Carrera de Especialización en Sistemas Embebidos
Sistemas Operativos en Tiempo Real 2.

Trabajo Práctico - Parte 1:

El objetivo de esta práctica es aprender a trabajar con algoritmos de asignación de memoria en un contexto de ejecución de tiempo real. Estos algoritmos están orientados a tener un tiempo de ejecución corto y determinista.

No se pretende que el alumno escriba un asignador de memoria dado que hay numerosas implementaciones disponibles en la red. Se espera que el alumno justifique su elección del algoritmo de asignación de memoria.

Además, el trabajo práctico permite aprender a trabajar con procesamiento asincrónico de eventos.

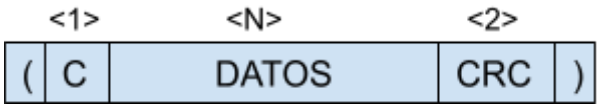
Descripción conceptual de la práctica:

Basándose en el driver de USART de sAPI, se deberá implementar un driver de mayor nivel, que permita separar paquetes entrantes.

El protocolo establecido indica que los paquetes están separados por delimitadores. Se utilizará '(' para abrir un paquete y ')' para cerrarlo.

Sin embargo, el "driver" deberá contemplar que luego de cierto tiempo relativo respecto del último byte recibido, el paquete se descarte si no recibe el EOM (esto se hace por si un usuario comienza a enviar un paquete, y no lo termina, o que ocurra algo similar ante alguna condición de error).

Cada paquete poseerá, además de los delimitadores, un campo de 'código' (C), un campo de datos (DATOS) y otro de comprobación (CRC). Todo el paquete estará codificado en ASCII.



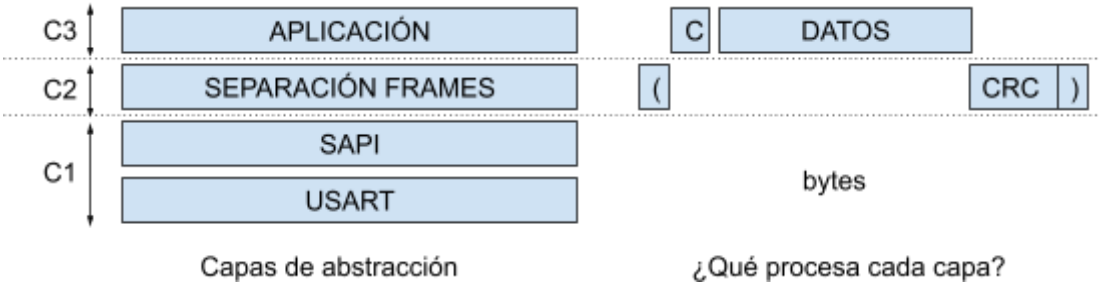
El algoritmo para el cálculo del CRC será el de crc8 (usando 0x00 como semilla. Ver test online) y se calculará sobre el campo de C y el de DATOS. El algoritmo de crc8 arrojará un byte, pero su representación en el paquete será ascii en hexadecimal, de manera que el 1er byte del campo CRC estará representado por un caracter de '0' a 'F' que identifique los 4 bits más significativos del valor del crc, y el 2do byte que identifica a los 4 bits menos significativos.(ej: crc= 0x1B => CRC = "1B").

La capa de abstracción C2, encargada de separar frames, deberá garantizar que la separación de los mismos ocurra en el contexto de la interrupción de recepción. Además, esta capa será la responsable de validar el CRC. Al recibir un paquete con el formato correcto, ésta capa de separación informará a la aplicación de la llegada de un paquete nuevo.

La aplicación, capa C3, deberá tener la única responsabilidad de recibir paquetes, validarlos, procesarlos y enviar una respuesta.

Carrera de Especialización en Sistemas Embebidos

Sistemas Operativos en Tiempo Real 2.



La aplicación deberá validar el contenido del paquete. Los paquetes deberán contener solo texto. El campo de C identificará la acción a realizar con los DATOS. Si C es 'M' la acción consistirá en transformar DATOS en letras mayúsculas, y si C es 'm' la acción consistirá en transformar DATOS en letras minúsculas. Los datos procesados deberán posteriormente ser enviados a la capa C2 y ésta, en el sentido contrario será la responsable de agregarle el campo de verificación, los delimitadores y de enviarla a la SAPI para que lo envíe al periférico.

En caso de que algún caracter recibido no sea válido, la aplicación deberá responder un mensaje de error.

Requerimientos:

Del TP:

Código	Requerimiento
R_TP-1	Todas las justificaciones deberán estar plasmadas en un archivo readme.md (markdown). Deberá estar versionado con el código fuente.
R_TP-2	El grupo deberá justificar la elección de las arquitecturas que utilice.
R_TP-3	El grupo deberá justificar la elección del esquema de memoria dinámica utilizada.
R_TP-4	El grupo deberá poseer un repositorio que los docentes puedan consultar libremente. Las entregas podrán ser utilizando la opción de empaquetar un release, o simplemente teniendo una rama llamada release.

Generales:

Código	Requerimientos
R_G-5	Cada capa de abstracción deberá estar diseñada de forma tal que se pueda instanciar (y eventualmente se pueda reutilizar varias instancias dentro de un mismo Firmware, por ej, para utilizarla con varias USARTs)

Carrera de Especialización en Sistemas Embebidos
Sistemas Operativos en Tiempo Real 2.

Capa separación de frames (C2).

Código	Requerimientos
R_C2-1	La cantidad máxima de bytes de cualquier paquete de datos será de 200 caracteres.
R_C2-2	Se deberá procesar paquetes que comienzan con SOM = "(" y finalizan con un EOM= ")"
R_C2-3	Deberá procesar en contexto de ISR todos los bytes entrantes y los salientes para todas las operaciones de la C2
R_C2-4	Los bytes entrantes, deberán almacenarse en un bloque de memoria dinámica
R_C2-5	Deberá tener control sobre la máxima cantidad de bytes recibidos
R_C2-6	Al elevar un paquete recibido a la capa de aplicación (C3), la C2 deberá poder seguir recibiendo otro frame en otro bloque de memoria dinámica, distinto al anterior
R_C2-7	En caso de no haber memoria, la recepción de datos del driver de la C1 deberá anularse
R_C2-8	El tiempo para cancelar un frame de datos inconcluso será de T = 60ms
R_C2-9	El timeout deberá implementarse con un timer de FreeRTOS
R_C2-10	En caso de que se haya cumplido el timeout, el frame no haya sido cerrado, el paquete deberá descartarse.
R_C2-11	El paquete deberá calcular el código de comprobación del paquete utilizando el algoritmo CRC8 con semilla = 0.
R_C2-12	En caso de que el código de verificación no sea validado, el paquete deberá descartarse
R_C2-13	Al recibir un mensaje correcto, se deberá señalizar a la aplicación de su ocurrencia, para ser procesada.
R_C2-14	Cuando la aplicación (C3) desee enviar un mensaje por el canal de comunicación, C2 deberá agregarle el código de comprobación y los delimitadores
R_C2-15	La transmisión de las respuestas al canal, deban cumplir la misma premisa que en la recepción: Entre paquetes debe existir un tiempo muerto de T=60ms

R_C2-16	Al finalizar la transmisión, se deberá liberar la memoria dinámica utilizada para la transacción.

Capa de aplicación (C3)

Código	Requerimientos
R_C3-1	El campo C deberá ser solamente 'M' o 'm' y estará asociado a la acción mayusculizar y minusculizar, respectivamente.
R_C3-2	El campo DATOS deberá procesar solo paquetes de texto a-z , A-Z y espacios
R_C3-3	Si hay paquetes con caracteres que no cumplan, deberán descartarse, y enviar a la capa C2 el texto ERROR1
R_C3-4	Si el campo C no es válido, el paquetes deberá descartarse y enviar a la capa C2 el texto ERROR2
R_C3-5	El texto entrante deberá pasarlo a mayúscula o minúscula según el campo C

Opcionales:

Código	Requerimientos
R_OP-1	En ocasiones, los protocolos que separan frames por tiempo, definen timeouts de frame en función del baudrate (ej MODBUS RTU). En algunos casos, el ms no es una unidad válida porque el TO resulta ser más chico. Implementar el ejercicio SIN timers de FreeRTOS, utilizando un timer de hardware

Trabajo Práctico - Parte 2:

Una vez finalizados los requisitos plasmados en la parte 1, el firmware deberá modificar algunas partes para hacer uso de objetos activos.

En la parte 1, la capa de aplicación se implementó mediante una tarea “tradicional” del RTOS. Para la parte 2 habrá que transformar la capa de aplicación en un objeto activo.

Por otro lado, en la parte 1, el procesamiento de los datos que se hizo dentro de la tarea de la capa de aplicación.

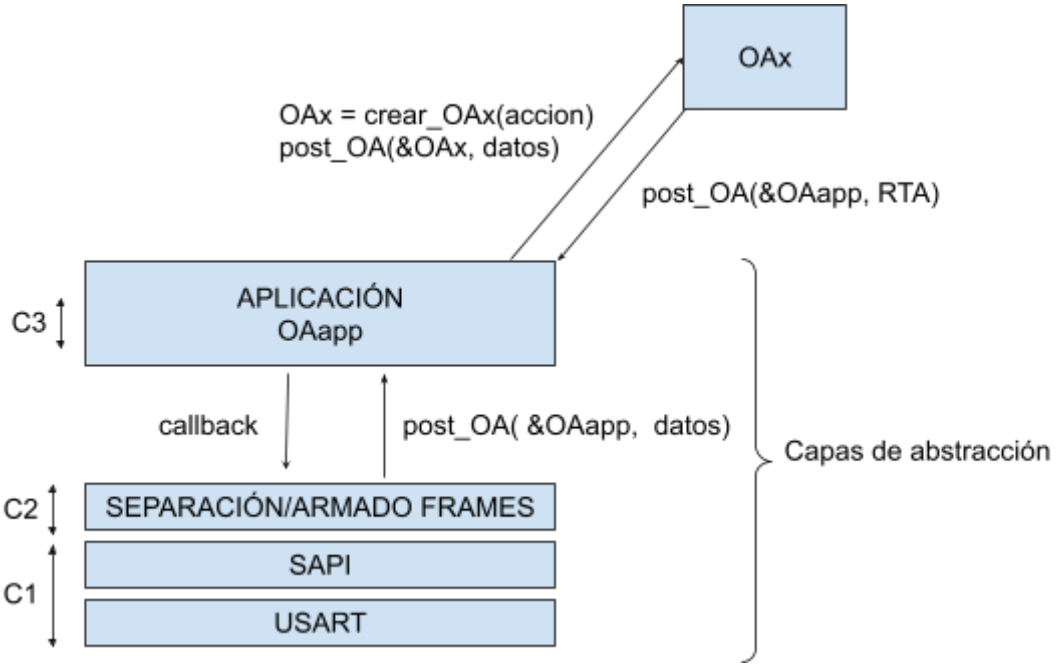
Carrera de Especialización en Sistemas Embebidos

Sistemas Operativos en Tiempo Real 2.

Para la parte 2, se delegará el procesamiento del paquete a otro objeto activo según el campo C que se reciba, dentro de la capa de aplicación.

Para cada frame recibido, la capa de aplicación creará un objeto activo, al que le delegará el procesamiento de las acciones de mayusculizar o minusculizar.

Al finalizar el procesamiento, el OA enviará por callback a la aplicación la respuesta, y ésta la enviará al driver.



Requerimientos nuevos

Capa de aplicación (C3)

Código	Requerimientos
R_AO-1	La aplicación deberá transformarse en un objeto activo. OAapp
R_AO-2	La aplicación deberá esperar dos tipos de evento: a- un evento proveniente del driver que signifique “llegó un paquete procesar”. b- un evento, con la respuesta procesada.
R_AO-3	El evento “a” desencadenara tres acciones: a1- validar el paquete a nivel C3 a2- si es válido, enviar un evento dinámico al OA asociado a la operación solicitada. a3- si es invalido, enviar la respuesta de error a C2.
R_AO-4	El evento “b” desencadenará una sola acción:

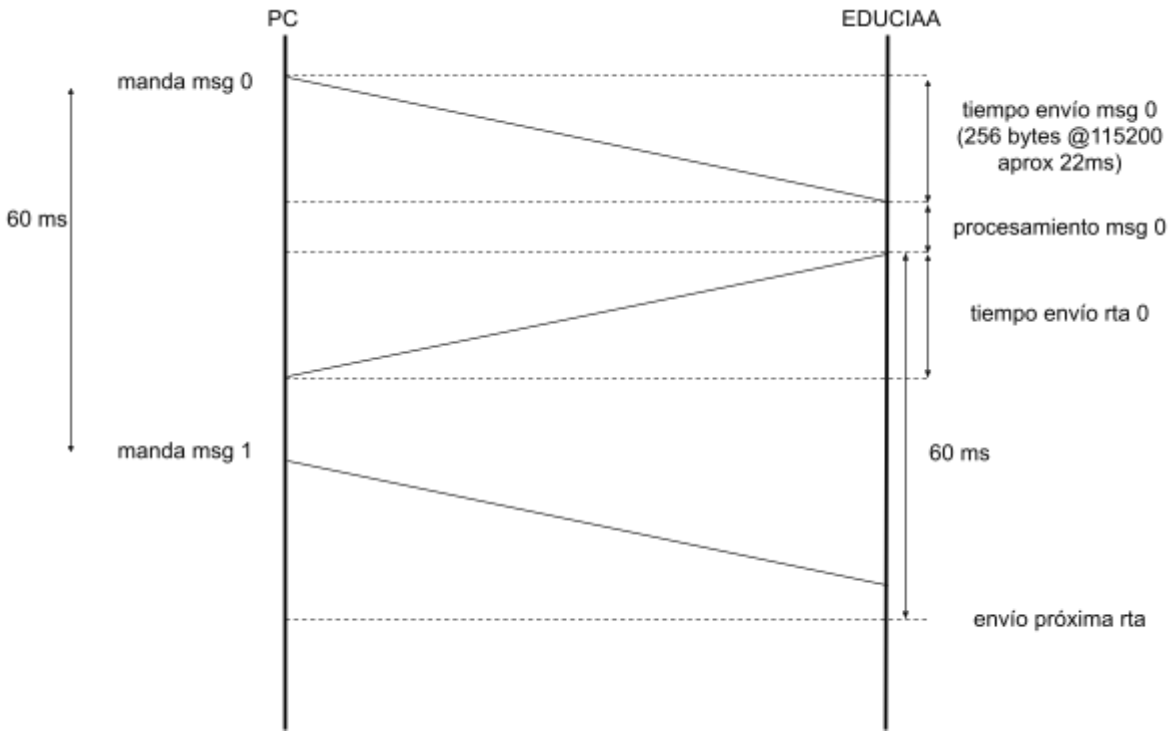
	b1- Enviar la respuesta procesada a C2.
R_AO-5	Existirán dos OSs de procesamiento: el de mayusculizar y el de minusculizar. (OAM y OAm)
R_AO-6	En caso de que cualquiera de las OAM o OAm no existan, deberán instanciarse en tiempo de ejecución
R_AO-7	Al finalizar la operación, cada OAM o OAm deberá enviar la respuesta a traves de un evento a OAapp.
R_AO-8	Al finalizar la operación, cada OAM o OAm, si no quedan elementos para procesar en la cola de entrada, deberá destruirse.

Sugerencias:

- Deténgase a pensar la arquitectura papel, modularizado y asignando responsabilidades a cada módulo.
- Implementar los requisitos de a uno y verificar el funcionamiento de los mismos antes de pasar al siguiente.
- Implementar el procesamiento de datos con una FSM.

Diagrama en secuencia

Se presenta un diagrama en secuencia que marca los límites temporales en el envío de comandos y recepción de respuestas.



Carrera de Especialización en Sistemas Embebidos

Sistemas Operativos en Tiempo Real 2.

Historial de cambios

Rev	Fecha	Autor	Detalle
0	2020/02/20	Franco Bucafusco	Creación del documento
1	2020/07/02	Franco y Martin	Modificaciones para 11va cohorte
2	2020/07/10	Franco	Aclaración sobre la semilla a utilizar para el cálculo de CRC. Agregado de límite de bytes por paquete. Agregado de aclaración en R_C2-2. Agregado de aclaración de branch de repos.
3	2020/10/28	Franco	Se actualizan y ordenan requerimientos. Se cambia la app a AO.
4	2021/05/02	Franco	Se agrega diagrama de secuencias