Programación en múltiples archivos de código fuente

Si hay constantes, definición estructuras, etc. necesarias en varios .c :

- Declarar en un .h
- Todos los .c que las necesitan incluyen a ese .h

Si hay variables globales necesarias en varios .c :

- Uno de ellos las declara “normal”, y si lo desea las inicializa:

uno.c

```
int _modo_control=MODO_CONTROL_REF;
…
```

- El resto las declara “extern”, y no las puede inicializar:

otro.c

```
extern int _modo_control;
…
```

Esquema solución ampliación 1:

- Ejecutar Distrib_Python/server.py (requiere instalar python 3)

- Variable global tipo SOCKET (incluir winsock2.h) con valor inicial INVALID_SOCKET

- Ante comando "TCP CONNECT" → si hay un carácter '/' en el argumento tras el '=':
  extraer dirección IP y número de puerto, y con ellos realizar los pasos para conectar con el servidor similares al cliente de: http://isa.uniovi.es/~ialvarez/Descargas/pagina_de_prueba.htm
  A continuación, esperar la recepción de un mensaje y a partir del texto recibido se determina qué variable enviar en cada Tm

- En cada paso de control (interrupción), si la vble global tipo SOCKET no es INVALID_SOCKET: componer texto estilo jSON (utilizar sprintf) y enviar mediante send(). Si hay error en envío, cerrar socket con closesocket() y poner valor a INVALID_SOCKET

- Si se recibe comando "TCP CONNECT" → si el argumento es DISCONNECT, hacer lo mismo que ante error de envío del paso anterior

Esquema solución ampliación 2: (ver documentación curses en página del trabajo)

2.1) Variables globales tipo WINDOW* para cada una de las ventanas (se usan en main y en la función de interrupción)

```
WINDOW *_ventana_cmd,*_ventana_estado, …. ;
```

2.2) Inicializar curses y ventanas en main()

```
main()
{
        …
        Llamadas a funciones para iniciar curses

         // Creacion de una ventana
        _ventana_cmd=newwin(…..);
        wattron(_ventana_cmd, colores para la ventana);
        resto de inicialización de la ventana (borrar, poner marco, etc.)
        wrefresh(_ventana_cmd);

        … Resto de inicializaciones
}
```

Esquema solución ampliación 2:

2.3) Escribir en una ventana cuando sea necesario

```
...
    wmove(id ventana, posición deseada del cursor en la ventana);
    wclrtoeol(id ventana); //  Si se desea borrar hasta fin de linea
    wprintw(id ventana, resto como en printf);
…
    wrefresh(id ventana);
```

2.4) Esperar por cadena en una ventana

```
...
    wmove(id ventana, posición deseada del cursor en la ventana);
    wclrtoeol(id ventana); //  Si se desea borrar hasta fin de linea
    wprintw(id ventana, resto como en printf);
    wrefresh(id ventana);
    wgetstr(id ventana, cadena de caracteres);
```

Esquema solución ampliación 3:

3.1) Obtención de la posición en grados a partir del encóder

int enc_count=Simulator_ReadCounter(0); // Leer contador encóder 0
pos_k = conversión de unidades enc_count a grados, teniendo en cuenta que:
enc = 64 pulsos x 4 cuentas/pulso
enc está asociado al eje motor. El eje de salida se mueve con relación 1:8

3.2) Derivación de la posición para obtener la velocidad

$Vel_k=(pos_k-pos_{k-1})/Tm$ (Ojo unidades deg/ms no son rpm)

4

Esquema solución ampliación 4:

4.1) Variables globales necesarias

```
entero _modo_limpia  -> TRUE/FALSE
Tabla 2 enteros  _estado_SW0  -> para detectar cambios en el pulsador 0
entero _time_last_activation_SW0_ms -> tiempo desde la última activación del pulsador 0
entero _count_activation_SW0_3s -> número de activaciones en los últimos 3 segundos
```

4.2) En función de interrupción:

```
 DesplazaTablaInt(_estado_SW0,2);
 _estado_SW[0]= nuevo estado pulsador 0;
 if (SW0 ha cambiado de 0 a 1)
 {
        _time_last_activation_SW0_ms=0;
 }
else if (SW0 se mantiene en 1)
{
       _time_last_activation_SW0_ms+=Tm_ms;
}
else if (SW0 ha cambiado de 1 a 0)
{
        _count_activation_SW0_3s++;
        Según _count_activation_SW0_3s   Y _time_last_activation_SW0_ms:
              ACTIVAR LIMPIA: _modo_limpia=TRUE; _modo_control=MODO_POS;    _ref_teclado=70;
              DESACTIVAR LIMPIA: _modo_limpia=FALSE; _count_activation_SW0_3s=0;
}
…
```

Esquema solución ampliación 4:

4.2) Continuación

```
…

        if (_limpia_activado)
        {
             if (posk[0] > 69.5)
                  ref teclado = -70

             if (posk[0] < -69.5)
                  ref teclado = 70
         }
      ....  Resto del lazo de control, el control de posición hará el trabajo ....

}
```