Considere un sistema de refrigeración de una sala de reuniones compuesto por dos dispositivos conectados a una computadora que ejecuta un sistema operativo Linux: (i) un ventilador y (ii) un sensor de temperatura ambiente con cronómetro incorporado. Cada dispositivo debe ser manejado por un driver independiente.

Se pide:

1. Proponer un diseño, en donde debe indicar cuántos y qué tipo de registros tendría cada dispositivo, e indicando también para qué se utilizarían. Indicar y justificar el tipo de interacción con cada dispositivo (interrupciones, polling, dma, etc.).

2. Una vez que tenga el diseño, escribir los drivers correspondientes a ventilador y temperatura, de modo tal que cumplan los siguientes objetivos:

   - Al iniciarse, la aplicación de usuario deberá recibir tres parámetros de configuración por entrada estándar: un umbral de temperatura mínima, un umbral de temperatura máxima, y un tiempo T (todos enteros).
   - El sensor de temperatura deberá poder informar con un número entero el promedio de la temperatura de los últimos T segundos.
   - Si la temperatura promedio de los últimos T segundos se encuentra por debajo del valor mínimo, el ventilador deberá apagarse.
   - Si la temperatura promedio de los últimos T segundos se encuentra por arriba del valor máximo, el ventilador deberá encenderse.
   - Cualquier temperatura que se encuentre entre la mínima y la máxima se deberá considerar dentro del rango normal de refrigeración, y no deberá tener ningún impacto en el estado del ventilador.
   - Para reducir el consumo energético del sistema, el ventilador solamente deberá cambiar de estado cuando la temperatura promedio se encuentre fuera del rango normal (menor a la temperatura mínima o mayor a la máxima).
   - No está permitido realizar sleep u otras operaciones similares.
   - Para cada driver se deberá implementar en código C las funciones mínimas necesarias para poder cumplir el objetivo planteado. El código deberá ser sintácticamente válido y respetar las buenas prácticas mencionadas durante las clases. Por simplicidad, siempre que esto no impacte en la solución, se permitirá omitir el chequeo de errores. Todas las decisiones implementativas deberán estar debidamente justificadas.

3. Además, se solicita explicar el funcionamiento de la aplicación de usuario, y su interacción con los drivers utilizando pseudocódigo lo más similar a C posible. Tenga en cuenta que la aplicación de control correrá a nivel de usuario. Indicar con código C cómo interactuará el software de control con los drivers. Cada operación usada debe estar justificada.

Implementar cualquier función o estructura adicional que considere necesaria (tener en cuenta que en el kernel no existe la libc). Se podrán utilizar además las siguientes funciones vistas en la práctica:

```
unsigned long copy_from_user(char *to, char *from, uint size)
unsigned long copy_to_user(char *to, char *from, uint size)
int IN (int regnum)
void OUT(int regnum, int value)
void *kmalloc(uint size)
void kfree(void *buf)
void request_irq(int irqnum, void *handler)
void free_irq(int irqnum)
void sema_init(semaphore *sem, int value)
void sema_wait(semaphore *sem)
void sema_signal(semaphore *sem)
void mem_map(void *source, void *dest, int size)
void mem_unmap(void *source)
```