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:

- Una aproximación discreta al regulador PID se puede obtener como:

$$R(z) = Kp + \frac{K_i.T_m.z}{z-1} + \frac{K_d.N.(z-1)}{(1+N.T_m).z-1}$$

(eq. 10 en https://www.scilab.org/discrete-time-pid-controller-implementation)

- Simplemente hay que realizar las operaciones requeridas para obtener:

$$R(z) = \frac{pol_{num}(z)}{pol_{den}(z)} = \frac{Kp.(z-1).\left((1+N.T_m).z-1\right) + K_i.T_m.z.\left((1+N.T_m).z-1\right) + K_d.N.(z-1)^2}{(z-1).((1+N.T_m).z-1)}$$

$$R(Z) = \frac{b'_0.z^2 + b'_1.z + b'_2}{a'_0.z^2 + a'_1.z + a'_2}$$

Dividiendo entre $a'_0.z^2$

$$R(Z) = \frac{b_0 + b_1.z^{-1} + b_2.z^{-2}}{1 + a_1.z^{-1} + a_2.z^{-2}}$$

$b'_0 = Kp.(1+N.T_m)+K_i.T_m.(1+N.T_m)+K_d.N$
$b'_1 = \cdots$
…

En PA-03 se realizó un ejemplo para producto de polinomios, que se puede utilizar para estos cálculos

$b_0 = {}^{b'_0}\!/_{a'_0}$
$b_1 = \cdots$
…

Una vez en esta forma, se aplica simplemente a los parámetros m,n,b[i],a[i] del regulador correspondiente

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) Constantes y variables globales necesarias

Nuevo estado para _modo_control : MODO_CONTROL_POS_LIMPIA
entero _elapsed_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
Tabla _valor_DI[2] para detectar cambios en pulsadores (ya se debería usar para otros apartados)

4.2) En la función de interrupción:

DesplazaTablaInt(_valor_DI,2);
_valor_DI[0]= nuevo estado pulsadores;
if (bit peso 0 de _valor_DI ha cambiado de 0 a 1) → pulsación: comienza cuenta tiempo
{
    _elapsed_last_activation_SW0_ms=0;
}
else if (bit peso 0 de _valor_DI está en 1) → mantenimiento pulsación: incrementa cuenta tiempo
{
    _elapsed_last_activation_SW0_ms += Tm_ms;
}
else if (bit peso 0 de _valor_DI ha cambiado de 1 a 0) → fin pulsación: chequea tiempos
{
    _count_activation_SW0_3s++;
    SI elapsed_last_activation_SW0_ms > 3 seg:
        ACTIVAR: _modo_control= MODO_CONTROL_POS_LIMPIA; _ref_teclado=70; count_activation_SW0_3s=0;
    SINO, SI count_activation_SW0_3s>2 veces
        DESACTIVAR : _modo_control= MODO_CONTROL_POS; _count_activation_SW0_3s=0;
}
…

Esquema solución ampliación 4:

4.2) Continuación

… Tomar en cuenta nuevo estado MODO_CONTROL_POS_LIMPIA…
Comparte toda la lógica de MODO_CONTROL_POS / MODO_REF_TECLADO, simplemente añade al final el cambio de la consigna según la posición actual :

```
if (posk[0] está cerca/ha sobrepasado 70°)
    ref teclado = -70;

if (posk[0] está cerca/ha sobrepasado -70°)
    ref teclado = 70
```