

Curso de Especialización en Sistemas

Embebidos

Programación de microprocesadores

6ta Cohorte

2018

Trabajo practico N 3: Blinky LED

Alumno:

Ing. Rodrigo Alejandro Tirapegui

Docentes:

Ing. Patricio Bos

Ing. Luis Chico

Enunciado:

Re-implementar el ejercicio entregable 2 con las siguientes modificaciones:

1. Incorporar mejores prácticas de programación al código en base a la puesta

en común hecha en clase.

2. Definir 3 tareas en el S.O. de Pont visto en clase:

a. blinkLedTask, con periodicidad de acuerdo al enunciado anterior y 0ms de retardo.

b. debounceTec1Task, con periodicidad 20ms y un retardo de 1ms.

c. debounceTec2Task, con periodicidad 20ms y un retardo de 2ms.

Luego, responda las siguientes preguntas:

1. ¿Se gana algo utilizando esta arquitectura time-triggered? ¿Se puede administrar mejor y más claramente la temporización de tareas?

Si, la principal ganancia consiste en que el control de parpadeo del LED y el control antirebote de las teclas nunca suceden en el mismo instante, haciendo que el sistema sea efectivamente Real-Time.

En definitiva, la arquitectura time-triggered permite administrar mejor y más claramente la temporización de tareas.

2. ¿Siguen siendo necesarios los delay bloqueantes y no bloqueantes, o se puede evitar su uso?

Ahora se puede evitar el uso de los delays tanto bloqueantes como no bloqueantes, aprovechando el intervalo de tiempo entre llamadas a las tareas provisto por el dispacher del S.O. de Pont.

3. Dado que este S.O. no permite pasar parámetros a las funciones que implementan las tareas, qué es mejor:

a. Duplicar el código de la MEF antirebote.

b. Utilizar una función “intermedia” que a su vez llame a la MEF antirebote con el parámetro adecuado.

Es conveniente utilizar una función “intermedia” que a su vez llame a la MEF antirebote con el parámetro adecuado.

4. ¿Números mágicos en el código? ¿Indentación y comentarios adecuados?

En el ejercicio actual, al igual que en el anterior, no existen números mágicos, dado que cada literal requerido por el programa es usado mediante un Tag generado con la directiva “#define”.

La indentación del código se realizó basado en la guía de estilo “[CIAA firmware coding guidelines](https://github.com/epernia/sAPI/blob/develop/documentation/docs/working-methodology.md#code-style)”.