2º INGENIERÍA INDUSTRIAL

TEORÍA DE CIRCUITOS Y SISTEMAS

LABORATORIO 3 SISTEMAS: SERVOMOTOR C.C. COMANDADO POR COMPUTADOR

OBJETIVOS DE LA PRÁCTICA

- Identificar sobre un montaje real los componentes de un sistema muestreado o controlado por computador:
  - Sistema físico continuo (motor eléctrico).
  - Sistema discreto (PC).
  - Muestreador y bloqueador (tarjeta de entradas/salidas para PC).
- Manejar el servomotor de la práctica a través del computador:
  - Enviar comandos desde el PC al motor.
  - Recoger desde el PC los datos de los sensores del motor.

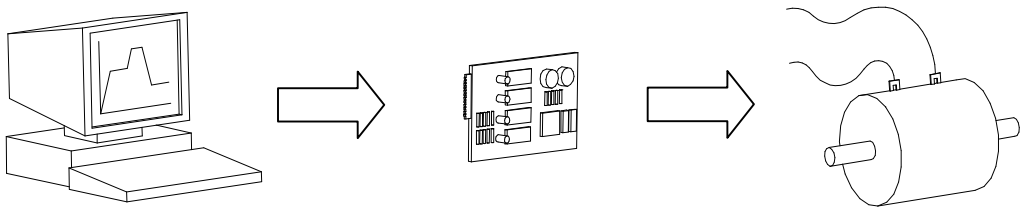
1. INTRODUCCIÓN

En esta práctica se aprenderá a manejar el servomotor de corriente, continua conocido de otras prácticas, a través de un computador. Las operaciones a realizar serán básicamente dos:

- Transmisión de órdenes de mando desde el PC hacia el motor.
- Recogida de datos de funcionamiento desde el motor hacia el PC.

En ambos casos el elemento que servirá de conexión entre el motor (sistema continuo) y el PC (sistema discreto) será una tarjeta de adquisición de datos, que realizará las conversiones A/D (de analógico a digital) y D/A (de digital a analógico) necesarias.

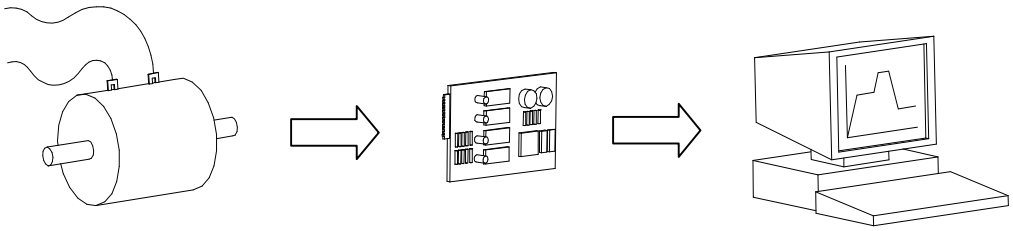
La siguiente figura muestra la forma de enviar comandos desde el PC hacia el motor:



PC: secuencia de tensiones a aplicar al motor $\{v_K\}$	Tarjeta: convierte la secuencia en una señal continua **v(t)**	Motor: recibe como entrada la señal continua **v(t)**

En este caso la tarjeta actúa como un bloqueador, transforma una secuencia discreta en una señal continua. El elemento que realiza este proceso se denomina convertidor digital/analógico o convertidor D/A.

A continuación se muestra la forma de recibir datos sobre velocidad del motor, posición, etc. en el PC:



Sensores motor: generan señales continuas, p. ej. de velocidad w**(t)**	Tarjeta: convierte las señales en secuencias discretas, p. ej. $\{w_K\}$	PC: recibe como entrada la secuencia discreta $\{w_K\}$

En este caso la tarjeta actúa como un muestreador, transforma una señal continua en una señal discreta. El elemento que realiza este proceso se denomina convertidor analógico/digital o convertidor A/D.

Como referencia se detallan algunos datos acerca de la tarjeta de adquisición de datos a utilizar en esta práctica.

Características de la tarjeta ACL – 8112PG

- Bus AT (interfaz ISA)
- 16 canales de entrada con masa común o bien 8 diferenciales
- Ganancia programable de x1, x2, x4, x8 y x16
- Circuito integrado de muestreo y retención (sample&hold)
- 2 canales de salida analógicos de 12 bits monolíticos multiplexados
- 16 canales de entrada digital y 16 de salida digital
- 3 contadores independientes de cuenta atrás de 16 bits
- Frecuencia de muestreo programable de hasta 100KHz
- Tres modos de disparo del convertidor A/D: disparo por software, disparo programable mediante temporizador/contador y disparo mediante pulso externo.
- Capacidad para trabajar mediante interrupción: 11 niveles de interrupción (IRQ3 a IRQ15) seleccionables mediante jumper
- Conector de 37 pines tipo D
- Tamaño compacto (half – size PCB)

Especificaciones de la tarjeta ACL – 8112PG

- Entrada analógica (A/D)
  - Convertidor A/D: B.B. ADS774 (aproximaciones sucesivas)
  - Canales de entrada: 16 con masa común o bien 8 diferenciales
  - Resolución: 12 bits
  - Rango de entrada (controlados por sofware): Bipolar de ±10V, ±5V, ±2.5±, ±0.625V y ±0.3125V (a menor rango mayor resolución)
  - Tiempo aproximado de conversión: 8μs
  - Protección de sobretensión: ±35V DC
  - Precisión: 0.015% de la lectura ±1 bit
  - Impedancia de entrada: 10MΩ
  - Modos de disparo: por software, por temporizador/contador y disparo externo
  - Modos de transferencia: por software, por interrupción y por DMA
  - Velocidad de transferencia de datos (frecuencia de muestreo): 100KHZ (máximo)

- Salida analógica (D/A)
  - Canales de salida: 2 salidas analógicas con doble buffer
  - Resolución: 12 bits
  - Rango de salida: unipolar de 0 a 5V o 0 a 10V mediante referencia interna y unipolar de +10V o –10V con referencia externa
  - Convertidor D/A: B.B. DAC7541 de multiplexación monolítica
  - Tiempo de conversión: 30μs
  - Linealidad: ±1/2 bit LSB
  - Intensidad máxima: ±5mA

- Entradas y salidas digitales (DIO)
  - Canales: 16 de entrada y 16 de salida compatibles TTL
  - Tensión de entrada: mínimo de 0V y máximo de 0.8V para estado 0 y mínimo de +2.0V para estado 1
  - Carga admisible: +0.5V a –0.2mA máximo para estado 0 y +2.7V a +10mA máximo para estado 1
  - Tensión de salida: mínimo de 0V y máximo de 0.4V para estado 0 y mínimo de +2.4V para estado 1

- Contador programable
  - Dispositivo: 8254
  - Temporizador para conversión A/D: temporizador de 32 bits (dos contadores de 16 bits en cascada) con base de tiempos de 2MHz
  - Contador: un contador de 16 bits con base de tiempos de 2MHz
  - Frecuencia de salida del temporizador: de 0.00046 Hz a 0.5 MHz

Para realizar la conexión con los dispositivos se utiliza una tarjeta adicional conectada mediante un cable de 37 hilos a la ACL - 8112PG. Dicha tarjeta adicional consiste en una especie de regleta donde se puede acceder fácilmente a cada uno de los canales y que aporta la electrónica necesaria para acondicionar las señales. La disposición de los distintos canales y señales se muestra en la siguiente tabla:

PIN	DESCRIPCIÓN
1 a 8	Canal 0 a 7 de entrada analógica
9 – 10	Masa de los canales 0 a 7 de entrada analógica
11	Referencia de tensión externa 2
13	Salida de +12 V
14	Masa de los canales de entrada analógica
15	Masa de los canales de salida digital
16	Señal de salida del contador 0
17	Señal de disparo externo
18	No conectada
19	+5V
20 a 27	Canal 8 a 15 de entrada analógica
28 – 29	Masa de los canales 8 a 15 de entrada analógica
30	Canal 0 de salida analógica
31	Referencia de tensión externa 1
32	Canal 1 de salida analógica
33	Entrada de reloj para 8254
34	Entrada para 8254
35 – 36	No conectada
37	Entrada de señal de reloj externo
38	Masa de los canales de salida analógica

En la práctica que nos ocupa, los pines que nos van a interesar son los correspondientes a las entradas analógicas, 1 a 8 (y sus masas correspondientes, 9 o 10) para leer las señales de salida del sistema (datos de los sensores), y los de salida analógica 30 y 32 (y su masa correspondiente, 38) para aplicar tensiones al motor.

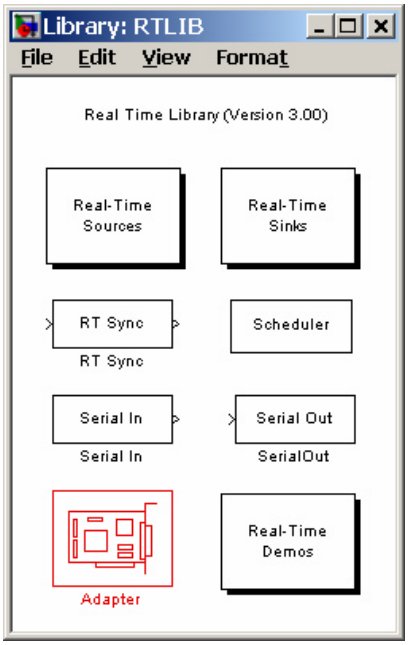
2. SOFTWARE PARA EL MANEJO DE LA TARJETA

La forma más sencilla de utilizar la tarjeta de adquisición (sin necesidad de programar) es mediante un toolbox de Matlab denominado 'Extended Real Time Toolbox'. Este paquete permite trabajar con la tarjeta de adquisición de datos directamente desde Matlab e incluso desde esquemas Simulink, ofreciendo así una gran potencia y facilidad en la etapa de desarrollo de un sistema de control.

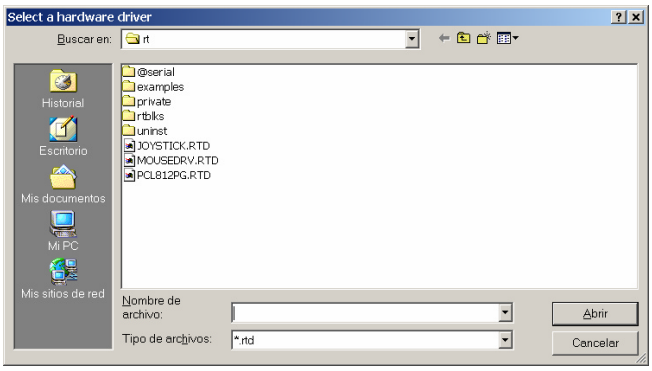
El Extended Real Time Toolbox (en adelante RTT) permite la comunicación directa entre el sistema de adquisición de datos y Matlab / Simulink; es decir, permite acceder a la tarjeta desde la línea de comando de Matlab o directamente desde un esquema Simulink. Por su simplicidad, se elegirá la segunda opción de forma que se pueda acceder directamente a la tarjeta desde Simulink. Para ello el RTT dispone de una librería de bloques Simulink que realizan dicha tarea. Dicha librería se denomina Real Time Toolbox Simulink Block Library (a partir de ahora RTLIB) y provee los bloques Simulink necesarios para comunicar el entorno (en nuestro caso el motor) con la tarjeta.

La utilización de la RTLIB en un esquema Simulink pasa por dos fases: primero la inclusión y configuración del driver del hardware de nuestra tarjeta y segundo la correcta utilización de los diferentes bloques de entrada/salida en tiempo real.

El acceso a dicha librería se puede realizar de dos formas: directamente desde Simulink o desde la línea de comandos de Matlab mediante el comando **`rtlib`**. Puesto que el acceso a la RTLIB desde Simulink difiere dependiendo de la versión de Matlab/Simulink, accederemos a dicha librería mediante el comando **`rtlib`**. Al ejecutar dicho comando aparecerá un ventana de Simulink como la que se muestra en la siguiente figura:

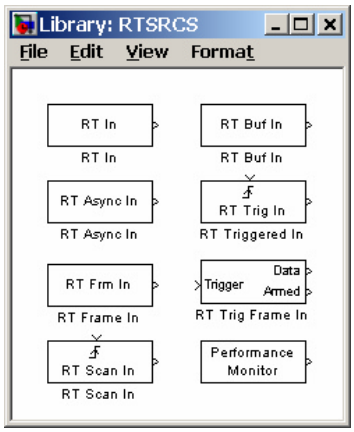


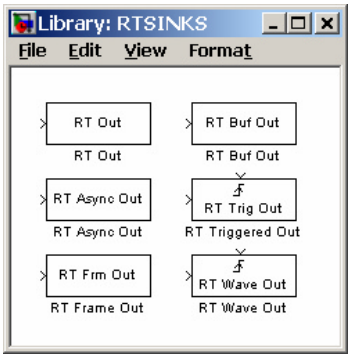
A partir de aquí abriremos un nuevo esquema Simulink y podemos comenzar a trabajar con los bloques de la RTLIB. El primer paso es incluir en nuestro esquema un bloque adaptador. El bloque A*dapter* es un bloque que representa la tarjeta de adquisición de datos y hace las veces de driver o interfaz entre Simulink y la tarjeta. Se utiliza para disponer dentro del esquema Simulink de los canales de la tarjeta de adquisición de datos. Una vez incluido dicho bloque en el esquema Simulink, lo primero que hay que hacer es configurar el mismo para que utilice el driver de la tarjeta que poseemos. Para ello haremos doble clic con el ratón sobre el mismo y se abrirá una ventana como la que se muestra:



En nuestro caso, el driver que debemos cargar es el correspondiente al fichero PCL812PG.RTD, para ello lo seleccionamos y pulsamos *'Aceptar'*. A partir de este momento ya tenemos cargado en nuestro esquema Simulink los drivers de la tarjeta ACL – 8112PG con las opciones por defecto. En el caso de querer cambiar dichas opciones haremos de nuevo doble clic sobre el bloque *Adapter* y se abrirá una ventana de selección de parámetros. En principio y a menos que se indique lo contrario, se trabajará con las opciones del driver por defecto por lo que no será necesario realizar este último paso (se pulsará OK sin cambiar ningún parámetro).

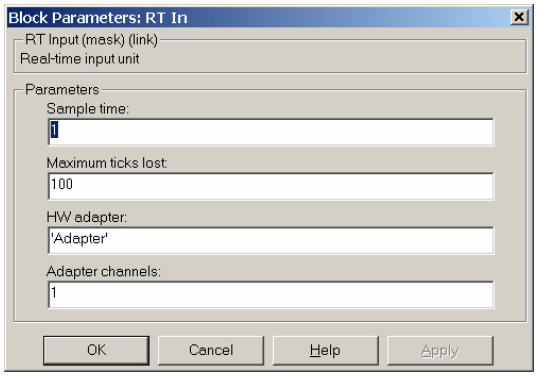
Ahora ya podemos empezar a utilizar los bloques de entrada/salida de la librería. Se dispone de dos tipos de bloques: *sources* y *sinks*. Para acceder a los mismos haremos doble clic sobre los bloques *Real – Time Sources* o *Real – Time Sinks* respectivamente. De esta forma aparecerán las ventanas correspondientes tal y como se muestra en la siguiente figura:





Como se puede apreciar aparecen multitud de bloques tanto en una como en otra ventana. En este documento sólo se van a analizar aquellos que se van a utilizar en la práctica. Dichos bloques son *'RT In'* y *'RT Out'* ya que son los mejores para realizar el control en tiempo real.

El bloque *'RT In'* está diseñado para la adquisición de datos en tiempo real. Para configurar este bloque haremos doble clic sobre el mismo, apareciendo así una ventana como la que se muestra a continuación:



Veamos los parámetros configurables:

- *Sample Time:* especifica el periodo de muestreo al cual se debe realizar la adquisición de los datos.
- *Maximum ticks lost:* especifica el máximo número de 'tics' (muestreos) que se pueden perder antes de mostrar un mensaje de error. Esto es un mecanismo de "seguridad" para asegurar que la adquisición se ha realizado de forma correcta (no ha habido excesivas pérdidas de datos) ya que hay que tener en cuenta que este sistema en tiempo real se ejecuta sobre Matlab y Simulink, y éstos a su vez sobre Windows por lo que cabe la posibilidad de que en un determinado instante el sistema esté tan cargado que no se pueda realizar la adquisición con la frecuencia correcta. Esto, claro está, depende tanto de la potencia del microprocesador como de la cantidad de recursos de la máquina que están ocupados (aplicaciones, ventanas abiertas, etc.).
- *HW adapter:* indica el nombre del bloque *Adapter* incluido en el esquema Simulink. Esto es así pues cabe la posibilidad de tener en un mismo esquema más de un adaptador para gestionar dos sistemas de adquisición distintos.
- *Adapter channels:* especifica el canal analógico de entrada al que se quiere acceder. En este caso tendremos 8 canales numerados del 1 al 8 correspondiendo a los canales de entrada analógica 0 a 7 de la tarjeta de adquisición.

Los parámetros a fijar en el bloque *RT Out* son idénticos a los anteriores (y por tanto la ventana de configuración) con la salvedad de que en este caso se trata de un bloque de salida y se emplea para generar las señales necesarias con el mínimo retardo posible. En este caso, el parámetro *Sample Time* especifica el periodo al cual se actualiza el canal de salida correspondiente en la ACL – 8112PG, *Maximum ticks lost*, el número de veces que no se ha podido actualizar dichos canales debido a la carga del sistema, *HW adapter* es idéntico al caso anterior, y *Adapter channels* especifica el canal de salida analógica al que se accede que en este caso son 2, numerados del 1 al 2 correspondiendo a los canales de salida analógica 0 a 1 de la tarjeta ACL – 8112PG.

3. NIVELES DE TENSIÓN UTILIZABLES

Tal y como está configurada la tarjeta, los rangos de tensión utilizables son los siguientes:

- Tensiones de entrada aceptables: **entre –10 y +10 V**
  Esto permite medir las principales tensiones devueltas por los sensores de la tarjeta, tanto los potenciómetros como el tacogenerador.
- Tensiones de salida generables: **entre 0 y 10V**
  Esto limita un poco las posibilidades de accionamiento del motor, ya que sólo se podrá hacer girar en un sentido. Para hacerlo girar en ambos sentidos sería necesario disponer de señales de salida positivas y negativas. No obstante, para esta práctica será suficiente con la señal de la que se dispone.

En cualquier caso, el rango de tensiones de salida y de entrada soportables por la tarjeta se transforma en una rango de –1 a 1 en los bloques del RTT. Esto implica realizar transformaciones entre los valores que aparecen en Simulink y las tensiones reales que se obtendrán en la tarjeta.

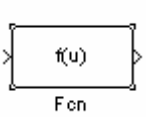
Así, si necesitamos generar una tensión de 2.5 voltios (en una escala de 0 a 10) en un canal de salida de la tarjeta, el valor que debemos indicar al bloque de salida *RT Out* de Simulink será un valor de -0.5 (el equivalente al valor anterior en una escala de –1 a 1). De esta forma es necesario realizar antes del bloque de salida la transformación siguiente:

$$V_{simulink} = -1 + \frac{V_{tarjeta}}{5}$$

De la misma forma, si leemos desde un bloque de entrada *RT In* un valor de 0.5 (en una escala de –1 y 1), éste corresponderá a una tensión de 5 V (en una escala de –10 a +10). Por tanto es necesario realizar después del bloque de entrada la transformación siguiente:

$$V_{tarjeta} = V_{simulink} \cdot 10$$

Estas transformaciones se realizarán en Simulink mediante un bloque de función *Fcn* (librería *'Functions & Tables'* o *'User-defined functions'*, según las versiones de Matlab):



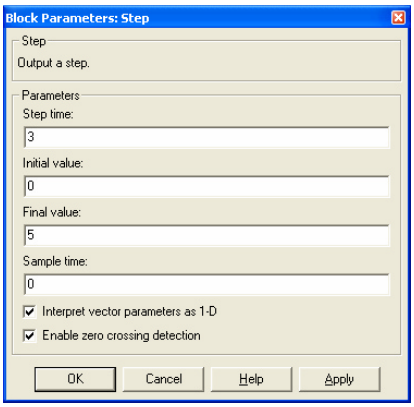
Este bloque realiza una operación matemática cualquiera entre la señal de entrada y la señal de salida. La señal de entrada se denomina **u**; en nuestro caso **u(1)** dado que es un dato unidimensional. Las funciones que se deben realizar son:

- Antes del bloque de salida: (u(1)/5)-1
- Después del bloque de entrada: u(1)*10

Además (por errores del driver de la tarjeta ACL – 8112PG) cada vez que se lanza una simulación (en este caso una simulación equivale a una ejecución sobre el sistema real) las salidas de la tarjeta no se ponen inicialmente a 0V sino a 5V. Esto puede suponer un problema si la salida de la tarjeta se utiliza, por ejemplo, para accionar el motor. En este caso, si se lanza una simulación en la que se desea que el motor inicialmente esté parado (tensión sobre el motor 0V), el efecto será el siguiente:

- se lanza la simulación
- antes de comenzar la misma, la tarjeta se inicializa automáticamente y genera una tensión de 5V en sus salidas, durante aproximadamente 1 segundo
- durante este segundo el motor adquiere una determinada velocidad de giro
- cuando termina la inicialización, comienza la simulación; en el caso que planteamos se aplican 0V al motor, pero éste no se detendrá inmediatamente, por lo que será necesario esperar durante aproximadamente 3 segundos antes de poder empezar a tomar datos con el motor parado.

En la práctica lo que haremos será dejar transcurrir un tiempo de 3 segundos antes de aplicar cualquier señal al sistema. Como lo más habitual será utilizar señales de tipo escalón, basta con hacer que el escalón actúe en el instante t=3sg. Por ejemplo para una referencia en escalón de 5 unidades la configuración de dicho bloque será:



De esta forma estamos indicando que el valor inicial será 0 y que a los tres segundos el valor inicial será el deseado, es decir, 5. El parámetro *Sample Time*, si apareciera, puede permanecer a 0 ya que el periodo de muestreo al que se va a realizar la conversión se especifica en el bloque *RT Out*.

4. PRIMER EJEMPLO: LECTURA DE DATOS DE SENSORES

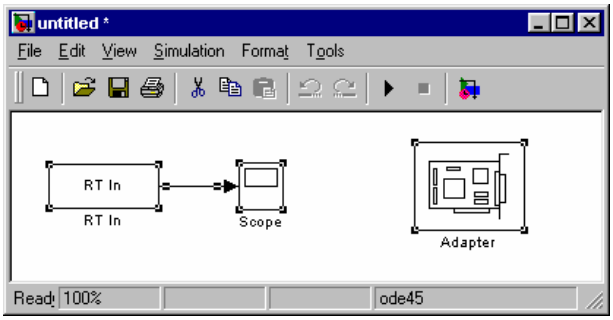
Como primer ejemplo, se construirá un esquema Simulink que permita recoger en el PC las señales ofrecidas por los sensores de la maqueta, a través de los canales de adquisición de datos de la tarjeta.

En particular, conectaremos la señal del potenciómetro de referencia (sensor de posición angular del eje que gira libremente) al canal cero de la tarjeta. Esto supondrá hacer las conexiones siguientes:

- El conector **'Potenciómetros: eje referencia'** de la regleta del motor se conectará al **pin 1** de la tarjeta (positivo del 1er canal de entrada analógica)
- El conector **'Fuente de alimentación: 0V'** se conectará al **pin 9** de la tarjeta (masa de las señales de entrada analógica)

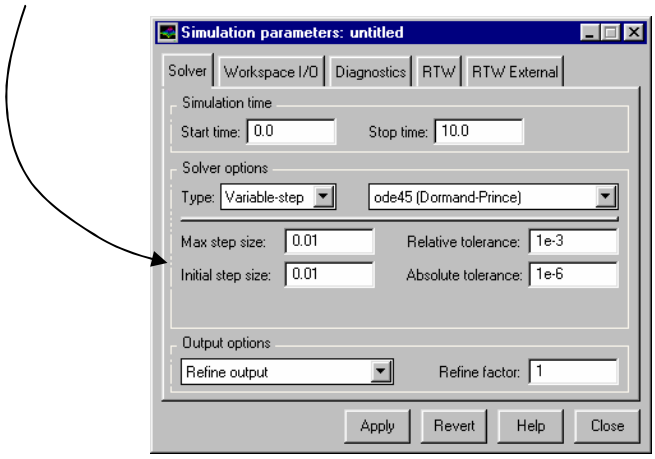
De este modo, el 1er canal de entrada analógica de la tarjeta estará adquiriendo continuamente los valores de tensión que le devuelva el sensor.

A continuación se debe construir el esquema Simulink que permita recoger esos datos desde la tarjeta y visualizarlos en el PC. El bloque a utilizar para recoger datos es el bloque RT In (entrada de datos desde la tarjeta al PC) y los datos los visualizaremos mediante un osciloscopio, tal y como muestra el siguiente esquema:



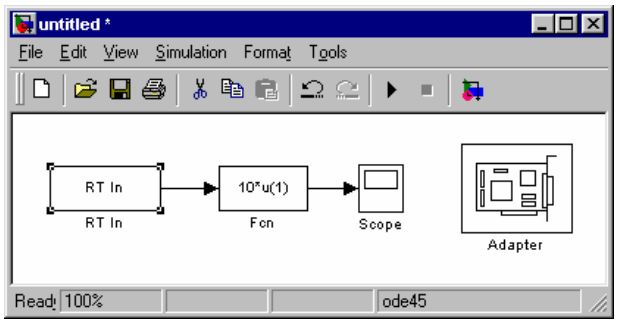
Antes de lanzar la simulación se deberán ajustar los parámetros de los bloques y de la propia simulación:

- Para el bloque **RT In** se fijarán los siguientes parámetros:
  - periodo de muestreo (sample time) = 0.01 seg. Se recogerán datos del sensor cada 0.01 seg.
  - Canal de entrada (adapter channel) = 1. Se recogerán datos del primer canal de entrada (es donde se ha conectado la señal)
- Para la **simulación**, se fijarán algunos parámetros particulares que supondrán un mejor funcionamiento:
  - **Max step size** se fijará a 0.01 seg (el mismo valor que el periodo de muestreo elegido)
  - **Initial step size** se fijará también a 0.01 seg.



Ahora encenderemos la fuente de alimentación de la maqueta y lanzaremos la simulación. Debemos observar cómo al girar a mano el eje correspondiente al potenciómetro de referencia, el osciloscopio del esquema Simulink refleja tales movimientos. Será conveniente lanzar simulaciones con tiempos de simulación del orden de 20 seg. o más para tener tiempo suficiente para realizar pruebas.

Si nos fijamos en los valores que aparecen el en osciloscopio podremos comprobar como no se corresponden con los valores reales de tensión que devuelve el potenciómetro. La razón es que no hemos incluido la función que transforma valores Simulink a valores reales sobre la tarjeta. A continuación modificaremos el esquema para incluir esta función. Recordemos que el bloque a utilizar es *Fcn* de la librería *Functions & Tables* y la función a introducir es (u(1)*10). El aspecto del esquema debe ser el siguiente:



Si se lanza de nuevo la simulación se podrá comprobar cómo ahora los datos si se reflejan correctamente en el osciloscopio (deben variar aproximadamente entre –10V y 10V para las posiciones extremas del eje).

Como segunda prueba, se mantendrá el mismo esquema Simulink pero se modificarán las conexiones (se recuerda que antes de modificar ninguna conexión se deberá apagar la fuente de alimentación de la maqueta). Lo que pretendemos en esta segunda prueba es leer no la señal de posición del eje sino la señal de velocidad del motor. Se hará una conexión similar a la anterior pero en lugar de utilizar el conector **'Potenciómetros: eje de referencia'** se utilizará **'Tacogenerador: señal positiva'**.

En este caso, al lanzar la simulación deberemos mover el motor para poder comprobar sobre el osciloscopio de Simulink cómo realmente se están adquiriendo datos de velocidad de giro del mismo. Para este ensayo se moverá el motor haciendo uso de los botones de test.

5. PRIMER EJERCICIO: INTEGRACIÓN SEÑAL VELOCIDAD

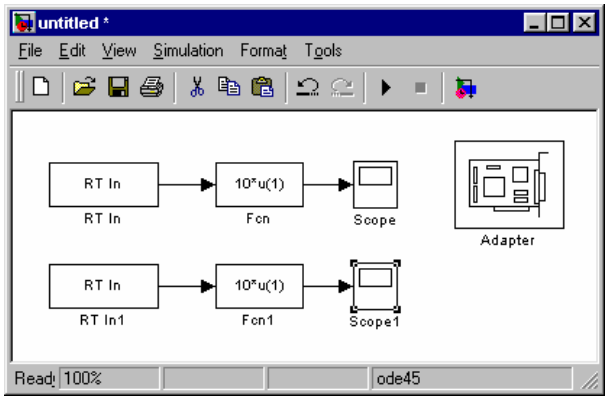
En los ejemplos anteriores se ha comprobado cómo es posible leer desde Simulink datos de los sensores de la maqueta. Como ejercicio se deberán leer dos valores simultáneamente:

- posición del eje del motor (conector **'Potenciómetros: eje motor'**)
- velocidad de giro del motor (conector **'Tacogenerador: señal positiva'**)

Para ello se harán las siguientes conexiones:

- El conector **'Potenciómetros: eje motor'**se conectará al **pin 1** de la tarjeta (positivo del 1er canal de entrada analógica)
- El conector **'Tacogenerador: señal positiva'**se conectará al **pin 2** de la tarjeta (positivo del 2er canal de entrada analógica)
- El conector **'Fuente de alimentación: 0V'** se conectará al **pin 9** de la tarjeta (masa de las señales de entrada analógica)

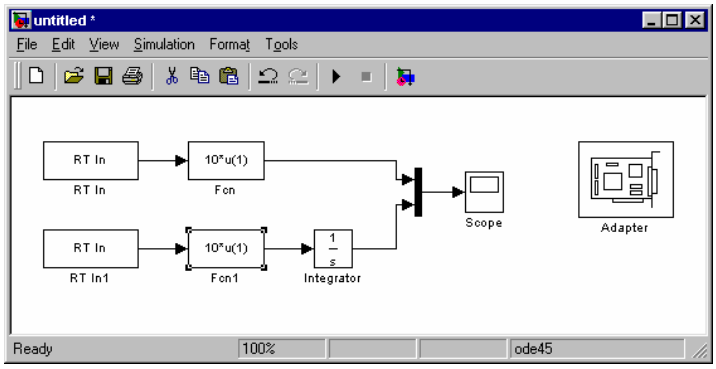
El esquema Simulink a crear será similar al anterior, con la diferencia de que se leen dos señales en vez de una, tal y como se muestra en la figura:



Es importante fijar correctamente los parámetros de los dos bloques *RT In*, uno de ellos debe referirse al *Adapter Channel* 1 y el otro al *Adapter Channel* 2.

Si el montaje funciona correctamente, en la simulación se debe apreciar cómo una señal (la posición) es aproximadamente la integral de la otra (la velocidad). Usaremos esta característica para generar una señal de posición angular ***'artificial'*** mediante la integración de la señal de velocidad angular, suponiendo que trabajásemos sobre un sistema que no dispusiera de sensores de posición.

Lo que debemos hacer es situar un bloque integrador a continuación de la señal de velocidad. Además se llevarán las dos señales al mismo osciloscopio (la señal de posición obtenida a partir del potenciómetro y la señal de posición generada a partir de la velocidad) mediante un bloque multiplexor (*'Mux'* de la categoría *'Signals and Systems'* o *'Signal Routing'*, según las versiones de Matlab) de modo que se puedan comparar sus valores. La siguiente figura muestra el diagrama creado:



Antes de lanzar la simulación, se moverá el motor con el botón de test hasta llevarlo a la posición cero (la aguja indicadora en el punto más alto). A continuación se lanzará la simulación y se moverá el motor mediante los botones de test en uno y otro sentido, con cuidado de no sobrepasar los extremos (+180º y –180º) ya que la señal del potenciómetro no es válida en esos puntos. Aún teniendo esta precaución, veremos que las señales no son iguales. Esto es debido a que las **constantes** del sensor de posición y del sensor de velocidad son completamente distintas. Como ejercicio se deberá introducir a continuación del integrador un bloque de ganancia (*'Gain'*, categoría *'Math'*) ajustando su valor hasta que las señales original y generada coincidan aproximadamente.

Ejercicio:

- Apuntar el valor de la ganancia para el cual la señal generada se corresponde con la real.
- Lanzar una simulación en la que se compruebe que el ajuste es correcto.

6. SEGUNDO EJEMPLO: ACCIONAMIENTO DEL MOTOR DESDE EL PC

Hasta ahora los experimentos realizados se han limitado a adquirir datos de la maqueta desde el PC. En este segundo ejemplo se hará la operación contraria: se enviarán datos hacia el motor (tensiones a aplicar entre sus terminales).

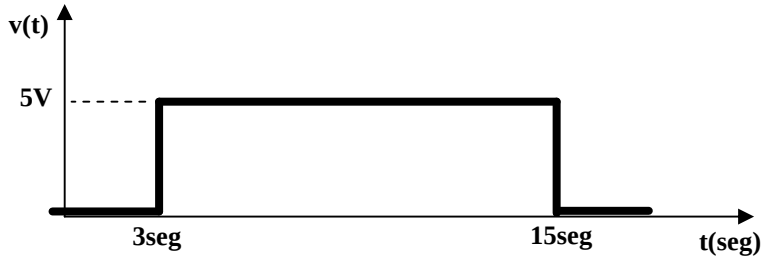
Lo que deseamos hacer es introducir entre los terminales del motor la tensión indicada en uno de los canales de salida de la tarjeta. Las conexiones a hacer para ello serán (se recomienda eliminar antes las conexiones del ejercicio anterior... y sobre todo apagar la fuente de alimentación de la maqueta):

- El conector **'Amplificador: entrada positiva'** se conectará al **pin 30** de la tarjeta (positivo del 1er canal de salida analógica)
- El conector **'Fuente de alimentación: 0V'** se conectará al **pin 38** de la tarjeta (masa de los canales de salida analógica)

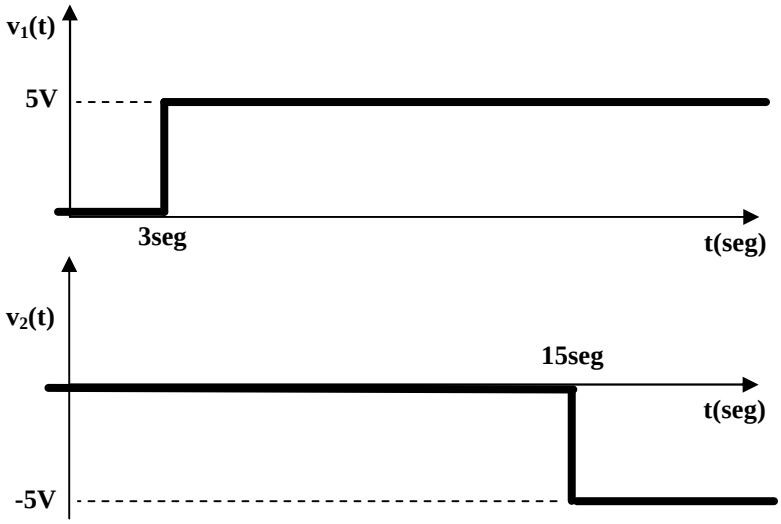
De este modo, el valor que se mande hacia la primera salida analógica de la tarjeta será el valor de tensión que se aplique al motor en cada momento.

A continuación se debe construir el esquema Simulink que permita enviar los datos deseados desde el PC hacia la tarjeta. El bloque a utilizar para enviar datos es el bloque RT Out (envio de datos desde el PC hacia la tarjeta).

También tenemos que añadir los bloques capaces de generar la señal a ser introducida al motor. Supongamos que en este primer ejemplo queremos introducir una señal de tensión como la mostrada en la siguiente figura:



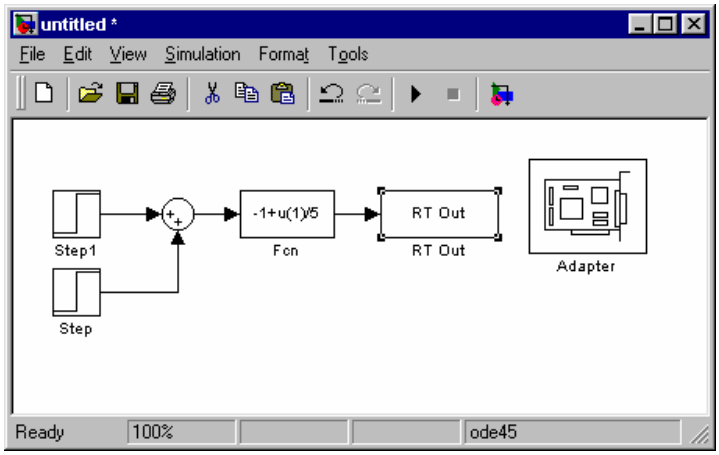
Esta señal representa empezar a enviar tensión al motor tres segundos después del arranque de la simulación y dejar de enviar tensión a los 15 segundos. La forma más sencilla de generar señales de este tipo es mediante descomposición en señales más sencillas. En este caso podemos obtener la señal como suma de dos escalones: $\mathbf{v(t)=v_1(t)+v_2(t)}$



Mediante este procedimiento o procedimientos similares podremos generar las señales que deseemos para introducir al motor. Se deberán respetar dos condiciones:

- Durante los primeros segundos se debe introducir una señal de cero voltios, para dar tiempo al motor a detenerse después de la inicialización de la tarjeta (este problema con el driver se ha comentado anteriormente)
- Antes del término de la simulación deberá volverse a situar la tensión en cero voltios, porque si no el motor seguiría girando indefinidamente (la tensión nunca volvería a cero)

En concreto, el diagrama Simulink necesario para generar la señal indicada e introducirla al motor incluiría los dos escalones más la función de conversión de valores adecuada. Recordemos que el bloque a utilizar es *'Fcn'* (de la librería *'Functions & Tables* o *'User-defined functions'*, según las versiones de Matlab*)* y la función a introducir en esta caso es (u(1)/5)-1. El aspecto del esquema debe ser el siguiente:



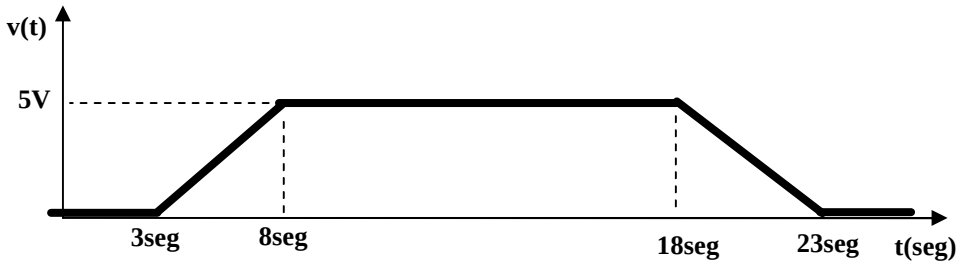
Antes de lanzar la simulación se deberán ajustar los parámetros de los bloques:

- Para el bloque **RT Out** se fijarán los siguientes parámetros:
  - periodo de muestreo (sample time) = 0.01 seg. Se enviarán tensiones al motor cada 0.01 seg.
  - Canal de salida (adapter channel) = 1. Se enviarán los datos al primer canal de salida (es donde se ha conectado el motor)
- Para cada uno de los bloques **Step**, se fijarán los parámetros de forma que la señal que produzcan sea la deseada. Se recomienda probar el resultado de la suma de los escalones lanzando una primera simulación con la fuente de alimentación del equipo apagada y conectando un osciloscopio (bloque *'Scope'*) en Simulink para visualizar la señal.

Una vez todo correcto, se lanzará la simulación e inmediatamente después se encenderá la fuente de alimentación de la maqueta (en este orden). El resultado debe ser que el motor girará de acuerdo con la señal enviada.

7. SEGUNDO EJERCICIO: ARRANQUE Y PARADA SUAVES

El objetivo es lograr que el motor responda a una curva de funcionamiento que le proporcione un arranque y una parada suaves. Para ello la señal aplicada a sus terminales no puede presentar cambios bruscos. La señal a aplicar será la siguiente:



Notas:

- La señal se puede generar como suma de cuatro señales rampa.
- Se deben representar en un mismo osciloscopio tanto la señal de la tensión aplicada al motor como la señal de la velocidad (obtenida a través del conector **'Tacogenerador: señal positiva'**)
- Por lo tanto será necesario incluir en el diagrama tanto un bloque RT Out como un bloque RT In, y hacer las conexiones correspondientes para la entrada y la salida.

Ejercicio:

- Conseguir el arranque y parada suaves del motor.
- Llevar el gráfico de las dos señales (velocidad y tensión) a Matlab.

ANEXO: SEÑALES DISPONIBLES EN LA REGLETA DE CONEXIONES

SEÑAL		**Nº conector**
Encoder absoluto:	bit 0	1
	bit 1	20
	bit 2	2
	bit 3	21
	bit 4	3
	bit 5	22
Encoder incremental:	señal desfasada	4
	señal sin desfasar	23
	referencia	5
Comando PWM		24
Activa/desactiva fallos:	señal+10V	6
	potenciómetro	25
	amplificador	7
	tacogenerador	26
	encoder incremental	8
	encoder absoluto	27
Fuente de alimentación:	-15 V	9
	+15 V	28
	-10 V	10
	+10 V	29
	0 V	11
	+5 V	30
Potenciómetros:	eje motor	12
	eje referencia	31
	referencia-motor (señal de error)	32
Amplificador:	entrada negativa	14
	entrada positiva	33
	señal cero	34
Tacogenerador:	señal negativa	16
	señal positiva	35
Generador de señales:	señal triangular	17
	señal cuadrada	36