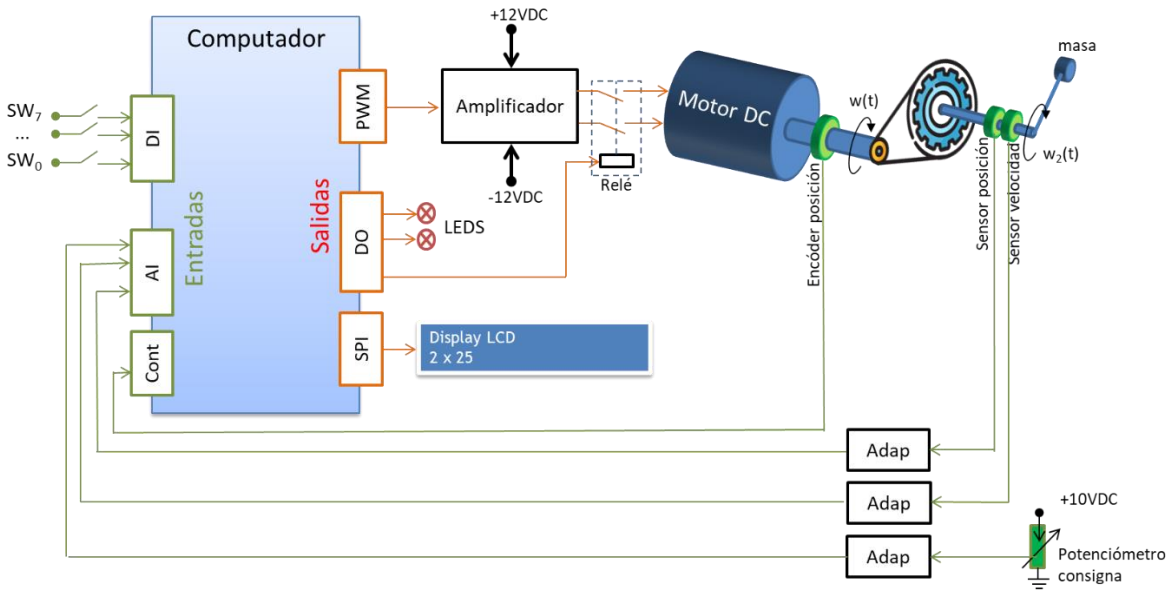
UNIVERSIDAD DE OVIEDO
Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas

Guía de Prácticas

ASIGNATURA:	Informática Industrial y Comunicaciones		
CENTRO:	Escuela Politécnica de Ingeniería de Gijón		
ESTUDIOS:	Grado en Ingeniería Electrónica y Automática		
CURSO:	3º	CUATRIMESTRE:	1
CARÁCTER:	Obligatoria	CRÉDITOS ECTS:	6
PROFESORADO:	Ignacio Alvarez, José Mª Enguita, Angel Navarro, Mariam Saeed		

PRACTICA 07: Control básico y E/S digital

1. Ejercicio a realizar. Se dispone de un computador conectado mediante E/S analógica y digital a un sistema formado por:
   - Un motor DC de 12V, que mueve mediante una correa dentada (con relación 1:8) un disco graduado que dispone de sensores para medida de la posición y velocidad angulares.
   - Un potenciómetro para generar señal de consigna (referencia), conectado a +10V.
   - Un computador de control, con entradas A/D, salida PWM, E/S digital, y display LCD.
   - La electrónica necesaria para adaptación y amplificación de señales



Se desea realizar sobre este sistema un control básico de velocidad en cadena abierta, con las siguientes especificaciones:

a) Periodo de muestreo Tm=100 ms.

b) Tensión a aplicar al motor, según entrada del usuario (-12 a 12 V), por señal PWM.

c) Activación de relé para accionamiento del motor mediante bit de peso 0 de salida DO.

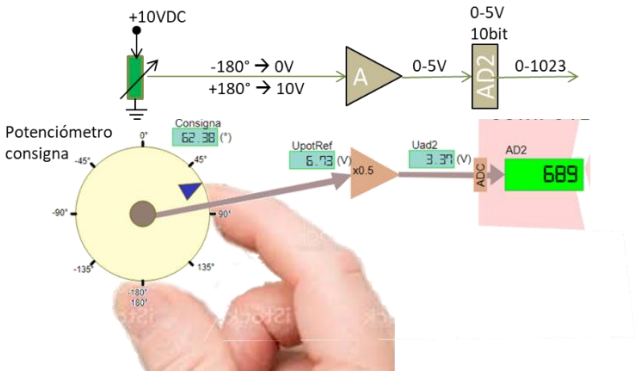
d) Se indicará en cada instante el valor de las magnitudes medidas en el display LCD: posición consigna, posición y velocidad del disco conducido por el motor

e) Se indicará en cada instante el sentido de giro del disco conducido por el motor activando los bits de peso 6 de salida DO (giro a derechas) y 7 (giro a izquierdas).

2. Para la realización de la práctica será necesario utilizar el simulador de sistema físico Feedback, cuya documentación para uso y descarga se encuentra al final de este documento. A este simulador se puede conectar un programa de usuario mediante la librería UserLibSimulator ya utilizada en la PL anterior (Pistón).

3. **Pasos en la realización del código** (realizar y probar cada uno de forma incremental):

3.a) Obtener del canal AD 2 el valor en grados del potenciómetro manual de consigna, según la interpretación de la figura siguiente, y escribir el valor en el display LCD cada 100 ms. Posicionando el cursor sobre cada elemento de la cadena de medición se muestran las transformaciones de esa etapa, que deben ser revertidas por el cálculo.



Función:
float
GetPotRef_deg( );

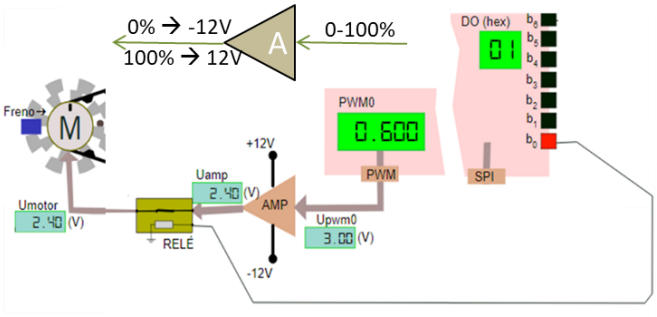
ad2=Leer canal AD 2 → calc u_{ad2}(V) → calc u_{sens2}(V) → calc pos_ref (deg)

```
int err=Simulator_ConnectWss(copiar/pegar de "Mostrar Info clientes");
Simulator_LCD_init();
while (1) {
   refk_deg=GetPotRef_deg();  // Función a realizar
   Simulator_LCD_gotoxy(1,1);  // Posiciona cursor en display
   Simulator_LCD_printf(...);  // Escribe dato en display
   Simulator_Delay(100);
}
```

3.b) Activar el bit de peso 0 de las salidas digitales, de forma que se cierre el relé que permite el paso de la tensión del amplificador hacia el motor:

```
int err=Simulator_ConnectWss(...);
...
Simulator_WriteDO(valor que activa el bit de peso 0 de DO);
while (1) {
   ...
```

3.c) Pedir por teclado el valor de tensión u_{mot} a aplicar al motor (-12 a 12V), y generar la salida PWM necesaria para que se aplique dicha tensión. Posicionando el cursor sobre cada elemento de la cadena de accionamiento se muestra la conversión que realiza.



Función:
void SetUmotor_V
(float u_m);

U_{mot}(V) → calc duty (tanto por 1) → calc Ton (Tpwm=cte=1000) → Aplicar PWM

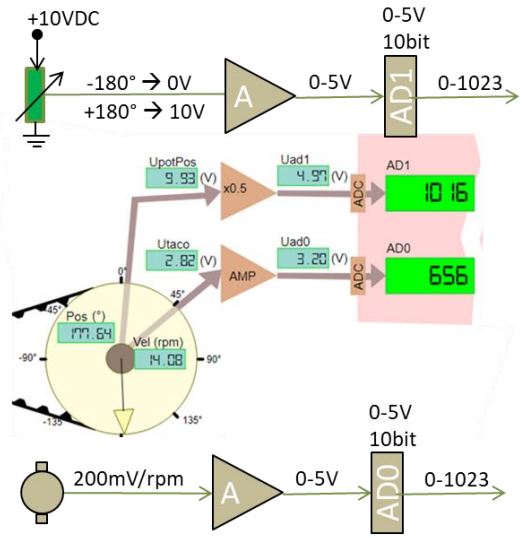
```
float u_motor;
...
Simulator_ConfigPWM(0,1000,500,1);
Simulator_WriteDO(valor que activa el bit de peso 0 de DO);
pedir u_motor por teclado
SetUmotor_V(u_motor);

while (1) {
   ...
```

(Comprobar que, al aplicar el freno, con la misma tensión decae la velocidad)

(Modificar el valor de la masa de carga excéntrica en Parámetros: Masa=100g. Comprobar el efecto de la masa en la velocidad a una tensión constante)

3.d) Modificar el anterior para la lectura cada 100 ms de la posición angular del motor (en grados) y velocidad angular (en r.p.m.), y escribir los mismos en el display LCD.



ad1=Leer canal AD1 → calc u_{ad1}(V) → calc u_{sens1}(V) → calc pos_{mot} (deg)

ad0=Leer canal AD0 → calc u_{ad0}(V) → calc u_{sens0}(V) → calc vel_{mot} (rpm)

Funciones:
float GetPosM_deg();
float GetVelM_rpm();

```
...
Pedir valor por teclado: umotor_V
AplicarTensionMotor V(umotor V);
while (1)
{
   refk_deg=GetPotRef_deg();
   posk_deg=GetPosM_deg();
   velk_rpm=GetVelM_rpm();

   escribir datos en display
   Simulator_Delay(100);
}
```

3.e) Añadir la comprobación del estado del bit 7 de pulsadores (leer DI) y accionar el relé en cada instante en función de su estado:

```
int valor_DI,valor_DO;
...
while (1)
{
   valor_DI=Simulator_ReadDI();
   if (bit de peso 7 de valor_DI está activo)
      valor_DO= valor que activa el bit de peso 0 de DO;
   else
      valor_DO= valor que desactiva el bit de peso 0 de DO;
   Simulator_WriteDO(valor_DO);
   ...
}
```

3.f) Añadir la activación de los bits 6 ó 7 de los LEDs en función del sentido de giro, modificando el valor del puerto de salida DO:

```
...
while (1)
{
   ...
   if (girando a derechas)     →   valor_DO = (LED7=0, LED6=1)
   else if (girando a izqdas)  →   valor_DO = (LED7=1, LED6=0)
   else (parado)               →   valor_DO = (LED7=0, LED6=0);
   Simulator_WriteDO(valor_DO);
   ...
}
```

4. Ampliaciones propuestas:

4.a) Mejorar la activación y desactivación de los LEDs de 3.e y 3.f para que la modificación de `valor_DO` no afecte al estado del resto de bits:

Ejemplo para activar bit 7:

valor_DO	B7	B6	B5	B4	B3	B2	B1	B0
¿OP?								
¿mask?	¿?	¿?	¿?	¿?	¿?	¿?	¿?	¿?
result	**1**	B6	B5	B4	B3	B2	B1	B0

Para comprobar, activar otros LEDs al principio y comprobar que se mantienen activos:

```
…
valor_DO = valor para activar LED4 y LED5;
while (1)
{

   ...
   if (giro a derechas)
      valor_DO = valor deseado (LED7=0, LED6=1, resto igual);
   else if (giro a izquierdas)
      valor_DO = valor deseado (LED7=1, LED6=0, resto igual)
   else // Parado
      valor_DO = valor deseado (LED7=0, LED6=0, resto igual);
   Simulator_WriteDO(valor_DO);
}
```

4.b) Cambiar la entrada para arrancar/parar motor al bit DI5, conectado a un pulsador, de manera que cada pulsación modifique el estado anterior del motor (primera pulsación arranca, segunda pulsación para, tercera pulsación vuelve a arrancar, …)

```
int valor_DI[2]; // Estados actual y anterior de pulsadores
int estado_motor[2]; // Estados actual y anterior de activación motor
…

estado_motor[0]=0;
valor_DI[0]= Simulator_ReadDI();
while (1)
{
   valor_DI[1]= valor_DI[0];          // O DesplazaTablaInt(valor_DI,2);
   estado_motor[1]=estado_motor[0];   // O DesplazaTablaInt(estado_motor,2);
   valor_DI[0]= Simulator_ReadDI();

   if (hay cambio 0 a 1 en B5 de valor_DI)
   {
      estado_motor[0]=!estado_motor[1];
      valor_DO= valor que activa bit de peso 0 de DO según estado_motor[0]
      Simulator_WriteDO(valor_DO);
   }
   ...
}
```

SIMULADOR FEEDBACK

El sistema físico y su conexión con el computador se encuentran simulados mediante una nueva descarga en AlMarSimulatorWSL. Para obtenerla:

- ❑ Ejecutar en ventana de comandos: wsl -d AlMarSimulatorWSL -u almar
- ❑ Abrir página web http://127.0.0.1:8080 → Menú Download
- ❑ Indicar y aplicar la dirección siguiente: http://isa.uniovi.es/~ialvarez/Curso/descargas/SimuladorWSL2/programacion_c.json
- ❑ Descargar: Simulador Feedback
- ❑ Volver a menú principal, seleccionar simulador Feedback.
- ❑ En el menú Parámetros, seleccionar PL = 7
- ❑ Ejecutar con Run. No detener la ejecución hasta el final de la práctica.
- ❑ Situar el ratón sobre cada elemento para ver sus características.
- ❑ Pulsar sobre interruptores y pulsadores para modificar entradas digitales
- ❑ Pulsar y arrastrar el triángulo del potenciómetro consigna para modificar. Id. para el freno.