Laboratorio 13:

Multitarea bajo un kernel de planificación no expropiativo

Programación de sistemas y dispositivos

José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Presentación

- Objetivo: Desarrollar una aplicación multitarea bajo un kernel de planificación no expropiativo.

- El punto de partida es una aplicación que realiza de 7 tareas :
  1. Cada 500 ms, alterna el led que se enciende.
  2. Cada 50 ms, lee el keypad y, si hay una tecla pulsada, envía su scancode a otras tareas.
     - Usa un maibox (variable global + flags) como mecanismo de comunicación.
  3. Cada segundo, envía por la UART0 la hora leída del RTC.
  4. Cada 10 s, muestra por la UART0 el número de ticks transcurridos desde el inicio.
     - Lleva localmente la cuenta de ticks.
  5. Envía por la UART0 cada una de las teclas pulsadas.
  6. Muestra en el display 7-segmentos cada una de las teclas pulsadas.
  7. Indica por la UART0 la detección (por interrupción) de la presión de un pulsador.
     - La RTI activa un flag cada vez que detecta presión.

- Todas la tareas son periódicas
  - Las tareas 5 y 6 chequean el mailbox cada 100 ms.
  - Las tarea 7 chequea el flag cada 50 ms

- Adicionalmente, todas las tareas envían un mensaje a la UART0 a su inicio.

Presentación

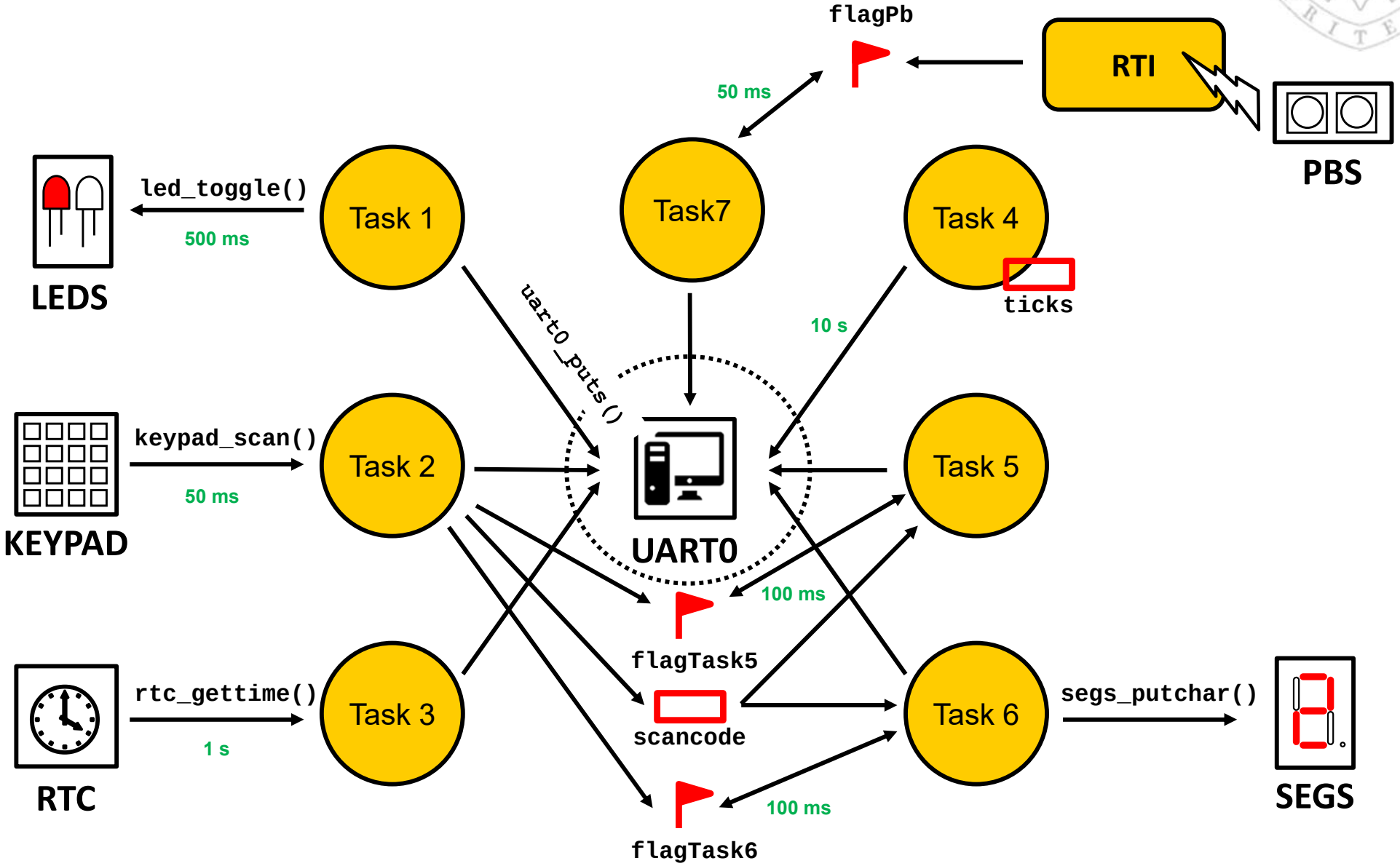
- Las tareas se ejecutan en un kernel de planificación no expropiativo.
  - El planificador es una hebra en foreground (RTI por temporizador) que se encarga de pasar a preparadas aquellas tareas cuyo periodo de ejecución haya vencido.
  - El despachador es una única hebra en background que ejecuta secuencialmente las tareas preparadas por orden de prioridad.
    - Dado que las tareas son ejecutadas sin expropiación por el despachador, los recursos compartidos son accedidos secuencialmente y no es necesario protegerlos.
  - Además, la RTI por presión de pulsador es una hebra en foreground adicional.
    - Cada vez que detecta una presión de cualquier pulsador, activa un flag para que la hebra en background envíe por la UART0 un mensaje sin conflicto con las otras tareas.
- En kernelcoop.c se facilitan todas las funciones del kernel (véase tema 6)
  - create_task(), delete_task(), scheduler_init(), scheduler() y dispacher()

- Esta aplicación deberá ampliarse con 2 tareas adicionales:
  8. Muestra en el LCD cada una de las teclas pulsadas.
  9. Cada segundo, muestra en el LCD los segundos transcurridos desde el inicio.

J.M. Mendias
2020

Aplicación multitarea

arquitectura software



Planificador no expropiativo

declaraciones

```
...

#define MAX_TASKS         (10)
#define TICKS_PER_SEC     (100)  ..... Entre tick y tick transcurren 10 ms

typedef struct
{
  void (* pfunction) (void);
  uint32 state;
  uint32 period;
  uint32 ticks;
  boolean ready;
} task_t;                          } TCB (task control block)

uint32 create_task( void (*pfunction)(void), uint32 period );
void delete_task( uint32 i );

void scheduler_init( void );
void scheduler( void ) __attribute__ ((interrupt ("IRQ")));
void dispacher( void );
```

Aplicación multitarea

declaraciones

```
uint8 scancode;
boolean flagTask5;
boolean flagTask6;

volatile boolean flagPb;

void Task1( void );
void Task2( void );
...
void Task7( void );

void isr_pb( void ) __attribute__ ((interrupt ("IRQ")));
```

Declara los recursos de sincronización y comunicación

Declara tareas

RTI por presión de pulsador

Aplicación multitarea

programa principal

```
void main( void )
{
  sys_init();
  timers_init();
  ...
  keypad_init();

  uart0_puts("\n\n Ejecutando kernel de planificación no expropiativo\n");
  uart0_puts("-----------------------------------\n\n");

  flagTask5 = FALSE;

  ...

  scheduler_init();

  create_task( Task2, 5 );
  create_task( Task5, 10 );
  ...
  create_task( Task4, 10000 );

  timer0_open_tick( scheduler, TICKS_PER_SEC );
  pbs_open( isr_pb );

  ...
```

Inicializa dispositivos

Inicializa flags

Inicializa el Kernel

Crea tareas (prioridad por orden de creación)

Se crean primero (mayor prioridad) las tareas más frecuentes

Durante la creación se ejecuta por primera vez las funciones para inicializarlas

Instala planificador

Instala RTI

Aplicación multitarea

programa principal / RTI / tareas

```
  ...

  while( 1 )
  {
    sleep(); ····· Entra en estado IDLE, sale por interrupción
    dispacher(); ····· Todas las tareas preparadas se ejecutan por orden de prioridad
  }

}
```

```
void isr_pb( void )
{
  flagPb = TRUE; ····· Activa flag cada vez que se pulsa cualquier pulsador
  EXTINTPND = BIT_RIGHTPB | BIT_LEFTPB;
  I_ISPC = BIT_PB;
}
```

- Todas las tareas son idénticas a las de la arquitectura cyclic executive del laboratorio 12.

Acerca de *Creative Commons*

- Licencia CC (Creative Commons)

  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/