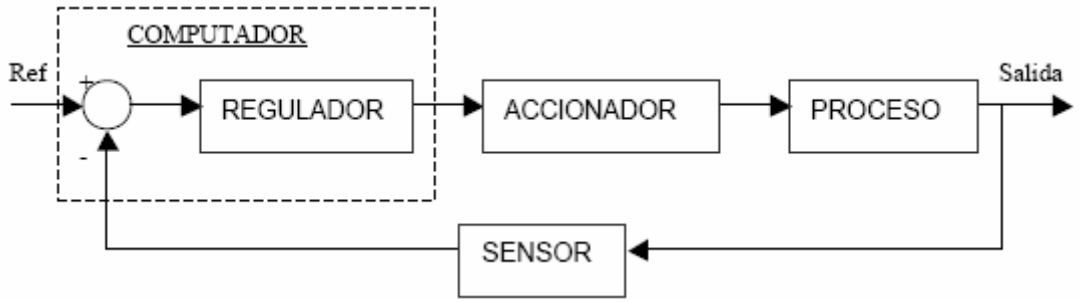
Práctica 2

Identificación de un servomotor con computador

Descripción del sistema experimental

Esquemáticamente, el diagrama de bloques general de un sistema de control por computador es el siguiente:



El funcionamiento del lazo de control anterior es el siguiente: el operador genera una señal de consigna o referencia que será utilizada por el regulador para que la salida del sistema coincida con dicha señal. El regulador a partir de dicha consigna, de la salida del sistema y con el algoritmo de control adecuado, genera una acción de control que pasa al accionador que se encargará de adecuar dicha señal a las características de la entrada del proceso. Un sensor lee la salida del sistema y la traslada al controlador.

Tomando como referencia el esquema de bloques anterior, identifiquemos los componentes físicos correspondientes que se van a utilizar en esta práctica:

a) Proceso: el proceso es el servomotor de corriente continua Feedback.

b) Accionador: está incluido en el servomotor y es la electrónica necesaria generar la señal necesaria que actúa sobre el motor.

c) Sensor: el sensor en este caso también se encuentra englobado en el mismo proceso. Los sensores de que se dispone son entre otros un encoder que nos proporcionan la posición del eje del servomotor y un tacogenerador que nos proporciona la velocidad angular de giro del eje del motor. Las señales proporcionadas por ambos sensores son valores de tensión, que se pueden medir directamente con el sistema de adquisición de datos. Además, dichos sensores son de ganancia unitaria y no presentan ninguna dinámica asociada por lo que tampoco es necesario contemplarlos en la identificación del proceso.

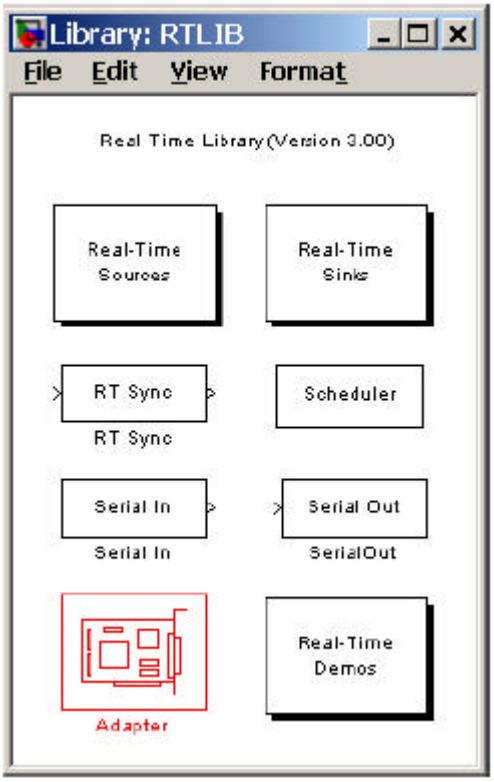
d) Regulador: el regulador en este caso está formado por el computador (con su software asociado) y el sistema de adquisición de datos, que en este casó será una tarjeta de adquisición de datos ACL-8112PG.

Comunicación con la tarjeta de adquisición de datos

El paquete *Extended Real Time Toolbox* (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, elegimos 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* (RTLIB) y provee los bloques Simulink necesarios para comunicar el entorno 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, accedemos a dicha librería mediante el comando **rtlib**.



Por tanto, desde la ventana de Matlab, ejecutamos el comando:

```
>>rtlib
```

y aparece la ventana de la izquierda.

El bloque *Adapter* es el que vamos a utilizar y representa la tarjeta de adquisición de datos, haciendo las veces de driver o interfaz entre Simulink y la tarjeta.

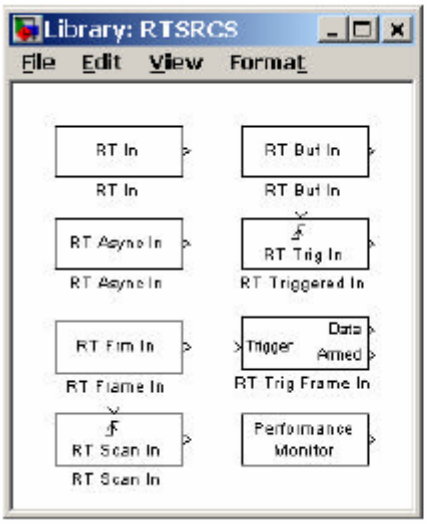
Pinchamos el citado bloque con el ratón y lo arrastramos hacia la ventana de Simulink, que hemos abierto previamente.

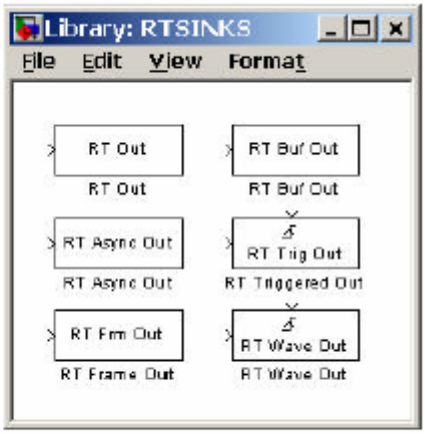
A continuación, debemos configurar este bloque para que utilice el driver de la tarjeta que tenemos.

Para ello, hacemos doble clic con el ratón sobre el bloque y se abre una ventana con una serie de ficheros. En nuestro caso, seleccionamos el fichero PCL812PG.RTD y pulsamos “Aceptar”. A continuación se abre una ventana de selección de parámetros.

Puesto que vamos a trabajar con las opciones del driver por defecto, no es necesario cambiar ningún parámetro.

Una vez hecho esto, ya podemos utilizar los bloques de entrada/salida de la librería. Existen dos bloques: sources y sinks, a los que se puede acceder haciendo clic sobre los bloques Real ~ Time Sources o Real ~ Time Sinks, respectivamente. De esta forma, aparecen las siguientes ventanas.





De todos estos bloques los que vamos a utilizar son: *RT IN* y *RT OUT*.

Configuración de *RT IN*

El bloque *'RT In'* está diseñado para la adquisición de datos en tiempo real, el procesamiento de señales, y las aplicaciones de control donde los datos han de ser procesados tan rápido como sea posible. Para configurar este bloque haremos doble clic sobre el mismo, apareciendo así una ventana como la que se muestra a continuación:

Los parámetros configurables son:

Sample Time: especifica el periodo de muestreo al cual se debe realizar la adquisición de los datos. Se permite especificar un vector en el caso de que se quiera acceder a más de un canal con el mismo bloque. Asimismo permite establecer un *offset*. En nuestro caso, el periodo de muestreo es 0.01 segundos, lo que equivale a 100 muestras por segundo.

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.

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 (por ejemplo uno de entradas analógicas y otro de salidas analógicas independientes).

Adapter channels: especifica el canal analógico de entrada al que se quiere acceder. En este caso, puesto que son canales de entrada analógica tendremos 8 (ya que la ACL – 811PG tiene 8) numerados del 1 al 8 correspondiendo a los canales de entrada analógica 0 a 7 de la tarjeta de adquisición. Es posible especificar un vector de canales por si se quiere acceder a más de uno de ellos desde el mismo bloque. En nuestro caso, seleccionamos el canal 1, que corresponde a la entrada analógica 0 de la tarjeta de adquisición.

Configuración de *RT OUT*

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.

Los parámetros con los que debemos configurar este bloque son exactamente los mismos que antes, incluido el "*channel Adapter*".

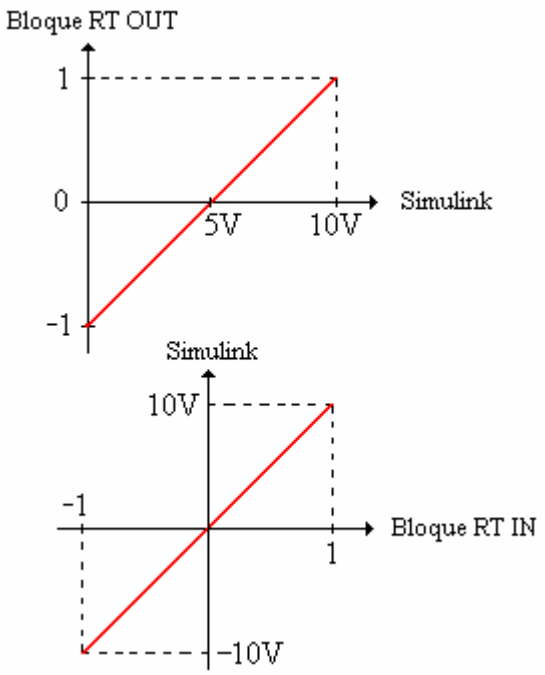
Sample Time: 0.01
Maximum ticks lost: 100
HW adapter: 'Adapter'
Adapter Channel: 1 (Salida analógica 0 de la tarjeta)

Aspectos a tener en cuenta

Antes de realizar las pruebas de adquisición y generación de señales es importante tener en cuenta algunos aspectos.

a) En primer lugar, hay que decir que el rango de tensiones de salida y de entrada soportables por la tarjeta (en nuestro caso 0 a 10V y –10V a 10V) se transforma en un rango de –1 a 1 en los bloques del RTT. Esto implica realizar una transformación tanto de la señal que leemos del bloque *RT IN* como de la señal que enviamos al bloque de salida *RT OUT*.

Evidentemente, esta transformación es lineal.



$$y = -1 + \frac{X}{5}$$

$$y = 10 \cdot X$$

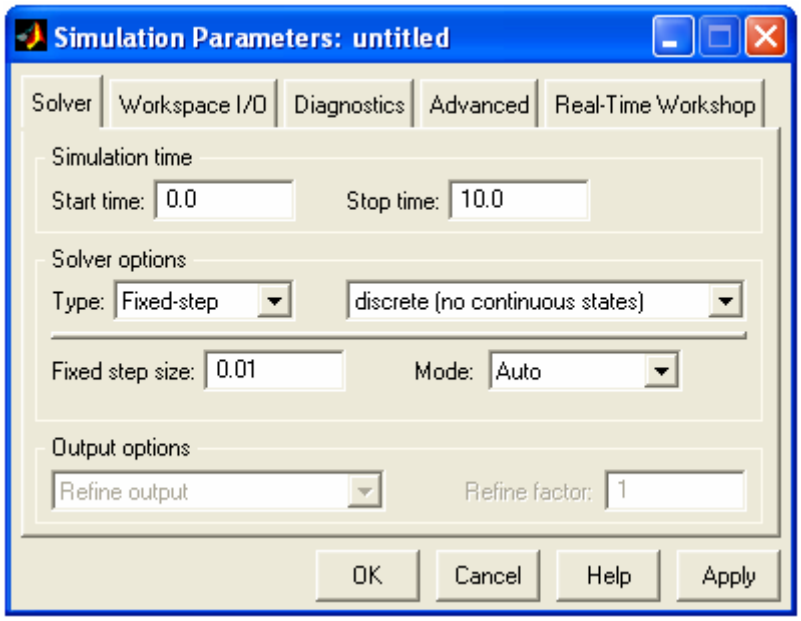
Estas transformaciones se realizan en el esquema Simulink mediante un bloque de función *Fcn*, situado dentro de la librería *Functions & tables*, de forma que las funciones que se deben realizar son, como hemos visto antes, las siguientes:

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

b) Por problemas, errores o imperfecciones del driver de la tarjeta ACL – 8112PG, el modo en que el sistema inicializa la tarjeta cuando se inicia una simulación (en este caso la ejecución en tiempo real del esquema Simulink) es poniendo las salidas analógicas de la misma a una tensión de 5 voltios en vez de a los 0 voltios esperados. Por esto, cuando ejecutemos el esquema Simulink, es necesario que la señal de referencia empiece a actuar unos dos segundos después de iniciar el programa, cuando el valor de la salida se haya anulado. Para ello, los bloques que generan esta señal de referencia (por ejemplo un escalón que es lo más habitual) deben comenzar con un valor inicial de 0 y a los dos segundos generar el valor de referencia deseado.

c) Todos los bloques del esquema Simulink son de naturaleza discreta, por lo que deben tener el mismo periodo de muestreo que los bloques de entrada y salida.

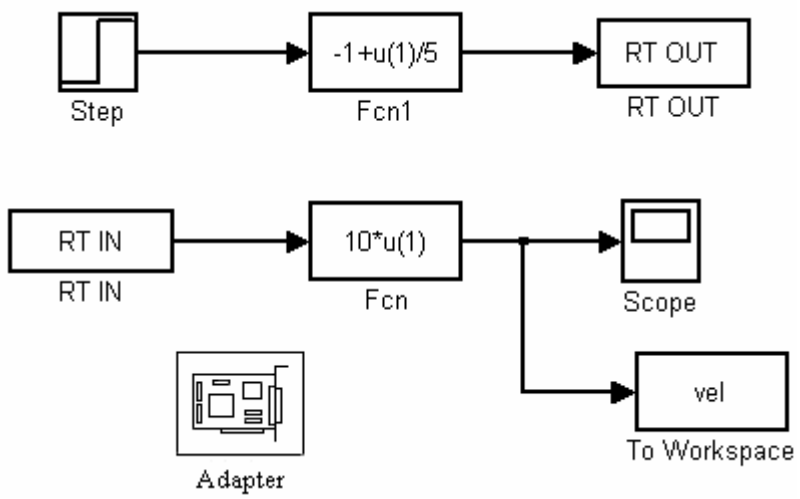
RT IN: Step time: 0.01
RT OUT: Step time: 0.01
To workspace: Step time: 0.01
Scope: Step time: 0.01
Además, los parámetros de la simulación son los siguientes:



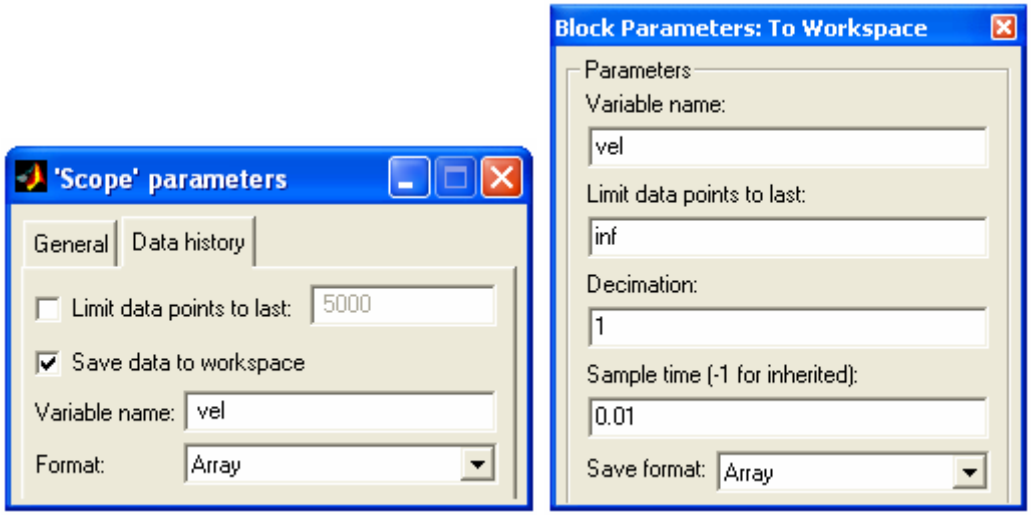
Ejemplo

Este primer ejemplo consiste en adquirir datos por el canal 0 de entrada analógica y generar una tensión de 3 voltios por el canal 0 de salida analógica (recordemos que ambos canales se denominan 1 en los bloques de RTT). El periodo de muestreo debe ser de 0.01 seg. tanto para la entrada como para la salida de datos. Además, se desea visualizar la señal adquirida mediante un osciloscopio.

El esquema Simulink es el siguiente:



Los parámetros que debemos configurar, además de los comentados anteriormente, son los del bloque *Scope* y *To Workspace*.



Para visualizar los resultados tenemos dos opciones:

Forma A: Utilizando el bloque *Scope*.

Con este método, el array "vel" es una matriz de dos columnas: en la primera columna se almacena el tiempo y en la segunda, el valor de la salida correspondiente a dicho instante de tiempo.

- Si queremos visualizar el valor de la salida en cada instante de muestreo (tiempo).

```
>>plot(vel(:,1), vel(:,2));
```

- Si queremos visualizar el valor de la salida en cada índice de muestreo (número de muestra).

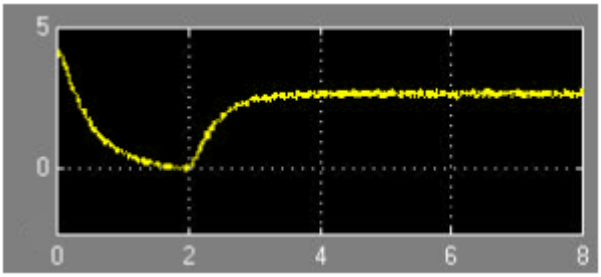
```
>>plot(vel(:,2));
```

Forma B: Utilizando el bloque *To workspace*.

Con este método, el array "vel" es una única fila que contiene los valores de la salida en el orden en el que se van muestreando, por lo que sólo podemos visualizar el valor de la salida en función del orden de las muestras y no en función del tiempo.

```
>>plot(vel);
```

Resultados

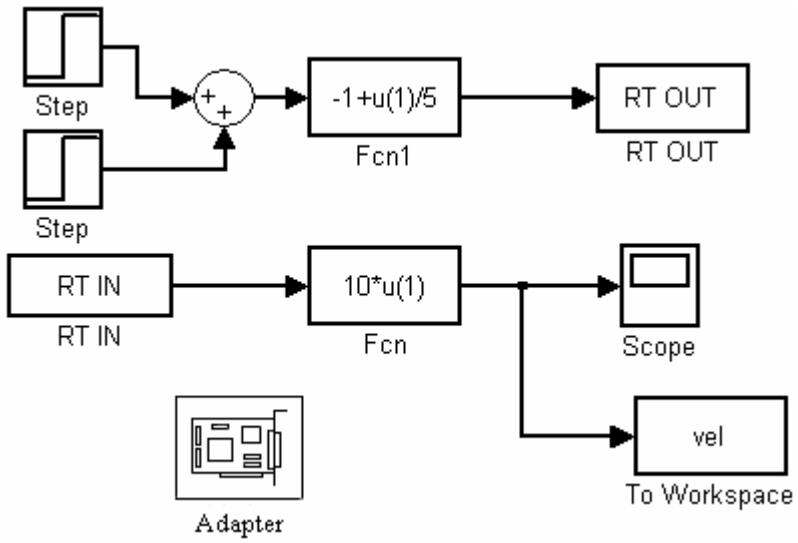


Como vemos, debido a un error en los drivers, inicialmente el valor de la salida es la correspondiente a la un escalón de entrada de 5 voltios. Casi inmediatamente dicho escalón pasa a cero voltios de valor inicial de la referencia y el motivo de esperar dos segundos a generar el escalón de 3 voltios es esperar a que el motor pare. Transcurridos dos segundos aparece el escalón de 3 voltios a la entrada del motor y se genera la salida que se observa en la gráfica anterior. Si recogemos dicha señal en un bloque *To Workspace* podemos representar la señal mediante el comando *plot*, como hemos comentado antes. Además, puesto que sabemos el número de muestras que se recogen en los 2 segundos iniciales, podemos eliminar del vector en el que se almacena la señal los datos correspondientes a los 2 primeros segundos. En este caso, el número de muestras recogidas en los dos primeros segundos es 2/0.01 = 200. Puesto que la ejecución ha durado 8 segundos, se han recogido en total 800 muestras. Para eliminar las 200 primeras muestras de la variable en la que están almacenadas, desde la línea de comandos de Matlab haremos lo siguiente:

```
>>vel = vel(201:800);
```

De esta forma ya se pueden analizar y representar los datos como si se hubiera realizado el escalón de referencia en el instante cero de la ejecución.

Otro aspecto a destacar es que la tarjeta de adquisición de datos, una vez ejecutado el programa, mantiene en sus canales de salida el último valor (en el caso anterior 3 voltios) y es recomendable (por no decir necesario) dejar el valor de la salida de estos canales a 0 voltios. Para ello, debemos modificar la señal de referencia. Suponiendo que la ejecución (tiempo de simulación) dura 8.5 segundos, la referencia a generar será la suma de las señales de dos bloques escalón. El primero será idéntico al anterior, y el otro será un escalón con valor inicial 0, y a los 8 segundos generará un valor final opuesto en signo al valor final del primero (en nuestro ejemplo –3 voltios). De esta forma, al ser la referencia la suma de ambos hará que a los ocho segundos la referencia sea 0 y por tanto se permita parar el motor. El esquema Simulink resultante sería:

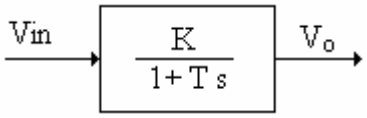


Al igual que en el caso anterior, puesto que conocemos el número de muestras que se toman, podemos eliminar aquellas que no nos interesan.

No obstante, puesto que únicamente nos interesa el modo de operación normal, tener en cuenta este tipo de aspectos sólo es importante cuando se desea analizar los datos obtenidos, ya que para el modo de operación normal, los instantes iniciales y finales apenas importan y más cuando el tiempo de ejecución es muy elevado o infinito.

Identificación de un sistema de primer orden

El sistema que vamos a identificar es el siguiente:



Esquema de conexiones

El esquema de conexiones es el siguiente:

Tarjeta ordenador (ACL–8112PG)	Tarjeta servomotor (ADAM-3937)
pin 1 (canal 0 de entrada analógica)	pin 35 (TACHO+)
pin 10 (GND de entrada analógica)	pin 11 (0 V)
pin 30 (canal 0 de salida analógica)	pin 33 (PA+INPUT)
GND (GND de salida analógica)	pin 14 (PA–INPUT)

La señal TACHO+ proporciona una tensión entre 0 y 10 voltios proporcional a la velocidad del motor. La masa de dicha señal se tomará del propio servomotor ya que de esta forma se obtiene una señal mucho más “limpia” pues la electrónica de la caja siempre introduce algo de ruido.

La señal PA+ INPUT (pin 33) es la entrada al servomotor y es donde se aplica, por tanto, la acción de control. La masa de dicha señal se tomará del terminal PA–INPUT (pin 14).

Identificación de K

$K = \lim_{s \to 0} G(s)$ = Ganancia en régimen permanente

Para identificar la ganancia del sistema almacenamos el resultado de la simulación en una variable ('vel', por ejemplo) mediante el bloque *To workspace* y realizamos la media de los valores de la señal de salida entre los instantes de muestreo en los que la dicha señal se ha estabilizado (muestras 300 y 700, aproximadamente).

Para ello, ejecutamos el siguiente comando:

```
>>V = mean(vel(300:700));
```

De esta forma, obtenemos el valor de la salida en régimen permanente, que coincide con la ganancia del sistema en el caso de que la entrada sea un escalón unitario. Para el resto de señales de entrada, la ganancia será el valor V dividido entre la amplitud del escalón.

$$k = \frac{V = mean(vel(300:700))}{amplitud\ \ escalón(Step)}$$

La ganancia obtenida de esta forma, relaciona la tensión de entrada con la tensión de salida que ofrece el tacogenerador. En el caso de que queramos conocer la correspondencia entre las tensiones que ofrece el tacogenerador y la velocidad de giro del motor (rpm), utilizaremos la constante obtenida en la práctica 1.

$$\frac{\omega}{Vo} = \frac{84{,}6 \cdot 32\ \ rpm}{12\ \ V} = 225{,}6\ \ rpm/V$$

Por último, decir que realizamos el mismo experimento para escalones de distinta magnitud (1, 3, 5 ,7 y 9) y calculamos la media de los cinco experimentos. Evidentemente, este procedimiento se debe realizar con freno y sin freno, obteniendo así dos funciones de transferencia distintas, cada una de ellas resultante de promediar cinco valores.

Los resultados obtenidos son:

Amplitud escalón	Sin freno		Con freno	
	Vo	K	Vo	K
1	0,877	0,877	0,531	0,531
3	2,319	0,773	1,4355	0,48
5	3,9021	0,78	2,35	0,47
7	5,5262	0,79	2,9427	0,42
9	6,106	0,68	2,934	0,326
Valor medio	Vmedia = 0,78		Vmedia = 0,4454	

Identificación de T

Para obtener el valor de la constante de tiempo T debemos tener en cuenta que dicho parámetro es el tiempo transcurrido desde el inicio del escalón (2 segundos) hasta que la salida alcanza el 63 % del valor final.

Para ello, observamos la señal de salida en la pantalla del Scope y, mediante un zoom, vemos el instante en el que la señal alcanza el 63% del valor en régimen permanente. A dicho instante de tiempo le restamos 2, ya que el escalón comienza dos segundos después de iniciarse la simulación.

Los resultados obtenidos son:

Amplitud escalón	Sin freno		Con freno	
	63% Vo	T	63% Vo	T
1	0,877	0,13	0,531	0,1
3	2,319	0,13	1,4355	0,11
5	3,9021	0,16	2,35	0,1
7	5,5262	0,22	2,9427	0,13
9	6,106	0,24	2,934	0,13
Valor medio	Tmedia = 0,176		Tmedia = 0,114	

Según los resultados obtenidos, las funciones de transferencia buscadas son:

Sistema sin freno	Sistema con freno
$G(s) = \frac{0{,}78}{1 + 0{,}176 \cdot s}$	$G(s) = \frac{0{,}4454}{1 + 0{,}114 \cdot s}$

Funciones equivalentes discretas

Una vez obtenidas las funciones de transferencia buscadas, debemos calcular las equivalentes discretas.

Para ello podemos utilizar el comando de Matlab **c2dm** (Continuous To Discrete with Method). Dicho comando tiene la siguiente sintaxis:

```
[numd, dend] = c2dm(num, den, Ts, 'method')
```

de esta forma se convierte la función de transferencia continua *G(s) = num(s)/den(s)* en la función de transferencia discreta *G(z)=numd(z)/dend(z)* a un periodo de muestreo *Ts* utilizando el método de discretización *'method'*, donde *num*, *den*, *numd*, *dend* son los vectores cuyos elementos son los coeficientes en orden decreciente de los polinomios numerador y denominador de las funciones de transferencia continua y discreta respectivamente. En cuanto a los métodos de dicretización, en nuestro caso utilizaremos el que antepone un retenedor de orden cero a la entrada del sistema, *'zoh'* , que es el que nos interesa en este tipo de aplicaciones.

Sistema sin freno

```
[numd, dend] = c2dm(0.78, [0.176 1], 0.01, 'zoh')
```

$$G(z) = \frac{0{,}0431}{z - 0{,}9448}$$

Sistema con freno

```
[numd, dend] = c2dm(0.4454, [0.114 1], 0.01, 'zoh')
```

$$G(z) = \frac{0{,}0374}{z - 0{,}916}$$