energética Vol. XXVIII, No. 2 / 2007

Aplicación de RT-Linux en el control de motores de pasos. Parte I

Ernesto Duany
Javier Muñoz
Mario Morera

Recibido: Julio del 2007
Aprobado: Septiembre del 2007

Resumen / *Abstract*

La idea fundamental de este artículo es mostrar cómo controlar un motor de paso empleando el puerto paralelo de una computadora, y demostrar la eficiencia temporal de las aplicaciones que se desarrollan en sistemas preparados para ejecutar tareas de precisión; aprovechar al máximo la capacidad de tiempo real estricto que brinda RT-Linux para el control de accionamientos eléctricos. Se desarrolla un pequeño software en lenguaje C que envía las señales al puerto paralelo en el tiempo necesario. Este software no está diseñado con fines comerciales, solo permite realizar pruebas sobre el circuito de control diseñado para este propósito.
Palabras clave: Motor de pasos, RT-Linux, tiempo real

The fundamental idea of this article is to control a steps motor using the parallel port of a computer and demonstrate the temporary efficiency of the applications that are executed in prepared systems to execute tasks of real time. To take advantage of to the maximum capacity of strict real time that RT-Linux offers us for the control of electric workings. A small software is developed in language C that sent the signs to the parallel port in the necessary time. This software is not designed with commercial, only allows to carry out tests on the control circuit designed for this purpose.
Key words: Steps motor, RT-Linux, real time

INTRODUCCIÓN

En este artículo se realiza un control de un motor de pasos por medio de una microcomputadora, empleando el sistema operativo Linux como base, y su *kernel* de tiempo real RT-Linux. En la actualidad se hace necesario tener un control más estricto del tiempo en que se ejecutan las tareas y aun más cuando se trata de un sistema de control.

El trabajo trata de manera general de un sistema de tiempo real enviando las órdenes correctas al circuito de control del motor de pasos, de forma que este realice cada paso en el tiempo estimado por el controlador. Para esto fue necesario diseñar un hardware básico sin introducir retrasos en el sistema de control.

En esta primera parte del artículo se trabaja con una tarea de control en tiempo real que se encarga de enviar las órdenes adecuadas para que el motor se mueva según los parámetros deseados. Para una mejor comprensión se describirá el hardware construido para este fin.

CIRCUITO DE CONTROL

El circuito para controlar el motor de pasos se muestra en la figura 1 y consta de cuatro etapas. La primera está compuesta por el sistema de optoacopladores, los que permiten aislar a la computadora empleada del resto del esquema. De esta manera quedan separadas las tensiones y corrientes manejadas en la etapa de potencia el hardware externo en general, y se protege al ordenador. Seguidamente las señales

son enviadas al circuito integrado L297 que es el encargado de generar la secuencia necesaria por sus terminales de salida para que el motor opere correctamente.

El motor seleccionado para realizar el estudio es un motor bipolar de dos fases, alimentado con una tensión de 4,2 V, con un consumo de corriente por fase máximo de 0,6 A y un ángulo de paso de 1,8°. Para esto fue necesario el integrado L298 H que es una configuración de dos puentes H con su correspondiente censado de corriente. En la parte superior de la figura se tiene el sistema de alimentación necesaria para gobernar el circuito completamente.

Las señales que llegan al circuito de control provenientes del puerto paralelo de la computadora son las siguientes:
- Habilitación.
- Señal de movimiento.
- Sentido de giro.
- Modo de trabajo.

La señal de movimiento es un tren de pulso proveniente del terminal 3 del puerto paralelo, este pulso dependerá del tiempo *on/off* seleccionado por el operador. En el terminal 2 se habilita o no el circuito de control. El sentido de giro es seleccionado mediante una señal en el terminal 4 aunque este dependerá en gran medida del conexionado de los enrollados del motor. El motor seleccionado, conjuntamente con el circuito de control está preparado para trabajar en los modos, *half* o *full* en dependencia de la señal aplicada 1 o 0 respectivamente.[1]

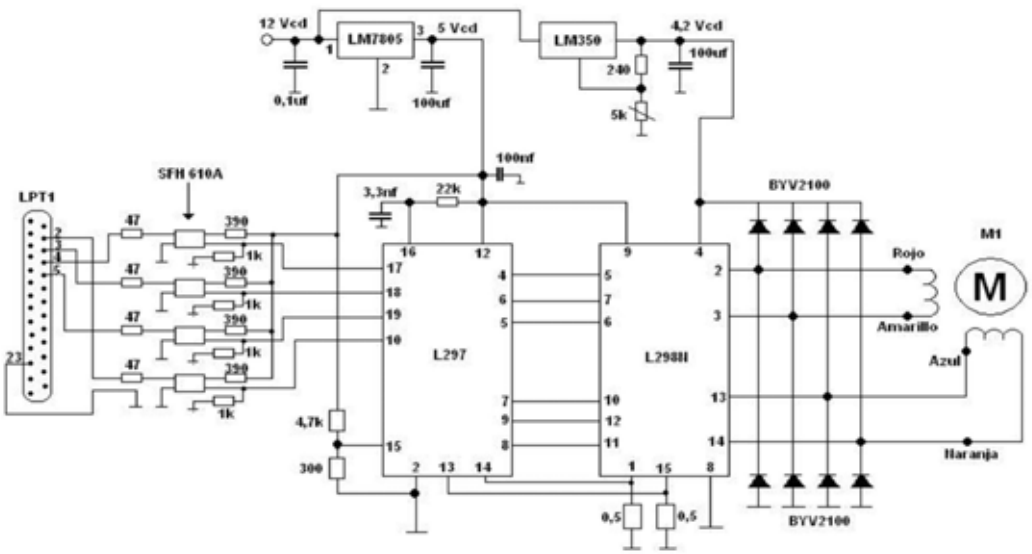
SOFTWARE

El software empleado en el control del motor de pasos es el objetivo fundamental del estudio realizado. Para demostrar la utilidad práctica que representa para la ingeniería eléctrica en los sistemas de control, se emplea una computadora de pocos recursos y sistema operativo Linux como plataforma de trabajo. El sistema empleado es un SuSE Linux 9.0 con *Kernel* 2.4.29, parcheado para utilizar el microkernel de RT-Linux versión 3.1.

La figura 2 muestra un diagrama general del principio de funcionamiento de Linux una vez que se le ha agregado *el microkernel* de RT-Linux.

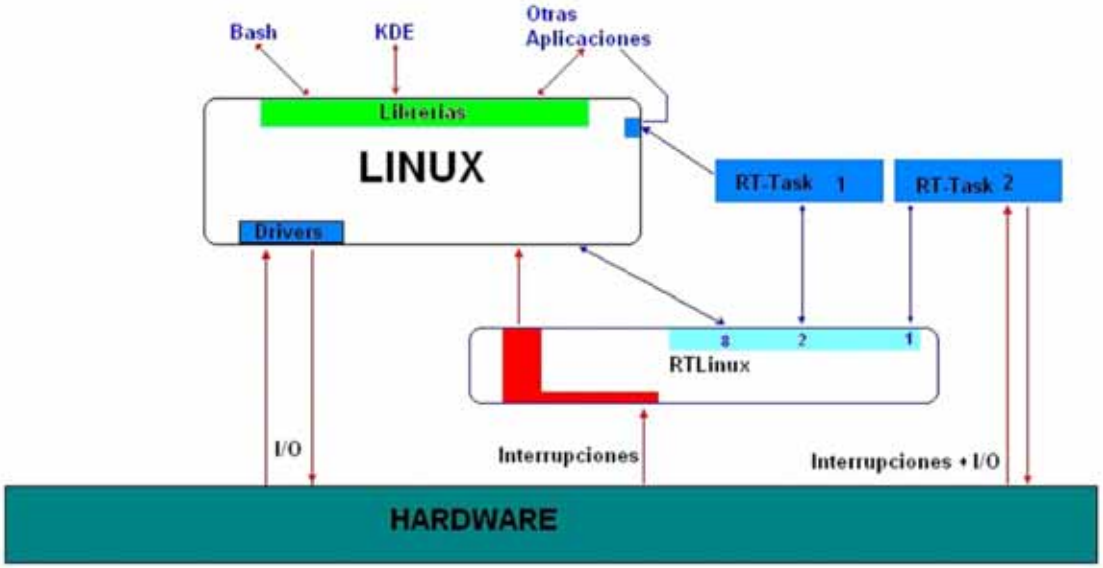
El sistema representado constituye en su conjunto un sistema de tiempo real. En este caso el *kernel* interactúa con el *microkernel* de RT-Linux, pero no directamente con las interrupciones. Las tareas de tiempo real tendrán acceso directo a los periféricos de entrada y salida y a las interrupciones del hardware si se deseara, no siendo así en el caso del sistema Linux. El sistema no pierde el control de los periféricos, solo que la prioridad para acceder a ellos cambia totalmente. En este caso las interrupciones son enviadas al núcleo de RT-Linux y este será quien las administre directamente de forma adecuada.

Las características más significativas de RT-Linux para el control de aplicaciones son: Constituye un sistema de tiempo real estricto y acceso directo al hardware (puertos e interrupciones).[2]



Esquema del circuito de control del motor de pasos.

1



Esquema del sistema operativo Linux con la posibilidad de ejecutar tareas en tiempo real .

2

El programa que se encarga de controlar el motor en tiempo real, esta compuesto por tres ficheros, dos programas y un fichero de cabecera que contiene la estructura de todos los datos que circulan a través de los FIFOs. El primer programa, que se compila como módulo se llama motor.o y contiene los procedimientos para crear la tarea de RT que es la encargada de mover el motor. Es de carácter obligatorio crear las tareas de RT y dispositivos de RT desde módulos.

Una vez que la tarea es creada, se escribe en la tabla de prioridades de RT-Linux el intervalo que va a ejecutar dicha tarea. Básicamente lo que hace la tarea es leer un valor de habilitación y tiempo del FIFO / dev/rtf0 y basado en esto escribe en el puerto su correspondiente número, lo que se traduce en general en un pulso en cada uno de los terminales del puerto paralelo.[3]

La aplicación que no se ejecuta en tiempo real, cuyo nombre es control, es la encargada de preguntarle al usuario los tiempos y el valor que desea en el terminal de habilitación y son enviados a la tarea empleando el FIFO de comunicación rtf0. Como se puede observar el flujo en este FIFO es de la aplicación que no corre en tiempo real hacia la aplicación que sí se ejecuta en tiempo real.

Hay que señalar que a la tarea de tiempo real se le agregó dos esperas no activas, para permitir ejecutar otras aplicaciones en esos intervalos de inactividad. También se puede notar cómo es posible emplear comandos y funciones que pertenecen al lenguaje de Shell de Linux y a los estándares POSIX que han sido implementados en lenguaje C, sin que esto afecte el trabajo en tiempo real. Otro detalle importante es el trabajo con los relojes de tiempo real, en este caso se emplea CLOCK_MONOTONIC que presenta la misma resolución que el CLOCK_REALTIME y otros más de los que posee el sistema.[4]

A continuación se explican algunos segmentos de código del modulo motor.o. Para mejor comprensión se le ha agregado algunos comentarios al código.

```
motor.o
#include <rtl.h>
.
.                    //Declaración de Cabeceras Necesarias
.
#include "msg.h"        //Cabecera Privada. Control de
Datos
#define SIZE 8192
#define fifo_nr 0
#define LPT_Addr 0x378  //Dirección Base del Puerto
Paralelo LPT1
#define periodo 10
RT_TASK task ;
void Real_Time_Speed_Task (void);

// Procedimiento para la Tarea de Tiempo Real
void Real_Time_Speed_Task (){
```

```
int tiempo, enable, resto;
struct msg_control datos;
struct timespec inicio,fin;
while (1) {
rtf_get(fifo_nr,&datos, sizeof(datos));
enable=datos.enable; //Este valor depende el Sentido,
Modo y Enable
tiempo=datos.speed; // período en usegundos
clock_gettime(CLOCK_MONOTONIC,&inicio);
outb(2+enable, LPT_Addr);
usleep(tiempo);
outb (enable, LPT_Addr);
usleep(tiempo);
clock_gettime(CLOCK_MONOTONIC,&fin);
rt_task_wait();
}
}
// Procedimiento para Cargar el Módulo
int init_module (void) {
RTIME now = rt_get_time ();
rtf_create (fifo_nr, SIZE);
rt_task_init(&task, Real_Time_Speed_Task, 1, 3000,
4);
rt_task_make_periodic (&task, now+3000,periodo);
rtl_printf ("Módulo Montado");
return 0;
}
// Procedimiento para liberar el Módulo
void cleanup_module (){
rt_task_delete (&task);
rtf_destroy (fifo_nr);
rtf_destroy (fifo_nr2);
rtl_printf ("Módulo Desmontado");
}
```

La cabecera msg.h contiene el código siguiente:

```
msg.h
struct msg_control{
        int enable;
        int speed;
};
```

La Aplicación denominada control presenta el siguiente código:

```
control
#include <stdio.h>
#include <sys/io.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include "msg.h"

//Procedimiento de Velocidad del Motor
int main (){
struct msg_control datos1;
int num, estado;
int fd;
fd= open ("/dev/rtf0", O_WRONLY);
while (1){
printf ("Entre el estado de enable(0 - 1 - 4 - 8):");
scanf ("%d",&estado);
printf ("Entre el tiempo deseado en 75 - 500  us:");
scanf ("%d",&num);
datos1.speed =  num*10;//tiempo;
datos1.enable = estado;
write(fd,&datos1,sizeof(datos1));
};
```

La figura 3 muestra un esquema en bloque del principio de funcionamiento del módulo que se ejecuta en tiempo real.

Diagrama en bloque del módulo de tiempo real.

3

RESULTADOS OBTENIDOS

Se realiza una comparación de las señales obtenidas mediante equipos de gran precisión con aplicaciones de tiempo real. Para ello se emplean voltímetros, un osciloscopio marca Tektronix y puntas de pruebas para detectar estados lógicos. Aunque estas últimas fueron construidas durante el proyecto para comprobar las secuencias de salida del integrado L297 empleado en el circuito de control.

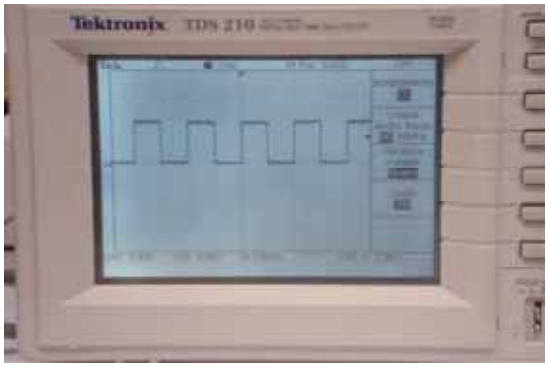
Los programas que se desarrollan en este proyecto fueron compilados, ejecutados y probados en una computadora marca Intel Pentiun II con una velocidad del microprocesador de 350 MHz y una memoria Ram de 256 MB. Estas características son importantes ya que influyen en los resultados obtenidos, la velocidad

y la marca desempeñan un papel muy importante con respecto a las resoluciones de tiempo que se pueden obtener.

Con el objetivo de medir la eficiencia del software de control, se conectó al puerto paralelo de la máquina, exactamente en el pin 3, la entrada del osciloscopio para observar el tren de pulso generado por la tarea en tiempo real, y al mismo tiempo se imprimía por pantalla mediante la programación, los intervalos de tiempo adquiridos desde el reloj de tiempo real CLOCK_MONOTONIC.

La figura 4 muestra la forma de onda de la señal obtenida a la salida del terminal 3 del puerto paralelo ejecutando la tarea de tiempo real mediante el módulo **motor.o.** Por medio de la aplicación **contro**l se le pasan los parámetros a la tarea de tiempo real y los cambios se observan de manera inmediata. En el momento en que se leen del FIFO todos los datos de sentido, modo y enable, este es restaurado en espera de nuevos datos. Para este caso el motor se encontraba trabajando en modo *full* y el período de ejecución de la tarea era de 2 m. Esto se logra escribiendo un uno en el software de control y seguidamente el valor de los estados *on/off* del tren de pulso.

Teniendo en cuenta los datos conocidos del motor empleado en el estudio, con paso de 1,8° se necesitan un total de 200 pasos para lograr una revolución. Al aplicarse pulsos con diferentes frecuencias, se pudo observar que el mismo comenzaba a perder paso y entrar en vibraciones mecánicas para tiempos inferiores a 750 µs, este sería el tiempo *on*/*off* mínimo que se le pudiera aplicar a los motores de esta categoría, este dato está en dependencia del tipo de motor. La velocidad estimada para este caso serian 3,3 r/s, aunque en este motor su máxima potencialidad esta en el posicionamiento y no en la velocidad.



Tren de pulsos obtenido en el terminal 3 del puerto paralelo.

4

CONCLUSIONES

La elección del Linux junto con la extensión de tiempo real RT-Linux ha resultado para este trabajo una gran ventaja, ya que se ha utilizado herramientas de desarrollo muy potentes, tales como el compilador de c GNU, y todo esto sin ningún costo.

Este sistema ha, demostrado que aprovecha los recursos de la máquina al máximo, para la ejecución de las tareas de tiempo real. Como se puede apreciar, todo lo necesario para realizar este trabajo es de dominio público lo cual facilita la programación de las aplicaciones.

La utilización de RT-Linux como sistema de control en tiempo real estricto permite asignar prioridades de ejecución en las tareas de tiempo real, respetar temporizaciones y controlar ampliamente el hardware de una computadora personal que es una herramienta muy poderosa para el cálculo y el control de procesos de desarrollo rápido.

REFERENCIAS

1. *DataSheet del Circuito Integrado L297 y L298H.*

2. Barabanov, Michael and Victor Yodaiken: *Getting Started with RTLinux*, April, 2001.

3. Duany Renté, Ernesto: "Control de motores de paso en Linux, Técnicas de tiempo real", Tesis de grado, Fachhochschule Giessen- Friedberg, Friedberg (Hessen), Alemania, junio, 2006.

4. Ripoll Ripoll, José Ismael: *Real-Time Linux*, Departamento de Informática y Sistemas Computadores (DISCA), España, 1998.

Sitios Web:
- http://www.rtlinux.com
- http://www.fsmlabs.com
- http://www.rtlinux.org

AUTORES

Ernesto Duany Renté
Ingeniero Electricista, Departamento de Ingeniería Eléctrica, Centro de Investigaciones y Pruebas Electroenergéticas (CIPEL), Instituto Superior Politécnico Jose Antonio Echeverría, Cujae, Ciudad de La Habana, Cuba
e-mail:duany@electrica.cujae.edu.cu
eduanyr@yahoo.es

Mario Morera Hernández
Ingeniero Electricista, Doctor en Ciencias Técnicas, Profesor Titular, CIPEL, Instituto Superior Politécnico José Antonio Echeverría, Cujae, Ciudad de La Habana, Cuba
e-mail:marmor@electrica.cujae.edu.cu

Javier Muñoz Álvarez
Ingeniero Electricista, Departamento de Ingeniería Eléctrica, CIPEL, Instituto Superior Politécnico José Antonio Echeverría, Cujae, Ciudad de La Habana, Cuba
javierm@electrica.cujae.edu.cu