

Computadores y Programación

TRABAJO CURSO 2020/2021

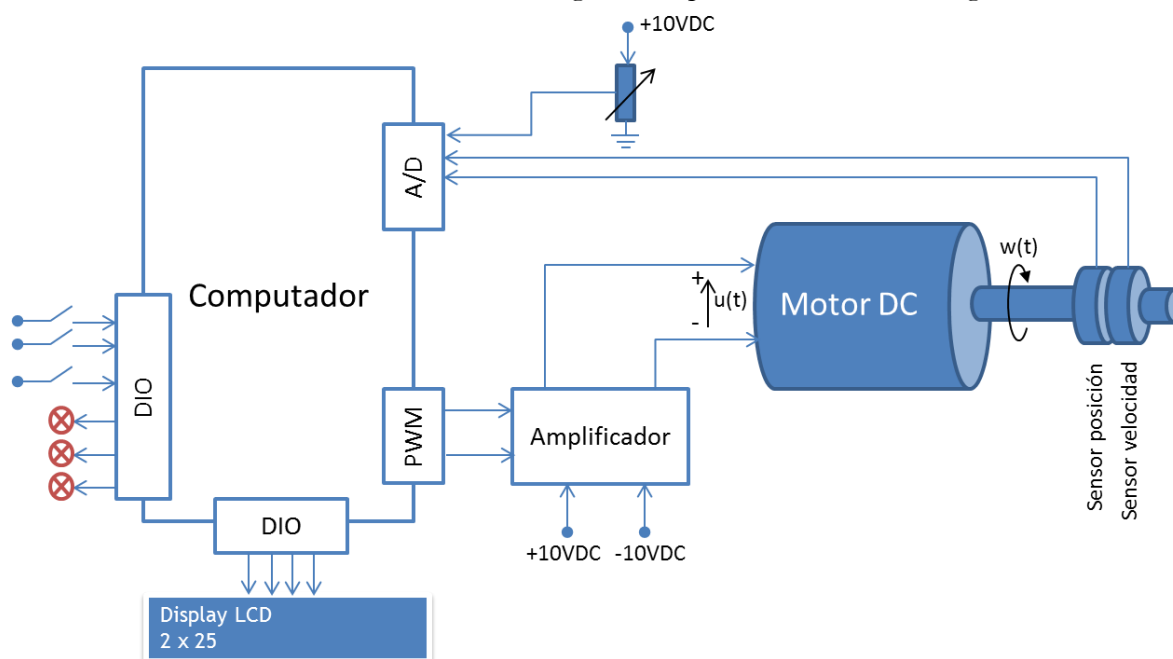
1. Introducción

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 7/Windows 10. Se utilizará como sistema a controlar el simulador de sistema mecatrónico disponible en la página web de la asignatura (última versión descargable desde <http://isa.uniovi.es/~ialvarez/Curso/infindycom/practicas/SimuladorMotorConESDigital.rar>).

2. Especificaciones

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



El funcionamiento es el siguiente:

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

Bit peso	Selección	Estado 1	Estado 0
0	Arrancar y parar el control del motor	Parar ($u_k=0$)	Arrancar
1	Gestión del problema de paso por 180° (corregir $pos_k[0]$ en función de $pos_k[0]-pos_k[1]$, sólo modo posición+closed-loop)	Gestionar	No gestionar
2	Movimiento por el camino más corto (corregir $e_k[0]$ para que esté entre -180 y 180 , sólo modo posición+closed-loop)	Gestionar	No gestionar
4	Selección del modo de control (I)	Ver bit 5	Usar RZ
5	Selección del modo de control (II): sólo si bit 4 a 1	P (pos) PI (vel)	Todo/nada

- Mediante salidas digitales hacia indicadores luminosos se indicará:



- Bit de peso 6: motor girando a derecha.
 - Bit de peso 7: motor girando a izquierda.
 - Bit de peso 5 (parpadeo cada 0.5 seg): actualmente en modo control posición.
 - Bit de peso 4 (parpadeo cada 0.5 seg): actualmente en modo control velocidad.
- Mediante teclado, el usuario podrá introducir en cualquier momento los siguientes comandos, que deben producir el efecto indicado:
- **POS=valor_en_grados** → pasa a control de posición; la posición objetivo es la indicada en el comando.
 - **POS=POT** → pasa a control de posición; la posición objetivo es fijada por el potenciómetro.
 - **VEL=valor_en_rpm** → pasa a control de velocidad; la velocidad objetivo es la indicada en el comando.
 - **VEL=POT** → pasa a control de velocidad; la velocidad objetivo es fijada por el potenciómetro ($-180^\circ = -60\text{rpm}$, $+180^\circ = +60\text{rpm}$).
 - **OPEN LOOP=valor_en_volt** → aplicar directamente la tensión indicada, sin realizar ningún control.
 - **TENSION=valor_en_volt** → tensión a utilizar en control todo/nada (sólo se usa si el interruptor indica uso de este modo), el mismo valor para control de posición y de velocidad.
 - **KP=valor** → constante a utilizar en control P/PI (sólo se usa si el interruptor indica uso de este modo), valor aplicado sólo al modo de control en vigor.
 - **SLEEP=tiempo_en_ms** → detener la lectura y ejecución de comandos durante el tiempo indicado (activar LED de peso 0 durante el tiempo de espera).

Los nombres de comando se introducirán siempre en mayúscula.

El programa debe admitir que el usuario introduzca uno o varios espacios en blanco entre las palabras.

- Periódicamente ($T_m=100\text{ ms}$) bajo interrupción se ejecutará un paso de un lazo de control, con los siguientes pasos:
- Si SW0 indica motor parado: aplicar tensión 0
 - Si SW0 indica control activo:
 - Obtención de la referencia (según último comando VEL ó POS).
 - Lectura de la posición actual del motor en grados y de la velocidad en rpm.
 - Desplazamiento de tablas temporales: e_k y u_k .
 - Corrección de la posición del motor, si procede, en función del bit de entrada 1 (sólo modo de control de posición)
 - Cálculo del error actual (según modo POS/VEL), corrección si procede (bit de entrada 2, sólo modo POS) e introducción en tabla e_k .
 - Cálculo de salida actual u_k (en voltios) a través del algoritmo correspondiente al modo de control seleccionado según el modo y el estado de switches.
 - Cálculo y generación de señal PWM para cambiar la entrada del motor a partir del valor u_k actual.
 - Escritura en display LCD de los valores de consigna (1ª línea) y posición/velocidad actual (2ª línea).
 - Activación de los indicadores luminosos correspondientes.

- Los valores iniciales de los diferentes parámetros se encuentran en el anexo.
- Es posible utilizar la función siguiente para ajustar el tamaño de la ventana de comandos, de forma que resulten más cómodas las pruebas (incluir <windows.h>):

```
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);
    ....
}
```

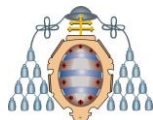
- Disponible para descargar un ejemplo que satisface todas estas especificaciones (incluidas las ampliaciones propuestas posteriormente) en la dirección:

<http://isa.uniovi.es/~ialvarez/Curso/Mecatronica/C1-ComputadoresYProgramacion/Practicas/TrabajoFuncionando.rar>

Se deberá comprobar que el comportamiento del programa desarrollado es similar a este ejemplo, en cuanto a las diferentes respuestas (a comandos de teclado y a interruptores), la dinámica cuando se aplican los mismos controladores, etc.

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.
- Realizar el programa en al menos 3 módulos de código fuente (con sus correspondientes archivos de encabezado):
 - **principal.c** (sólo main).
 - **rutinacontrol.c** , **.h** (función de servicio de la temporización del lazo de control, y funciones necesarias únicamente para ella).
 - **funciones_auxiliares.c** , **.h** (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, constantes, etc.) según lo indicado en el apartado 3.	6 pts
El trabajo cumple las especificaciones del apartado 2 pero tiene algunos problemas leves de funcionamiento o de programación inadecuada.	5 pts
Faltan algunos contenidos, o algunos problemas de funcionamiento o programación inadecuada son más graves.	4 pts (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 pts (repetir)

Las siguientes adiciones sumarán calificación hasta 12/10:

Contenido adicional	Calificación
<p>① Añadir soporte para el comando:</p> <ul style="list-style-type: none"> ○ $RZ=[b_0 \ b_1 \ ... \ b_m] / [a_0 \ a_1 \ ... \ a_n]$ Cambia el regulador del lazo de control para el modo activo (posición/velocidad). Los polinomios del regulador podrán tener cualquier longitud (usar asignación dinámica de memoria). Los valores de cada polinomio podrán estar separados indistintamente por comas y/o por espacios. 	+ 2 pto
<p>② Utilizar la librería curses (descargar y ver documentación en http://isa.uniovi.es/~ialvarez/Curso/infindycom/trabajos.shtml) 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:</p> <ul style="list-style-type: none"> ○ Modos de funcionamiento (tipo de ref, tipo control activo, etc.) ○ Valores de referencia y salida <p>Ejemplo:</p> 	+ 2 pto
<p>③ Si el switch de peso 7 está activo, no utilizar el sensor de velocidad. En este caso, la velocidad se obtendrá por derivación de la señal de posición.</p> <p>Si el switch de peso 6 está activo, no utilizar el sensor de posición. En este caso, la posición se obtendrá por integración de la señal de velocidad (para un correcto funcionamiento, es necesario que previamente a la activación de este modo el motor esté en la posición 0).</p> <p>En ambos casos: ¡¡¡ atención a las unidades !!! (deg/ms \neq rpm)</p>	+1 pto
<p>④ Si se produce la secuencia de activación de switches 0,1,2 (por este orden), realizar un movimiento alternativo de -70° a $+70^\circ$ (estilo limpiaparabrisas). No atender otros comandos hasta que se haya modificado algún switch.</p>	+1 pto



Se dispone en el enlace siguiente:

http://isa.uniovi.es/~ialvarez/Curso/infindycom/practicas/Ayuda_Ampliaciones.pdf

de información adicional para realizar las ampliaciones.

!!! ATENCION !!! La ampliación 1 no se debe tomar de dicho documento, si no de <http://isa.uniovi.es/~ialvarez/Curso/infindycom/practicas/practica9-2020.pdf>

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.

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 PROFESOR.

5. Entrega

Comprimir los archivos de código fuente y encabezados (.c y .h) en un solo 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 CyP20-21. Alumno=Nombre y apellidos", con solicitud de confirmación de recepción.

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

Fecha límite de entrega: El día anterior al examen de la convocatoria correspondiente.

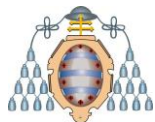


6. Coordinación con otras asignaturas

Todo lo realizado es utilizable para el trabajo de la asignatura de electrónica utilizando el PIC y el sistema mecatrónico real. Sólo algunas funciones (E/S analógica y digital, temporización, interrupción) deben ser modificadas de uno a otro caso, así como los reguladores.

Algunas diferencias que se deben tener en cuenta son:

Item	PC con Qt Creator (gcc) (ANSI-C estándar)	PIC (no completamente estándar)
Tamaño de enteros	8 bits: char 16 bits: short 32 bits: int 32 bits: long	8 bits: char, short, int 16 bits: long 32 bits: long long
Signo de enteros	signed por defecto	unsigned por defecto Usar signed long long para entero 32 bits equivalente a int de QtCreator
Disponibilidad de memoria	De sobra para el programa a realizar, sin restricciones	Restringida, es posible que haya problemas de espacio para las variables.
Desplazamiento a derecha con signo	Desplaza correctamente con bit de signo: int x=-32,y; y=x>>2; // y pasa a valer -8	No desplaza bit de signo: signed int x=-32,y; y=x>>2; // y pasa a valer 48 Solución para desplazamiento a derecha: #define DD(x,q) (((x)>=0)? ((x)>>(q)): (~((x)>>(q))))
PWM	El duty 0% se corresponde con la tensión mínima de salida, y el duty 100% con la tensión máxima.	Está invertido: el duty 0% se corresponde con la tensión máxima de salida, y el 100% con la tensión mínima.
Regulador	Usar los indicados a continuación	Determinar reguladores adecuados según lo estudiado en otras asignaturas

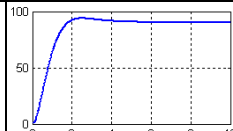


Anexo: reguladores

FUNCION DE TRANSFERENCIA DEL MOTOR, POSICION/TENSION:

$G(s) = \frac{\Theta(s)}{U(s)} = \frac{K'}{s \cdot (1 + \tau \cdot s)}$	Valor	Simulador
	K'	72 °/V.s
	τ	2.5 s

REGULADOR EN MODO POSICION:

$R(z) = \frac{U(z)}{E(z)} = \frac{b_0 + b_1 \cdot z^{-1} + \dots + b_m \cdot z^{-m}}{1 + a_1 \cdot z^{-1} + \dots + a_n \cdot z^{-n}}$ $u_k = b_0 \cdot e_k + b_1 \cdot e_{k-1} + \dots + b_m \cdot e_{k-m} - (a_1 \cdot u_{k-1} + \dots + a_n \cdot u_{k-n})$	Valor	Simulador
	T_m	200 ms
	m	1
	b_0	0.13 V/°
	b_1	-0.11 V/°
	n	1
	a_1	-0.42
Resultados esperados ante escalón de referencia 90°		

FUNCION DE TRANSFERENCIA DEL MOTOR, VELOCIDAD/TENSION:

$G(s) = \frac{\Omega(s)}{U(s)} = \frac{K}{1 + \tau \cdot s}$	Valor	Simulador
	K	12 rpm/V
	τ	2.5 s

REGULADOR EN MODO VELOCIDAD:

$R(z) = \frac{U(z)}{E(z)} = \frac{b_0 + b_1 \cdot z^{-1} + \dots + b_m \cdot z^{-m}}{1 + a_1 \cdot z^{-1} + \dots + a_n \cdot z^{-n}}$ $u_k = b_0 \cdot e_k + b_1 \cdot e_{k-1} + \dots + b_m \cdot e_{k-m} - (a_1 \cdot u_{k-1} + \dots + a_n \cdot u_{k-n})$	Valor	Simulador
	T_m	200 ms
	m	2
	b_0	0.85 V/rpm
	b_1	-1.36 V/rpm
	b_2	0.55 V/rpm
	n	3
	a_1	-1.91
	a_2	1.16
	a_3	-0.25
Resultados esperados ante escalón de referencia 60 rpm		