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

Autor: Alejandro Celery
Fecha de creación: 30/09/2018

Práctica de programación orientada a eventos:

El objetivo de esta práctica es aprender a trabajar con sistemas reactivos. No se pretende que el alumno escriba un framework para tal fin.
Para tal efecto se proveerá el código básico para empezar a trabajar según este paradigma.

Descripción conceptual de la práctica:

Se propone agregar al programa generado en las prácticas 1 y 2 un despachador de eventos. Los eventos a manejar serán “botón oprimido” y “botón liberado” para los cuatro botones de usuario de la EDU-CIAA y la acción a tomar será encender/apagar el LED que está encima de cada botón

Arquitectura de la aplicación:

Se mantendrá la arquitectura de la práctica (2) y se agregará lo necesario para resolver los requerimientos funcionales en forma reactiva.

- Se agregará una tarea despachadora de eventos con su correspondiente cola.
- Se agregará un servicio de timers para generar los eventos de timeout.
- Se agregará un módulo de manejo de los pulsadores que hará el manejo con antirebote de los botones y publicará los eventos detectados.
- Se agregará un módulo de manejo de los LEDs que responderá a los eventos recibidos en los botones.

Paquetes de datos: **(en negrita los campos diferentes de la práctica 2)**

1B	1B	1B	T BYTES	1B
STX	OP	T	DATOS	ETX

Delimitación de paquete:

STX: Carácter 0x55
ETX: Carácter 0xAA.

Campos:

OP (Operación):
0: Convertir los datos recibidos a mayúsculas.
1: Convertir los datos recibidos a minúsculas.
2: Reportar stack disponible.
3: Reportar heap disponible.
4: Mensajes de estado de la aplicación.
5: Medir performance del sistema.
6: Eventos de botones oprimidos/liberados.

T (Tamaño): El tamaño del texto recibido.
DATOS: Texto a procesar. Deben ser caracteres ASCII legibles.

Requerimientos de software: (adicionales a los de las prácticas 1 y 2):

R1: La aplicación tendrá un servicio de timers.

R2: La aplicación tendrá un módulo que detecte las opresiones y liberaciones de los 4 botones de la EDU-CIAA y publicará los eventos correspondientes.
R2.1: El módulo manejará los eventos “SIG_TIMEOUT”.
R2.2: El módulo consultará el estado de los botones cada vez que reciba “SIG_TIMEOUT”.
R2.3: El módulo manejará los 4 botones con una única MEF (tendrá una estructura asociada al manejo de cada botón).
R2.4: Al detectar un botón pulsado se publicará el evento (“SIG_BOTON_PULSADO”, N), donde N es el número de botón pulsado (de 1 a 4 como están marcados en la EDU-CIAA.
R2.4: Al detectar un botón liberado se publicará el evento (“SIG_BOTON_LIBERADO”, N), donde N es el número de botón liberado (de 1 a 4 como están marcados en la EDU-CIAA.

R3: La aplicación tendrá un módulo que maneje los LEDs de la EDU-CIAA en respuesta a los eventos de botón pulsado/liberado.
R3.1: Al recibir el evento (“SIG_BOTON_PULSADO”, N) se prenderá el LED que está justo encima del botón.
R3.2: Al recibir el evento (“SIG_BOTON_LIBERADO”, N) se apagará el LED que está justo encima del botón.
R3.3: En el caso del LED RGB se encenderá un solo color, sin importar cual.

Requerimientos opcionales:

R4: La aplicación tendrá un módulo que informará el tiempo de pulsación de cada botón.
R4.1: Al recibir el evento (“SIG_BOTON_PULSADO”, N) se tomará el valor del tick del scheduler.
R4.2: Al recibir el evento (“SIG_BOTON_LIBERADO”, N) se enviará un mensaje por UART informando el tiempo de pulsación y el botón que fue pulsado.
R4.2.1: El mensaje de información tendrá código de operación 6.

Sugerencias

- Se puede tomar como punto de partida el ejemplo “ControlCasa” mostrado en clase.
- Implementar primero el timer service.
- Generar luego el módulo de manejo de los pulsadores y verificar que reciba los eventos de timeout.
- Empezar por un solo pulsador/LED y extenderlo luego a los demás.
- Reutilizar el código de la práctica 2 para informar el estado de las pulsaciones por UART.

Historial de cambios