# Ejercicio 3 Gusano full

Proyecto de base: **p1a_tarea4_xxx.pds.prj** (Práctica_1a, tarea 4).

Partiendo del proyecto base (contador de dos dígitos), implementar una nueva funcionalidad (**modo gusano**) consistente en desplazar la visualización de un segmento según un bucle formado por los 4 dígitos del display de 7-segmentos, tal y como se indica en la figura.

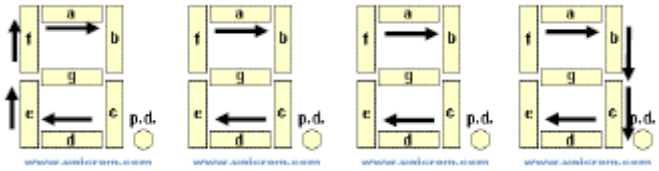
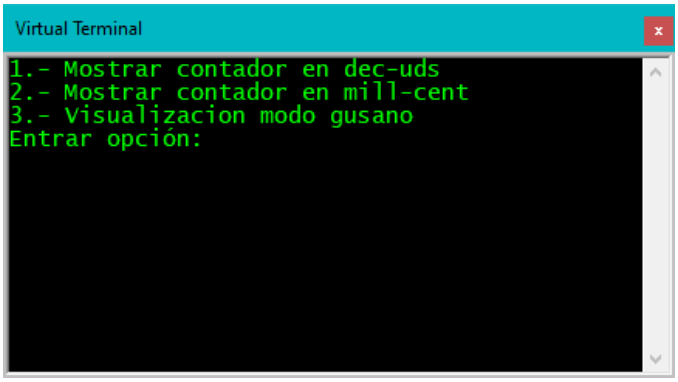



**Figura. Ej. movimiento hacia la derecha/izquierda (modo gusano)**

Para entrar en este modo de visualización se utilizará una nueva opción del menú (opción 3) que aparecerá en el "virtual terminal". Las opciones 1 y 2 devuelven el sistema al modo básico o contador de la tarea 4.




Para implementar esta nueva funcionalidad será necesario reconfigurar la conexión de los pulsadores "pright" y "pleft" que también afectará a la tarea 4 (modo contador) ya que ahora usarán las interrupciones como método de sincronización.

- Pulsador "pright" → al pin 20 (interrupción externa INT1)
- Pulsador "pleft" → al pin 21 (interrupción externa INT0)

**Modo contador:**En las opciones 1 y 2 el sistema trabaja en modo contador (tarea 4 de base). Realice los cambios que estime oportunos para que el contador siga funcionando correctamente ya que los pulsadores han cambiado de pin y usan interrupciones como mecanismo de sincronización.

**Modo gusano:**Cuando se entra en opción 3, la variable "sentido" (modificada por los pulsadores "pright" y "pleft")indicará si el desplazamiento del segmento encendido se hace hacia la derecha o hacia la izquierda. Cada pulsación generará una interrupción que será atendida por la correspondiente ISR() que actualizará la variable "sentido":

Si se pulsa "pright" entonces sentido = 0 → movimiento a la derecha

Si se pulsa "pleft" entonces sentido = 1 → movimiento a la izquierda

La estructuración del programa se hará en base a las siguientes funciones mínimas:

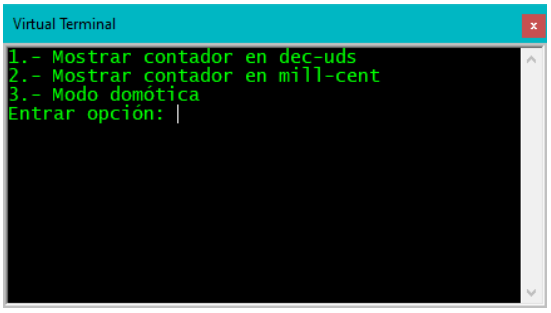
```
// Inicialización de variables globales

voidSetup(){

// inicialización de puertos
//inicializar interrupciones externasINT0 (pin21, pleft), INT1(pin 20, pright), INT2 (pin 19, 100Hz)
```

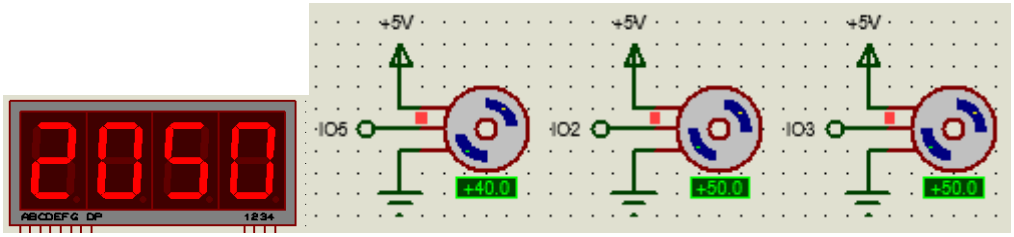
```
}
ISR(INT2_vect){
// gestión del barrido entrelazado del display de 7-segmentos → tarea 4
}
ISR(¿?) {
// gestión del pulsador “pright” (pin 20) para los modos contador y gusano
}
ISR(¿?) {
// gestión del pulsador “pleft” (pin 21) para los modos de contador y gusano
}
voidloop() {

// Si (opción == 1 || opción == 2) → funcionamiento modo  contador

// Si (opción == 3)→ funcionamiento en modo gusano (un bucle completo antes de volver a loop()). El “sentido” se cambia con los pulsadores “pright” y “pleft”.

}
```

# Ejercicio 5 Domótica

Proyecto de base: **p1a_tarea4_xxx.pds.prj** (Práctica_1a, tarea 4).

Partiendo del proyecto base (contador de dos dígitos), implementar un nuevo modo de funcionamiento, llamado **modo domótica,** consistente en controlar el grado de apertura de las láminas de una persiana motorizada mediante el uso de servomotores. El sistema tendrá capacidad para controlar hasta tres servomotores (tres persianas). El modo de funcionamiento se selecciona a través de una nueva opción (opción 3) que aparece en el menú visualizado en el "virtual terminal".




**Descripción del funcionamiento en modo domótica.**

En el dígito más significativo del display debe aparecer el número de persiana o servomotor (0, 1 o 2) que se va a controlar y, en los 3 dígitos menos significativos, el ángulo en el que se encuentra el motor (posición de las láminas de la persiana). El gobierno de los servomotores se realizará con ayuda de los pulsadores que tendrán la siguiente funcionalidad:

- Pulsador "pright": Incrementa el ángulo del servomotor en 10º. Máximo 180º.
- Pulsador "pleft": Decrementa el ángulo del servomotor en 10º. Mínimo 0º
- Pulsador "pup": Selecciona el siguiente servomotor (0,1,2). Máximo 2
- Pulsador "pdown": Selecciona el anterior servomotor (2, 1,0). Mínimo 0

El servomotor seleccionado para cada una de las persianas tiene las siguientes características:

- Señal de control PWM de frecuencia 250 Hz (T = 4 milisegundos).
- La posición angular del eje del motor se controla mediante el ciclo de trabajo (duty cycle) de la señal PWM (ancho del pulso) de acuerdo a las siguientes especificaciones:
  - ✓ Ancho del pulso 0.5 ms: 0º; Ancho del pulso 1.5 ms: 90º; Ancho del pulso 2.5 ms: 180º
- Dispositivo Proteus: MOTOR-PWMSERVO (Hobby servo)

Utilizar el Timer 3 para generar las señales de mando de los servomotores de acuerdo a las siguientes especificaciones:

- Modo 14: Fast PWM, TOP = ICR3
- Señales PWM para el control de los 3 servomotores: OC3A (pin 5), OC3B (pin 2) y OC3C (pin 3)
- Frecuencia de la señal PWM: 250 Hz (T = 4 ms)

- TOP: Escoger el valor de TOP más pequeño que se pueda representar con un número entero (sin decimales). Esta especificación nos permitirá seleccionar el valor N del prescaler.
- Ancho del pulso (duty cycle): desde 0.5 ms (0º) hasta 2.5 ms (180º)

A continuación, se proporciona una posible organización del software:

```
// Inicialización de variables globales

void setup(){

// inicialización de puertos y canal serie (práctica base)
//inicializar interrupciones externas: INT2 (pin 19, 100Hz)
// inicializar pines del timer3 (5, 2, 3)
// inicializar timer 3 (modo 14, fast PWM, TOP =ICR3, f = 250 Hz)
}
ISR(INT2_vect){
// if (domo == 0) gestión del barrido entrelazado del display de 7-segmentos → tarea 4
// if (domo == 1) visualización de nº de servo y ángulo
}
void loop() {

// leer canal serie. Si hay dato de entrada actualizar "opción"{

        // Si (opción == 1 || opción == 2) → modo  contador (domo =0)

        // Si (opción == 3)→ modo domótica (domo=1).

        }

// si (domo == 0) -> funcionalidad contador

// si (domo == 1)  -> funcionalidad domótica. Lee pulsadores y actualiza variables: "servo" (0,1,2) y "angulo[servo]". Actualiza registros de comparación OCR3x, si procede, para cambiar la posición angular de un servo.

}
```