21 20:52:14.202435   conectarRobot
Servidor  2020-11-21 20:52:16.809907   getcolor
Servidor  2020-11-21 20:53:44.865364   homing
Servidor  2020-11-21 20:55:29.430688   modoautomatico
Servidor  2020-11-21 20:58:33.383198   pintar
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
|    Modo    |            Hora            |    x     |    y     |    z     |    q1    |    q2    |    q3    |   Efector    |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
|     Manual | 2020-11-21 20:53:44.865221 |     0.45 |     0.00 |     0.35 |     0.00 |     0.00 |     0.00 | No Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-21 20:55:25.797185 |     0.10 |     0.10 |     0.10 |     0.00 |     0.00 |     0.00 | No Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-21 20:55:26.711307 |    -0.10 |    -0.90 |     0.09 |     0.79 |     1.04 |     0.36 |    Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-21 20:55:27.481105 |     0.09 |     0.80 |     0.01 |    -1.68 |     0.41 |    -0.93 |    Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-21 20:55:27.926252 |     0.08 |     0.07 |     0.01 |     1.46 |     0.54 |    -0.93 |    Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-21 20:55:28.290619 |     0.70 |     0.06 |    -0.06 |     0.72 |     1.50 |    -0.93 |    Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-21 20:55:29.115883 |    -0.60 |    -0.50 |     0.08 |     0.09 |     0.68 |    -0.93 |    Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-21 20:55:29.430487 |     0.50 |     0.04 |     0.60 |    -2.45 |     0.48 |    -0.93 | No Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
|     Manual | 2020-11-21 20:58:33.365585 |     0.10 |     0.10 |     0.10 |     0.08 |    -0.34 |    -0.93 |    Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
|     Manual | 2020-11-21 20:58:33.383124 |     0.20 |     0.20 |     0.20 |     0.79 |     1.04 |     0.36 | No Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|

------------------------------------------------------------------------------
---------------------------- CMD SERVIDOR ----------------------------
------------------------------------------------------------------------------
CMD >>>
```

Este comando lo que hace es mostrarnos una serie de información muy útil del historial de acciones que se han realizado con el robot. Podemos ver estados del robot, tiempo de ejecución, comandos escritos por consola, y los movimientos del robot.

- Comando **setcolor**:

```
CMD >>> setcolor
------------------------------------------------------
Los colores disponibles son:

['Rojo', 'Verde', 'Azul', 'Amarillo', 'Naranja', 'Violeta', 'Blanco', 'Negro']
Ingrese uno:
? Rojo
Rojo
------------------------------------------------------------------------------
---------------------------- CMD SERVIDOR ----------------------------
------------------------------------------------------------------------------
CMD >>>
```

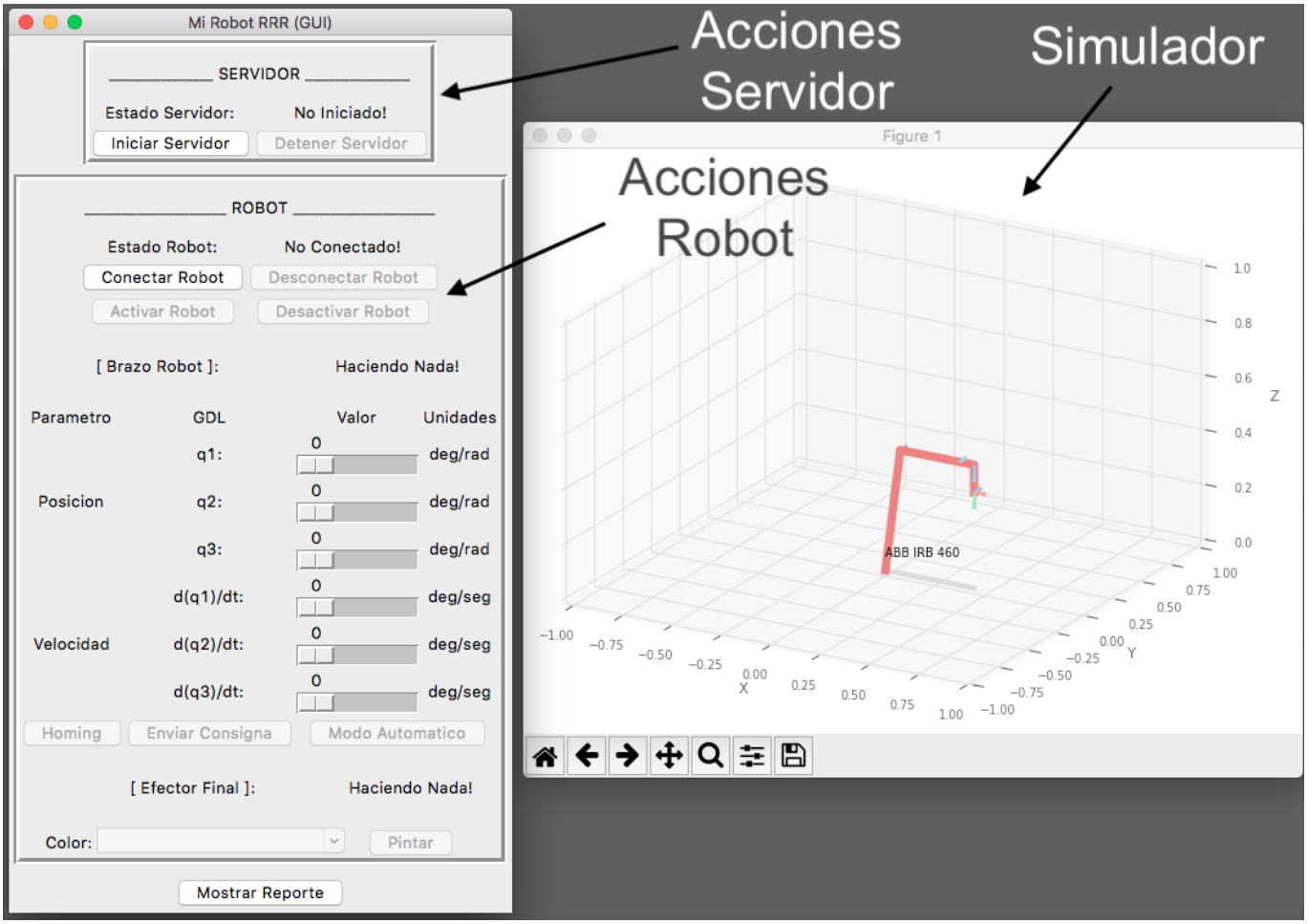
Con este comando nos ofrece una serie de colores disponibles y podemos ingresar como textos alguno de esos.

¿Cómo lanzar la interfaz gráfica desde consola?

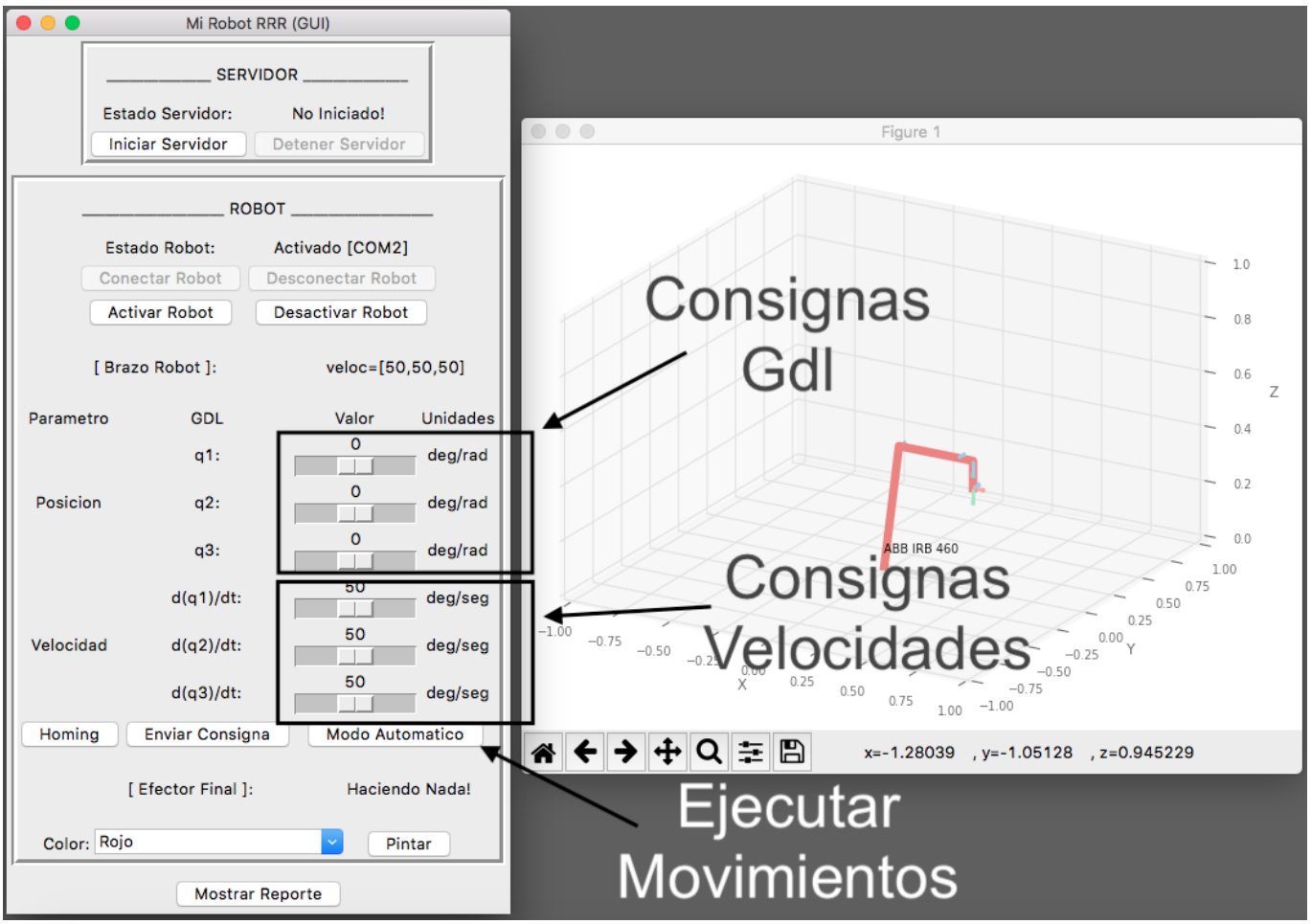
Comando Gui para lanzar la interfaz gráfica y luego de lanzarla se inicia la simulación

```
------------------------------------------------------------------------------
---------------------------------- CMD SERVIDOR ------------------------------
------------------------------------------------------------------------------
[CMD >>> Gui                                                                  ]
------------------------------------------------------
Comenzamos Simulacion
```

Algunos botones de la interfaz



Las primeras cosas a tener en cuenta en la interfaz es que contamos con una ventana que contiene un modelo del robot para simular los movimientos, luego en la ventana principal tendremos las acciones de **iniciar** y **detener** el **servidor**. Y también las acciones de **conectar**, **desconectar**, **activar** y **desactivar** el **robot**.



Una vez conectado y activado el robot vamos a tener acceso al resto de la interfaz con lo cual podremos mover los **Sliders** para acomodar el robot en una **posición** deseada, también podremos ajustar su **velocidad** y disponemos de 3 botones que son para **ejecutar movimientos** en el robot. Primero el **homing** envía al robot a la posición inicial [0,0,0], luego si tenemos una posición dada por los **Sliders** la enviamos al robot con el botón **enviar consigna**, y si queremos ejecutar una secuencia de comandos prefijada lo hacemos con **modo automático**.
Al final tendremos los controles del efector final y también la opción de mostrar el reporte.

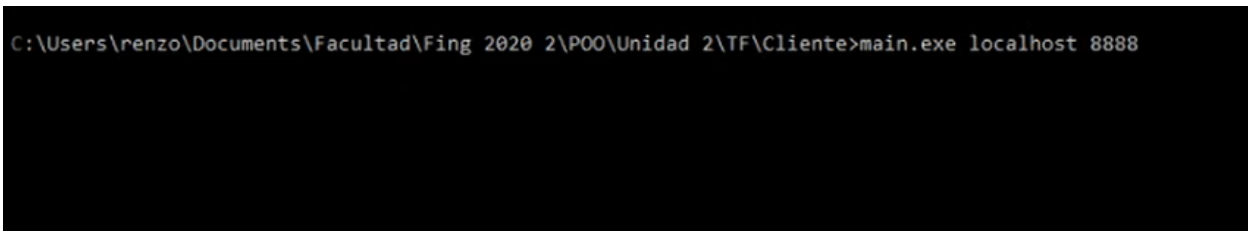
¿Cómo salir de la consola?

Comando **quit** para cerrar toda la aplicación

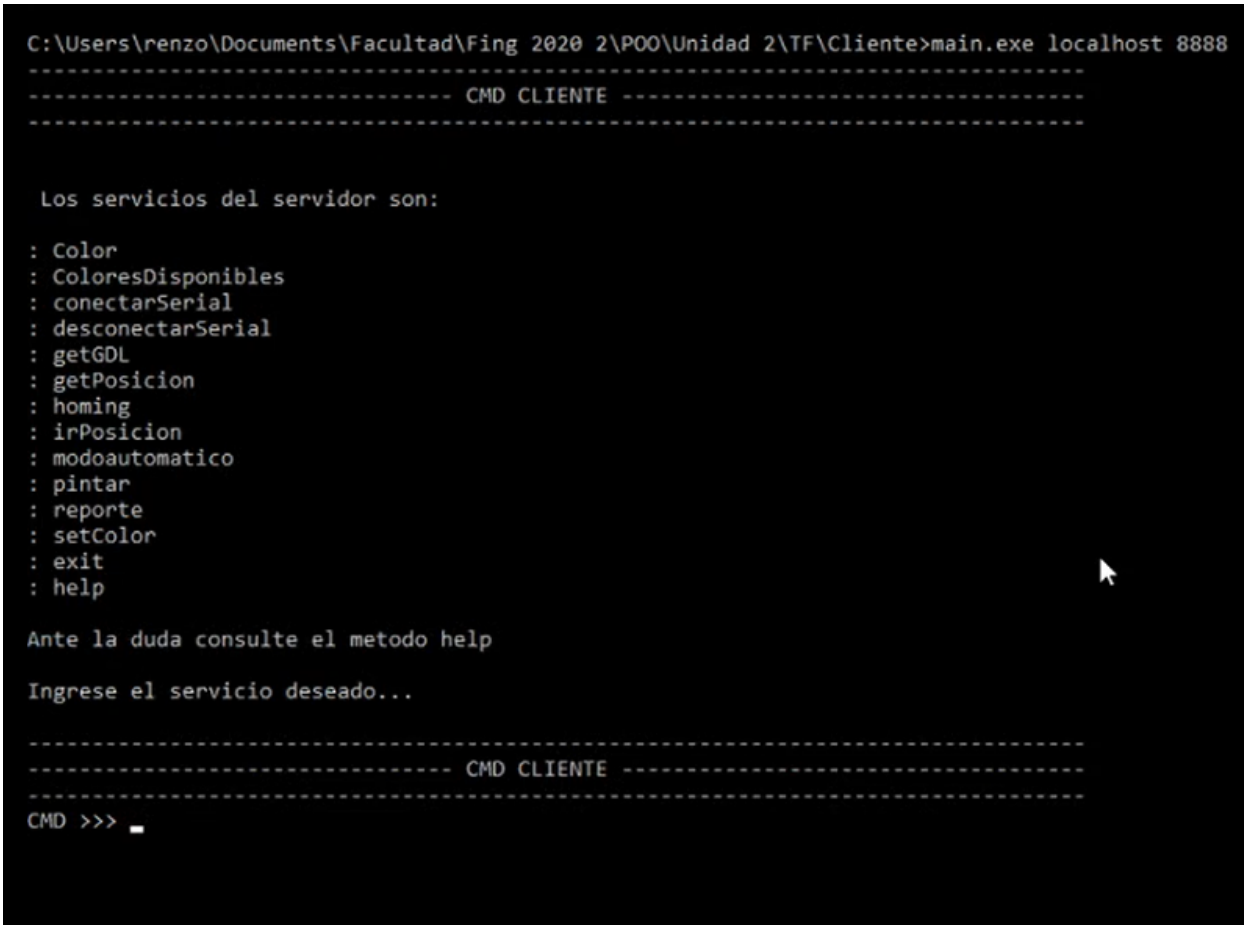
```
------------------------------------------------------------------------------
---------------------------------- CMD SERVIDOR ------------------------------
------------------------------------------------------------------------------
[CMD >>> quit                                                                 ]
------------------------------------------------------
Servidor ya detenido
None
Ejecucion consola terminada
(base) MacBook-Pro-de-Lucas:Servidor lucastrubiano$
```

Guia sobre el Cliente

Para lanzar el cliente ejecutamos main.exe seguido de la dirección IP del servidor y el puerto, como se ve en la siguiente imagen.



Si el servidor se encuentra activo nos aparecerá lo siguiente:



Vemos que se nos muestra todos los servicios disponibles los cuales son muy parecidos a los de la consola del lado servidor y además funcionan de manera muy similar. Para utilizar cualquiera de ellos solamente es necesario ingresar el comando deseado y en caso de dudas se puede recurrir al comando help.

Como vemos en la siguiente figura el comando help nos devuelve una lista de los comandos disponibles, y nos pide que ingresemos el comando del cual se requiere información.

```
--------------------------------------------------------------------------------
------------------------------- CMD CLIENTE ------------------------------------
--------------------------------------------------------------------------------
CMD >>> help
: Color
: ColoresDisponibles
: conectarSerial
: desconectarSerial
: getGDL
: getPosicion
: homing
: irPosicion
: modoautomatico
: pintar
: reporte
: setColor

Ingrese el nombre del comando
? Color

Ayuda : Devuelve el color establecido

--------------------------------------------------------------------------------
------------------------------- CMD CLIENTE ------------------------------------
--------------------------------------------------------------------------------
CMD >>>
```

El comando conectarSerial nos permite conectar y activar el robot por puerto serial. El puerto puede pasarse como argumento o se establece el que está por defecto.

```
--------------------------------------------------------------------------------
------------------------------- CMD CLIENTE ------------------------------------
--------------------------------------------------------------------------------
CMD >>> conectarSerial

Comando enviado exitosamente
Conectado [COM9]
Activado [COM9]
--------------------------------------------------------------------------------
```

En la siguiente imagen se puede apreciar las distintas formas de pasar los argumentos requerido por los servicios. En este caso el servicio setColor:

```
--------------------------------------------------------------------------------
CMD >>> setColor Rojo

Comando enviado exitosamente
El color establecido es: Rojo
--------------------------------------------------------------------------------
------------------------------- CMD CLIENTE ------------------------------------
--------------------------------------------------------------------------------
CMD >>> setColor

Ingrese el color
? Amarillo

Comando enviado exitosamente
El color establecido es: Amarillo
```

Por último se muestra lo desplegado por el comando reporte:

```
CMD >>> reporte

Comando enviado exitosamente
Estado del robot:Activado
Tiempo de ejecucion: 1102.1980624198914

Lista de comandos usados

Servidor  2020-11-22 12:11:16.925240   help
Servidor  2020-11-22 12:11:25.052450   iniciarServidor
Cliente  2020-11-22 12:18:34.511912   getcolor
Cliente  2020-11-22 12:18:43.458753   setcolor
Cliente  2020-11-22 12:18:49.638416   setcolor
Cliente  2020-11-22 12:19:11.366750   setcolor
Cliente  2020-11-22 12:22:01.950752   modoautomatico
Cliente  2020-11-22 12:22:22.408124   getcolor
Cliente  2020-11-22 12:22:31.304260   modoautomatico
Cliente  2020-11-22 12:23:17.970599   modoautomatico
Servidor  2020-11-22 12:23:27.379539   modoautomatico
Servidor  2020-11-22 12:23:32.663196   modoautomatico
Servidor  2020-11-22 12:25:44.609805   modoautomatico
Cliente  2020-11-22 12:26:05.168836   conectarSerial
Cliente  2020-11-22 12:29:00.456060   modoautomatico
Cliente  2020-11-22 12:29:38.640600   reporte
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
|    Modo    |            Hora            |    x     |    y     |    z     |    q1    |    q2    |    q3    |    Efector   |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-22 12:29:28.977120 |     0.10 |     0.10 |     0.10 |     0.79 |     1.04 |     0.36 |  No Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-22 12:29:30.873727 |    -0.10 |    -0.90 |     0.09 |    -1.68 |     0.41 |    -0.93 |     Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-22 12:29:31.962141 |     0.09 |     0.80 |     0.01 |     1.46 |     0.54 |    -0.93 |     Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-22 12:29:32.616956 |     0.08 |     0.07 |     0.01 |     0.72 |     1.50 |    -0.93 |     Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-22 12:29:33.271778 |     0.70 |     0.06 |    -0.06 |     0.09 |     0.68 |    -0.93 |     Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-22 12:29:34.644174 |    -0.60 |    -0.50 |     0.08 |    -2.45 |     0.48 |    -0.93 |     Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
| Automatico | 2020-11-22 12:29:35.245141 |     0.50 |     0.04 |     0.60 |     0.08 |    -0.34 |    -0.93 |  No Pintando |
|------------|----------------------------|----------|----------|----------|----------|----------|----------|--------------|
```

Conclusión:

Para resumir un poco el trabajo realizado, la orientación a objetos nos ayudó mucho ya que sin ese enfoque desarrollar el código hubiera sido demasiado complicado. También el uso de las librerías de XmlRpc nos permitieron comunicar lenguajes diferentes a través de mensajes que manejan la misma estructura. También hicimos uso de los Threads o hilos en python que permiten a varias partes del código correr en simultáneo.

En nuestro caso por cuestiones de tiempo no llegamos a implementar una interfaz del lado cliente pero nos hubiera gustado poder implementarla. Fue un proyecto en el que integramos herramientas tanto de esta materia como de otras y la posibilidad de hacerlo en lenguaje Python nos facilitó mucho esta integración entre contenidos de diferentes materias.

Referencias

- https://new.abb.com/products/robotics/es/robots-industriales/irb-460
- https://stackoverflow.com/
- DEITEL, H. M. y DEITEL, P. J. (2009): Cómo Programar en C/C++, 6ª edicion, Prentice-Hall
- Apuntes cátedra