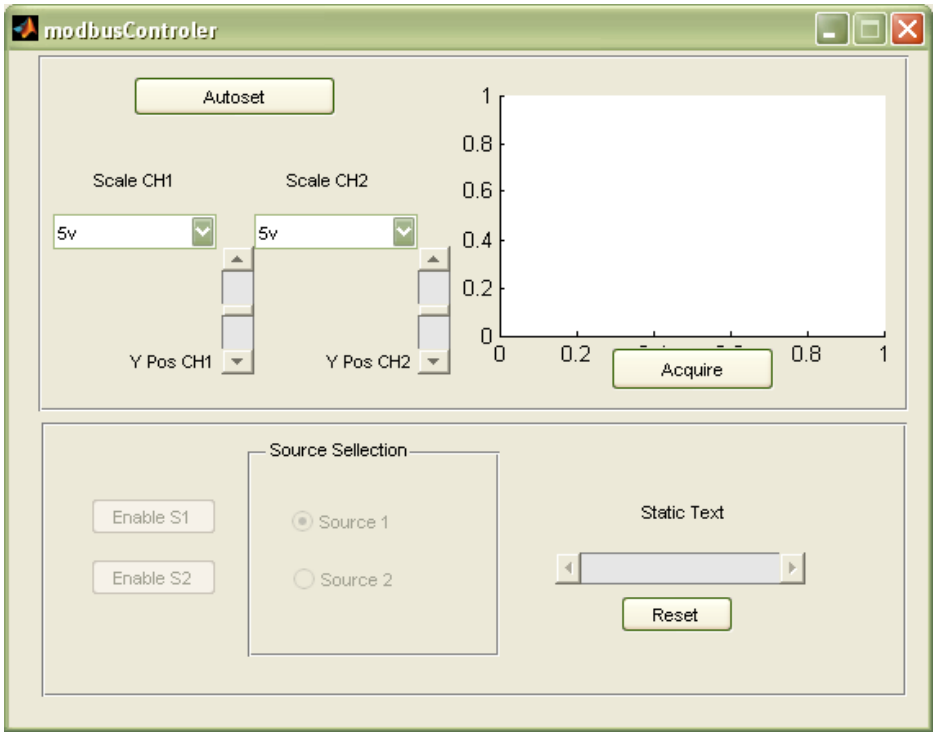
2 SCADA MATLAB

2.1 Objetivos

El objetivo de esta sesión es la construcción de un pequeño sistema SCADA completamente parametrizable.

Tal y como se vió en la sesión anterior, un SCADA es un paquete informático que permite el control y la supervisión remota de una planta. A través del SCADA el operador es informado del estado de la planta, mediante representaciones sinópticas, formadas por iconos que cambian de valor en función de la información recibida de la planta. También a través de dichas pantallas el operador puede mandar órdenes a la planta.

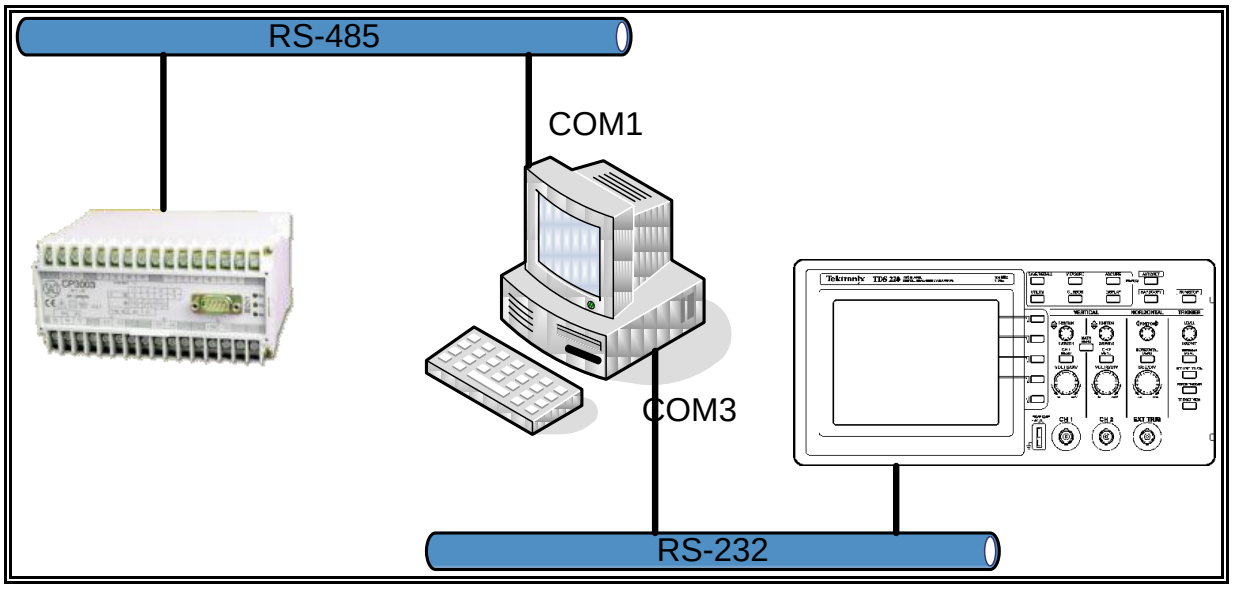
En función del tamaño del sistema a controlar, el SCADA se puede repartir entre varios ordenadores y equipos de comunicaciones (son los puestos centrales de operación; ejemplo: despacho de REE en Madrid) o estar todo contenido en un mismo equipo. A diferencia de la sesión anterior, donde se utilizó un elemento comercial de SCADA, en la presente se construirán algunos elementos fundamentales de un SCADA utilizando el entorno gráfico de programación de MATLAB.



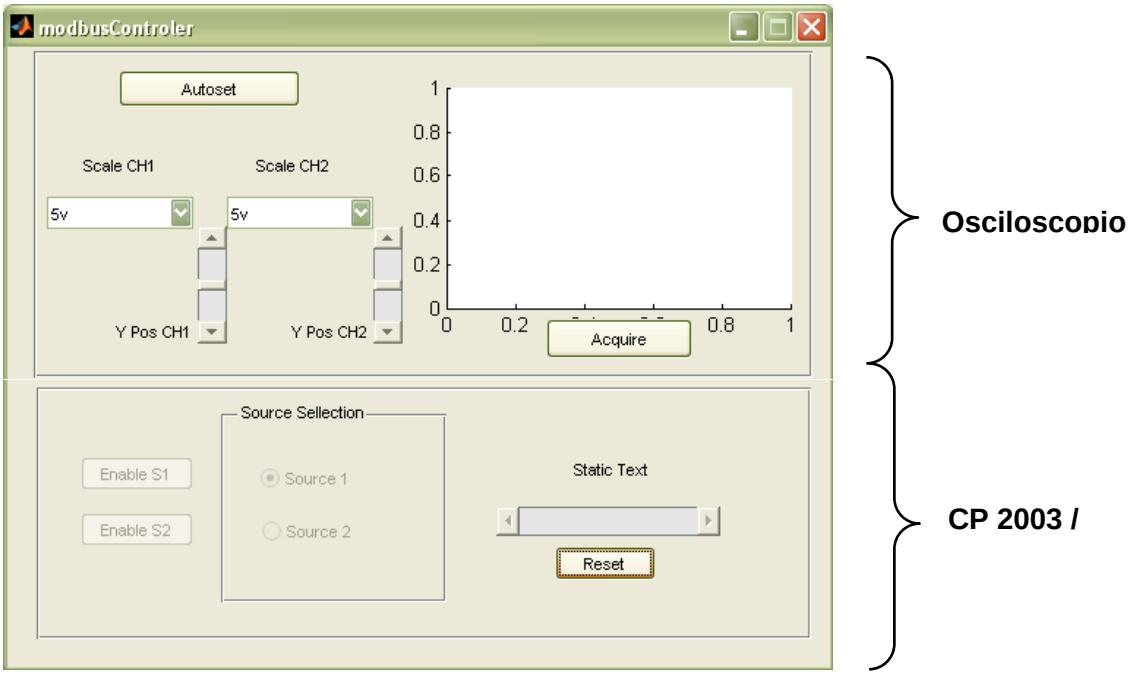
De esta manera se cubrirá toda la pila de comunicaciones en sistemas de comunicaciones sobre RS232 y RS485. En la primera parte de la asignatura se hizo más hincapié en las capas físicas y el canal, mientras que en estas sesiones se trabaja la capa de aplicación.

2.2 Descripción del escenario.

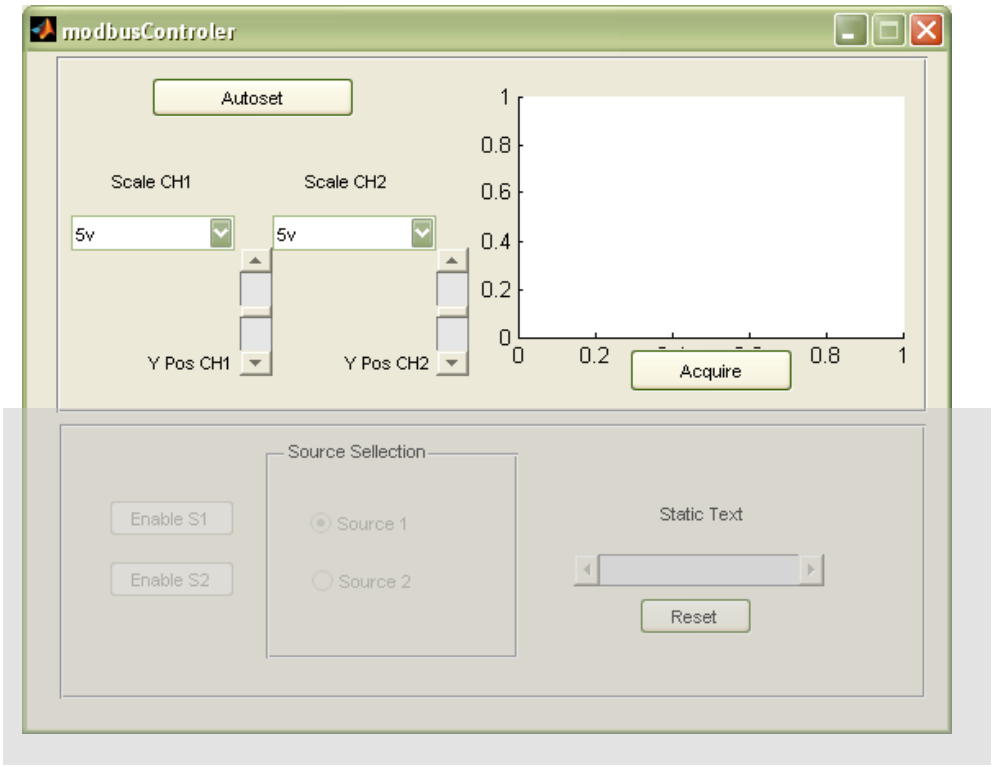
El escenario a simular es el mostrado en la imagen inferior. En él, la aplicación que corre sobre Matlab ofrecería a un operador el estado de una o varias señales de control (escaneadas mediante un osciloscopio) al mismo tiempo que le serviría para ajustar otros relés o señales analógicas mediante un convertidor CP2003.



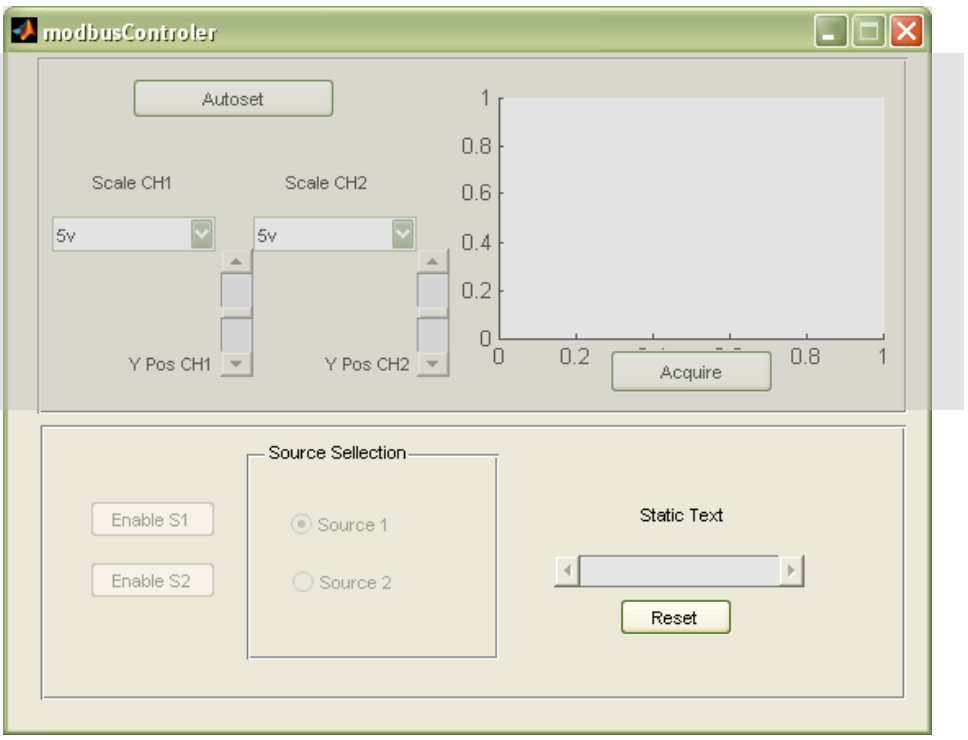
La aplicación que controlará ambos dispositivos será la siguiente:



Tal y como se muestra en la figura anterior, se diferencian dos zonas de control en la aplicación. La interfaz con el osciloscopio cuenta con las siguientes funcionalidades.



Contamos con un botón que “Autoconfigura” el osciloscopio. En cada canal podemos elegir la representación que nos haga en la pantalla. Además podemos variar el nivel de referencia del equipo con dos slider’s independientemente. También podemos tomar leer los datos de las medidas de ambos canales y hacer una representación de los datos.



En cuanto a la parte referida al CP2003, la aplicación ha de ser capaz de habilitar y deshabilitar ambas salidas analógicas de manera independiente; solo si una de las salidas de la fuente está activa, se progamará para que el valor de corriente a la salida sea el indicado por el slider horizontal y además se mostrará como texto. En una primera instancia todos los controladores estarán inhabilitados hasta presionar el botón de reset, el cual configurará inicialmente el dispositivo con 10mA por cada salida analógica, aunque las salidas se mantendrán inhabilitadas.

Adicionalmente, se montará un circuito separado donde se activarán dos indicadores luminosos que darán a conocer el estado de cada una de las salidas de la fuente (Habilitada/Deshabilitada).

2.3 Introducción al diseño de GUIs con Matlab.

MATLAB cuenta con un editor de Interfaces Gráficas de Usuario (Graphic User Interfaces o GUI): guide.

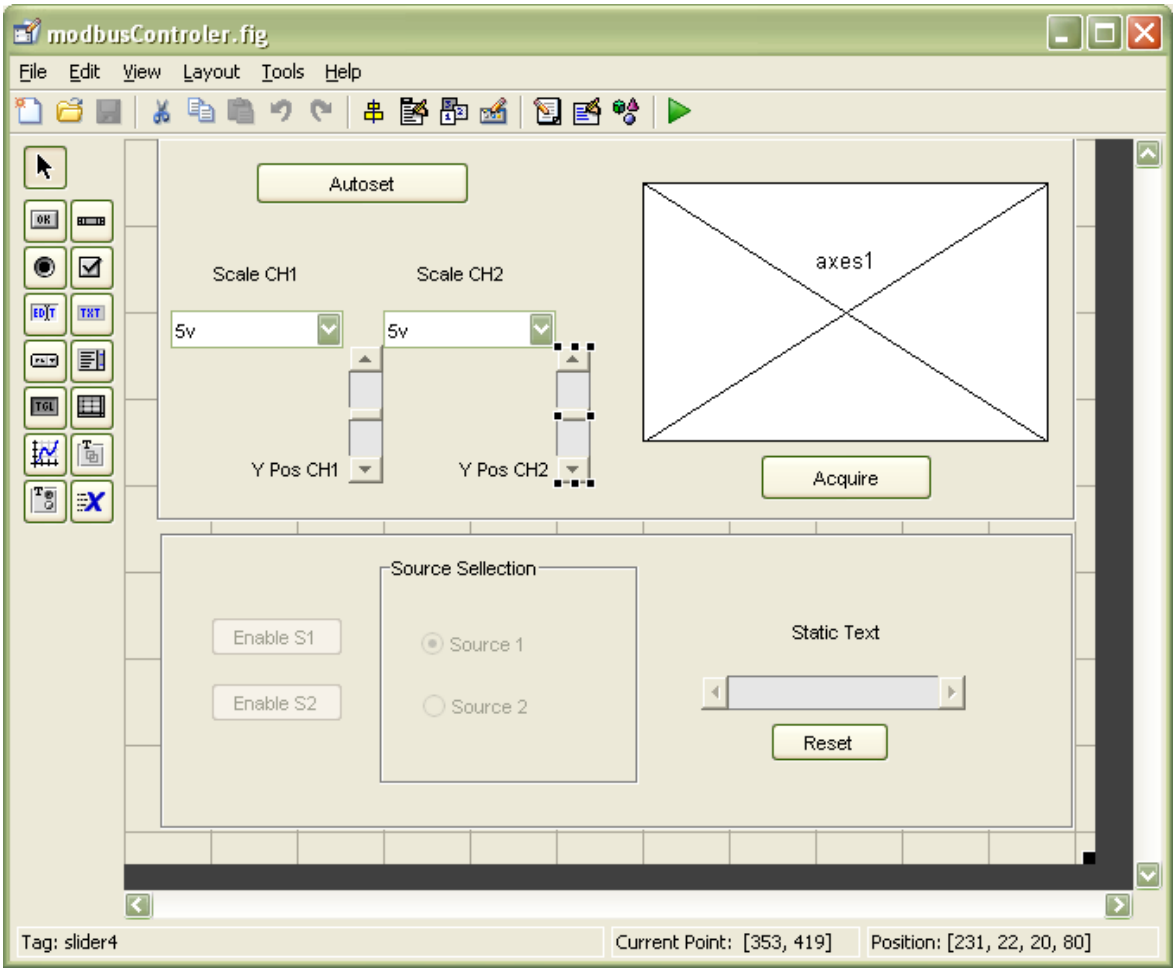
Una aplicación GUIDE consta de dos archivos: .m y .fig. El archivo .m es el que contiene el código con las correspondencias de los botones de control de la interfaz y el archivo .fig contiene los elementos gráficos. Cada vez que se adicione un nuevo elemento en la interfaz gráfica, se genera automáticamente código en el archivo .m

Para la presente sesión se proporcionan ambos archivos. Se encuentran disponibles en la web de la asignatura.

Una vez descargados, se pueden abrir con

```
>> guide modbusControler.fig
```

Al abrir la figura con el editor de GUI's veremos lo siguiente:



Ésta es la ventana de edición. Aquí podemos añadir más controladores o configurar los ya existentes.

En el esqueleto proporcionado para la práctica ya se encuentran todos los controladores creados. Únicamente hay que configurarlos y programarlos para que realicen las funciones descritas.

Haciendo click derecho sobre cualquiera de los objetos accedemos a algunas de sus propiedades. Una de las más interesantes es la de acceder a los *Callbacks* (View Callbacks/Callbacks). Los *callbacks* son las funciones con las rutinas a ejecutar cuando se interactúa con alguno de los elementos, es decir, la función que se ejecutará al hacer clic sobre un botón, por ejemplo.. Accediendo a ellos, Matlab nos lleva directamente a la zona del código donde está definida dicha función.

Por ejemplo para el botón de “Acquire” nos llevaría a la siguiente parte del código:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Obviamente, la primera vez que se accede a la función ésta está vacía. La tarea del alumno es programar las acciones que son requeridas para conseguir la funcionalidad deseada.

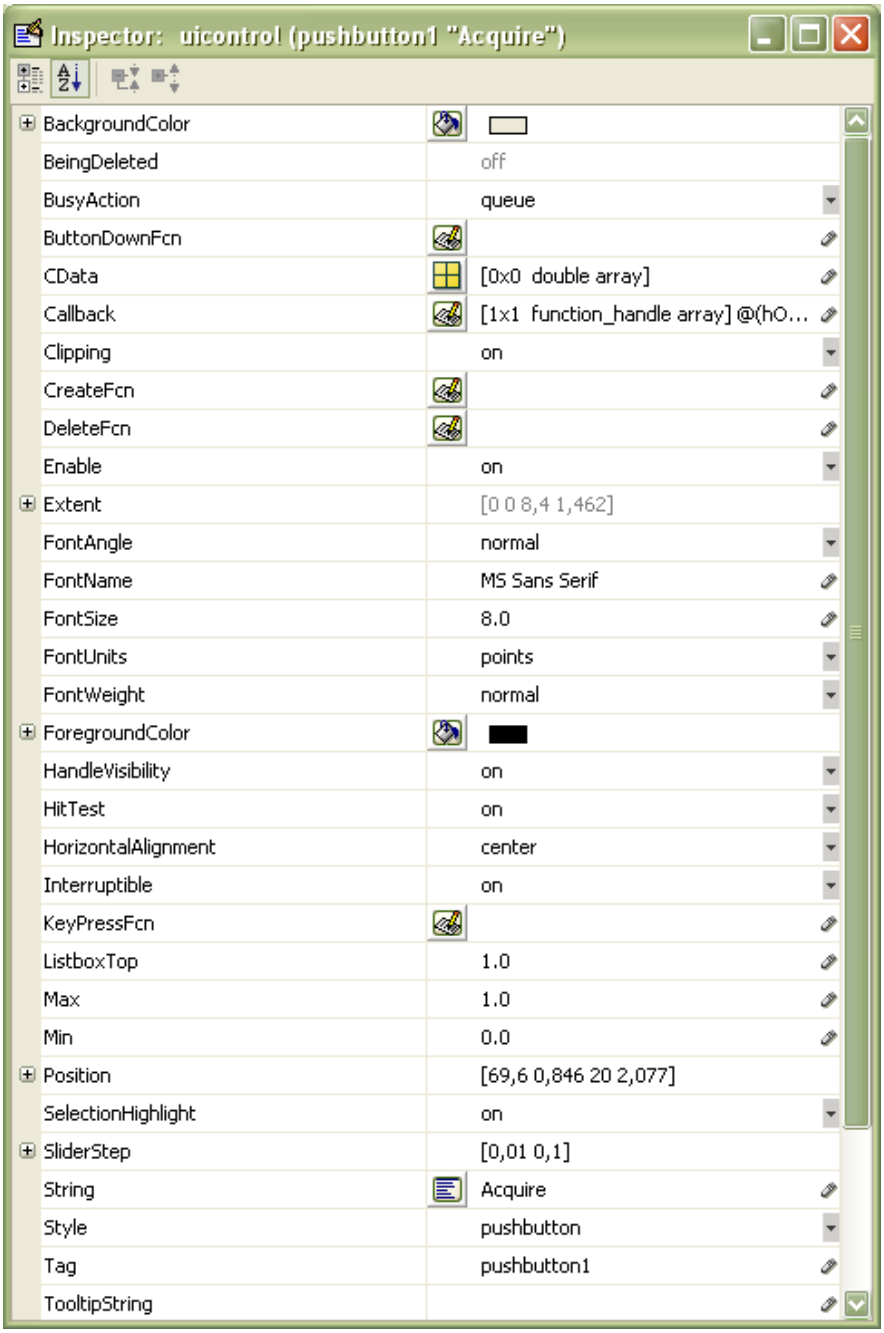
Merece la pena resaltar los diferentes parámetros de entrada que se pasan a las funciones *callback*.

El primer parámetro (“*hObject*”) es un manejador al objeto concreto que creó el evento. En este caso el objeto es un “botón” cuyo nombre es “pushbutton1”. Este parámetro es una estructura con todas las propiedades referentes a dicho objeto. Para consultar o modificar dichas propiedades se usan las funciones de matlab ***get*** y ***set***. Ver la ayuda de Matlab referente a estas funciones para más detalles.

El objeto “*eventdata”*, como su propia descripción indica, se plantea para uso futuro.

Por último, “*handles”* es una estructura donde cada campo contiene un “*hObject*” de cada uno de los objetos de la GUI.

Adicionalmente al uso de las funciones **get** y **set**, también se puede acceder a las propiedades de los objetos haciendo doble click en cualquiera de ellos. El menú es el siguiente:



Los valores de estas propiedades son usados para la primera inicialización de la interfaz gráfica. A partir de ahí pueden ser modificados por los *callbacks*.

Algunas de las propiedades dependen del tipo de objeto al que se refieren. Durante la sesión de prácticas se hará un breve repaso de las más importantes.

Para ejecutar la interfaz gráfica se puede hacer clic en el icono  o desde Tools/Run.

2.4 Descripción de la práctica.

En la sesión de laboratorio se utilizará el esqueleto de GUI disponible en la web de la asignatura programando sobre él las funcionalidades necesarias. Debido a la brevedad de la sesión **únicamente se realizarán las funcionalidades relativas al osciloscopio**. Así mismo, será necesaria la preparación de un trabajo previo a la sesión, el cual se detalla en el punto 2.5.

La evaluación de esta parte de la práctica se llevará a cabo mediante la implementación de cada una de las funcionalidades descritas anteriormente y que se detallan a continuación:

Osciloscopio.

- AutoSet.
- Selección de escala.
- Selección de posición de referencia.
- Adquisición de datos manual.
- Adquisición de datos automática (frecuencia configurable).

Pregunta 1: *Enviar por correo electrónico el archivo .m y .fig indicando los puntos implementados (jmatanza@dea.icai.upcomillas.es)*

2.5 Trabajo Previo.

Utilizando el esqueleto de GUI disponible en la asignatura, sobreescribir la función callback del botón de “Autoset” para que, al pulsarlo, escriba la cadena “Hola mundo” en la consola de MATLAB.

2.6 Gestión de puertos serie con Matlab

Ya que la comunicación con el osciloscopio se hará mediante RS232, merece la pena hacer algunos comentarios sobre la gestión del puerto serie mediante Matlab. La gestión de los puertos serie con Matlab se hace mediante la función *serial*.

`obj = serial('port')` crea un "objeto Puerto serie" asociado al Puerto serie especificado por "port". Si "port" no existe o si se encuentra en uso no será posible su apertura. Matlab lo comunicará con un error.

El "objeto puerto serie" es una estructura con las caraterísticas del puerto abierto. Vease por ejemplo:

```
>> s = serial('COM1');
>> s
   Serial Port Object : Serial-COM1

   Communication Settings
      Port:               COM1
      BaudRate:           9600
      Terminator:         'LF'

   Communication State
      Status:             closed
      RecordStatus:       off

   Read/Write State
      TransferStatus:     idle
      BytesAvailable:     0
      ValuesReceived:     0
      ValuesSent:         0
>>
```

Este objeto puede ser utilizado como descriptor para una función de escritura. Únicamente hay que abrir el descriptor y comenzar a escribir sobre él. A continuaciones tenemos un ejemplo de código que escribe y escucha la respuesta del puerto:

```
s = serial('COM1');
fopen(s);
fwrite(s,'Hola Mundo'); %Función para escribir en el puerto
out = fread(s); %Función para escuchar en el puerto
fclose(s);  %Una vez terminad hay que cerrar el descriptor
delete(s); % Y cerrar el puerto
```