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 en donde se debe aprovechar al máximo la utilización del CPU.

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

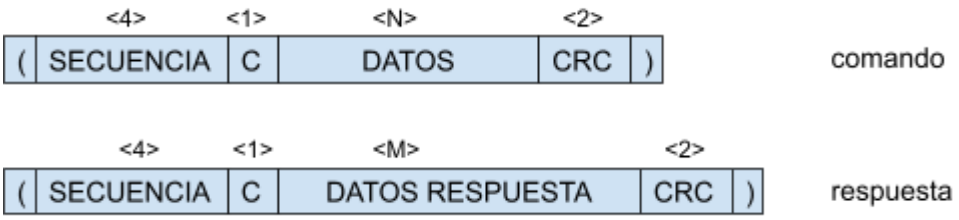
Descripción conceptual de la práctica:

Basándose en el driver de USART de sAPI, se deberá implementar un driver de mayor nivel, que permita separar paquetes entrantes. Además, deberá implementar una aplicación para ese driver, en donde se procese, y se permita obtener una respuesta.

El protocolo establecido indica que los paquetes están separados por delimitadores. Se utilizará '(' (Start Of Message, SOM) para abrir un paquete y ')' (End Of Message, EOM) 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). Además, recibir un nuevo SOM reiniciará el paquete, descartando lo recibido previamente.

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



El número de secuencia, servirá al host de comandos para saber si un cierto paquete se envió correctamente. Se representa a través de valores ASCII decimales de 4 caracteres, completando a izquierda con 0 de ser necesario (ej, "0234" = 0234).

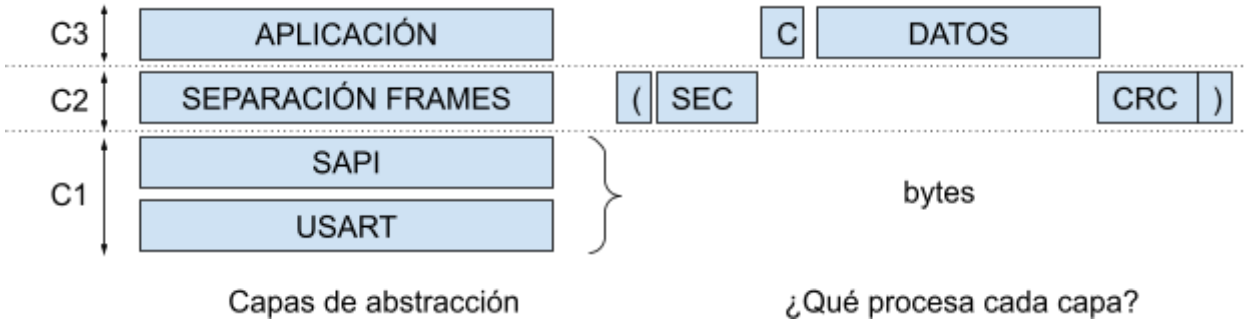
El algoritmo para el cálculo del CRC será el de crc8 (usando 0x00 como semilla. Ver test online o website par cálculo rápido) y se calculará sobre el campo de C, SECUENCIA y el de DATOS. El algoritmo de crc8 arrojará un byte, pero su representación en el paquete será ASCII en hexadecimal, de manera que 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").

Carrera de Especialización en Sistemas Embebidos

Sistemas Operativos en Tiempo Real 2.

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, C2 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.



La aplicación deberá validar el contenido del paquete.

Los paquetes deberán contener solo texto, espacios y guiones bajos '_'. El campo de C identificará la acción a realizar con los DATOS. Si C es 'C' la acción consistirá en transformar DATOS en un texto con estilo camelCase, si C es 'P' la acción consistirá en transformar DATOS en un texto con estilo PascalCase y si C es 'S' la acción consistirá en transformar DATOS en un texto con estilo snake_case. REFERENCIA

Ejemplos:

Entrada	camelCase	PascalCase	snake_case
hola mundo	holaMundo	HolaMundo	hola_mundo
holaMundo	holaMundo	HolaMundo	hola_mundo
hola_mundo	holaMundo	HolaMundo	hola_mundo
HolaMundo	holaMundo	HolaMundo	hola_mundo

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, secuencia y los delimitadores y de enviarla a la SAPI para que lo envíe al periférico.

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

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

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 privado 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	Requerimiento
R_G_1	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	Requerimiento
R_C2_1	El grupo deberá poseer un repositorio privado que los docentes puedan consultar libremente. Las entregas podrán ser utilizando la opción de empaquetar un release, o simplemente teniendo una rama llamada release.
R_C2_2	La cantidad máxima de bytes de cualquier paquete de datos será de 200 caracteres.
R_C2_3	Se deberá procesar paquetes que comienzan con SOM = "(" y finalizan con un EOM= ")"
R_C2_4	Recibir un caracter SOM reiniciará el paquete.
R_C2_5	Deberá procesar en contexto de ISR todos los bytes entrantes.
R_C2_6	Los bytes entrantes, deberán almacenarse en un bloque de memoria dinámica
R_C2_7	Deberá tener control sobre la máxima cantidad de bytes recibidos
R_C2_8	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_9	En caso de no haber memoria, la recepción de datos del driver de la C1 deberá anularse.
R_C2_10	El tiempo para cancelar un frame de datos inconcluso será de T = 4 ms desde el último byte recibido.
R_C2_11	El timeout deberá implementarse con un timer de FreeRTOS
R_C2_12	En caso de que se haya cumplido el timeout, el frame no haya sido cerrado, el paquete deberá descartarse.
R_C2_13	Para cada paquete, se deberá calcular el código de comprobación utilizando el algoritmo CRC8 con semilla = 0.
R_C2_14	En caso de que el código de verificación no sea validado, el paquete deberá descartarse.
R_C2_15	Al recibir un mensaje correcto, se deberá señalizar a la aplicación de su ocurrencia, para ser procesada.
R_C2_16	Cuando la aplicación (C3) desee enviar un mensaje por el canal de comunicación, C2 deberá agregarle el código de comprobación, la secuencia y los delimitadores
R_C2_17	Deberá procesar en contexto de ISR todos los bytes salientes en C2
R_C2_18	Cada byte enviado al canal, deberá ser enviado uno a continuación del otro, y al terminar un frame, esperar 1ms.
R_C2_19	Al finalizar la transmisión, se deberá liberar la memoria dinámica utilizada para la transacción.
R_C2_20	El campo de secuencia que llega en cada paquete estará formado por números ASCII y deberá incluirse en la respuesta asociada al paquete.
R_C2_21	Si el campo de secuencia posee un caracter (invalido no es un número imprimible de 4 caracteres), la trama se ignorará.

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

Capa de aplicación (C3)

Código	Requerimiento
R_C3_1	La cantidad máxima de palabras será de 15.
R_C3_2	La cantidad mínima de palabras será de 1.
R_C3_3	La cantidad máxima de caracteres por palabra será de 10.
R_C3_4	La cantidad mínima de caracteres por palabras será de 1.
R_C3_5	No existen paquetes con datos nulos.
R_C3_6	El campo C deberá ser solamente 'S , 'C' o 'P' y estará asociado a la conversión a snake_case, camelCase y PascalCase, respectivamente.
R_C3_7	El campo DATOS deberá procesar solo paquetes de texto a-z , A-Z, guiones bajo y espacios
R_C3_8	No se permitirán dobles “_” o dobles espacios “ “ entre palabras.
R_C3_9	No se permitirán “_” o espacios “ “ al final de una trama de datos.
R_C3_10	Si hay paquetes con caracteres que no cumplan o cuyo contenido sea inválido, deberán descartarse, y enviar a la capa C2 un mensaje de error con nro de error ERROR_INVALID_DATA
R_C3_11	Si el campo C no es válido, el paquetes deberá descartarse y enviar a la capa C2 un mensaje de error con nro de error ERROR_INVALID_OPCODE
R_C3_12	El texto entrante deberá procesarse según el campo C
R_C3_13	Los mensajes de error que devolverá la aplicación tendrán el formato Enn E de error, nn un número de error codificado en ascii (ver tabla). Notar que luego, C2 agregará la secuencia y el CRC resultando una respuesta del este estilo (SSSSEnnCC)

Opcionales:

Código	Requerimiento
R_O_1	Optimice la compilación del FreeRTOS desactivando en freertosconfig.h los elementos que NO utilice en su implementación.
R_O_2	Optimice la compilación de toda la imagen de su firmware, empleando optimización de nivel 2 en el compilador. ¿Mejoró la performance? Cambie config.mk la opción OPT=g por OPT=2. Ver volatile keyword

Carrera de Especialización en Sistemas Embebidos

Sistemas Operativos en Tiempo Real 2.

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.

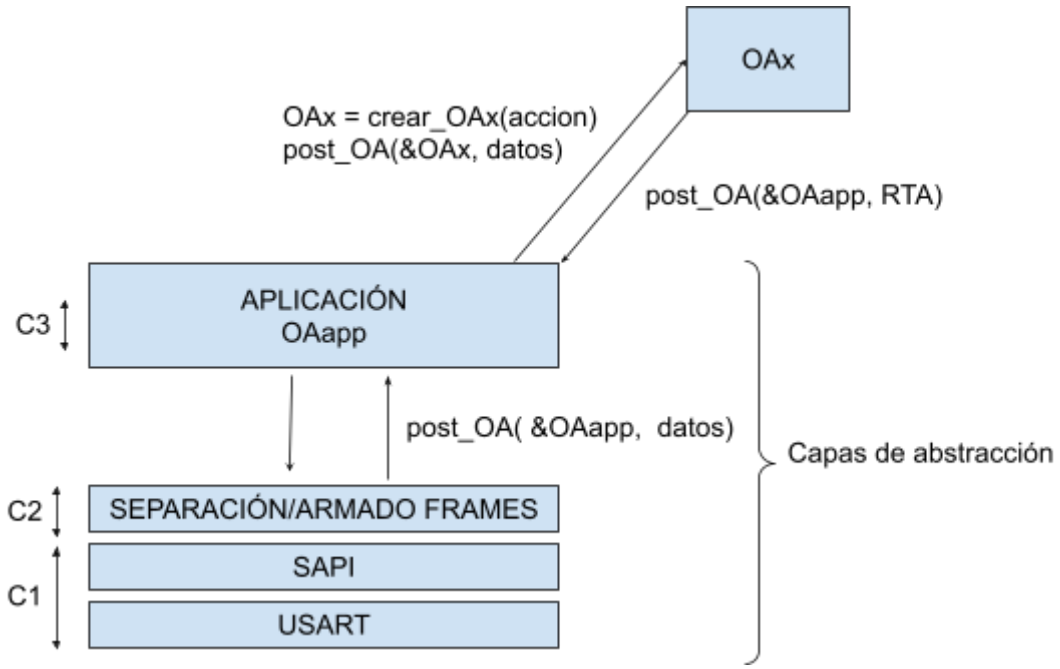
En la parte 1, la capa de aplicación se implementó mediante una tarea “tradicional” del RTOS. Para la parte 2 habrá que transformar la capa de aplicación en un objeto activo.

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

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.



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

Requerimientos nuevos

Capa de aplicación (C3)

Código	Requerimiento
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 tres tipos de OSs de procesamiento: el de snake_case, camelCase y PascalCase (OAs,OAc y OAp)
R_AO_6	En caso de que cualquiera de las OAs,OAc y OAp no existan, deberán instanciarse en tiempo de ejecución
R_AO_7	Al finalizar la operación, cada OAs,OAc y OAp deberá enviar la respuesta a través de un evento a OAapp.
R_AO_8	Al finalizar la operación, cada OAs,OAc y OAp, si no quedan elementos para procesar en la cola de entrada, deberá destruirse.
R_AO_9	Si algun proceso de creación de objetos fallará (falta de memoria por ejemplo) se deberá enviar a C2 un mensaje de error con código ERROR_SYSTEM

Tabla de errores

Nombre	Código
ERROR_INVALID_DATA	0
ERROR_INVALID_OPCODE	1
ERROR_SYSTEM	2

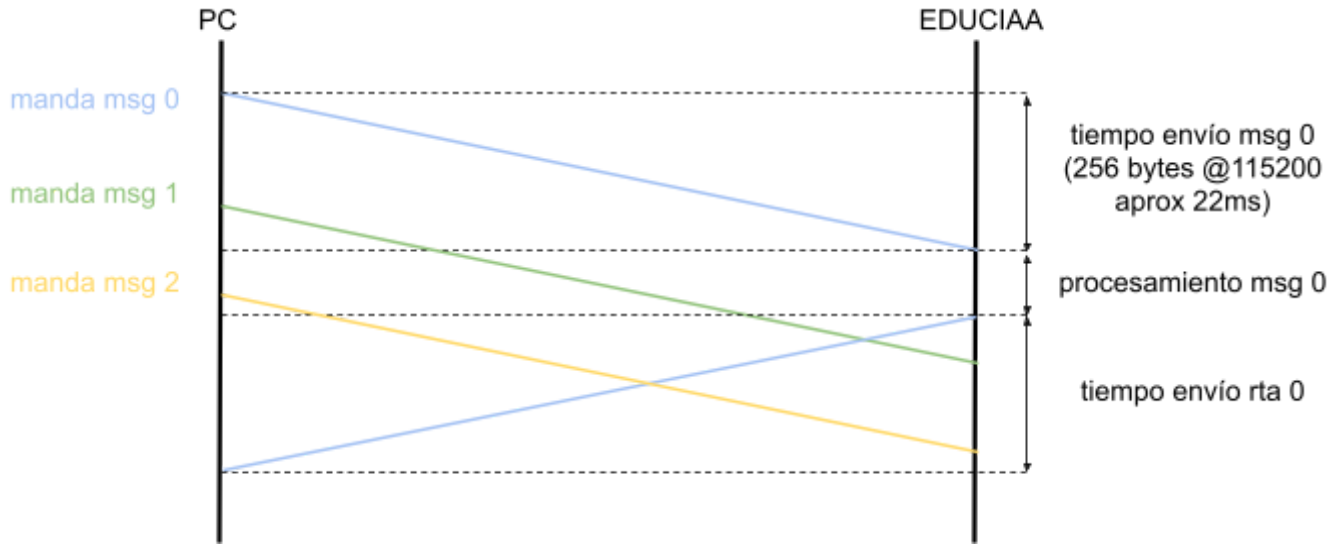
Carrera de Especialización en Sistemas Embebidos
Sistemas Operativos en Tiempo Real 2.

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 en el envío de comandos y recepción de respuestas.



Historial de cambios

Rev	Fecha	Autor	Detalle

Carrera de Especialización en Sistemas Embebidos

Sistemas Operativos en Tiempo Real 2.

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
5	2021/05/18	Franco	Se agregan requerimientos opcionales
6	2021/06/21	Franco	Redefinición del tp
7	2021/05/26	Franco y Martin	Correcciones y más reqs.