Práctica 5: Sistemas Operativos de Tiempo Real

Hecho por Vladislav Kravchenko

Durante la realización de la practica 5 he trabajado con el sistema operativo RTX. Para el correcto funcionamiento del trabajo, como pasos previos, he configurado el programa Keil según los anexos proporcionados por el profesorado. En estos anexos, concretamente en el anexo **Practica_5_RTX_Anexo IV- Buffer FIFO Circular,** viene explicado detalladamente como funciona un Buffer FIFO circular, y, además, se realizan varios ejemplos de código. Durante la realización de la práctica me he apoyado en uno de estos códigos para obtener un buffer FIFO circular que guarde y saque los números primos obtenidos por el programa. En esta práctica al trabajar con el sistema operativo RTX, debemos eliminar del programa el SysTick y realizar la temporización con la función **delay ()**, con la que ya hemos trabajado en la practica 3.

Definimos dos hilos que se encargaran de obtener y representar los números primos. Estos hilos son el hilo productor y el hilo consumidor. Para definirlos:

```
osThreadId productor_id;
osThreadId consumidor_id;

void productor(void const *argument);
void consumidor(void const *argument);

osThreadDef(productor,osPriorityNormal,1,0);
osThreadDef(consumidor,osPriorityNormal,1,0);
```

Para obtener la funcionalidad que nos piden en la práctica y para evitar los problemas de concurrencia entre los hilos, me ayudo de los mecanismos de sincronización. Para resolver el caso en el cual el hilo productor quiera introducir un dato en el buffer y se encuentre lleno, y cuando el hilo consumidor quiera sacar un dato del buffer y este se encuentre vacío, me ayudo de los semáforos. Con las sentencias:

- **osSemaphoreWait:** operación P, espera a que el semáforo se encuentre disponible.
- **osSemaphoreRelease:** operación V, libera un semáforo.

En el **main ()**, creo los semáforos:

```
meterfifo_id=osSemaphoreCreate(osSemaphore(meterfifo),8); //Posiciones libres al inicio=8
sacarfifo_id=osSemaphoreCreate(osSemaphore(sacarfifo),0); //Al inicio hay 0 datos que sacar al estar vacio
```

Para evitar la concurrencia entre hilos me ayudo del mutex con las sentencias:

- **osMutexWait:** Operación P, espera a que el mutex se encuentre disponible.
- **osMutexRelease:** Operación V, libera un mutex que fue obtenido mediante una llamada a osMutexWait.

En el **main ()**, creo el mutex con la siguiente sentencia:

- **mutex_id**=osMutexCreate(osMutex(**mutex**));

También en el main () creo los semáforos:

- **meterfifo_id**=osSemaphoreCreate(osSemaphore(**meterfifo**),8);
- **sacarfifo_id**=osSemaphoreCreate(osSemaphore(**sacarfifo**),0);

Por todo lo demás, la práctica mantiene la misma funcionalidad de la practica 4, con las interrupciones, el sentido de la cuenta, la asignación de los valores de máximo y mínimo, y el cambio de representación de binario a BCD y viceversa.