6.1.2 EJERCICIO DE COMUNICACIÓN

Se pretende desarrollar una aplicación en C# que comunique simultáneamente con los dos robots del laboratorio 016 para que les permita intercambiar mensajes.

El primero de ellos (el más cercano a las mesas de estudio) tiene la dirección IP: 10.172.17.201, el segundo de ellos presenta la dirección IP: 10.172.17.200.

En el primero de los robots se desarrollará una aplicación básica de control que, con movimientos punto a punto, genere una figura geométrica fácil de identificar (por ejemplo, un rectángulo que puede estar en cualquier plano).

Para el segundo de los robots se deberá diseñar un programa que vaya siguiendo, con un pequeño retraso, los movimientos del primero.

La solución pasa por:

- Crear en el ordenador una aplicación que sea cliente simultáneamente de los dos robots.
- Que la aplicación en C# lea, cada pocos milisegundos, la variable global `$POS_ACT` del primero de los robots (es una variable de tipo `E6POS`).
- La procese si hubiera lugar (para crear utilidades como espejo en X...).
- Y se la escriba al segundo robot en una variable global cuyo nombre vosotros decidiréis.

Obviamente el segundo de los robots tendrá un programa que moviéndose punto a punto irá a la posición de la variable que, cada pocos milisegundos, está siendo refrescada, reproduciendo de esta forma el movimiento del primero.

Hay una serie de aspectos a tener en cuenta:

- La configuración de los robots hace que no tengan orientados apuntando a la misma dirección los ejes X e Y. En el segundo de los robots habrá que establecer una base adecuada si se desea que el movimiento sea copia uno del otro.
- El orden de poner en marcha las aplicaciones será:
  - En primer lugar, el primer robot (se recomienda hacerlo en modo automático después de haber realizado las pruebas que garanticen el correcto funcionamiento del programa).
  - Luego la aplicación en C#.
  - Por último, el segundo robot (se recomienda hacerlo en modo automático después de haber realizado las pruebas que garanticen el correcto funcionamiento del programa).

6.2 COMUNICACIÓN CON UN ROBOT DE UNIVERSAL ROBOTS

Este tipo de robots permiten comunicarse con ellos mediante múltiples protocolos:

- Interfaces primario y secundario.
- RPC.
- RTDE.
- Comunicación mediante "sockets" con el programa que se ejecuta en el robot.
- Modbus/TCP, versiones cliente y servidor.

En este caso, las explicaciones se van a centrar en el uso de servidor Modbus/TCP incluido en todas sus familias de robots.

En esta sección solo se van a exponer las particularidades del ejercicio de comunicaciones propuesto. Las explicaciones sobre cómo implementar el protocolo Modbus/TCP se recogen en el capítulo 7 (que habrá que leer antes de comenzar esta programación).

6.2.1 PARTICULARIDADES DE LA IMPLEMENTACIÓN DEL PROTOCOLO MODBUS/TCP EN EL ROBOT UR3

Los principales aspectos a reseñar son:

- En la implementación por defecto de Universal Robots, el servidor Modbus/TCP presenta un temporizador[41] que, si en 22 segundos aproximadamente, no se le envían desde el cliente ninguna petición de datos, desconecta automáticamente la conexión; y si se desea continuar comunicando habrá que volver a pasar por la fase de establecer la conexión TCP. Esto no suele ser muy relevante en programas que capturan cíclicamente información, pero sí en aquellos que lo hacen esporádicamente (o cuando se está en la fase de depuración del programa).
- El conjunto de funciones implementadas en el robot es relativamente corto, solo aquellas que permiten leer o escribir registros internos y leer o escribir salidas discretas.
- MUY IMPORTANTE: al ser el robot una estructura móvil, hay que tener mucho cuidado cuando se escribe[42] en una determinada variable, puede que conduzca a un movimiento que cause un accidente.

6.2.2 EJERCICIO DE COMUNICACIÓN

Se pretende crear una aplicación gráfica en C# que lea diversos registros del robot, como mínimo los relacionados con las posiciones de las articulaciones de los 6 ejes del robot, las 6 posiciones del TCP y algún registro más que se considere importante[43].

También se incluirá la lectura de algunas salidas discretas como puedan ser si el robot está encendido, si se ha parado por una señal de emergencia…

[41] Es raro encontrar este comportamiento en servidores de otros fabricantes y la norma Modbus no especifica la obligatoriedad de tener que implementar algo así, pero es lo que hay.

[42] Al leer variables no se pueden dar esos problemas.

[43] Ver sección 6.2.3.

Dicha información se plasmará en sendas cajas de texto que se irán actualizando cada pocos milisegundos.

Para llevar a cabo las comunicaciones se diseñará una clase, en C#, que implemente parte del protocolo Modbus/TCP. La clase a desarrollar deberá ser realizada originalmente por el alumno, no empleando librerías ni código fuente de terceros.

Para darle una mayor vistosidad, se podrá utilizar una librería, proporcionada por el profesor, que permitirá visualizar el movimiento del robot en el simulador de Coppelia Robotics (ver sección 6.2.4).

Además, se añadirá una funcionalidad que nos permita conocer el tiempo transcurrido entre que se le ha enviado al servidor Modbus/TCP del robot una petición de datos (p.ej. la de leer la posición del TCP) y el momento en que ha retornado la respuesta.

6.2.3 DOCUMENTACIÓN ORIGINAL DE UNIVERSAL ROBOTS SOBRE EL MAPA DE REGISTROS DEL ROBOT QUE SON ACCESIBLES MEDIANTE MODBUS/TCP

Registros Internos:

Dirección	Pre 3.0	3.0	3.1	R	W	
0	x	x	x	*		Inputs, bits 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
1	x	x	x	*	*	Outputs, bits 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
2	x	x	x		*	SetOutputsBitsMask 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
3	x	x	x		*	ClearOutputsBitsMask 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
4	x	x	x	*		Analog input 0 (0-65535)
5	x	x	x	*	*	Analog input 0 domain e {0=current[mA], 1=voltage[mV]}
6	x	x	x	*		Analog input 1 (0-65535)
7	x	x	x	*	*	Analog input 1 domain e {0=current[mA], 1=voltage[mV]}
8	x	x	x	*		Analog input 2 (tool) (0-65535)
9	x	x	x	*	*	Analog input 2 (tool) domain e {0=current[mA], 1=voltage[mV]}
10	x	x	x	*		Analog input 3 (tool) (0-65535)
11	x	x	x	*	*	Analog input 3 (tool) range e {0=current[mA], 1=voltage[mV]}
16	x	x	x	*	*	Analog output 0 output (0-65535)
17	x	x	x	*	*	Analog output 0 output domain e {0=current[mA], 1=voltage[mV]}
18	x	x	x	*	*	Analog output 1 output (0-65535)
19	x	x	x	*	*	Analog output 1 output domain {0=current[mA], 1=voltage[mV]}
20	x	x	x	*	*	Tool output voltage (V) e {0V, 12V, 24V}
21	x	x	x	*		Tool digital input bits

Dirección	Pre 3.0	3.0	3.1	R	W	
22	x	x	x	*	*	Tool digital output bits
24	x	x	x	*		Euromap67 input bits (0-15)
25	x	x	x	*		Euromap67 input bits (16-32)
26	x	x	x	*		Euromap67 output bits (0-15) (read only!)
27	x	x	x	*		Euromap67 output bits (16-32) (read only!)
28	x	x	x	*		Euromap 24V voltage
29	x	x	x	*		Euromap 24V current
30	-	x	x	*		Configurable inputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
31	-	x	x	*	*	Configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
32	-	x	x		*	Bit mask configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
33	-	x	x		*	Clear configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
34-127	x	x	x			Reserved for future system variables
128-255	x	x	x	*	*	General purpose 16 bit registers
256-	x	x	x	*		Robot state (ver en filas inferiores)
512-	x	x	x	*		Reserved
768-	x	x	x	*		Tool states (ver en filas inferiores)
1024-	x	x	x	*		Reserved
2048-	x	x	-	*		RT Machine control (ver en filas inferiores)
	Robot state					
256	x	x	x			Controller version high number
257	x	x	x			Controller version low number
258	x	x	x			Robot mode: CB3 and 3.1: Disconnected=0, Confirm_safety=1, Booting=2, Power_off=3, Power_on=4, Idle=5, Backdrive=6, Running=7 CB2: No_controller=-1, Running=0, Freedrive=1,Ready=2, Initializing=3, Security_stoppde=4, Emergency_stopped=5, Fault=6, Not_connected=8, Shoutdown=9
260	x	x	x			isPowerOnRobot
261	x	x	x			isSecurityStopped
262	x	x	x			isEmergencyStopped
263	x	x	x			isTeachButtonPressed
264	x	x	x			isPowerPuttonPressed
265	x	x	x			isSafetySignalSuchThatWeShouldStop

Dirección	Pre 3.0	3.0	3.1	R	W	
270	x	x	x			Base joint angle (in mrad)
271	x	x	x			Shoulder joint angle (in mrad)
272	x	x	x			Elbow joint angle (in mrad)
273	x	x	x			Wrist1 joint angle (in mrad)
274	x	x	x			Wrist2 joint angle (in mrad)
275	x	x	x			Wrist3 joint angle (in mrad)
280	x	x	x			Base joint angle velocity (in mrad/s)
281	x	x	x			Shoulder joint angle velocity (in mrad/s)
282	x	x	x			Elbow joint angle velocity (in mrad/s)
283	x	x	x			Wrist1 joint angle velocity (in mrad/s)
284	x	x	x			Wrist2 joint angle velocity (in mrad/s)
285	x	x				Wrist3 joint angle velocity (in mrad/s)
290	x	x	x			Base joint current (in mA)
291	x	x	x			Shoulder joint current (in mA)
292	x	x	x			Elbow joint current (in mA)
293	x	x	x			Wrist1 joint current (in mA)
294	x	x	x			Wrist2 joint current (in mA)
295	x	x	x			Wrist3 joint current (in mA)
300	x	x	x			Base joint temperature (in C)
301	x	x	x			Shoulder joint temperature (in C)
302	x	x	x			Elbow joint temperature (in C)
303	x	x	x			Wrist1 joint temperature (in C)
304	x	x	x			Wrist2 joint temperature (in C)
305	x	x	x			Wrist3 joint temperature (in C)
310	x	x	x			Base joint mode List of Joint Modes: JOINT_SHUTTING_DOWN_MODE = 236; JOINT_PART_D_CALIBRATION_MODE = 237; JOINT_BACKDRIVE_MODE = 238; JOINT_POWER_OFF_MODE = 239; JOINT_NOT_RESPONDING_MODE = 245; JOINT_MOTOR_INITIALISATION_MODE = 246; JOINT_BOOTING_MODE = 247; JOINT_PART_D_CALIBRATION_ERROR_MODE = 248;

Dirección	Pre 3.0	3.0	3.1	R	W	
						JOINT_BOOTLOADER_MODE = 249; JOINT_CALIBRATION_MODE = 250; JOINT_FAULT_MODE = 252; JOINT_RUNNING_MODE = 253; JOINT_IDLE_MODE = 255;
311	x	x	x			Shoulder joint mode
312	x	x	x			Elbow joint mode
313	x	x	x			Wrist1 joint mode
314	x	x	x			Wrist2 joint mode
315	x	x	x			Wrist3 joint mode
320	-	-	x			Base joint revolution count (number of full turns, typically 0 or 1)
321	-	-	x			Shoulder joint revolution count
322	-	-	x			Elbow joint revolution count
323	-	-	x			Wrist1 joint revolution count
324	-	-	x			Wrist2 joint revolution count
325	-	-	x			Wrist3 joint revolution count
	TCP					
400	x	x	x			TCP-x in tenth of mm (in base frame)
401	x	x	x			TCP-y in tenth of mm (in base frame)
402	x	x	x			TCP-z in tenth of mm (in base frame)
403	x	x	x			TCP-rx in mrad (in base frame)
404	x	x	x			TCP-ry in mrad (in base frame)
405	x	x	x			TCP-rz in mrad (in base frame)
410	x	x	x			TCP-x speed in mm/s (in base frame)
411	x	x	x			TCP-y speed in mm/s (in base frame)
412	x	x	x			TCP-z speed in mm/s (in base frame)
413	x	x	x			TCP-rx speed in mrad/s (in base frame)
414	x	x	x			TCP-ry speed in mrad/s (in base frame)
415	x	x	x			TCP-rz speed in mrad/s (in base frame)
420	x	x	x			TCP-x offset in mm (in tool frame)
421	x	x	x			TCP-y offset in mm (in tool frame)
422	x	x	x			TCP-z offset in mm (in tool frame)
423	x	x	x			TCP-rx offset in mrad (in tool frame)

Dirección	Pre 3.0	3.0	3.1	R	W	
424	x	x	x			TCP-ry offset in mrad (in tool frame)
425	x	x	x			TCP-rz offset in mrad (in tool frame)
450	x	x	x			Robot current (in mA)
451	x	x	x			I/O current (in mA)
	Tool state					
768	x	x	x			Tool state
769	x	x	x			Tool temperature (in C)
770	x	x	x			Tool current (in mA)
	GUI state					
1024	x	x	x			Reserved
	RTMachine					
2048	x	x	-		*	Split time (latches the actual time to the registers 2049-2053) calculates the time from last time the controller was restarted
2049	x	x	-	*		Milliseconds
2050	x	x	-	*		Seconds
2051	x	x	-	*		Minutes
2052	x	x	-	*		Hours
2053	x	x	-	*		Days

Tabla 6.1.- Registros internos accesibles en el robot UR3.

Salidas discretas:

	Pre 3.0	3.0	3.1	R	W	
0-15	x	x	x	*		[BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
16-31	x	x	x	*	*	Outputs, bits 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
32-47	x	x	x		*	SetOutputsBitsMask 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
48-63	x	x	x		*	ClearOutputsBitsMask 0-15 [BBBBBBBBTTxxxxxx] x=undef, T=tool, B=box
64-79	x	x	x	*		Euromap67 input bits (0-15)
80-95	x	x	x	*		Euromap67 input bits (16-32)
96-111	x	x	x	*		Euromap67 output bits (0-15) (read only!)

	Pre 3.0	3.0	3.1	R	W	
112-127	x	x	x	*		Euromap67 output bits (16-32) (read only!)
128-135	x	x	x	*		Configurable inputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
136-143	x	x	x	*	*	Configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
144-151	x	x	x		*	Bit mask configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
152-159	x	x	x		*	Clear configurable outputs, bits [BBBBBBBBxxxxxxxx] x=undef, T=tool, B=box
260	x	x	x	*		isPowerOnRobot
261	x	x	x	*		isProtectiveStopped
262	x	x	x	*		isEmergencyStopped
263	x	x	x	*		isTeachButtonPressed
264	x	x	x	*		isPowerPuttonPressed
265	x	x	x	*		isSafetySignalSuchThatWeShouldStop

Tabla 6.2.- Salidas discretas accesibles en el robot UR3.

6.2.4 FUNCIONES PARA CONECTARSE Y ENVIAR INFORMACIÓN AL SIMULADOR DE COPPELIA ROBOTICS

Solo puede funcionar en aquellos ordenadores donde se haya instalado la versión educativa de CoppeliaSim. En concreto el ejecutable debe estar en la carpeta: "`C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\CoppeliaSim.exe`".

Para poder utilizarlo se deben tener en cuenta los siguientes requisitos:

- Se debe crear una aplicación en WPF de 64 bits. Para ello se seguiremos los siguientes pasos:
  - En primer lugar, accederemos a las propiedades del proyecto, esto se realiza pulsando con el botón derecho del ratón en el nombre del proyecto y accediendo a "Propiedades" del menú desplegable.
  - En ese momento se abre una nueva ventana, y en la sección "Compilación" del menú asociado, se elegirá como "Plataforma de destino": x64.

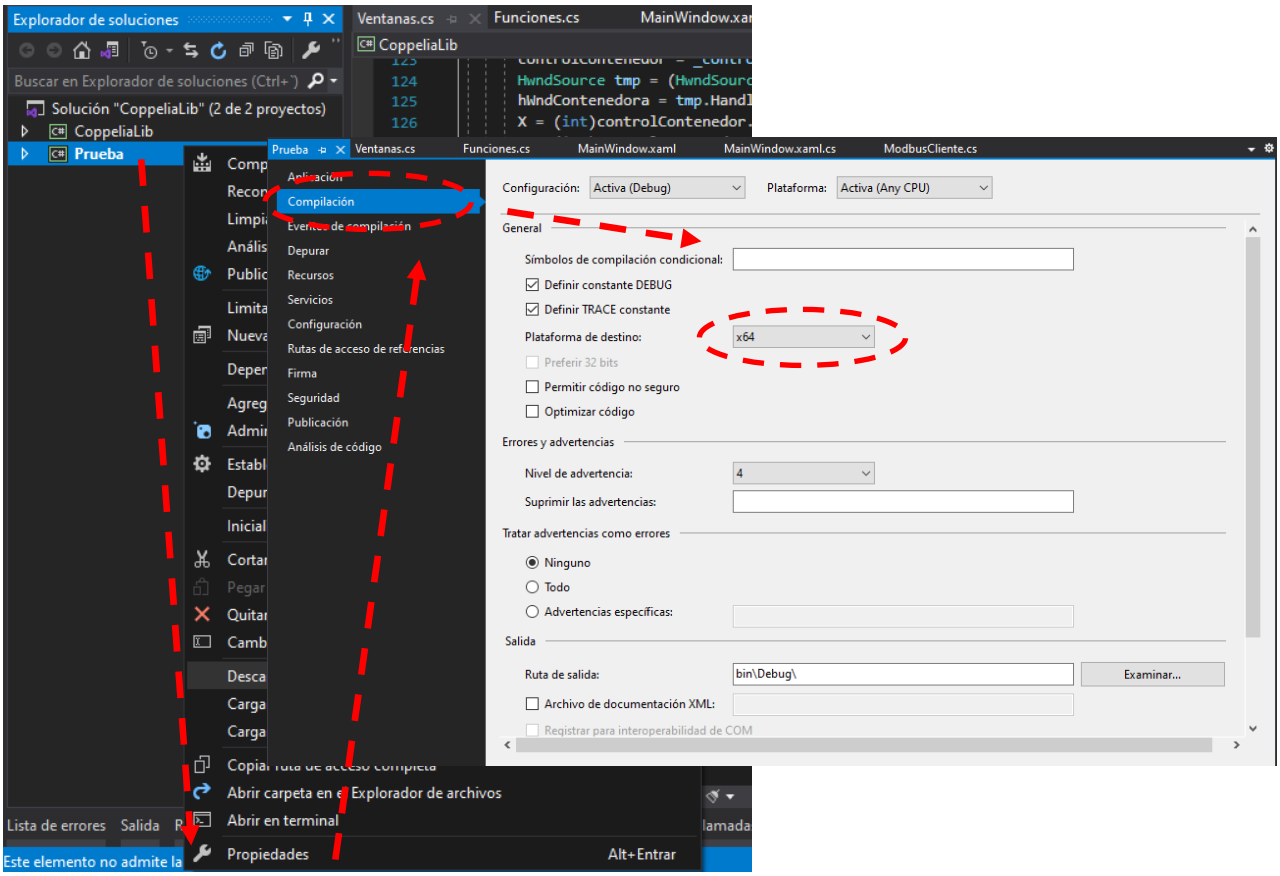

Figura 6.1.- Generación de una aplicación WPF de 64 bits.

- Se debe disponer de la librería "`CoppeliaLib.dll`" (está en Alud). Y con ella seguir los siguientes pasos.
  - Se la debe depositar en la carpeta donde se encuentra el programa ejecutable que creamos con Visual Studio (puede ser algo así como: "`<Ruta donde se encuentra en proyecto>\<nombre del proyecto>\bin\Debug`").

- Se debe incluir la librería en el proyecto que estemos desarrollando. Para ello se accede con el botón derecho del ratón al menú de "`Referencias`" del proyecto.

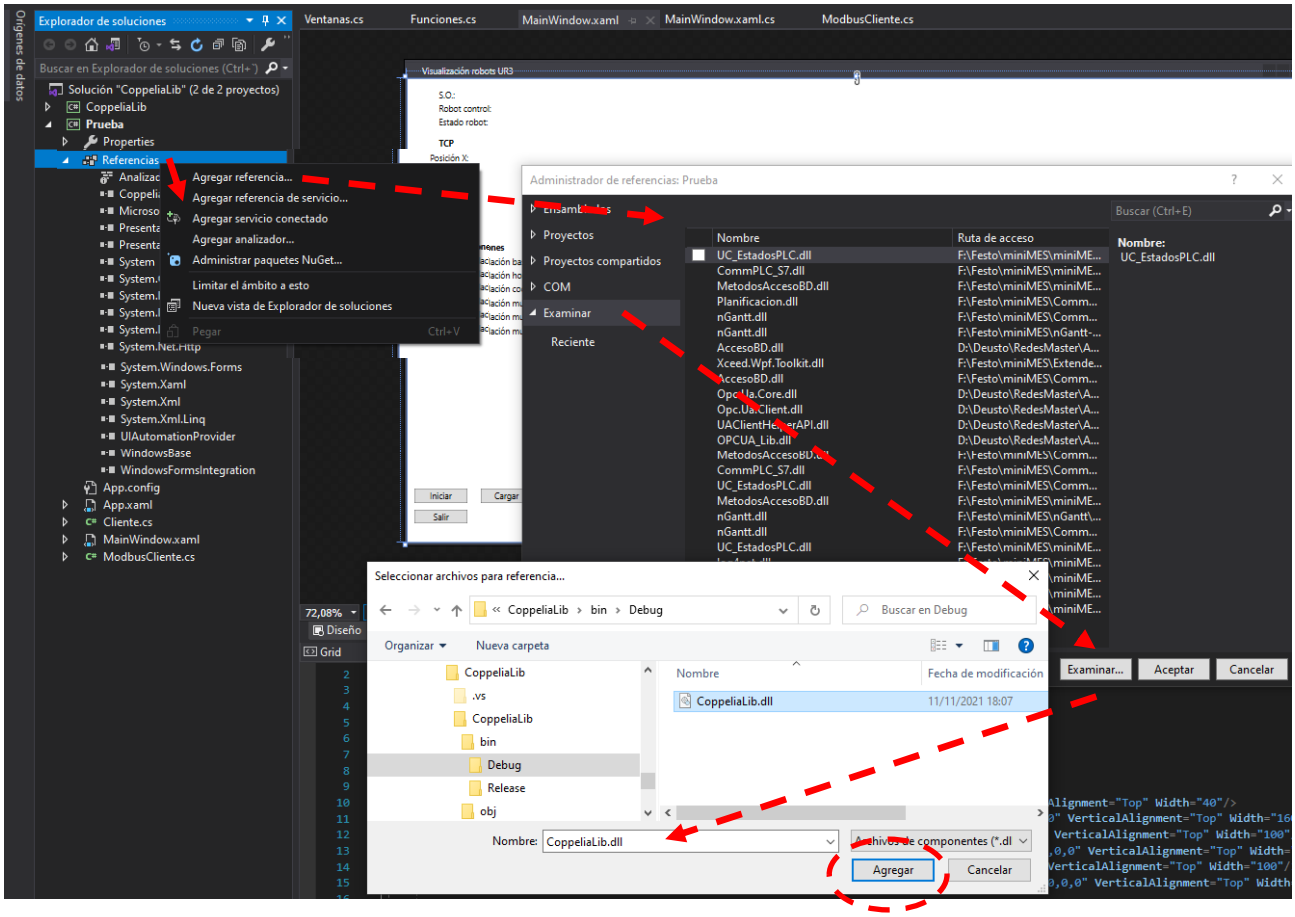

Figura 6.2.- Forma de incluir una librería en el proyecto.

- La clase incluida en este proyecto se denomina "`FunCoppelia`" y está en el espacio de trabajo "`CoppeliaLib`"[44]. Es una clase diseñada para trabajar en el entorno WPF .NET Framework 4.7.2.

- Dicha clase presenta los siguientes métodos:

  - `FunCoppelia()`.

    Es el constructor y no tiene argumentos. Un ejemplo de su uso podría ser:

```
//-----------------------------------------------------
private FunCoppelia sim = null;

//-----------------------------------------------------
public MainWindow()  {

  InitializeComponent();
  ...
  sim = new FunCoppelia();
  ...
}
```

[44] Para facilitar las cosas, este espacio de trabajo lo incluiremos con un `using` al principio del programa.

- `int inicializarCoppelia(System.Windows.Controls.Panel controlContenedora).`

  Se le pasa la referencia de un control derivado de clase `Panel` (por ejemplo, un control de la clase `Canvas`). Este control estará dimensionado, en la ventana del proyecto (y sin formar parte de otra unidad que lo integre), de forma que en él se mostrará la ventana del simulador de Coppelia.

  En XAML, este canvas podría ser algo similar a:

  ```
  <Canvas x:Name="cv_Contenedor" HorizontalAlignment="Left" Height="651" Margin="325,10,0,0" VerticalAlignment="Top" Width="957" />
  ```

  Si la función retorna 0 es que ha terminado correctamente.

  Un ejemplo de su uso podría ser:

  ```
  //-----------------------------------------------
  private void btn_Iniciar_Click(object sender, RoutedEventArgs e)  {

      res = sim.inicializarCoppelia(cv_Contenedor);
      res = sim.cargarCoppelia("<Ruta completa al fichero>/Inicio1.ttt");
      ...
  ```

- `int cargarCoppelia(string escena).`

  Una vez inicializada la ventana, se le carga al simulador el fichero con la escena a visualizar. En este caso se proporciona al alumno una escena formada por dos robots y que se encuentra en el fichero `Inicio1.ttt`. IMPORTANTE: hay que proporcionarle la ruta completa donde se encuentra el fichero.

  Si la función retorna 0 es que ha terminado correctamente.

  Un ejemplo de su uso se muestra en la función anterior.

- `int ponPosicionRobotUR3(int n, float[] girosEjes).`

  Cuando se quiera modificar la posición de alguno de los robots, se llamará a este método. Pasándole en el primer parámetro el número del robot a mover (1 o 2). Y en el segundo parámetro un array de 6 números reales `float` con las posiciones de las articulaciones expresadas en radianes.

  Los valores de los números reales que se le pasan deberán estar comprendidos en el margen $-\pi \rightarrow +\pi$.

  Hay que reseñar que en el simulador de Coppelia la posición de Inicio se corresponde con todas las articulaciones en 0 radianes[45].

  Si la función retorna 0 es que ha terminado correctamente.

  Un ejemplo de su uso podría ser:

  ```
  float[] ang=new float[6];
  ...
  res = sim.ponPosicionRobotUR3(1,ang);
  ```

[45] En el robot del laboratorio 006, la posición de Inicio se corresponde con todas las articulaciones en 0 radianes excepto la del hombro y la muñeca 1 que se encuentran en $3\pi/2$. Así pues, para que coincidan las posiciones real y simulada, a esas dos articulaciones habrá que restarles $3\pi/2$ y, posteriormente realizar el ajuste al margen comprendido entre $-\pi \rightarrow +\pi$.

- `int terminarCoppelia().`

  La llamada a este método cierra Coppelia y destruye la ventana donde se mostraba.

  Se debe salir siempre empelando este método, de otra forma es posible que la aplicación CoppeliaSim.exe siga ejecutándose y dé problemas en el siguiente arranque[46].

  Si la función retorna 0 es que ha terminado correctamente.

  Un ejemplo de uso:

```
private void btn_Finalizar_Click(object sender, RoutedEventArgs e)  {
    int res;

    res= sim.terminarCoppelia();
    sim = null;
    return;
}
```

- También deberemos disponer de la librería "`remoteAPI.dll`", esta librería solo hay que depositarla en la carpeta donde se encuentra el programa ejecutable que creamos con Visual Studio. No es necesario incluirla en las referencias del proyecto.

Una vez arrancada la aplicación, la ventana principal podría tener una apariencia similar a:

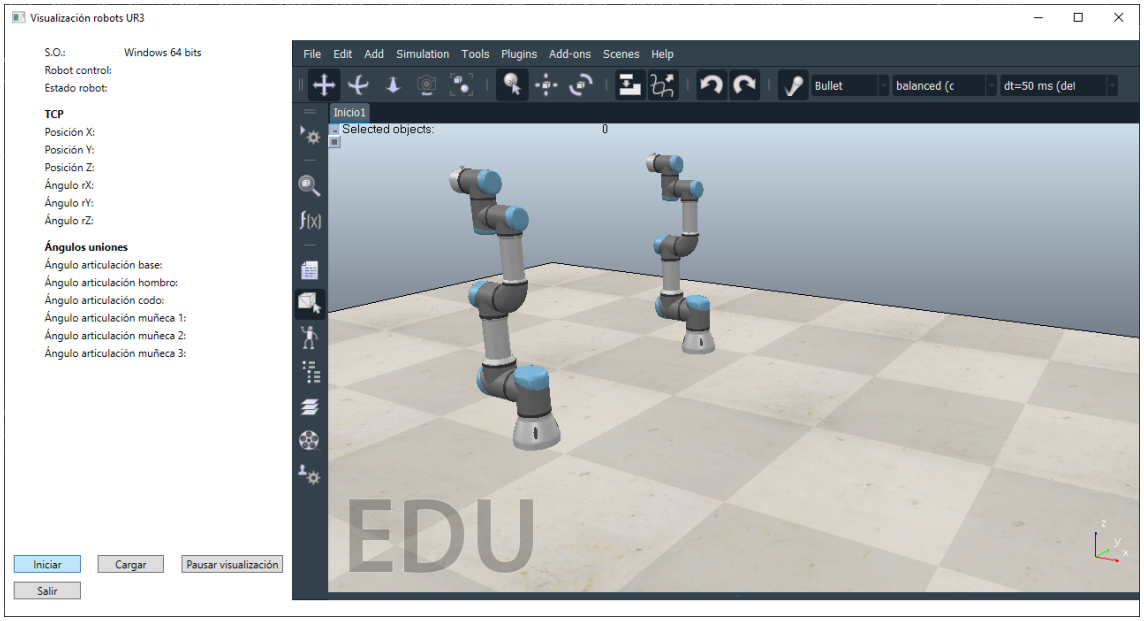

Figura 6.3.- Apariencia de la pantalla principal.

[46] En este caso se debe finalizar el programa CoppeliaSim.exe desde el administrador de tareas.