

**EV\_4\_1\_Identificar las interfaces de salida de robots industriales.**

Alumno: Fonseca Camarena Jonathan

Ingeniería Mecatrónica 6-A

Matricula 17311397

Tlajomulco de Zúñiga, Jalisco 26/06/2019

**El sistema estándar para la automatización:**

Una interfaz de máquina humana (HMI) se conecta a un controlador lógico programable (PLC) independiente. Estos sistemas han estado en vigor durante décadas entre la comunidad de automatización industrial. Ayudan al usuario a realizar un seguimiento de su sistema robot con una vista del panel PLC o interfaz HMI. Estos dispositivos de operador están diseñados para supervisar y controlar el sistema. Los operadores pueden indicar el estado del sistema, ajustar los parámetros de la aplicación, las posiciones de entrada y salida o ejecutar programas maestros.

**Controlador lógico programable (PLC)**

Los PLC son pequeños ordenadores digitales que controlan las posiciones de entrada y salida para asegurarse de que todos los componentes del sistema robot, como periféricos, robots, tocadiscos, etc., trabajen juntos en armonía.

[](https://www.robots.com/parts)

**Interfaz hombre-máquina (HMI)**

Una HMI es el panel de control que consta de un teclado numérico para introducir datos y una pantalla de control. Las estaciones de control del operador HMI se utilizan para la programación y la recopilación de datos. Las interfaces HMI permiten a las empresas ejecutar programas maestros y controlar posiciones de entrada/salida.

La tecnología está avanzando rápidamente y se están utilizando más unidades HMI/PLC combinadas, con una pantalla táctil todo en uno y un controlador lógico.

La combinación de un sistema PLC y HMI se está volviendo más popular y fácilmente disponible a medida que la tecnología de hardware y software continúa avanzando rápidamente. Todos los componentes del sistema de control están evolucionando rápidamente hasta el punto de una mayor funcionalidad donde un dispositivo combinado es una alternativa convincente.

**Beneficios de un sistema combinado:**

La tecnología combinada HMI-PLC se está convirtiendo en una alternativa más popular, ya que es menor en costos, causando la reducción de equipos, mientras que también facilita la funcionalidad del sistema. La inteligencia de la tecnología se está disparando a tasas exponenciales que está permitiendo a la mecánica de redes industriales interactuar a un nivel más avanzado y remoto. Además, las unidades combinadas ocupan menos espacio, minimiza el cableado, utiliza una sola fuente de alimentación y solo requiere una conexión de red.

**Beneficios de un sistema separado:**

En este momento, los sistemas separados permiten la selección de una gama más amplia de procesadores y capacidades de memoria mejoradas. Este sistema independiente también está disponible para aplicaciones de seguridad. Además, el control de movimiento y las comunicaciones son más extensos que una unidad combinada. El recuento de E/S y la E/S analógica también son más abundantes.

**Teach pendant para robot industrial**

El teach pendant en robótica industrial es un tipo de interfaz HMI diseñada para la programación y verificación de los programas a ejecutar por parte del robot industrial. De los diseños anteriores basados en un display que solo mostraba una línea de instrucciones, se ha pasado a displays que muestran menús desplegables.

Esta interfaz acoge la tarea del operario en robótica que debe programar el robot en la tarea industrial programada, por lo que la interacción entre el operario, el robot y diversos elementos de automatización lleva consigo la mejora de la interfaz en cuanto al manejo. Aquí se mezcla la aplicación de normativas de seguridad en máquinas junto al diseño ergonómico eficaz.

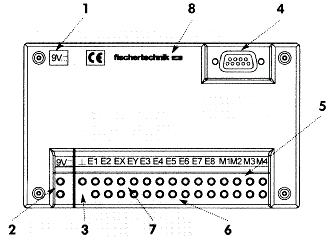
Una de las últimas funcionalidades añadidas a los teach pendant de robots industriales es la posibilidad de considerar la situación de pánico. Habitualmente el operario disponía de la palanca dead man, palanca que debía mantenerse presionada en el teach pendant mientras se hacía mover cada uno de los ejes del robot. Al dejar de presionar dicha palanca, el robot dejaba de moverse para prevenir una posible colisión entre operario programador y el robot. Considerando que, en una situación de emergencia, la tendencia del operario en situación de pánico es apretar con más fuerza dicha palanca, se descubrió experimentalmente que seguían produciéndose accidentes, por lo que a las dos posiciones anteriores se añade un tercer estado, el de pánico, de forma que cuando el operario aprieta desmesuradamente la palanca, el sistema interpreta que hay una inminente situación de accidente, por lo que se bloquea también los movimientos del robot industrial.

**Tablet PC.**

Las Tablet PC’s son equipos portátiles conectados al equipo principal mediante un sistema Wireless, y que permite que el operario, moviéndose libremente en planta, pueda acceder a la información, adquirirla, tratarla y compartirla o enviarla a la aplicación gráfica que se encuentra en la sala de control principal. El dispositivo como puede observarse en la siguiente figura, se trata básicamente de un panel de dimensiones más que aceptables (unas 14’, es decir, las mismas dimensiones que las antiguas pantallas de ordenadores de sobremesa) y suficientemente robustos para soportar entornos industriales. Su principal hándicap es su ergonomía, pero no tanto por lo que se refiere a su interfaz, porque de hecho funciona como un ordenador de sobremesa con una edición especial de Windows XP y por lo tanto dispone del mismo entorno gráfico que cualquier dispositivo SCADA de la aplicación, sino por el hecho de no disponer de teclado. Por eso, las soluciones que propone el fabricante se encaminan a asemejar al máximo la Tablet PC a un cuaderno, en el que se puedan tomar anotaciones mediante los clásicos bolígrafos de agenda electrónica.

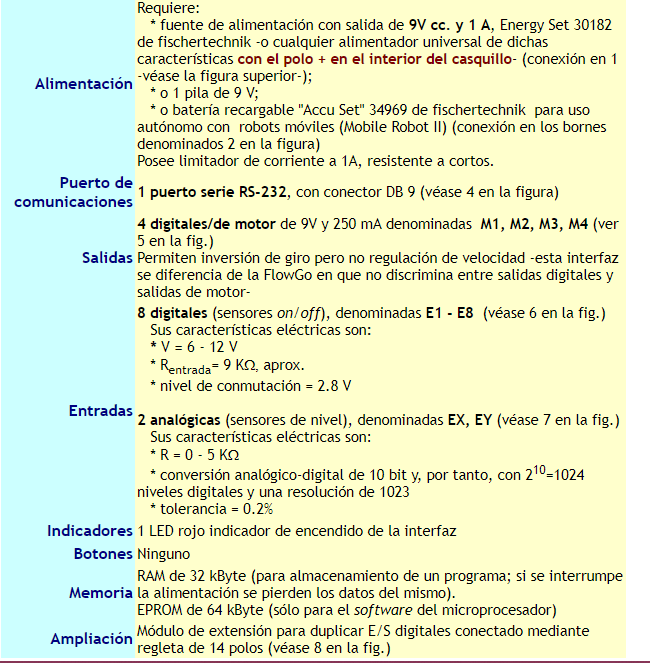
Para la introducción de parámetros, la Tablet PC permite introducir información mediante el bolígrafo o bien mediante la propia escritura del operador y que esta información sea interpretada. Por ejemplo, en la figura siguiente se puede observar como el dispositivo permite cambiar el valor de algún parámetro del sistema introduciendo el número 60 y que debe ser reconocido automáticamente por la aplicación de la Tablet

En lo que se refiere a la red inalámbrica cabe destacar la utilización de la tecnología Wi-Fi tan extendida en la actualidad en múltiples aplicaciones tanto profesionales como de gran consumo. La Tablet PC dispone de otras características ergonómicas como el cambio de orientación de la información en pantalla, es decir, permite pasar de formato vertical a formato apaisado. Esto puede ser útil porque puede ser más cómodo utilizar la Tablet en modo vertical cuando el operador se desplaza y en modo horizontal cuando se instala en su soporte.

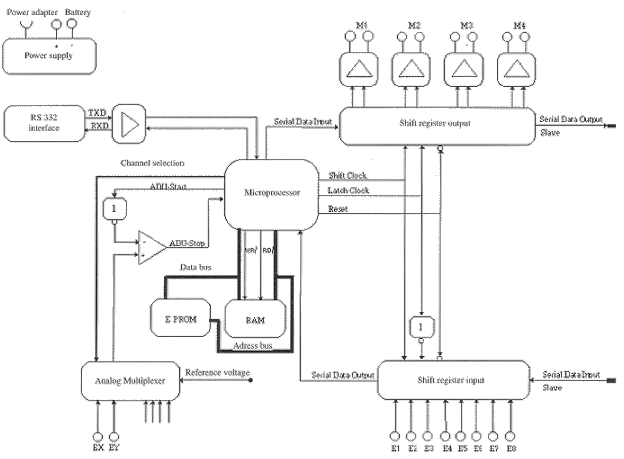
**La interfaz de control inteligente interface.**

Nos referiremos ahora a la interfaz de control (inteligent interface de fischertechnik), cuyo esquema se muestra en la figura siguiente:

Sus características principales son, para el modelo 30402:



En la siguiente figura se puede observar un diagrama de bloques de la composición interna de la interfaz:



Puede funcionar en dos modos:

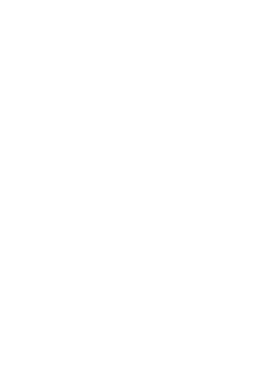
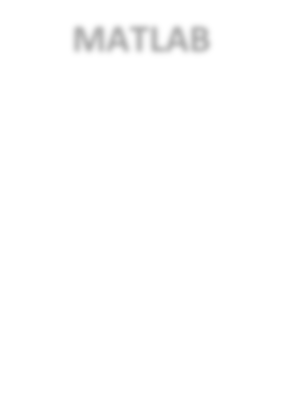
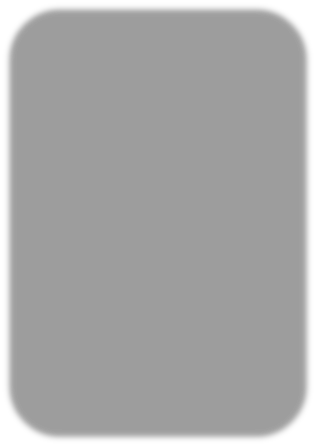
* modo en línea (online, gobernada en todo momento conectada al PC por medio del puerto de comunicaciones), o
* modo descarga: en este modo el procesamiento del programa tiene lugar únicamente en el microprocesador de la interfaz, estando ésta desconectada del PC. En una primera fase se escribe el programa de control en el PC, luego se descarga en la interfaz y, finalmente, ésta puede cortar el contacto con el PC (de modo que el cable serie puede ser desconectado). La memoria de esta interfaz es de tipo RAM, de modo que, si la interfaz es apagada, el programa descargado desaparece de la memoria.

La forma natural de programar la interfaz es el lenguaje LLWin, pero en modo "on-line" puede ser utilizada mediante cualquier lenguaje de programación. Los detalles acerca de este proceso se pueden encontrar en. En dicha página figuran algunos drivers diseñados para diferentes lenguajes de programación.

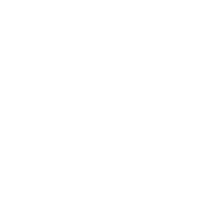
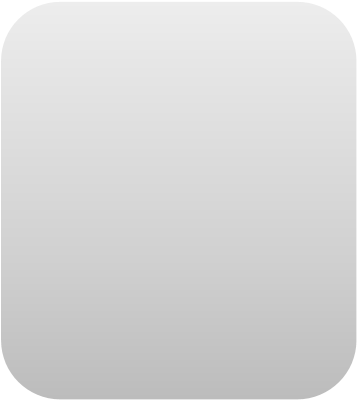
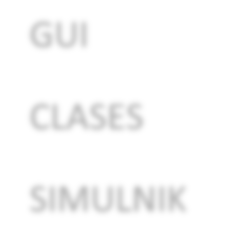
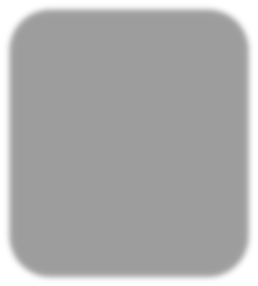
***Interfaces de salida de comunicación.***

***Visión general de las interfaces desarrolladas***

En este punto hablaremos en detalle de las interfaces desarrolladas para conseguir la comunicación con el brazo Robot. Se han realizado tres interfaces a través de tres herramientas de Matlab. En la siguiente figura observamos el bloque correspondiente y las tres herramientas mencionadas anteriormente:



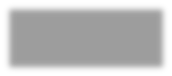
**MATLAB**



GUI

CLASES

SIMULNIK



**CLIENTE**

**FIGURA 3.1** Esquema básico de los bloques principales

La primera de la que hablaremos será la **interfaz GUI**. Una interfaz gráfica GUI tiene como característica principal que es *intuitiva*. Una persona que desconozca el funcionamiento de este tipo de interfaces podrá realizar fácilmente un uso de ella después de recibir unas breves instrucciones. También facilita al usuario el manejo de los procedimientos. Otra de sus ventajas es que proporciona mecanismos estándar de control, como ventanas y cuadros de diálogo.

Por tanto, este tipo de interfaces permiten al usuario interactuar con el sistema de una forma más fácil que otros sistemas.

Por otro lado, tiene alguna que otra desventaja, cómo su dificultad de desarrollo frente a otro tipo de interfaces o también otro punto a tener en cuenta a la hora de elegir este tipo de interfaz para desarrollar una aplicación, es que utiliza demasiados recursos del sistema y puede no interesarnos por complejidad y coste.

También debemos mencionar un problema importante que tenemos con esta interfaz, que no nos sirve para realizar un control automático del robot. Simplemente nos sirve para enviar órdenes específicas al brazo robótico para que éste realice ciertos movimientos o posicione el efector final en una posición concreta.

La segunda interfaz que describiremos, será la **interfaz a través de una clase de Matlab.** Como ya hemos descrito anteriormente, un Objeto es una entidad que contiene información y un conjunto de acciones que operan sobre los datos.

Para que un objeto realice una de sus acciones, se le manda una petición de invocación a una función que pertenece a un objeto en particular. Por tanto, podemos decir que la ventaja principal de este tipo de interfaces a través del desarrollo de clases en Matlab es la *encapsulación* de datos y operaciones que presenta y por tanto la inclusión en otros programas de Matlab (ficheros .m).

Otra de sus ventajas, es que la escritura del código es más fácil que la de otras interfaces, debido en gran parte a que las propias definiciones de la clase nos estructuran el código, por lo que si hemos desarrollado una clase con muchos objetos o métodos, no tendremos problema en manejar un código extenso.

Por otro lado, entre las desventajas que presenta podemos destacar, que el hecho de tener un único bloque de programa puede complicarnos sobre todo la detección de fallos y errores.

Por último, nos queda comentar **la interfaz desarrollada a través de un bloque de Simulink**. Una de sus ventajas más destacables en general, es que posee un entorno interactivo con un conjunto de librerías con bloques personalizables que permiten simular, implementar y probar una serie de sistemas variables con el tiempo.

Como desventaja (en comparación con las demás interfaces), podríamos destacar que no es tan sencilla de manejar y entender como aquellas, debido a la configuración de los parámetros de los bloques y de los parámetros de simulación.

***Interfaz a través de GUI***

La interfaz desarrollada a través de la herramienta de Matlab GUIDE, es uno de los puntos clave.

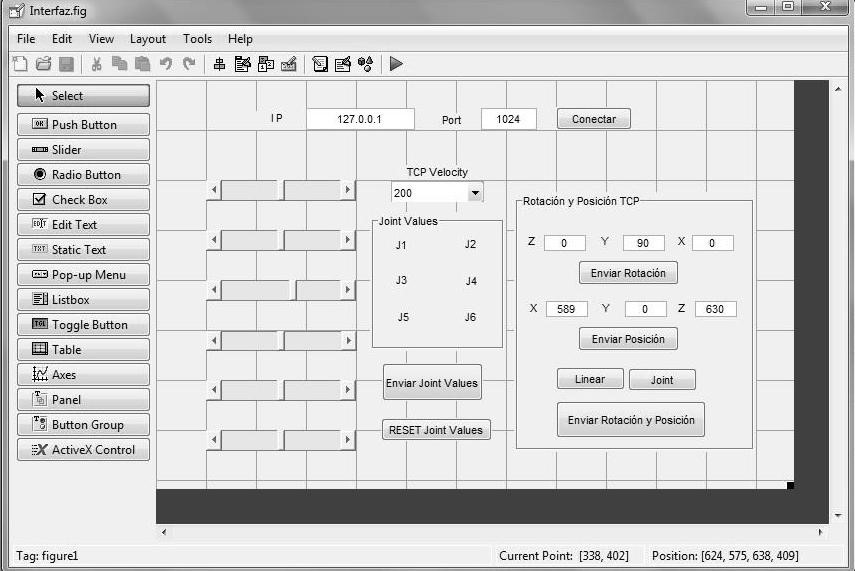
Ya conocemos el formato de datos que el cliente enviará al servidor (datagramas), pasaremos a explicar en detalle la interfaz y el entorno de funcionamiento de la misma.

Nuestra Interfaz Gráfica GUI, se compone de dos archivos: un archivo denominado Interfaz, en el que encontramos todo el código referente a la programación de los controles y el archivo Interfaz.fig, en el que encontramos los controles propiamente dichos.

Analizaremos los dos archivos a la par, ya que van ligados el uno al otro; es decir, si ponemos un control en nuestro archivo.fig en blanco, se generará una serie de instrucciones referentes a ese control seleccionado.

En la imagen siguiente podemos observar la apariencia final de la interfaz desde la pantalla de edición de la herramienta GUIDE. Ahí podemos hacer todas las modificaciones necesarias para conseguir la estética deseada para nuestra interfaz.

Además de modificaciones estéticas, también podemos realizar ciertas configuraciones y seleccionar que unciones que queremos implementar *(callbacks)* en el archivo.m.



**FIGURA 3.2** Interfaz Gráfica en pantalla de edición GUIDE

A partir de la figura anterior, comenzaremos a explicar los diferentes controles utilizados y el código implementado para cada uno de ellos.

***IP y Port***

En estos dos cuadros de texto, se le solicita al usuario que introduzca la dirección IP y el Puerto para realizar la comunicación TCP/IP con el servidor.

A continuación, analizaremos el código empleado. La funcionalidad de cada línea de código será comentada e irá precedida de % en letra cursiva para mejor visualización. Tenemos que mencionar también, que se ha hecho uso de variables globales, debido a que se deben utilizar ciertas variables para diversos controles y como ya sabemos, las variables utilizadas en una función (y no declaradas como globales), son consideradas por el programa como locales y por tanto no serían detectadas por otras funciones de nuestro código.

|  |
| --- |
| function edit1\_Callback(hObject, eventdata, handles)  *%Definición de variables globales utilizadas para este control.*  global IP global t    global Port  IP = get(hObject,'String'); *% guardamos en la variable IP el valor del edit text que introduzca el usuario* guidata(hObject, handles); *%Salvar datos de la aplicación* |

***Conectar***

El push button “Conectar” se encarga de crear la variable t,que es el manejador (handle) de la conexión TCP/IP. Esta variable se utiliza para realizar la comunicación mediante el protocolo TCP/IP con sus correspondientes comandos. El código asociado a este control se muestra a continuación:

function pushbutton1\_Callback(hObject, eventdata, handles)

*%Definición de variables globales*

*utilizadas para este control.*

|  |
| --- |
| global t global IP global Port    *% Establecemos la conexión mediante el protocolo TCP/IP utilizando las funciones de la toolbox de Matlab correspondiente.*  t=tcpip(IP, Port); fopen(t); msgbox('La Conexión se ha establecido'); *% Abrimos un cuadro de diálogo al haberse establecido de forma correcta la conexión* guidata(hObject, handles); *%Salvar datos de la aplicación* |

***Barras de desplazamiento Ji***

Las barras de desplazamiento (“Sliders”), se utilizan para que el usuario elija el número de grados que desea mover cada una de las seis articulaciones que posee el robot. El código correspondiente al primer slider se muestra a continuación:

|  |
| --- |
| function slider1\_Callback(hObject, eventdata, handles)    *%Definición de variables globales utilizadas para este control.*  global J1 global sj1    *% cogemos el valor del slider y nos quedamos con su valor absoluto y su signo para asignárselo a las variables J y sj*  ang =get(hObject,'Value'); signo = sign(ang);  J1 = uint8(round(abs(ang))); if signo == -1 sj1 = 0; else sj1=1; end |

angulo =round(ang);

*% actualizamos el valor del static text correspondiente a este Joint Value*  set(handles.text2,'String',angulo); guidata(hObject, handles); *%Salvar datos de la aplicación*

*Nota: Para los otros cinco valores de Joint restantes, se utiliza el mismo fragmento de código. Para evitar la redundancia, omitiremos la descripción de los demás.*

***Enviar Joint Values***

El push button “Enviar Joint Values” se encarga de crear el “Datagrama de Valores Articulares” y enviarlo por el socket. Para ello hace uso tanto de los valores obtenidos a través de los sliders, como de la variable global t, creada para establecer la comunicación TCP/IP. El código perteneciente a este botón es el siguiente:

|  |
| --- |
| function pushbutton2\_Callback(hObject, eventdata, handles)    *%Definición de variables globales utilizadas para este control.*  global J1 global J2 global J3 global J4 global J5 global J6 global t global sj1 global sj2 global sj3 global sj4 global sj5 global sj6 |
| *% Construimos el datagrama*  D1=uint8([1 sj1 J1 sj2 J2 sj3 J3 sj4 J4 sj5 J5 sj6 J6]);    *%Enviamos el datagrama realizando una pausa de dos segundos, para dar tiempo al robot a realizar la instrucción ordenada.*  fwrite(t, D1); pause(2); guidata(hObject, handles); *%Salvar datos de la aplicación* |

***Reset Joint Values***

Este control se ha implementado para llevar al robot a una posición de reposo y facilitar al usuario un posible reseteo de las posiciones de las diferentes articulaciones del robot.

El código que se presenta a continuación realiza las funciones descritas anteriormente:

|  |
| --- |
| function pushbutton3\_Callback(hObject, eventdata, handles) *%Definición de variables globales utilizadas para este control.*  global J1 global J2 global J3 global J4 global J5 global J6 global sj1 global sj2 global sj3 global sj4 global sj5 global sj6 global t  *% Enviamos el robot a la posición de Reposo* |

|  |
| --- |
| set(handles.slider1,'Value',0) ; set(handles.text2,'String',0); set(handles.slider2,'Value',0); set(handles.text3,'String',0);  […]  *%Y así sucesivamente con los demás valores Joint del robot.* sj1=1; J1=0; sj2=1; J2=0; sj3=1; J3=0; sj4=1; J4=0; sj5=1; J5=0; sj6=1;J6=0;  *% asignamos los varales de reset y enviamos el datagrama*    D1=uint8([1 sj1 J1 sj2 J2 sj3 J3 sj4 J4 sj5 J5 sj6 J6]); fwrite(t, D1); pause(2); guidata(hObject, handles); *% Salvar datos de la aplicación* |

***Cuadros de texto editable Z, Y, X***

En este cuadro de texto se le solicita al usuario que introduzca los valores en grados de los ángulos de Euler Z, Y, X que desea que gire el efector final del robot.

El código asociado al cuadro de texto Z es el siguiente:

|  |
| --- |
| function edit3\_Callback(hObject, eventdata, handles)    *%Definición de variables globales utilizadas para este control.*  global sz global Rz  *% Obtenemos de los cuadros de texto las variables de Rotación para construir el datagrama* vble = str2double(get(hObject,'String')); signo = sign (vble); if signo == -1 sz=0; else sz=1; end |

Rz= uint8(round(abs(vble))); *% Obtenemos la variable de rotación Rz*.

guidata(hObject, handles); *%Salvar datos de la aplicación*

Para el resto de variables de rotación (Y, X) se utilizan las mismas instrucciones, siendo la única diferencia, la denominación de las variables intermedias y de las variables finales obtenidas.

***Cuadro de texto editable X, Y, Z***

Este control es similar al anterior con la diferencia de que en este caso al usuario se le solicita la posición del efector final (en mm) en vez de la rotación.

El código asociado se presenta a continuación:

|  |
| --- |
| function edit6\_Callback(hObject, eventdata, handles)  *%Definición de variables globales utilizadas para este control.*  global spx global X global xr    *%Obtenemos las variables en este caso correspondientes a la coordenada x para construir el datagrama*  x=str2double(get(hObject,'String')); xr=rem(x,100); X=floor(x/100); signo = sign (x);    if signo == -1 spx=0; else spx=1; |

end guidata(hObject, handles); *%Salvar datos de la aplicación*

Para el resto de variables de posición (Y,Z) se utilizan las mismas instrucciones, siendo la única diferencia la denominación de las variables intermedias y de las variables finales obtenidas.

***Enviar rotación, Enviar Posición y Enviar rotación y posición***

Para estos controles nos bastará con explicar uno sólo de ellos, ya que el resto son similares.

Cada uno de los *push button* nombrados anteriormente tiene como función crear el datagrama correspondiente y enviarlo por el socket. El push button “ Enviar rotación” creará y enviará el datagrama dos, el push button “Enviar posición” el datagrama tres y el push button “Enviar rotación y posición” se encargará de crear y enviar el datagrama cuatro (que como ya sabemos es una mezcla de los datagramas dos y tres; es decir, posee variables tanto de rotación como de posición).

El código del control correspondiente a la orden “Enviar la rotación del TCP” es el siguiente:

|  |
| --- |
| function pushbutton4\_Callback(hObject, eventdata, handles)    *%Definición de variables globales utilizadas para este control.*  global sz global sx global sy global Rz global Rx global Ry global t    D2=uint8([2 sz Rz sy Ry sx Rx]); *% Construimos el datagrama* |

fwrite(t, D2); *%Enviamos datagrama realizando una pausa de 2 segundos entre medias* pause(2); guidata(hObject, handles); *%Salvar datos de la aplicación*

***TCP Velocity***

Este control es un “pop up” menu, es decir, una lista desplegable en la que podremos elegir la velocidad del efector final.

Para este datagrama no es necesario un control externo que se encargue de crear y enviar el datagrama, lo realizamos todo en el mismo control.

El cógido asociado a este control se muestra a continuación:

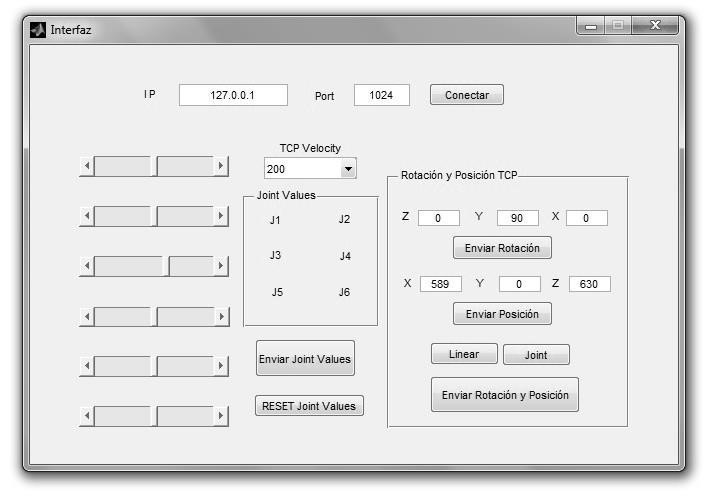
|  |
| --- |
| function popupmenu2\_Callback(hObject, eventdata, handles)    *%definimos las variables globales utilizadas en este control.*  global t; global Vt; global Vtr; global Vr; global Vrr; sxl=1; xl=0; xlr=0; szl=1; zl=0; zlr=0;  vel= get(hObject,'Value'); *% Ajustamos valores de velocidad a partir del control del popup menu*  Vtr=rem(vel,100);  Vt=floor(vel/100);  Vrr=rem(vel,100);  Vr=floor(vel/100);    D5=uint8([5 1 Vt Vtr Vr Vrr sxl xl xlr szl zl zlr]); *% construimos y enviamos en datagrama.*    fwrite(t,D5); |

pause(2); guidata(hObject, handles); *%Salvar datos de la aplicación*

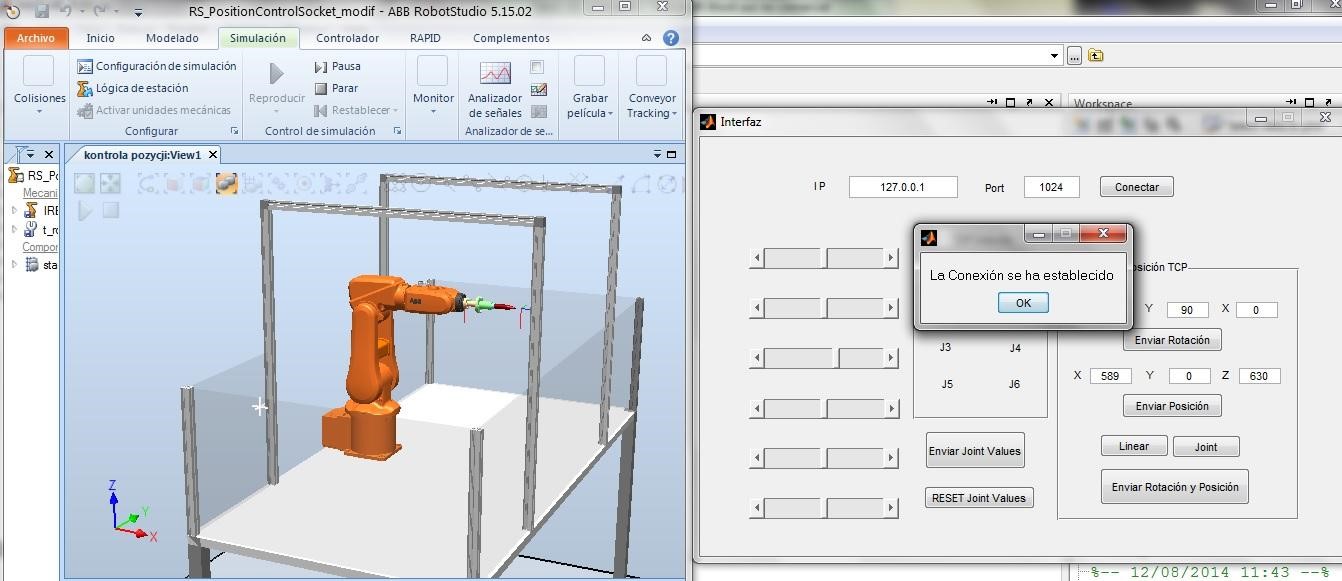
En la siguiente imagen, podemos ver la apariencia final de la interfaz ( la que verá el usuario) tras hacer clic en el botón de ejecutar.

Tras conocer el funcionamiento de la interfaz, procedemos a realizar una demostración gráfica. En las siguientes imágenes observaremos la apariencia final de nuestra interfaz gráfica GUI.

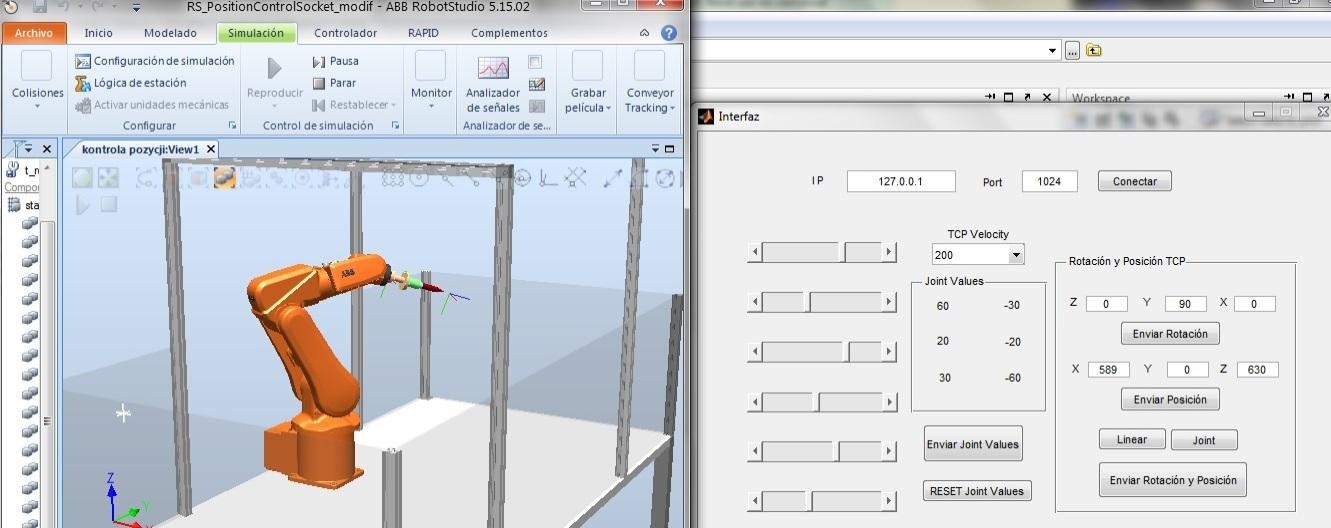
En figuras posteriores podremos observar el funcionamiento de la interfaz, para la que se ha hecho uso de la Estación de RobotStudio creada en el TFG [1] in RobotStudio for controlling ABB-IRB120 robot. Design and development of a palletizing station” [1] para poder así comprobar que el brazo robot realiza correctamente las instrucciones que le enviamos desde nuestra interfaz.



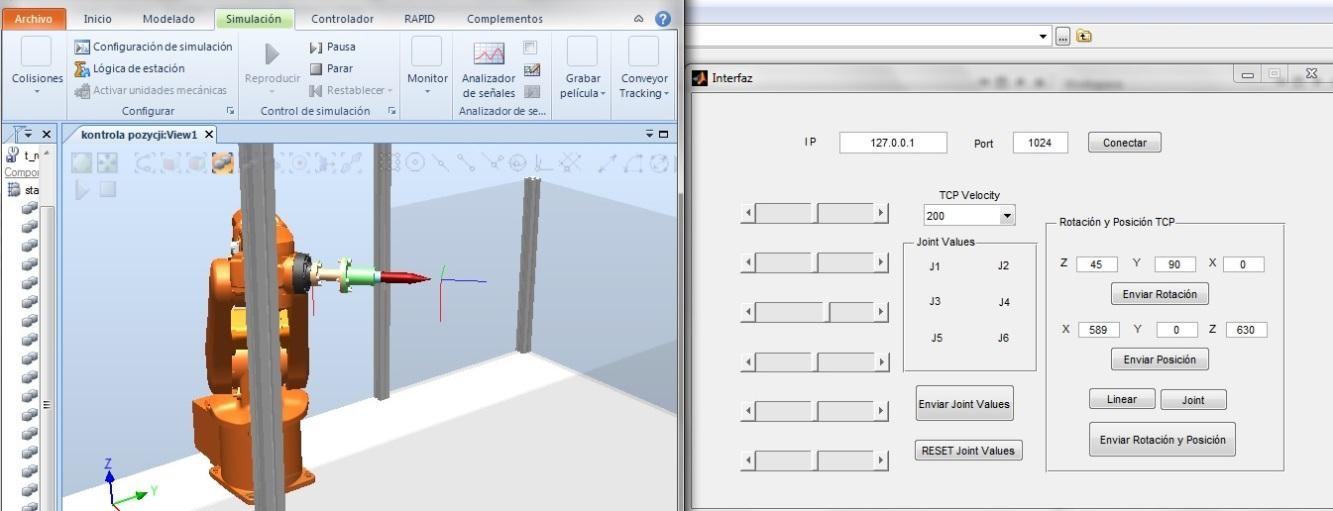
**FIGURA 3.3** Apariencia final Interfaz GUI



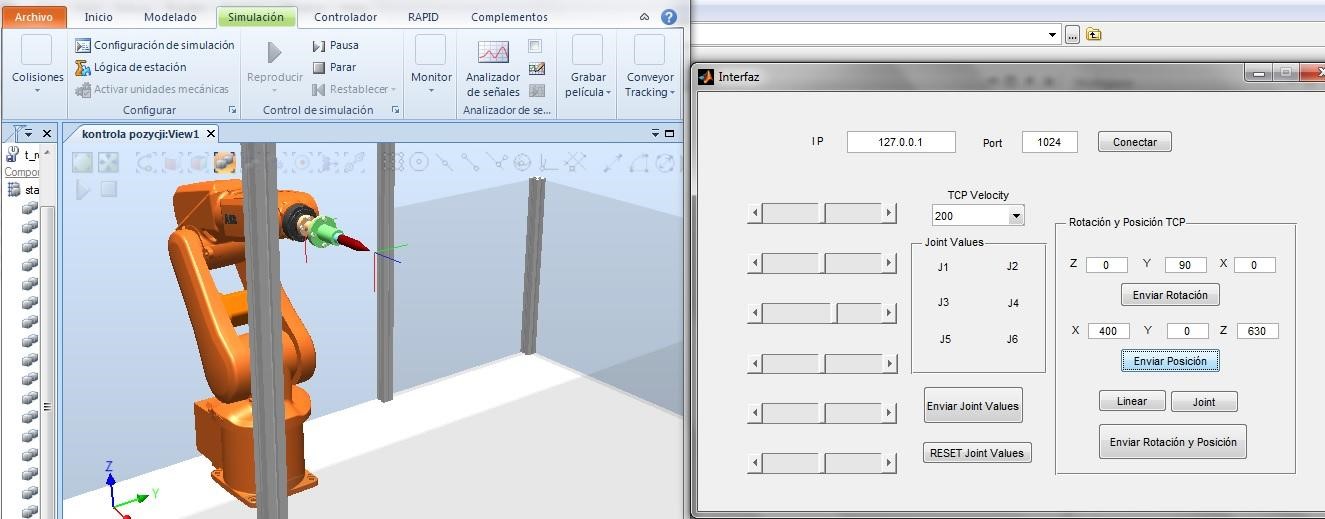
**FIGURA 3.4** Paso 1 de la demostración de la Interfaz GUI



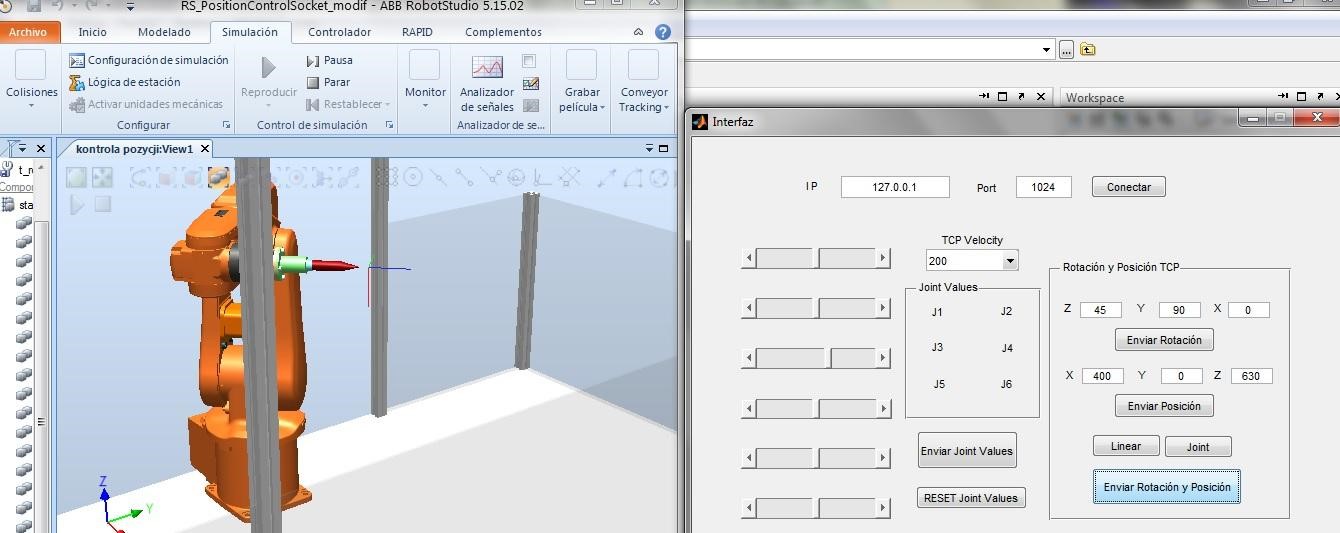
**FIGURA 3.5** Paso 2 de la demostración de la Interfaz GUI:



**FIGURA 3.6** Paso 3 de la demostración de la Interfaz GUI:



**FIGURA 3.7** Paso 4 de la demostración de la Interfaz GUI:



**FIGURA 3.8** Paso 5 de la demostración de la Interfaz GUI:

***Interfaz a través de clases en Matlab***

El objetivo principal de esta interfaz a través de la creación de clases en Matlab, es facilitar al usuario el manejo de la comunicación con el brazo robot y también poder utilizar la clase creada para usos posteriores desde cualquier fichero .m de Matlab. Como ya sabemos, la creación de clases en Matlab puede ser más o menos complicada, pero su utilización es de lo más sencilla para el usuario. De ahí que hayamos elegido este método para crear nuestra interfaz.

Por tanto, para desarrollar esta interfaz, hemos definido una clase denominada “irb120”. Esta clase posee una serie de propiedades y unos métodos, compuestos por funciones que serán los encargados de realizar las acciones asociadas a esta clase. En primer lugar, utilizando el comando habitual, hemos definido la clase:

classdef irb120 < handle

Nuestra clase pertenece a la clase handle, que es la clase superior en Matlab por definición. No ha sido necesario implementar ningún tipo de subclases para esta interfaz, con la clase principal ha sido suficiente.

A continuación, si seguimos analizando el código desarrollado, nos encontramos con la definición de las propiedades:

Properties

IP; port; conexion; end

Las propiedades las utilizamos para asignar valores a los campos que posee nuestra clase y que son necesarios para que las funciones definidas posteriormente los utilicen.

A partir de este punto, entramos en el bloque de los métodos en el que tenemos definidas una serie de funciones que comentaremos a continuación:

 **Función que obtiene los valores de IP y Puerto**

Esta función es el denominado constructor de la clase. La clase la creamos escribiendo el nombre de la misma e introduciendo los valores que sean necesarios para su creación.

Más adelante observaremos un ejemplo de cómo crear la clase paso a paso. A continuación mostramos el código correspondiente al constructor que crea la clase:

function r=irb120(ip,puerto)

r.IP=ip;

r.port=puerto;

end

Esta función se encarga de obtener el valor de la dirección IP elegida por el usuario y el valor del puerto por el que se va a establecer la conexión con el servidor.

Tras estas líneas de código, ya estaría creada nuestra clase y tendría el siguiente aspecto:



**FIGURA 3.9** Creación de la clase

**FIGURA**

**3.10**

Clase irb120



 **Función que realiza la conexión y la desconexión**

Esta función es la encargada de realizar la conexión o desconexión de la comunicación según requiera la aplicación.

El código correspondiente es el que aparece a continuación:

function connect(r)

r.conexion=tcpip(r.IP, r.port); fopen(r.conexion); end function disconnect(r) fclose(r.conexion); end

En estas dos funciones, abrimos y cerramos respectivamente la comunicación con el robot. Para realizar la conexión, basta con seguir la línea de comando básica de la comunicación TCP/IP en Matlab, ajustando los parámetros a los de nuestra clase.

El código ejecutado en la ventana de comandos de Matlab sería el siguiente:



**FIGURA 3.11** Uso de funciones de la clase creada

 **Función que envía la configuración articular**

Esta función es la encargada de enviar los valores de los ángulos de cada una de las seis articulaciones del robot. Para explicar cómo se ha implementado, nos basta con analizar el primer valor de articulación, ya que los siguientes son similares salvo por la denominación de las variables (J1, J2…etc).

function jointvalues(r,q)

J1=uint8(round(abs(q(1))));

|  |
| --- |
| signo = sign(q(1));    if signo == -1 sj1 = 0;  else sj1=1; end  […]  end |

El usuario introduce desde la ventana de comandos de Matlab los seis ángulos necesarios para crear el datagrama de Valores Articulares en forma de vector. Este fragmento de código, recoge el primer elemento del vector, obtiene su valor absoluto y su signo y los guarda en las variables correspondientes (J1, sj1 respectivamente).

Para los valores de los ángulos restantes, el proceso es el mismo, pero accediendo a elementos sucesivos del vector de ángulos introducido por el usuario.

Una vez obtenidos todos los valores de los ángulos, creamos y enviamos el datagrama:

|  |
| --- |
| D1=uint8([1 sj1 J1 sj2 J2 sj3 J3 sj4 J4 sj5 J5 sj6 J6]);    *%Enviamos el dato*  fwrite(r.conexion,D1); |

La manera de ejecutar en Matlab esta función es como se muestra en la figura siguiente:



**FIGURA 3.12** Uso de funciones de la clase creada

Con la instrucción anterior estaríamos solicitando al robot que realice un giro de 20 grados positivos en la primera articulación, y otro de cuarenta grados negativos en al segunda articulación. El resto de articulaciones se quedarían en la posición de reposo.

 **Función que envía la Rotación del TCP**

Como su nombre indica, esta función se encarga de enviar los valores de los ángulos de rotación del TCP. Al igual que ocurre con la configuración articular, el usuario introduce en la ventana de comandos un vector de tres elementos correspondientes a los tres ángulos Z, Y, X (en ese orden y en grados).

Analizamos el código referente al ángulo Z y obviamos por similitud el código de los restantes. Tras obtener los tres valores, se crea y envía el datagrama.

function TCProtation(r,zyx) Rz=uint8(round(abs(zyx(1)))); signoz = sign (zyx(1)); if signoz == -1 sz=0; else sz=1; end

[…]

D2=uint8([2 sz Rz sy Ry sx Rx]); fwrite(r.conexion, D2);

La manera de ejecutar correctamente ese comando en Matlab sería la siguiente:



**FIGURA 3.13** Uso de funciones de la clase creada

Con la instrucción anterior estaríamos pidiéndole al robot que gire 45 grados positivos respecto a su eje Z. Los otros dos ángulos se quedarían en la posición de reposo.

* **Función que envía los valores de posición del TCP**

Para la posición del TCP, se realiza el mismo proceso que para la rotación: un vector de tres elementos (en este caso la posición viene dada en mm) y el código que veremos a continuación (correspondiente a la coordenada X) se encargará de seleccionar el primer elemento de ese vector y obtener la posición X, Y, Z del TCP.

|  |
| --- |
| function TCPposition(r,xyz) xr=rem(xyz(1),100); X=floor(xyz(1)/100); signox = sign (xyz(1));    if signox == -1 spx=0; else spx=1; end  […]    D3=uint8([3 spx X xr spy Y yr spz Z zr 0]);    fwrite(r.conexion,D3); |

Con la instrucción que se muestra a continuación, realizamos un cambio de la posición del efector final. En concreto estaríamos pidiéndole al robot que se desplace 400 mm en X (positivo), y que mantenga el reposo para sus otras dos coordenadas.



**FIGURA 3.14** Uso de funciones de la clase creada

* **Función que envía la rotación y la posición del TCP**

Esta última función, al igual que el datagrama del mismo nombre, se encarga de enviar de manera simultánea la rotación y la posición del TCP. El usuario introduce en la ventana de comandos de Matlab un vector en este caso de seis elementos: [rotación Z, rotación Y, rotación X, posición X, posición Y, posición Z].

El código de esta función, es un conjunto de las dos anteriores. La creación y envío del datagrama se realiza de la misma manera que los demás datagramas.

function TCProtandpos(r,tcp)

[…]

D4=uint8([4 sz Rz sy Ry sx Rx spx X xr spy Y yr spz Z zr]);

fwrite(r.conexion,D4);

En la figura siguiente ejecutaremos el comando realizando las dos instrucciones anteriores de manera simultánea:



**FIGURA 3.15** Uso de funciones de la clase creada

A continuación, veremos un ejemplo gráfico a través de imágenes del funcionamiento de esta interfaz junto con el simulador de RobotStudio para poder comprobar los resultados con el Robot.

**Paso 1** de la demostración de la Interfaz a través de clases en Matlab:

* *Creamos la clase, establecemos la conexión, y enviamos un valor de Joint*

**Paso 2** de la demostración de la Interfaz a través de clases en Matlab:

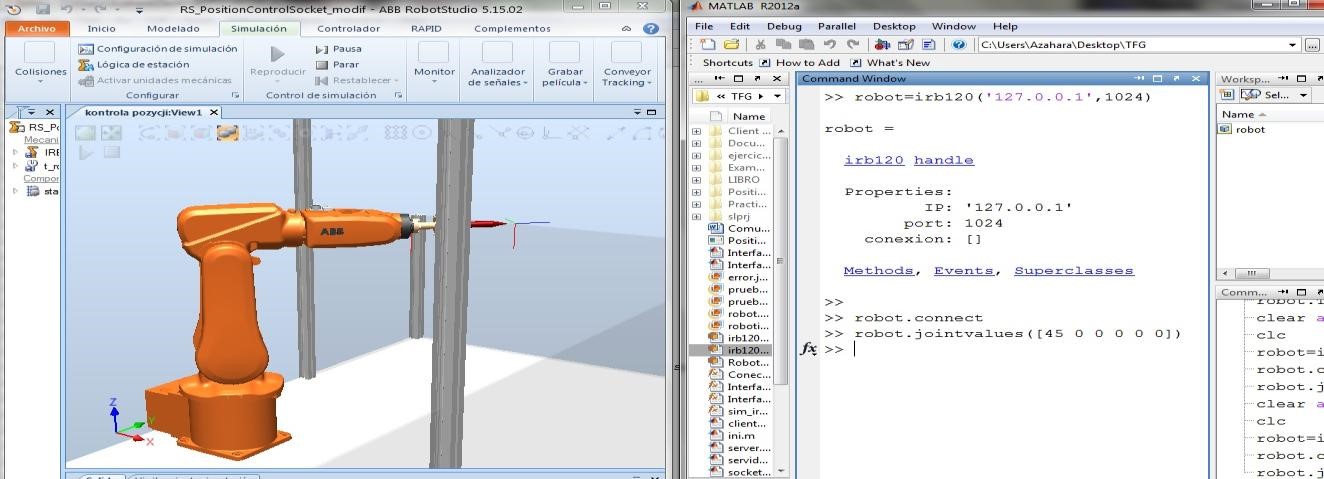
* *Enviamos un valor de Joint para cada uno de los ángulos*

**Paso 3** de la demostración de la Interfaz a través de clases en Matlab:

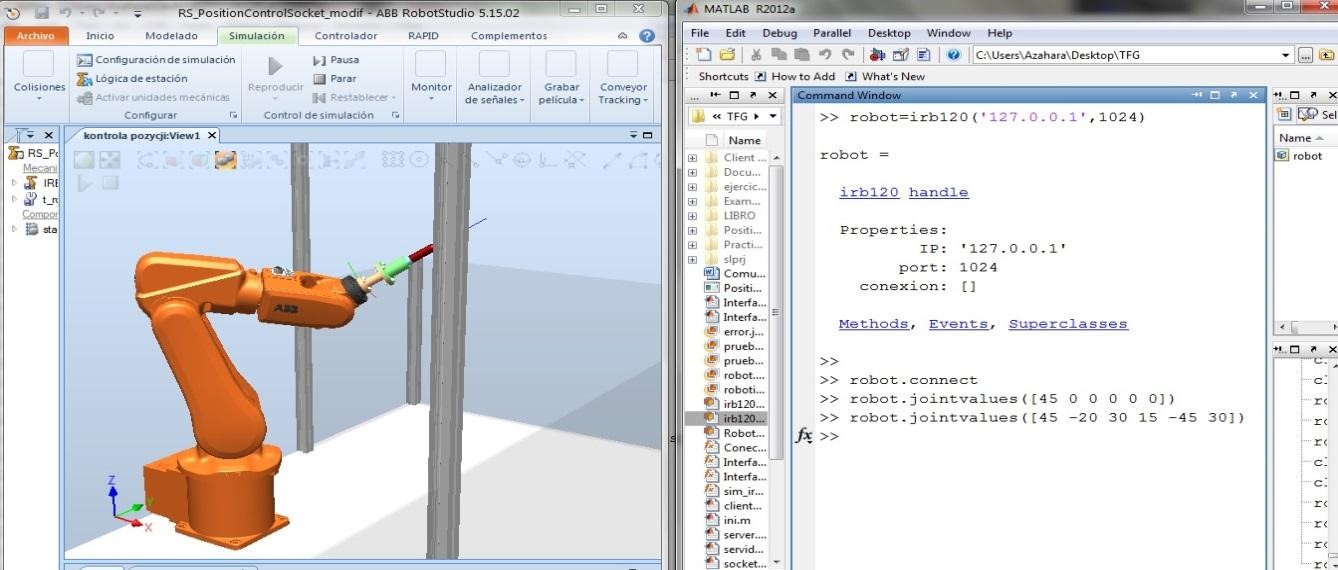
* *Elegimos unos valores de Rotación del TCP y se los mandamos al Robot* **Paso 4** de la demostración de la Interfaz a través de clases en Matlab: - *Elegimos unos valores de Posición del TCP y se los mandamos al Robot*

**Paso 5** de la demostración de la Interfaz a través de clases en Matlab:

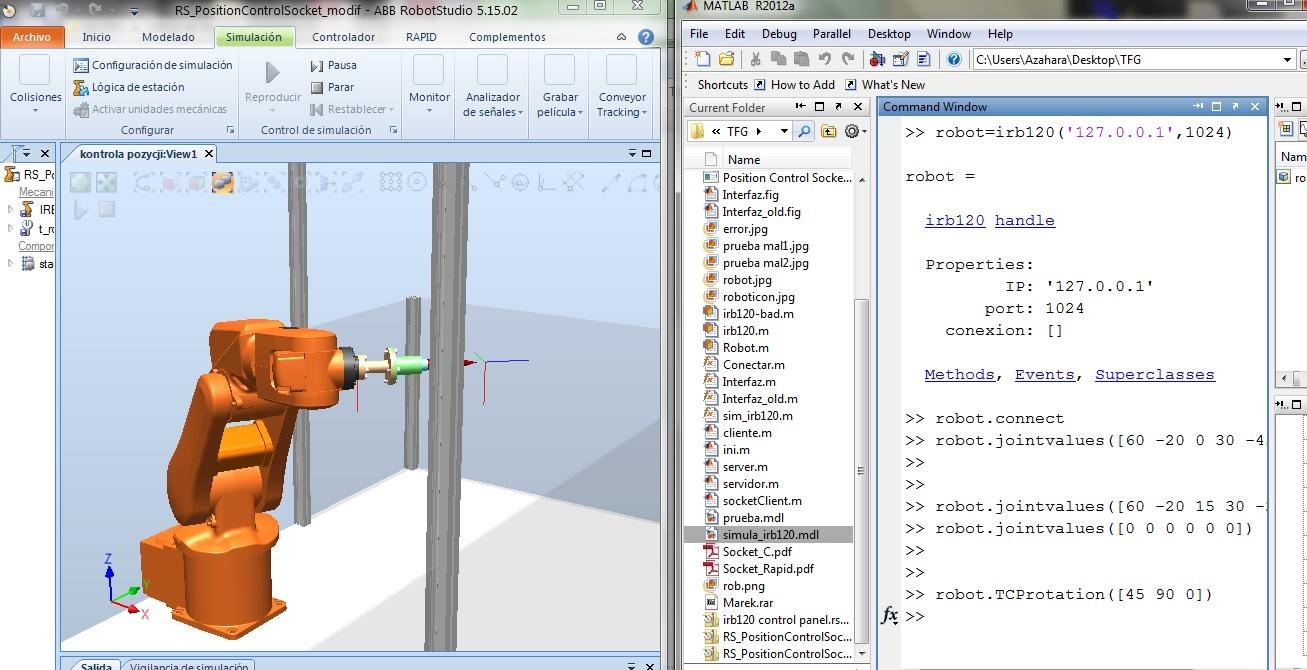
* *Elegimos unos valores de Rotación y Posición del TCP y se los mandamos al Robot de manera simultánea*



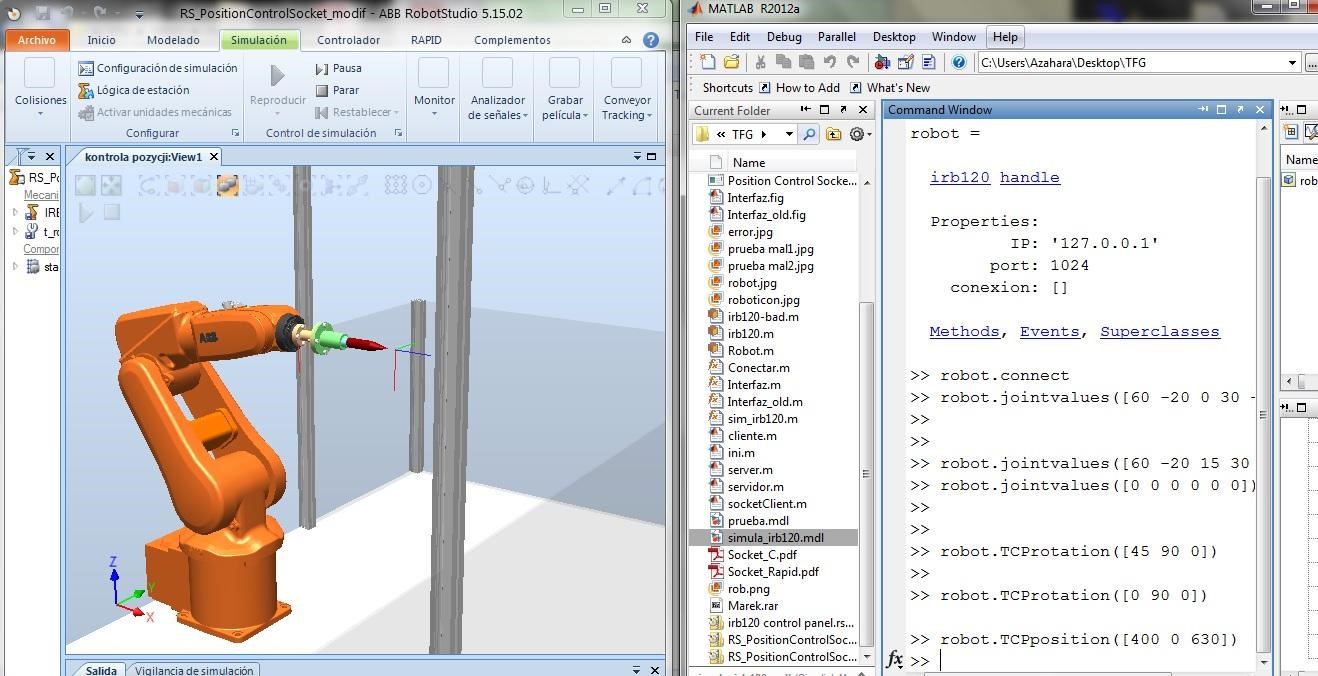
**FIGURA 3.16** Paso 1 de la demostración de la Interfaz a través de clases en Matlab



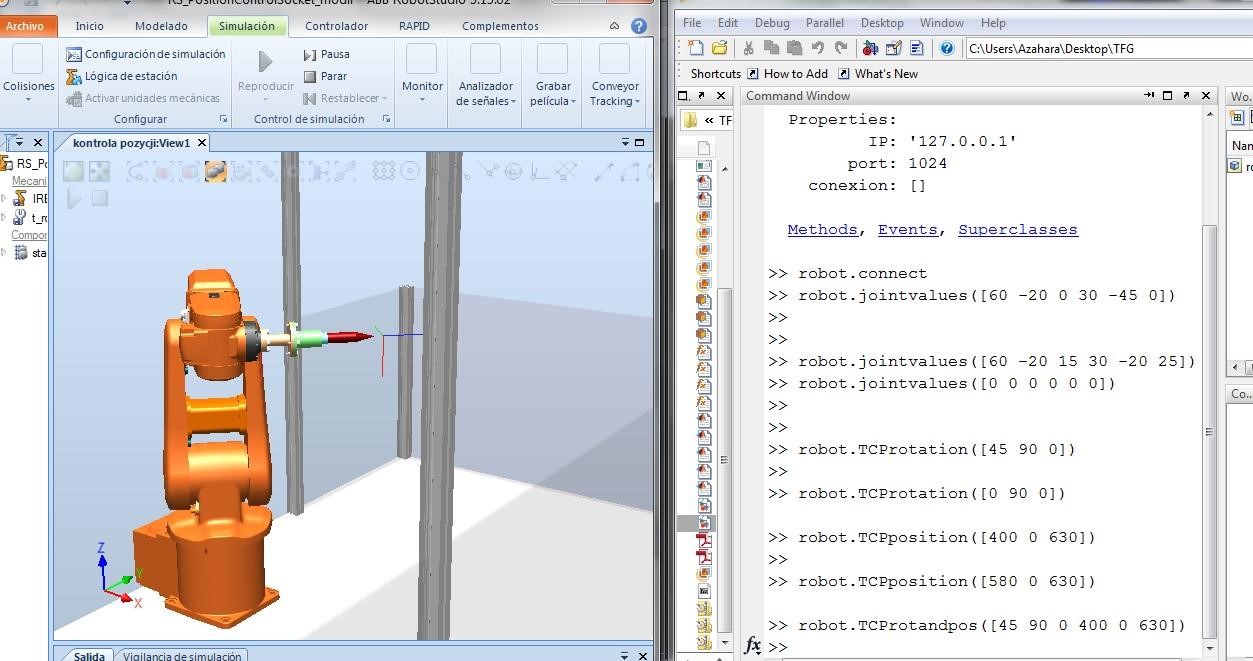
**FIGURA 3.17** Paso 2 de la demostración de la Interfaz a través de clases en Matlab



**FIGURA 3.18** Paso 3 de la demostración de la Interfaz a través de clases en Matlab



**FIGURA 3.19** Paso 4 de la demostración de la Interfaz a través de clases en Matlab



**FIGURA 3.20** Paso 5 de la demostración de la Interfaz a través de clases en Matlab

***Interfaz a través de un bloque de Simulink***

Para la realización de esta interfaz, utilizaremos la herramienta de Matlab, Simulink y un archivo de inicialización .m.

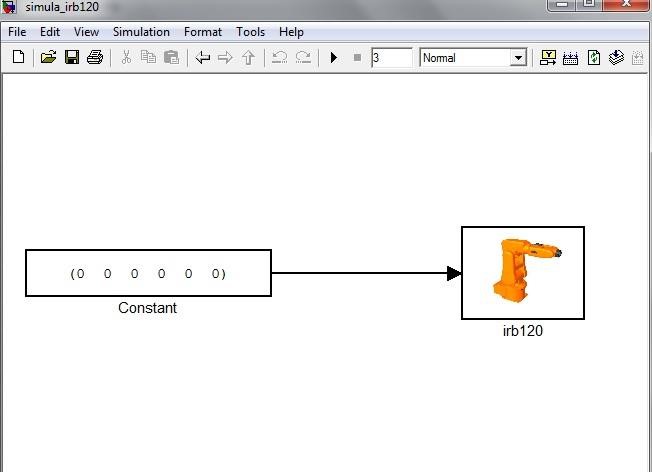
Esta interfaz consta por tanto, de dos archivos principales: el archivo .m de inicialización y el archivo .mdl en el que se encuentra el bloque de Simulink creado. En el primero, definimos una serie de variables globales que se utilizarán después y en el archivo de Simulink encontramos el bloque que hemos creado para conectar con el robot y enviarle los datos que creamos convenientes.

También debemos mencionar que en el archivo de inicialización, se realiza la apertura de la comunicación con el servidor. A través de Simulink, procesamos y enviamos la información.

En la siguiente figura, observamos el contenido del archivo de Simulink creado. Consta de dos elementos: el primero es un vector de constantes en el que introduciremos los valores de los ángulos Joint que queremos enviar al robot y en el segundo encontramos el bloque irb120 creado.

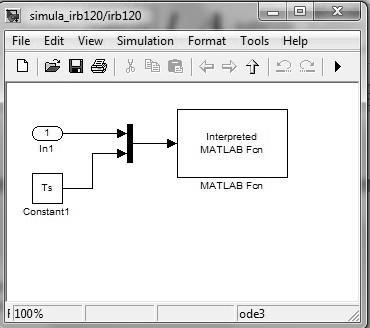
El bloque irb120 puede utilizarse en cualquier otro diagrama de bloques de Simulink.

Esta interfaz ha sido diseñada para realizar sólo el envío de la configuración articular del robot.



**FIGURA 3.21** Contenido del archivo .mdl

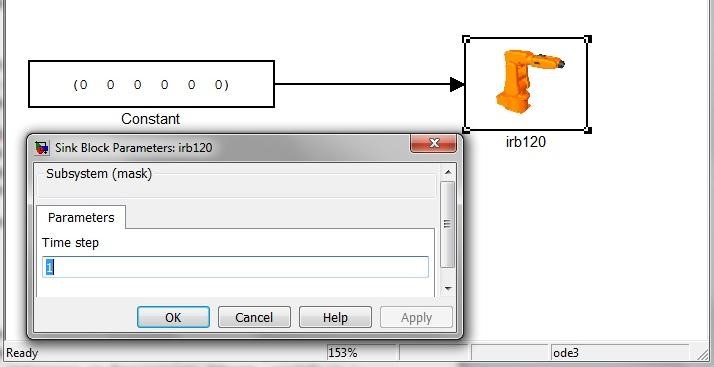
El bloque mencionado anteriormente ha sido enmascarado y posee en su interior un subsistema formado a su vez por otro bloque denominado “*Interpreted Matlab Function*”.



**FIGURA 3.22** Contenido del interior de la máscara del bloque irb120

El bloque *Interpreted Matlab Function* posee un archivo .m asociado, en el que hemos implementado un código que realiza el procesamiento de los datos, la creación del datagrama y el envío del mismo.

El parámetro *Ts* (Time Step), lo utilizaremos para unificar el tiempo de muestreo de cada bloque y para ajustar los tiempos de simulación de nuestro programa. Gracias al enmascaramiento del bloque, accedemos de forma rápida y sencilla a este parámetro simplemente con hacer doble clic sobre el bloque irb120.



**FIGURA 3.23** Parámetro Ts de la máscara

En la imagen anterior podemos comprobar la manera de acceder al parámetro “Time Step”.

El desarrollo actual de esta interfaz, implica que el bloque de Simulink obtiene la configuración articular directamente del vector de constantes que le estamos proporcionando a su entrada.

En apartados posteriores, analizaremos un funcionamiento diferente, orientado ya a nuestra aplicación final de pintar números.

El bloque de Simulink recibirá la configuración articular de un conjunto de bloques con el que realizamos un control diferencial.

A continuación, el funcionamiento de esta interfaz:

**Paso 1** de la demostración de la Interfaz a través de Bloque de Simulink: - *Ejecutamos el archivo de inicialización y se establece la comunicación*

**Paso 2** de la demostración de la Interfaz a través de Bloque de Simulink:

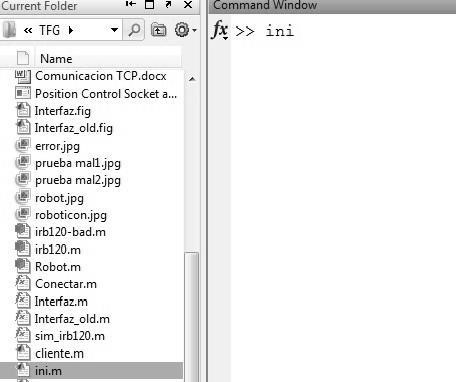
* *Elegimos un vector de valores y ejecutamos nuestro .mdl*

**Paso 3** de la demostración de la Interfaz a través de Bloque de Simulink:

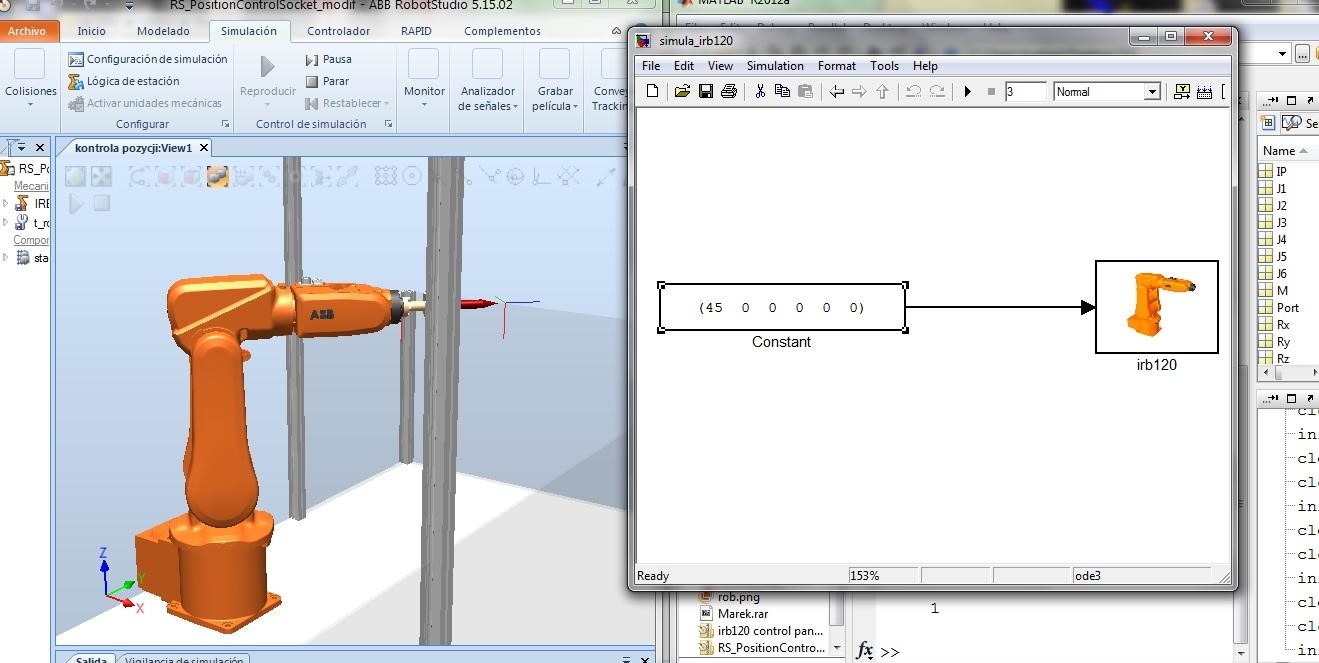
* *Modificamos el vector de valores y añadimos más ángulos*

**Paso 4** de la demostración de la Interfaz a través de Bloque de Simulink:

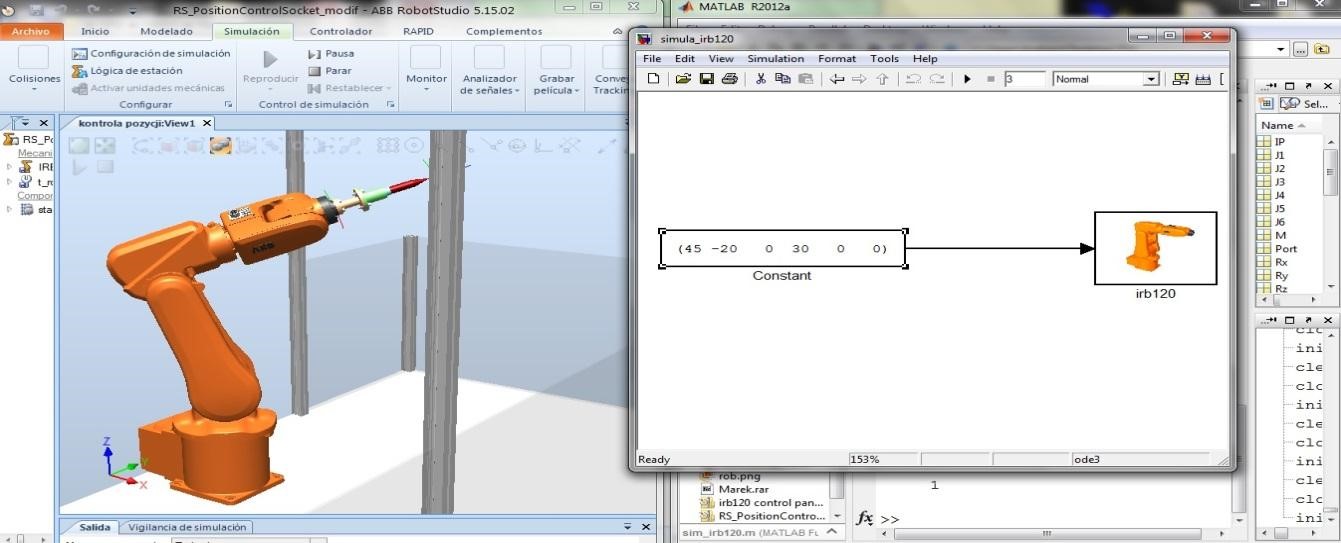
* *Volvemos a modificar los valores y ejecutamos el archivo comprobando el resultado*



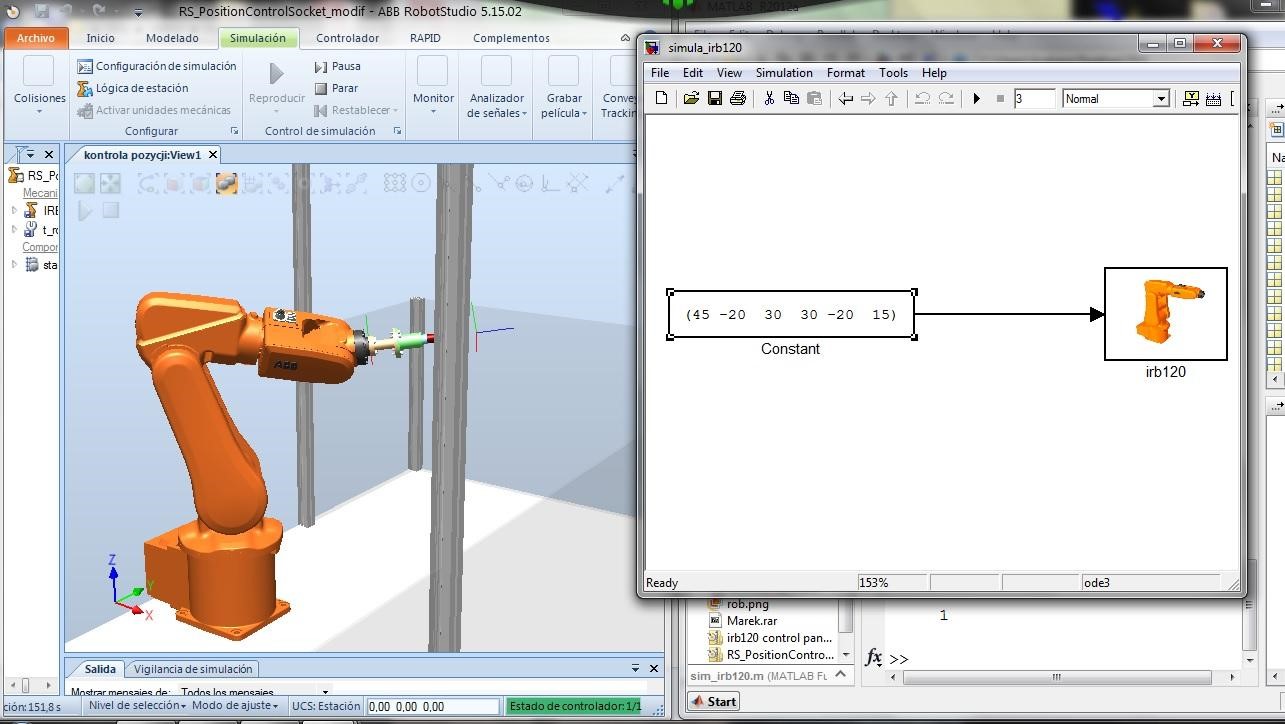
**FIGURA 3.24** Paso 1 de la demostración de la Interfaz a través de Bloque de Simulink:



**FIGURA 3.25** Paso 2 de la demostración de la Interfaz a través de Bloque de Simulink:



**FIGURA 3.26** Paso 3 de la demostración de la Interfaz a través de Bloque de Simulink:



**FIGURA 3.27** Paso 4 de la demostración de la Interfaz a través de Bloque de Simulink

Una sola interfaz de programación para robot y máquina-herramienta

La automatización de las máquinas-herramientas requiere la integración de robots industriales en células de mecanizado de tal modo que las modificaciones que tenga que realizar el operario, como por ejemplo equipamientos posteriores, tengan efecto inmediato sobre el robot o cumplan al menos las premisas ergonómicas para el manejo de la máquina. Asimismo, todas las operaciones de manejo del robot de una célula deben poder realizarse desde una sola unidad de operación siguiendo el principio del "Single Point of Operation". Esto incluye interacciones como el manejo en situaciones normales, el desbloqueo, la programación por aprendizaje y el diagnóstico.

LOS PROGRAMAS DE LA MÁQUINA-HERRAMIENTA Y DEL ROBOT PUEDEN MANEJARSE EN CANALES PARALELOS DESDE LA INTERFAZ DE USUARIO SINUMERIK OPERATE

KUKA Roboter y Siemens Drive Technologies se enfrentaron al reto de presentar en la EMO 2011 un entorno simple y eficiente. Para ello, KUKA desarrolló la biblioteca de módulos de funciones mxAutomation que permite programar la trayectoria de movimiento del robot KUKA de forma íntegra en la Sinumerik. De este modo, y sin disponer de conocimientos especiales de programación de robots, es posible crear el programa del robot con las herramientas de programación de PLC STEP 7 de Siemens y prepararlo de forma que pueda completarse en el marco de la programación de ciclos CN de la Sinumerik. Los programas de la máquina-herramienta y del robot pueden ejecutarse y controlarse en canales paralelos desde la interfaz de usuario Sinumerik Operate.

TRANSFERENCIA DE PARÁMETROS CONVENCIONALES AL ROBOT

Con ayuda del panel de operación manual HT8 y de la interfaz de usuario de la Sinumerik no sólo es posible realizar la programación por aprendizaje de tareas de preparación de la máquina-herramienta, sino también de funciones y puntos de la trayectoria del robot. De este modo, se puede prescindir de distintos y complejos paneles de operación manual. El nexo lo forman los módulos de mxAutomation, que se utilizan en el marco de las utilidades PLC gestionadas por la Sinumerik y que sirven para controlar el robot a través de una interfaz Profinet. El sistema de control KUKA KR C4 se encarga de ejecutar los movimientos del robot y de poner a disposición nuevas y conocidas funciones de seguridad, así como otras funciones específicas del robot, garantizando máximo rendimiento y fiabilidad.

GRACIAS A LA ACEPTACIÓN, SE CUMPLEN LAS PRETENSIONES EN CUANTO A RENDIMIENTO DE LA INVERSIÓN

En comparación con la carga y descarga manuales, las ventajas del empleo de un robot son obvias. Por ejemplo, la elevada precisión de posicionamiento de las piezas, la posibilidad de realizar un tercer turno completamente automatizado o los tiempos cortos de reequipamiento que hacen que la producción sea más flexible. El único impedimento para la automatización mediante robots industriales era, a menudo, la falta de aceptación por parte de los operarios. Con esta solución de manejo común, basada en mxAutomation y las interfaces abiertas de las unidades de control Sinumerik, KUKA y Siemens han establecido nuevas pautas.

SOBRE EL GRUPO KUKA ROBOTER

KUKA Roboter GmbH con sede en Augsburgo es una empresa de la sociedad anónima KUKA y uno de los más importantes fabricantes de robots industriales en todo el mundo. Sus principales competencias son el desarrollo, la producción y la venta de robots industriales, unidades de control y software. Es líder del mercado en Alemania y Europa, y ocupa el tercer puesto a nivel mundial. KUKA Roboter GmbH cuenta con unos 2350 empleados repartidos por todo el mundo. El volumen anual de ventas en 2010 ascendió a 435,7 millones de euros. La empresa cuenta con más de 25 filiales en los mercados más importantes de Europa, EE.UU. y Asia.

Bibliografía:

http://www.infoplc.net/noticias/item/857-una-sola-interfaz-de-programacion-para-robot-y-maquina-herramienta