# SISTEMAS OPERATIVOS AVANZADOS
# AÑO 2007
# TRABAJO PRACTICO N°4

## 1. Titulo: Evaluación de Puntualidad de un Sistema de Tiempo Real

**Descripción:**

El objetivo es evaluar la puntualidad de ejecución de tareas periódicas de un sistemas operativo de tiempo real en desarrollo denominado MINIX4RT contra un sistema operativo de tiempo real RTLinux.

**Formato de entrega**

Deberá entregar:

1. Documento IMPRESO y en carpeta en formato A4 conteniendo las siguientes secciones:
   - 1.1. Carátula: Debe contener como MINIMO:
     - 1.1.1. Materia
     - 1.1.2. Numero y Título de trabajo práctico
     - 1.1.3. Apellido y Nombres de los integrantes
     - 1.1.4. Direcciones de correo electrónico (de todos los que tengan una cuenta habilitada)
     - 1.1.5. Año de cursado
   - 1.2. Cuerpo: Con la resolución de los Ítems del TP. Cuando el TP implique la modificación o el desarrollo de nuevo código, debe incluirse dicho código. Asumir que el documento se utilizará para un eventual debug. Incluir todas las aclaraciones que se crean necesarias para ayudar en la corrección
   - 1.3. Descargos: Indicar aquí las razones por las que no se pudo concluir o realizar alguno de los Items del TP. Si existiesen dudas acerca de la forma en que se resolvió algún ítem, mencionarlo también en esta sección.
2. Diskette:
   - 2.1. Formato: ext2
   - 2.2. Contenido:
     - 2.2.1. src : Conteniendo todo el código fuente desarrollado y/o modificado.
     - 2.2.2. obj : Conteniendo todos los ejecutables.
     - 2.2.3. doc: Conteniendo toda la documentación. Incluya aquí una copia del documento del que se habla en el punto 1.
     - 2.2.4. src/Makefile: En este archivo de reglas para *make*, incluya los comandos necesarios para que quien corrija el documento pueda correr los comandos:
       - 2.2.4.1. *make*: Esto debe compilar todos los programas necesarios y dejar los objetos en el directorio *obj*.
       - 2.2.4.2. *make clean*: Esto debe eliminar todos los programas objetos correspondientes del directorio *obj*
       - 2.2.4.3. make copy: Esto debe copiar TODO el diskette al directorio /home/soa/2007/tp4/NOMBREGRUPO, donde NOMBREGRUPO es el nombre asignado al grupo. El árbol de directorios debe ser creado si no existiese.
3. **Importante:** Las entregas de TPs son “en persona”. No se aceptan carpetas dejadas en el Departamento Sistemas o en cualquier otro lugar

**Implementación:**

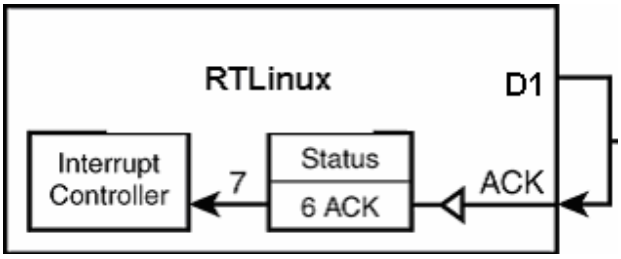
El sistema MINIX4RT generará una señal pulso en uno de los PINs del puerto paralelo en forma periódica. El RTLinux recibirá esa señal y tomará nota de la hora en que ha recibido la misma.
Se desea medir las variaciones en la precisión del período de los pulsos generados por el MINIX4RT los cuales se ven afectados por el diseño del propio sistema operativo y por la ejecución concurrente de otros procesos.
Dado que no se cuenta con elementos de precisión electrónica, se utilizara al RTLinux como sistema Monitor, pero para ello deberá medirse el error del propio sistema Monitor.

El trabajo práctico consta de 3 partes:

1. El desarrollo del programa Monitor en el sistema RTLinux que el alumno o grupo pueden desarrollar en su casa y realizar la prueba de *Medición de Latencia de interrupciones del sistema Monitor*.
2. Una vez realizada la parte anterior, coordinará con el Jefe de Trabajos Prácticos, fecha y hora para desarrollar las mediciones sobre el sistema MINIX4RT, para lo cual solo deberá llevar en diskette (múltiples copias) los programas desarrollados para instalar en un sistema RTLINUX del laboratorio de conectividad, el cual dispondrá también del sistema MINIX4RT y los cables necesarios para realizar las mediciones.
3. El grupo/alumno documentarán los tests realizados y completarán el presente TP.

### 1.1. Medición de latencia de interrupciones del sistema Monitor

Primeramente se debe medir la latencia de interrupciones de RTLinux. Para ello se debe utilizar un conector DB25 macho, donde se conectará el pin D1 con el pin ACK y se enviarán datos por el **puerto paralelo** a distintas frecuencias. El escenario es el siguiente:




Todas las mediciones se realizarán sobre 500 muestras.
A fin de que el sistema RTLinux no se vea afectado por la interacción con dispositivos de hardware, los valores de los Timestamps de las muestras serán almacenados en memoria, y al finalizar la prueba, serán transferidos a un archivo para luego ser procesados.

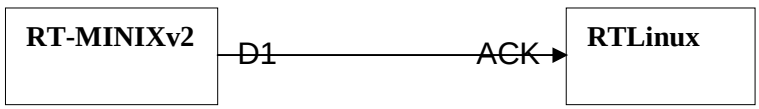
Se deberá crear una tabla como la siguiente donde se registrarán los períodos medidos.

| | FRECUENCIAS | | |
|---|---|---|---|
| | 10000 | 5000 | 1000 |
| MEDIA | | | |
| STD | | | |
| MIN | | | |
| MAX | | | |
| DEADLINES | | | |

### 1.2. Medición de puntualidad de MINIX4RT

Se conectará el pin D1 de la PC MINIX4RT con el pin ACK de la PC RTLinux.
Las mediciones se realizarán de la siguiente manera:

- ✓ La PC MINIX4RT emite pulsos en tiempo real a una frecuencia dada según que se muestra a continuación.
- ✓ La PC RTLinux al recibir el pulso registra el tiempo y contabiliza el mismo hasta la llegada del próximo pulso.




Generár gráficas con los resultados para las distintas frecuencias.
Se realizarán dos conjuntos de pruebas:

- ✓ Prueba en VACIO: no se ejecutarán procesos en la PC MINIX4RT
- ✓ Prueba con carga de CPU: se ejecutará el script que se muestra mas adelante para cargar la CPU de la PC MINIX4RT durante las mediciones.

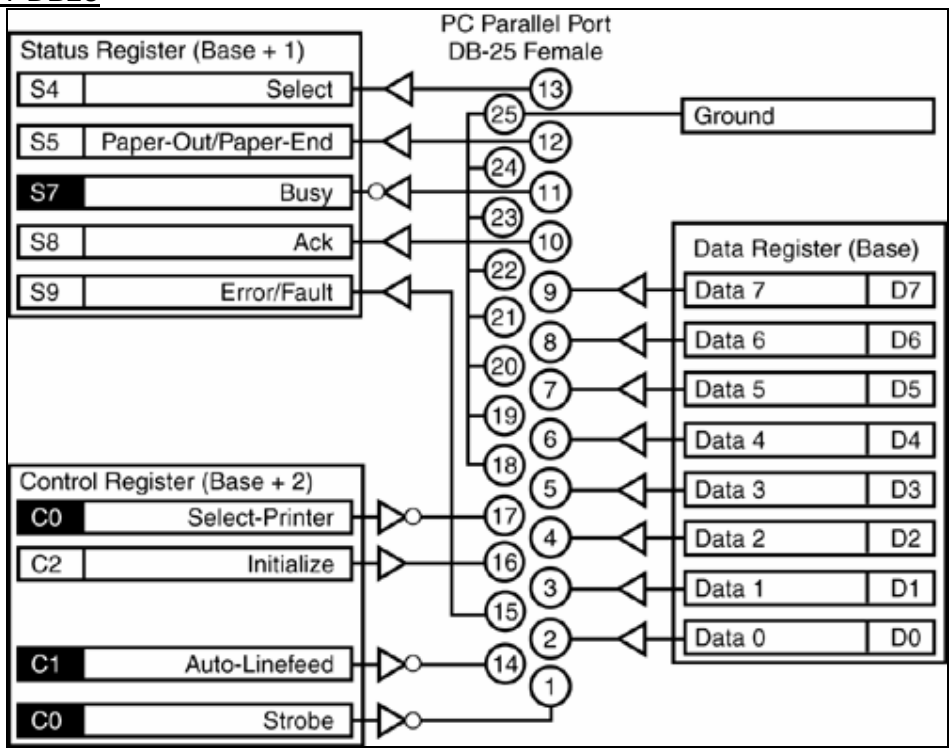
Se deberá crear una tabla como la siguiente donde se registrarán los períodos medidos.

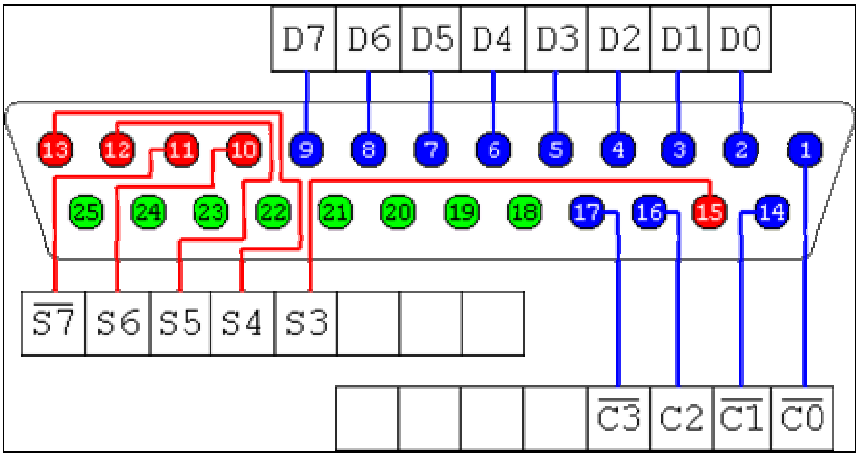
| | FRECUENCIAS EN VACIO | | | FRECUENCIAS CON CARGA CPU | | |
|---|---|---|---|---|---|---|
| | 10000 | 5000 | 1000 | 10000 | 5000 | 1000 |
| MEDIA | | | | | | |
| STD | | | | | | |
| MIN | | | | | | |
| MAX | | | | | | |
| DEADLINES | | | | | | |

*Cable LapLink para conectar las dos PCs:*

**1 no usado**
**2 <-> 15**
**3 <-> 13**
**4 <-> 12**
**5 <-> 10**
**6 <-> 11**
**7 no usado**
**8 no usado**
**9 no usado**
**10 <-> 5**
**11 <-> 6**
**12 <-> 4**
**13 <-> 3**
**14 no usado**
**15 <-> 2**
**16 no usado**
**17 <-> 19**
**18 <-> 18**
**19 <-> 17**
**20 no usado**
**21 <-> 21**
**22 <-> 22**
**23 <-> 23**
**24 no usado**
**25 <-> 25**

*PIN OUT DB25*


PC Parallel Port
DB-25 Female
Status Register (Base + 1)
S4 Select
S5 Paper-Out/Paper-End
S7 Busy
S8 Ack
S9 Error/Fault
Control Register (Base + 2)
C0 Select-Printer
C2 Initialize
C1 Auto-Linefeed
C0 Strobe
Data Register (Base)
Data 7 D7
Data 6 D6
Data 5 D5
Data 4 D4
Data 3 D3
Data 2 D2
Data 1 D1
Data 0 D0
Ground
13 25 12 24 11 23 10 22 9 21 8 20 7 19 6 18 5 17 4 16 3 15 2 14 1


D7 D6 D5 D4 D3 D2 D1 D0
13 12 11 10 9 8 7 6 5 4 3 2 1
25 24 23 22 21 20 19 18 17 16 15 14
S7 S6 S5 S4 S3
C3 C2 C1 C0

**Programa para Carga de CPU (en MINIX4RT)**

```
#!/bin/sh
while [ 1 ]
  do
   yes > /dev/null
done
```

**Deberán presentarse:**

- ✓ Características del Hardware utilizado para la prueba: CPU, MHz, RAM, etc
- ✓ Versiones del Software Utilizado
- ✓ Listado de los programas utilizados para los 2 sistemas operativos
- ✓ Resultados completos de las mediciones
- ✓ Tablas de resumen de los resultados
- ✓ Gráficos de los resultados

**Sistemas Operativos:** RT-Linux, MINIX4RT

http://www.ece.osu.edu/~cglee/ECE694Z/rtlinux/rtlinuxInstallation2005Sp.pdf

**Lenguaje:** C

**Bibliografia:**


- CD-ROM de la Cátedra
  - Oreilly Pc Hardware In A Nutshell 3rd Edition Ebook-Lib.chm
  - Embedded Linux - Hardware,Software,and Interfacing.chm
- http://www.electro.fisica.unlp.edu.ar/rt/index.html
- http://www.rtlnx.com/
- http://www.electro.fisica.unlp.edu.ar/rt/index.html
- http://www.lvr.com/parport.htm
- http://www.torque.net/linux-pp.html
- http://et.nmsu.edu/~etti/fall96/computer/printer/printer.html
- http://www.galeon.com/jcgr/index.html
- http://www.mnis.fr/opensource/ocera/rtos/c1450.html
- ftp://jano.unicauca.edu.co/cursos/SistTReal/docs/rtlinux-tutorial.pdf
- http://bernia.disca.upv.es/~iripoll/rt-linux/fancy-examples/examples/scripts/html/index.html
- http://packages.debian.org/testing/devel/rtlinux.html

# 1.3. ANEXOS

**Ejemplo de código MONITOR RTLinux**

```
#include <rtl.h>
#include <time.h>
#include <pthread.h>
#include <linux/module.h>
#include <rtl_sched.h>
#include <asm/io.h>
#include <rtl_core.h>

#define LPT_PORT 0x0378
MODULE_AUTHOR("Sistemas Operativos Avanzados");
MODULE_DESCRIPTION("Monitor");
MODULE_LICENSE("GPL");
static int frec = 1;   // frecuencia en Hz.
MODULE_PARM(frec,"i");   // especifica que frec es un parametro entero.

// irq del puerto paralelo (0x378/9/A)
#define IRQ 7
#define MAX 500

pthread_t thread;
hrtime_t tiempo_inicio, tiempo_fin, delta;
int i,a;
int error;
int j=MAX;
int mat[MAX];

char in_parallel_port(void)
{
      int n, res;

      n = inb(LPT_PORT+1);
      if (n & 0x80)           // si el bit 5 == 1...
          n = n & 0x7F; // lo ponemos a 0,
      else               // sino
          n = n | 0x80; // lo ponemos a 1 (logica invertida)
      res = ((n & 0x38)>>3)+((n & 0x80)>>4);   // arma el numero leido
del
                                 // puerto con los bits 4,5,6 y 8.
      if (res == 0xF)
          return 'Q';        // detectado fin de secuencia.

      return res + '0';
}


inline unsigned int intr_handler(unsigned int irq_num, struct pt_regs
*regs)
{
      tiempo_fin = gethrtime();
      delta = (tiempo_fin - tiempo_inicio) ;
```

```
      mat[j] = delta;
      rtl_hard_enable_irq(IRQ);         // habilita interrup. irq 7
        return 0;
}

void * start_routine(void *arg)
{
      struct sched_param p;
      long periodo = 1000000000/frec;

      p . sched_priority = 1;
      pthread_setschedparam (pthread_self(), SCHED_FIFO, &p);
      pthread_make_periodic_np (pthread_self(), gethrtime(),  periodo);

      for (j=MAX;j > 0;j--) {
            tiempo_inicio = gethrtime();
          outb(0xFF,LPT_PORT);
          rtl_delay(1000);
          outb(0x00,LPT_PORT);
          pthread_wait_np ();
      }
      rtl_printf("Listo\n");
      return 0;
}

int init_module(void)
{
      error = rtl_request_irq(IRQ, intr_handler); // instala el handler
      if(error < 0)                             // del irq 7.
      {
          rtl_printf("Error en rtl_request_irq (%d)\n", error);
          return -1;
      }

      outb_p(inb_p(LPT_PORT + 2) | 0x10, LPT_PORT + 2); // habilita
interrup. del puerto.
      rtl_hard_enable_irq(IRQ);         // habilita interrup. irq 7
      rtl_printf("Modulo monitor inicializado exitosamente\n");
      rtl_printf("port 0x%x irq %d\n *** frecuencia
%d\n",LPT_PORT,IRQ,frec);

      for (a = MAX; a >= 0; a--) {
        mat[a] = -1;
      }
      j=MAX;
      return pthread_create (&thread, NULL, start_routine, 0);
}

void cleanup_module(void)
{
      error = rtl_free_irq(IRQ);   // libera el manejador de interrup.
      if(error < 0) {
          rtl_printf("Error en rtl_hard_free_irq (%d)\n", error);
          return;
      }
        error = pthread_delete_np (thread);
      if(error != 0)
```

```
      {
          printk("Error en pthread_delete_np (%d)\n", error);
          return;
      }

      for (a = 1; a < MAX ; a++) {
        rtl_printf("Delta[%d]: %ld\n",a,mat[a]);
        rtl_delay(1000000);
      }

      rtl_printf("Modulo monitor finalizado exitosamente\n");
}
```