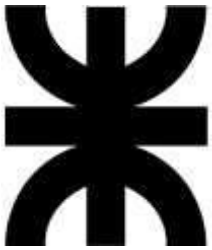

Universidad Tecnológica Nacional

Facultad Regional Avellaneda

Departamento de Ingeniería Electrónica

Cátedra: Técnicas Digitales III

Título del Trabajo Práctico:	Trabajo Práctico Nº2: FreeRTOS - Colas
Autores:	RIVAS, Martín

Resumen:
Contenidos analizados: • Gestión y API correspondiente a Colas • Gestión de ADC • Gestión de Interrupciones

Sistemas Operativos en Tiempo Real: FREERTOS

OBSERVACION: La entrega del presente trabajo se realizará mediante GitHub, en una carpeta llamada tp2_colas dentro de la sección 2.trabajos_practicos. El correspondiente branch será nombrado tp2_colas y el commit tp2_colas_entrega. La parte teórica debe ser entregada en formato PDF.

Parte Teórica

1) Desarrolle como es el almacenamiento de datos en el recurso "Cola" proporcionado por el Kernel.

2) Explique utilizando un ejemplo con código el porqué es común que el recurso Cola posea múltiples tareas escritoras y una sola tarea lectora.

3) Explique el proceso de bloqueo por escritura y bloqueo por lectura en el recurso Cola.

4) A la hora de enviar datos compuestos: explique cómo se lleva a cabo dicho proceso y por qué no genera conflictos con la definición de la macro del recurso. ¿Cuál sería la implementación si los datos fueran "grandes", lo que ocasionaría un problema con la memoria RAM?

1) Una cola puede contener un número finito de elementos de datos, todos del mismo tamaño fijo. El número máximo de elementos que una cola puede contener se llama su "longitud".
Tanto la longitud y el tamaño de cada elemento de datos se establecen cuando la cola es creada.
Normalmente las colas son usadas como buffers primero en entrar-primero en salir (FIFO) donde los datos se escriben en el final de la cola y se retira de la parte frontal de la cola. También es posible escribir en la parte delantera de una Cola. La escritura de datos a una cola se lleva a cabo copiando byte por byte de los datos a ser almacenados en la cola en sí misma.
La lectura de datos de una cola provoca una copia de los datos que se eliminaran de la cola.

2) Es común que el recurso cola tenga múltiples tareas escritoras y una sola lectora debido al uso de la comunicación asincrónica y la necesidad de compartir información entre tareas de manera eficiente y sincronizada.
Por ejemplo, podría ser el caso de múltiples tareas que cada una cumpla la función de leer un parámetro en la cola, tal vez proveniente de un sensor y haya una única tarea lectora (la controladora) que va a ser la que controle la escritura de una pantalla LCD.

```
void writeTask1 (void *pvParameters){
        xRead value;

        while(1){
                leerSensor1();
                xQueueSendToBack(queue, &value, portMAX_DELAY;
                vTaskDelay(200/portTICK_PERIOD_MS);
        }
}
```

```
void writeTask2 (void *pvParameters){
        xRead value;

        while(1){
                leerSensor2();
                xQueueSendToBack(queue, &value, portMAX_DELAY;
                vTaskDelay(200/portTICK_PERIOD_MS);
        }
}
void readTask (void *pvParameters){
        xRead value;

        while(1){

                xQueueReceive(queue, &value, portMAX_DELAY);
                escribirLCD();
        }
}
```

3) El proceso de bloqueo por escritura se produce cuando una tarea va a escribir sobre una cola y la misma ya está llena. La tarea va a pasar a estado Lista en 2 casos. Cuando se cumpla el tiempo máximo de bloqueo aclarado o cuando una tarea libere un dato de la cola leyéndolo.
El proceso de bloqueo por lectura se produce cuando una tarea va a obtener un dato sobre una cola y la misma esta vacía. La tarea pasará a estado Lista cuando otra tarea escriba un dato sobre la misma o cuando se cumpla el tiempo máximo de bloqueo establecido.
En el primer caso, se puede dar que más de una tarea este necesitando el dato, en ese caso se liberará la tarea de mayor prioridad que estaba esperando más tiempo. Lo mismo puede pasar en el caso de escritura, si más de una tarea estaba esperando para escribir, en el momento que se libere la cola, escribirá sobre ella la de mayor prioridad que estaba esperando más tiempo.

4) Para transferir datos compuestos, lo que se hace es definir una estructura y transferir el valor de los datos y la fuente de los datos estén contenidos en los campos de la estructura.
Cuando los datos son grandes, lo que se puede hacer es mandar por la cola un puntero al dato. El cual va a ser más fácil de copiar byte a byte y más liviano. Lo que hay que tener en cuenta al pasar punteros, es que la tarea receptora pueda acceder a la memoria apuntada por el puntero y sea la misma que cuando se guardo el dato. Una tarea debe ser la responsable de liberar la memoria porque sino la memoria apuntada sigue siendo válida.