# 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, Borja Millán | | |

PRACTICA 10: Servicio de interrupción y E/S en archivo

1. Reorganizar el programa realizado en prácticas anteriores para que el lazo de control se ejecute bajo interrupción del temporizador 0, de forma que el programa principal se pueda dedicar a la espera de comandos por teclado y el servicio de los mismos:

```
#include...

Declaración variables globales:
•  Todas aquellas que necesitan compartir main() y MiFnControl()
•  Todas aquellas que sólo usa MiFnControl() pero necesitan mantener su valor

Algunas a modo de ejemplo:
struct RZ  rz pos, rz vel;
float * ek pos,* ek vel,* uk...;
int  modo control, modo ref;

void MiFnControl()
{
   Declaración variables locales a MiFnControl() – sólo usadas aquí y que
   no necesitan mantener su valor anterior

   Actualizacion tablas temporales: Desplazar tablas uk, ek, …
   Leer nuevos valores posk,velk
   if (Switch 7 activo)
       uk[0]=0;
   else
   {
      if (modo control es POS)
      {
          Calcular _ek[0]
           uk[0]=valor control POS según  modo ref y switches
      }
      if (modo control es VEL)
      {
          Calcular _ek[0]
          _uk[0]=valor control VEL según _modo_ref y switches
      }
   }
   AplicarTensionMotor(_uk[0]);
   Actualizar LCD y bits de indicación de sentido de movimiento
}

main()
{
   char cmd[80];
   Otras vbles locales
   Dar valores iniciales a vbles globales: reguladores, ek, uk, modo control, etc.
   Simulator ConnectWss(...);
   Simulator_SetTimerInterrupt(0,100,MiFnControl);

   while (1)
   {
      printf(">>");
      gets(cmd);
      ProcesarCadena(cmd);  // → Actualiza vbles globales según contenido de cmd
   }
}
```

2. Añadir al comienzo de programa (después de inicializar el simulador y arrancar el temporizador de control, antes del bucle de solicitud por teclado) lectura de comandos del archivo de entrada "init.txt", que contendrá las líneas siguientes:

init.txt

```
RZ = [0.15, -0.11] / [1 -0.43]
POS = 90
SLEEP = 2500
POS = -90
SLEEP = 5000
POS = POT
```

Como resultado, el programa debe realizar al comienzo la asignación del RZ de posición y ejecutar los movimientos indicados, con una espera entre las dos consignas de posición.

3. Añadir a un archivo de texto "comandos.log" los diferentes comandos recibidos en el control del simulador, incluyendo la fecha y hora de cada uno. Utilizar funciones siguientes de <time.h> (ver ayuda con buscador Internet "man nombre_de_funcion"):

time_t time(time_t* pt_time); Obtiene la fecha y hora actual, como un entero que indica el tiempo en segundos que ha pasado desde 1/Ene/1970

struct tm* localtime(const time_t* pt_time); Obtiene la fecha y hora en una estrctura con campos año, mes, día, etc

Ejemplo de uso:

```
#include <time.h>

time_t ahora_sec;
struct tm* ahora;
…
ahora_sec=time(NULL);
ahora=localtime(&ahora_sec);
printf("La hora es: %02d:%02d\n",ahora->tm_hour,ahora->tm_min);
```

Ejemplo de salida deseada en el archivo comandos.log:

Comandos.log

```
02/12/2018 - 10:27:19 >> Arrancado programa de control
02/12/2018 - 10:27:25 >> RZ = [0.13,-0.11] /[1 -0.43]
02/12/2018 - 10:27:29 >> POS = 90
02/12/2018 - 10:27:30 >> SLEEP =3000
02/12/2018 - 10:27:33 >> POS =POT
```