Informática Industrial y Comunicaciones
TRABAJO CURSO 2023/2024

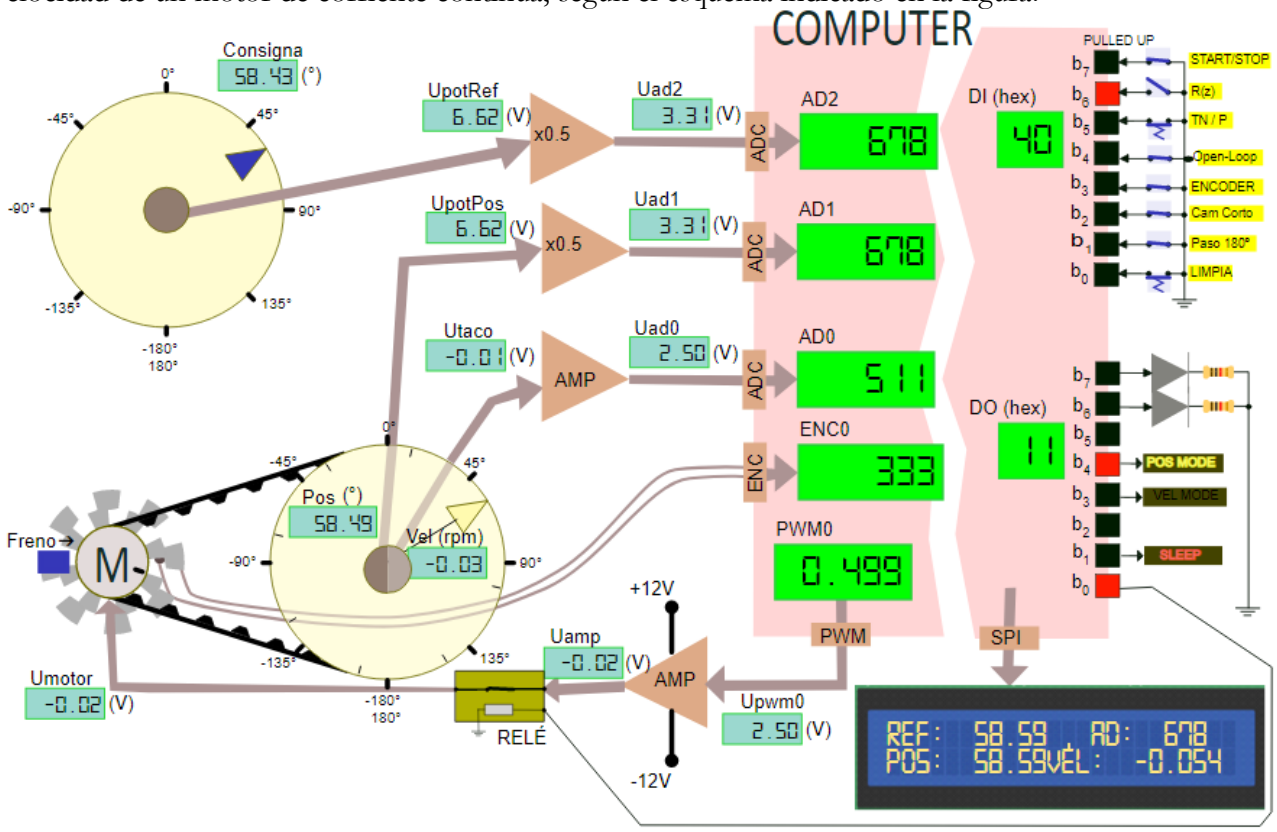
1. Introducción

Realizar un programa para el control de posición y velocidad de un motor DC.

El programa será realizado en modo consola, utilizando Qt Creator bajo Sistema Operativo Windows. Se utilizará como sistema a controlar el simulador de sistema mecatrónico Feedback actualizado (descargar y utilizar según información en apartado 5-Descargas).

2. Especificaciones

El trabajo consistirá en desarrollar el software necesario para controlar en lazo cerrado la posición o la velocidad de un motor de corriente continua, según el esquema indicado en la figura.



El motor es accionado mediante una señal PWM 0-5V que activa un amplificador de -12 a 12 V. Un relé conectado al bit 0 de la salida digital DO permite que la tensión amplificada alcance la entrada del motor.

Se dispone de sensores analógicos para la velocidad y posición de la rueda movida por el motor a través de una correa (relación de transmisión 1:8).

Un freno manual simula la variación de las condiciones de carga del motor.

El funcionamiento deseado es el siguiente:

- Mediante los interruptores de entrada digital, conectados al puerto DI, el usuario podrá establecer **en cualquier momento** las siguientes condiciones de funcionamiento:

Bit peso	Selección
7, 6, 5,4	Establecer modo de control del motor: ▫ DI_7 activado: parar el motor (aplicar tensión 0V, **no modificar** relé) ▫ DI_7 no activado: calcular tensión de control (**no modificar** relé) ○ DI_4 activado: aplicar directamente tensión al motor (Cadena Abierta) ○ DI_4 no activado: calcular tensión a aplicar al motor (Cadena Cerrada): • DI_6 no activado: usar regulador RZ • DI_6 activado: cada pulsación de DI_5 cambia el modo de control: • Por defecto, control todo/nada con U=±2V (modo posición) ó ΔU=±0.05V (modo velocidad) • Control proporcional con K=0.05 V/deg (modo posición) o proporcional-integral con K=0.05 V/rpm (modo velocidad)
2	Si está activado: realizar el desplazamiento por el camino más corto (sólo en control de posición, según indicaciones en punto 5 de PL-8)
1	Si está activado: gestión del problema de paso por 180° (sólo en control de posición, según indicaciones en punto 5 de PL-8)
0	Reservado modo limpia-parabrisas (ampliación opcional 4).

- Mediante salidas digitales hacia indicadores luminosos, conectados al puerto DO, se indicará:

Bit peso	Significado	Uso
7	Motor girando a izquierda	Activación LED
6	Motor girando a derecha	Activación LED
4	Modo actual: control de posición	Parpadeo LED 0.5 seg
3	Modo actual: control de velocidad	Parpadeo LED 0.5 seg
1	Procesando SLEEP: no se aceptan comandos	Activación LED

- Mediante salida digital hacia la bobina de un relé conectada al puerto DO, se permitirá que la tensión amplificada alcance al motor.

Bit peso	Significado	Uso
0	Permitir la activación del motor	Valor 1: motor activo Valor 0: motor inactivo

- Mediante teclado, el usuario podrá introducir en cualquier momento los siguientes comandos, que deben producir el efecto indicado:
  - **POS=valor_en_grados** → cambia a modo control posición; la posición consigna es la indicada en el comando.
  - **POS=POT** → cambia a modo control posición; la posición consigna es fijada por potenciómetro.
  - **VEL=valor_en_rpm** → cambia a modo control velocidad; la velocidad consigna es la indicada en el comando.
  - **VEL=POT** → cambia a modo control velocidad; la velocidad consigna es fijada por potenciómetro (-180º = -40 rpm , 180º=+40rpm).
  - **TENSION=valor_en_V** → establece la tensión a aplicar al motor en modo Cadena Abierta (DI_4 activo).
  - **RELAY=ON/OFF** → activa (ON) /desactiva (OFF) el bit DO_0 que actúa sobre la bobina del relé que corta el paso de tensión al motor.

- **RZ=[b_0,b_1, …, b_m] / [a_0,a_1, … ,a_n]** → cambia el regulador del lazo de **control para el modo activo** (posición/velocidad). Los polinomios del regulador podrán tener cualquier longitud. Los valores de cada polinomio podrán estar separados indistintamente por comas y/o por espacios. Se aplicarán los reguladores por defecto del anexo si no se ha introducido este comando.
- **SLEEP=valor_en_ms** → detener la lectura y ejecución de comandos durante el tiempo indicado (activar bit DO_1 durante el tiempo de espera).
- **FILE=init.txt** → Leer línea por línea el archivo indicado, ejecutando los comandos indicados en las líneas correspondientes. Un ejemplo de archivo de prueba se encuentra en el directorio Client/Test.

Los nombres de comando se introducirán siempre en mayúscula.
El programa debe admitir que el usuario introduzca uno o varios espacios entre las palabras.

El programa guardará en un archivo de texto "comandos.log" los comandos introducidos junto a su fecha y hora, ej: 08/12/2017 - 13:12:05 >> POS = 45

- Periódicamente, por interrupción del temporizador (Tm=100 ms), se ejecutará un paso de un lazo de control, que deberá calcular la tensión um_k a aplicar al motor (unidades: V de entrada al motor) en ese instante según el algoritmo siguiente:
  - Desplazamiento de tablas temporales: e_k , um_k, pos_k.
  - Detección del modo de funcionamiento deseado según el estado de las E/S digitales.
  - Si relé motor (estado DO_0)no activado: no hacer nada
  - Si modo parar motor (estado DI_7): um_k=0
  - Si modo arrancar (estado DI_7)+open loop (estado DI_4): um_k = tensión indicada en el último comando TENSION.
  - Si modo arrancar (estado DI_7)+closed loop (estado DI_4):
    - Obtención de la referencia (de comando de usuario o de potenciómetro en el simulador, según último tipo de referencia introducido por comando).
    - Lectura de la posición actual del motor en grados y de la velocidad en rpm.
    - Corrección de la posición del motor si procede, en función de DI_1 (sólo modo de control de posición - POS)
    - Cálculo del error actual (según modo POS/VEL); corrección del error si procede en función de DI_2 (sólo modo de control de posición - POS).
    - Cálculo de acción um_k (en voltios) a través del algoritmo adecuado al tipo de control según estado de DI_6 y última activación de DI_5.
  - Cálculo y generación de señal PWM para cambiar la entrada del motor a partir del valor um_k calculada.
  - Escritura en display LCD de la información relevante según el modo de control: valores de consigna y posición/velocidad actual, tipo control (RZ/TN/P/…).
  - Activación de los indicadores luminosos correspondientes.

- Es posible utilizar la función siguiente para ajustar el tamaño de la ventana de comandos, de forma que resulten más cómodas las pruebas:

```
void SetConsoleSize(int rows_total,int cols_total,int rows_seen,int cols_seen)
```

```
{
   HANDLE hConsole=GetStdHandle(STD_OUTPUT_HANDLE);
   COORD const bufferSize={cols_total,rows_total};
   SMALL_RECT const windowRect={0,0,cols_seen-1,rows_seen-1};
   SMALL_RECT const minimal_window = { 0, 0, 1, 1 };

   SetConsoleWindowInfo(hConsole, TRUE, &minimal_window);
   SetConsoleScreenBufferSize(hConsole,bufferSize);
   SetConsoleWindowInfo(hConsole,TRUE,&windowRect);
}

main()
{
   ....
  SetConsoleSize(60,120,24,40);
   ....
}
```

- Las funciones siguientes permiten que la E/S de consola se realice en la misma pantalla del simulador, resultado más cómodo para las pruebas. La ayuda de ambas se encuentra en la ayuda general del simulador:

  Simulator_Console_GetString( ) → Espera cadena introducida en la línea inferior de la consola de la página web del simulador

  Simulator_Console_Printf( ) → Escribe el texto (con formato similar a printf) en la consola de la página web del simulador

  Si se utiliza esta opción, indicarlo mediante un texto en la propia consola al inicio del programa:

  Simulator_Console_Printf("Introduzca comandos en esta consola\n");

3. Requerimientos de programación

- Dar nombres adecuados a variables, constantes y funciones, que reflejen claramente su cometido en el programa.
- Realizar funciones para las partes del programa que puedan ser reutilizables.
- Utilizar interrupciones para la temporización de control.
- Incluir en la cabecera de cada función comentario que informe sobre su tarea, sus parámetros, valor devuelto, y otras consideraciones (asignación dinámica de memoria que debe liberar el llamador, modificación de variables apuntadas por puntero, uso de variables globales).
- Utilizar #define para las constantes no triviales que sean necesarias.
- Definir y utilizar struct RZ para manejar los datos de cada regulador: tablas a,b y sus tamaños.
- Realizar el programa en **al menos** 3 módulos de código fuente (con sus correspondientes archivos de encabezado):
  - principal.c (sólo main).
  - rutinacontrol.c (función de servicio de la temporización del lazo de control, y funciones necesarias únicamente para ella).
  - funciones_auxiliares.c (resto de funciones a utilizar).
- Utilizar asignación dinámica de memoria para las tablas cuyo tamaño no se conozca en tiempo de compilación.

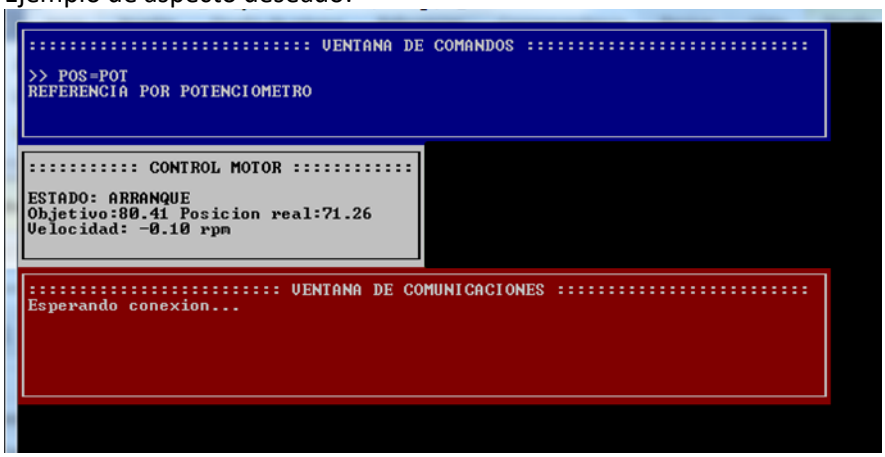
4. Calificación

La calificación del trabajo se realizará del modo siguiente:

Contenido del trabajo	Calificación
El trabajo cumple las especificaciones del apartado 2 funcionando correctamente y con una programación adecuada (uso de funciones, asignación dinámica de memoria, etc.) según lo indicado en el apartado 3.	6 ptos
El trabajo cumple las especificaciones del apartado 2 pero tiene algunos problemas leves de funcionamiento o de programación inadecuada.	5 ptos
Faltan algunos contenidos, o algunos problemas de funcionamiento o programación inadecuada son más graves.	4 ptos (compensable)
El trabajo no cumple las especificaciones del apartado 2, o tiene graves fallos de funcionamiento o de concepto, o partes sustanciales han sido copiadas.	0…3 ptos (repetir)

Las siguientes adiciones sumarán calificación hasta 12/10:

Contenido adicional	
① Crear un socket cliente TCP que, a solicitud del usuario mediante un comando, se conecte con el servidor en la dirección IP 127.0.0.1 (localhost), puerto 55455, donde un servidor realizado en Python está a la espera de conexiones. La conexión se realizará mediante el comando siguiente, la dirección IP y el puerto deberían poder cambiarse: **TCP CONNECT = 127.0.0.1 / 55455** Tras conectarse, el servidor enviará el nombre de la variable en la que está interesado, que puede ser: "REF" "POS" "VEL" "UMOTOR" En cada periodo de muestreo, el cliente enviará el valor de la variable en formato texto jSon, de forma que el servidor pueda mostrar la evolución de esta variable. El texto deberá contener el valor actual, por ejemplo para UMOTOR (tensión motor): **{ "name" : "UMOTOR", "value": 2.54 , "units": "Volt" }** Se terminará la conexión cuando se reciba el comando de consola:	+2 ptos

	TCP CONNECT = DISCONNECT Se dispone de un servidor Python para probar la comunicación en el enlace indicado en el apartado 5. Descargas	
②	Utilizar la librería curses (descargar y ver documentación en apartado anexos) para gestionar la entrada/salida por pantalla, dividiendo la misma en al menos dos ventanas: una para gestión de comandos, otra para visualización de estado: ○ Modos de funcionamiento (potenciómetro/referencia fija, modo de control activo, paso de 180°, etc.) ○ Valores de referencia y salida Ejemplo de aspecto deseado: 	+2 ptos
③	Si el switch de peso 3 está activo, no utilizar los sensores analógicos de velocidad y posición. En este caso, la posición se obtendrá a través de la cuenta de encóder. El encóder es de 64 pulsos por vuelta. Hay que tener en cuenta que está situado en el eje del motor (relación de transmisión 1:8 respecto al eje de salida) y que al ser en cuadratura genera 4 cuentas por pulso. La velocidad se obtendrá por derivación de la señal de posición. ¡¡¡ Atención a las unidades !!! (deg/ms ≠ rpm)	+1 pto
④	Si se produce la activación del pulsador DI_0 durante más de 3 segundos, comenzar un movimiento alternativo de -70° a +70° (estilo limpiaparabrisas). Se termina este modo cuando se activa el pulsador DI_0 dos veces seguidas en el periodo de 3 segundos.	+1 pto

4.1. Control de copia

El trabajo es individual, y por tanto el contenido entregado debe ser original de cada alumno, reflejando su desarrollo a lo largo de las prácticas de la asignatura.

Existen métodos para determinar con un alto grado de certeza si el trabajo es original o ha sido copiado/modificado a partir del trabajo de un compañero.

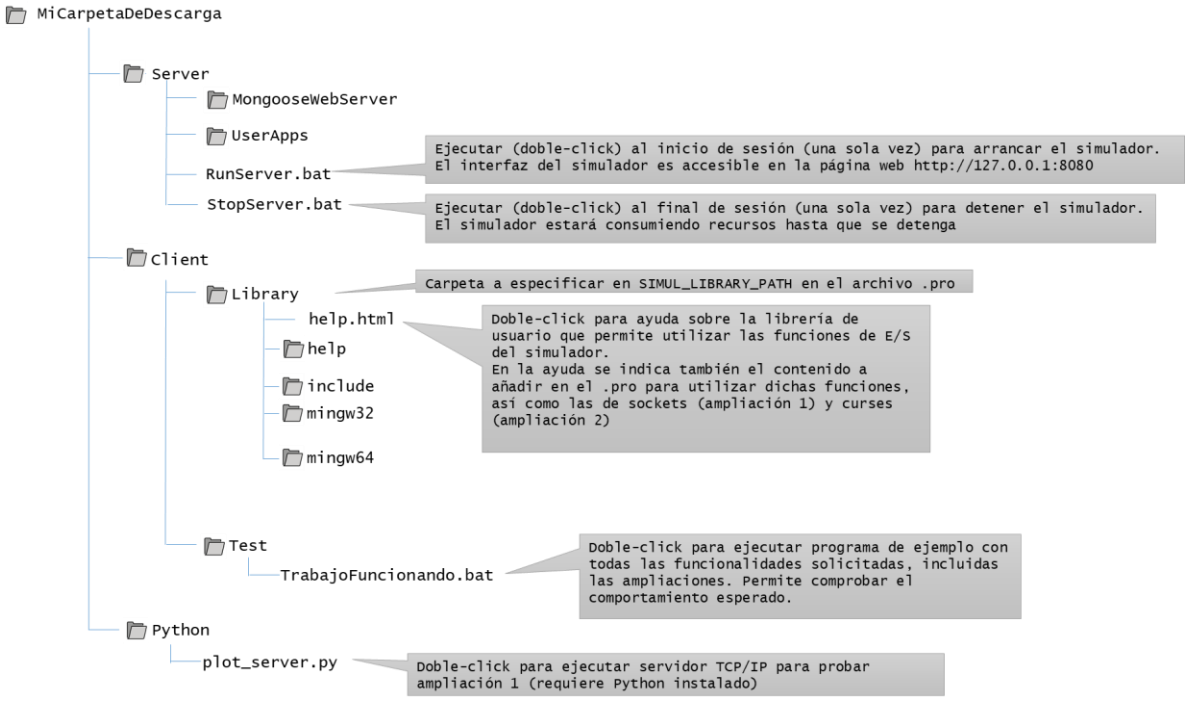
En caso de duda se podrá convocar al alumno a una sesión en la que deberá ejecutar su programa en modo depuración y comprobar junto al profesor su funcionamiento.

El trabajo será rechazado (y por tanto la convocatoria suspensa) en caso de detectarse la copia de cualquier parte del mismo.

El examen evaluará lo realizado en el trabajo, por lo que la copia redundará también en baja calificación en el examen. NO MERECE LA PENA ARRIESGARSE. REALICE EL TRABAJO DE FORMA INDIVIDUAL. SI TIENE DUDAS, CONSULTE PRESENCIALMENTE O POR E-MAIL A SU(S) PROFESOR(ES).

5. Descargas

Disponible descarga de archivo .zip en la dirección
http://isa.uniovi.es/~ialvarez/Curso/descargas/Simulador_Trabajo_IIyC_2023.zip
con los contenidos siguientes:



6. Entrega

Comprimir **únicamente** los archivos de código fuente y encabezados (.c y .h) en un archivo (.zip ó .rar) con el nombre y apellidos del alumno.

Enviar el archivo comprimido por e-mail a la dirección: ialvarez@isa.uniovi.es , indicando en el campo "Asunto" el texto "Trabajo IIyC23-24 Alumno=Apellidos, Nombre".

Si se ha(n) realizado alguna(s) ampliación(es), debe indicarse en el cuerpo del e-mail para que sea(n) evaluada(s).

Fechas límite de entrega: Las indicadas en la página de la asignatura:
http://isa.uniovi.es/~ialvarez/Curso/infindycom/trabajos.shtml

Anexo: reguladores

Los siguientes reguladores serán aplicados por defecto, si no se introduce ningún otro a través del comando RZ.

REGULADOR POR DEFECTO EN MODO POSICION:

$$R(z) = \frac{U(z)}{E(z)} = \frac{b_0 + b_1.z^{-1} + ... + b_m.z^{-m}}{1 + a_1.z^{-1} + ... + a_n.z^{-n}}$$

$$e_k = c_k - y_k$$

$$u_k = b_0.e_k + b_1.e_{k-1} + ... + b_m.e_{k-m} - \\ - (a_1.u_{k-1} + ... + a_n.u_{k-n})$$

Valor	
Tm	100 ms
m	1
b0	0.15 V/deg
b1	-0.11 V/deg
n	1
a1	-0.43

REGULADOR POR DEFECTO EN MODO VELOCIDAD:

$$R(z) = \frac{U(z)}{E(z)} = \frac{b_0 + b_1.z^{-1} + ... + b_m.z^{-m}}{1 + a_1.z^{-1} + ... + a_n.z^{-n}}$$

$$u_k = b_0.e_k + b_1.e_{k-1} + ... + b_m.e_{k-m} - \\ - (a_1.u_{k-1} + ... + a_n.u_{k-n})$$

Valor	
Tm	100 ms
m	2
b_0	6.0 V/rpm
b_1	-9.3 V/rpm
b_2	3.48 V/rpm
n	3
a_1	-0.3
a_2	-0.78
a_3	0.08

OTROS REGULADORES:
Si se desean calcular / aplicar otros reguladores, la función de transferencia del sistema motor es, en unidades del Sistema Internacional (S.I.):

$$G(s) = \frac{W(s)}{U(s)} = \frac{rad/s}{V} = \frac{K_t}{(Js + B)(Ls + R) + K_e.K_t}$$

Kt=0.01 N.m/A
J=0.0025 Kg . m^2
B=0.001 N . m . s
L=0.1 H
R=5 Ω
Ke=0.01 V . s/rad

Teniendo en cuenta las unidades, y que hay una relación de 1:8 entre el eje del motor y la salida, se pueden utilizar las funciones de transferencia para calcular los reguladores continuos con Matlab:

GS (velrpm_eje_salida / Vmotor) = GS (S.I.)* cte1 (rad/s → rpm) * cte2 (1/8)
GS (deg_eje_salida /Vmotor) = GS(S.I.) * 1/s (integrador) * cte1 (rad/s → deg/s) * cte2 (1/8)

Con estos valores se puede calcular R(s), y a continuación discretizar para obtener R(z). Los coeficientes de R(z) se pueden aplicar directamente en la ecuación en diferencias del regulador:

```
>> RZ=c2d(tf,0.1,'tustin');
```