Trabajo Práctico N°4: RTOS

El proyecto inicia desde el archivo *main.c*, en el cual se declara un mensaje de bienvenida que se imprime utilizando la interfaz UART. Se inicializan las tareas, y las colas de mensajes que éstas utilizaran. Por último, se da paso al scheduler.

Las tareas creadas son la siguientes:

vStartSensorTask

Tal como lo indica el nombre, es la tarea encargada de simular el trabajo de un sensor. Utiliza la función *rand()* y arroja un valor aleatorio entre las constantes previamente definidas *MAX_TEMP* y *MIN_TEMP*. El valor generado se envía a través de una cola de mensajes al filtro pasabajos. Se genera un valor cada cien milisegundos.

vStartFilterTask

Esta es la siguiente tarea en inicializarse en el *main.c*, y su objetivo es leer las últimas N mediciones del sensor y devolver el promedio. Para ello se utiliza un buffer circular que se va actualizando a medida que se leen nuevas mediciones, asegurándonos de siempre tener las últimas N. El valor de N se puede modificar desde el código o a través de la interfaz de UART. La tarea monitorea constantemente si el valor de N cambio para así modificar el tamaño del buffer circular. Una vez obtenido el promedio, se completa la estructura TempData_t que tiene la forma:

```
// Estructura para pasar datos del filtro al display
typedef struct {
  int temperature;
  long time_ms;
} TempData_t;
```

Esta estructura es útil ya que nos permite ponerle un lugar en el tiempo al promedio obtenido para luego poder graficar. Completada la estructura, se envía a la siguiente tarea a través de otra cola de mensajes.

vStartDisplayTask

Esta es la tarea encargada de graficar los valores de temperatura generados por el filtro pasabajos en función del tiempo. También utiliza un buffer circular que se va completando a medida que se leen medidas y luego rotando una vez lleno. Esto nos sirve para lograr que el gráfico se desplace hacia la izquierda en función del tiempo.

El gráfico tiene la siguiente forma:

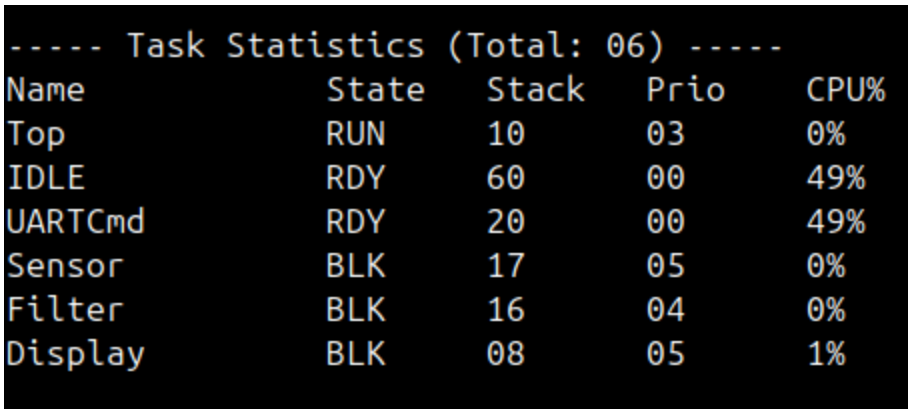
Sobre la izquierda se puede ver el valor de temperatura y el tiempo correspondiente a ese valor.

vStartUartCmdTask

Es la tarea que debe escuchar por UART los valores recibidos para ver si tiene que modificar el valor de N que se usará para la cantidad de muestras a filtrar. El formato esperado es N=X, donde X es el valor que deseamos tener de cantidad de muestras. Si la tarea detecta que llego un nuevo valor de N, entonces modifica una variable global a la tarea vStartUartCmdTask y vStartFilterTask llamada requestedN. vStartFilterTask compara constantemente si requestedN es distinto de currentN, y si es así ajusta el buffer haciéndolo más grande o pequeño según sea necesario.

vStartTopTask

Es la última de las tareas y hace provecho de la API de FreeRTOS con las funciones uxTaskGetNumberOfTasks() y uxTaskGetNumberOfTasks() para poder obtener información de manera dinámica de todas las tareas corriendo en el sistema. La información proporcionada para cada tarea es: el nombre, el estado (RUN, RDY, BLK, SUS), la prioridad y el uso de CPU:



Sobre prioridades en las tareas

Las tareas con más prioridad son las del sensor y el display, por criterio de diseño. Se asume que la importancia recae en la lectura de la información y la posterior publicación de la misma, para poder realizar una toma de decisiones. El filtrado si bien es importante tiene una prioridad más baja que las anteriores dos. Luego viene la tarea del Top, que puede servir en modo de monitoreo del sistema pero no es información crucial. Y por último la tarea que menos prioridad tiene es la del UART, ya de vuelta su aplicación tampoco es crítica y podría esperar si fuese necesario.

Análisis de Stack

Mediante el uso de la función uxTaskGetStackHighWaterMark() pudimos monitorear la cantidad de palabras restantes en el stack de cada función. De una cantidad inicial de 70 palabras, en todos los casos vimos que había un gran sobrante al final de cada función, por lo que se pudo reducir la cantidad inicial. Se dejó un 20% de margen de sobra para cada tarea. El análisis se puede ver en la rama **stack-hwm** del repositorio si lo clonamos y ejecutamos:

```
cd SO2-RTOS
```

```
git checkout stack-hwm

cd FreeRTOS/Demo/CORTEX_LM3S811_GCC

./make.sh

qemu-system-arm -machine lm3s811evb -cpu cortex-m3 -kernel gcc/R
```

Los valores de stack inicial para cada tarea quedaron como:

```
// Tamaños de stack para cada tarea. Se deja un 20% de margen
#define SENSOR_TASK_STACK_SIZE 42
#define FILTER_TASK_STACK_SIZE 58
#define DISPLAY_TASK_STACK_SIZE 60
#define UART_CMD_TASK_STACK_SIZE 48
#define TOP_TASK_STACK_SIZE 50
```