APLICACIÓN DE LA COMPUTACIÓN

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

Ernesto Duany
Javier Muñoz
Mario Morera

Recibido: Julio del 2007
Aprobado: Septiembre del 2007

Resumen / ***Abstract***

Este trabajo complementa al presentado anteriormente: "Aplicación de RT-Linux en el control de motores de pasos. Primera parte", de manera que se puedan relacionar a las tareas de adquisición y control para la obtención de un sistema lo más exacto posible. Las técnicas empleadas son las de tiempo real aprovechando las posibilidades del microkernel RT-Linux y los software libres contenidos en sistemas Unix/Linux. Las señales se obtienen mediante un conversor AD y mostradas en pantalla empleando el Gnuplot.
Palabras clave: Gnuplot, RT-Linux, conversor AD, tiempo real

The work presented in this paper is a complement to the control and acquisition tasks which were explained in "Application of RT-Linux in the Control of Steps Motors. First Part", so that those both real time tasks can be fully related in order to make the whole control system more accurate. The employed techniques are those of Real Time Taking advantage of the possibilities of the micro kernel RT-Linux and the free software distributed in the Unix/Linux operating systems. The signals are obtained by means of an AD converter and are shown in screen using Gnuplot.
Key words: Gnuplot, RT-Linux, converte AD converter, real time

INTRODUCCIÓN

Con el objetivo de sensar la corriente que circula por cada una de las fases del motor, se construyó un sistema de adquisición de datos con un conversor AD configurado en corrida libre, con el circuito integrado ADC0804LCN.[1] Este es de tipo CMOS, 8 bits y aproximaciones sucesivas, con un tiempo de conversión de aproximadamente 100 µs según los cálculos realizados. Además de ser de bajo costo y fácil implementación.

Su principal función es trasformar una tensión analógica que encuentre entre sus terminales (6-7), y dar su correspondiente valor en palabra digital de 8 bits. El numero máximo que se puede obtener con este conversor AD es 2^8=256 en términos digitales seria desde 00000000 hasta 11111111.

HARDWARE

Con el conversor AD no se puede obtener un verdadero valor de la tensión analógica, ya que dependiendo de la resolución para un mismo valor digital, se tienen diferentes valores analógicos (figura 1). Esto está dado por el error de cuantificación que precisamente es igual al tamaño del subintervalo de tensión analógica.

En este caso con el conversor a 8 bits y un intervalo de variación de la tensión analógica de 0 hasta 5 el

tamaño del subintervalo esta dado por la expresión siguiente:

$$\frac{Va}{2^n} = \frac{5}{256} = 0{,}00195V$$

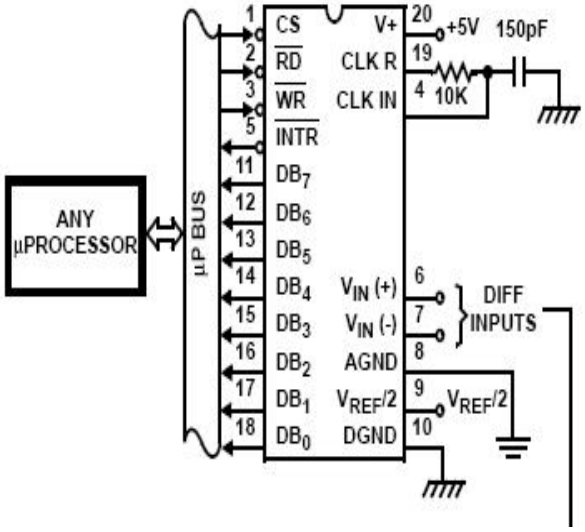


Diagrama del conversor AD.

1

SOFTWARE

El software para la adquisición de datos en tiempo real, se diseñó siguiendo la misma filosofía que la tarea elaborada para el control del motor de pasos.[3,4] En este caso en particular se tiene una tarea que deposita los valores capturados en un FIFO, y una tarea que no es en tiempo real que se encarga de extraer los datos del FIFO y almacenarlos en un fichero.[2] En la figura 2 se presenta un diagrama secuencial, en bloques, de la RT-Task de captura.

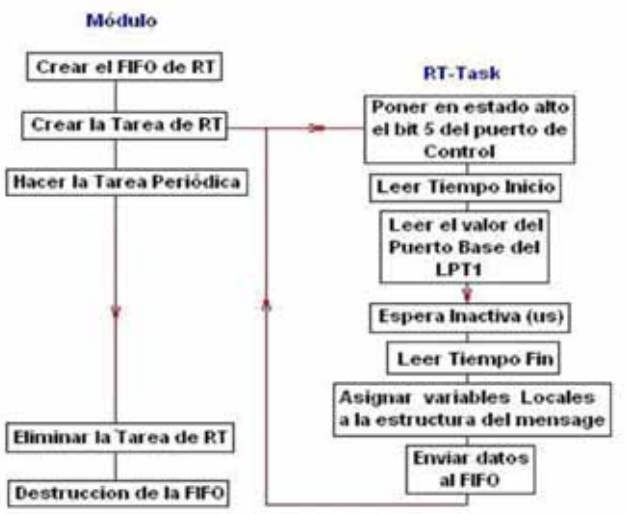


Diagrama en bloque de la tarea de adquisición de datos.

2

TAREA DE TIEMPO REAL PARA ADQUIRIR LOS VALORES DE TENSIÓN POR EL PUERTO PARALELO

A continuación se muestra el código en lenguaje C empleado para la confección de la tarea de adquisición, en tiempo real.

vart.c

```
----------------------------------------Inicio----------------------------
#include <rtl.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <rtl_fifo.h>
#include <sys/io.h>
#include <rtl_sched.h>
#define SIZE 8192
#define fifo_nr 0       //Fifo de Salida de Datos
#define Addr_Base 0x378
#define Addr_Control Addr_Base+2
#define periodo 10
#include <stdio.h>
#include <time.h>
#include "msg.h"        //Cabecera Privada. Permite control de Mensajes


RT_TASK task;


void Real_Time_Captura_Task (void);


void Real_Time_Captura_Task (){
int tiempo=0, enable, resto;
struct msg_control datos;
struct timespec inicio,fin;

//Escribimos en el bit 5 del puerto de control para activar el puerto Base como lectura
outb(32,Addr_Control);

while (1) {

tiempo=tiempo+100; //periodo en usegundos
clock_gettime(CLOCK_MONOTONIC,&inicio);

//Se Adquiere el valor del puerto Base del LPT1
resto=inb(Addr_Base);//#include <fifo.h>

//Espera inactiva de 100us
usleep(100);
clock_gettime(CLOCK_MONOTONIC,&fin);
datos.speed=resto;
datos.enable=tiempo;

//Se escriben las datos en el FIFO de Tiempo Real
rtf_put(fifo_nr,&datos, sizeof(datos));
rtl_printf("Estoy escribiendo: %d , %d \n",fin.tv_nsec-inicio.tv_nsec, resto);
```

```
rt_task_wait();
}

}


//Procedimiento para Cargar el Modulo
int init_module (void) {
RTIME now = rt_get_time ();
rtf_create (fifo_nr, SIZE);
rt_task_init(&task, Real_Time_Captura_Task, 1, 3000, 4);
rt_task_make_periodic (&task, now+3000,periodo);
rtl_printf ("Modulo Montado");
return 0;

}

// Procedimiento para liberar el modulo

void cleanup_module (){
rt_task_delete (&task);
rtf_destroy (fifo_nr);
rtl_printf ("Modulo Desmontado");
}


----------Fin ----------
leer.c
----------Inicio----------

#include <stdio.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
#include "msg.h"

int main() {
FILE *f1;
int fd0;
float res;
int n,i=1;
struct msg_control lectura;
fd0=open("/dev/rtf0",O_NONBLOCK);
f1=fopen("datos.dat","wt");
while (1){
i=i++;
read(fd0,&lectura, sizeof(lectura));
res=(lectura.speed*5) / 256.0;
printf("LaLecturaes:%d,%d,
%.4f\n",lectura.enable,lectura.speed,res);
fprintf(f1,"%d\t%.4f\n",lectura.enable,res);
if (lectura.enable >= 500000) { break;};
}

fclose (f1);
return 0;
}
----------Fin ----------
```

El fichero cabecera msg.h es el mismo explicado en la referencia 2, por esta razón no es necesario poner el código.

Al aparecer el fichero datos.dat con los datos leídos de la FIFO de tiempo real, se procede a graficar con cualquier software útil para tales fines, como lo es por ejemplo el Gnuplot de Linux. Este es un programa para trabajo matemático y representación vectorial en 2 y 3 dimensiones.

RESULTADOS OBTENIDOS: SISTEMA DE ADQUISICIÓN DE DATOS

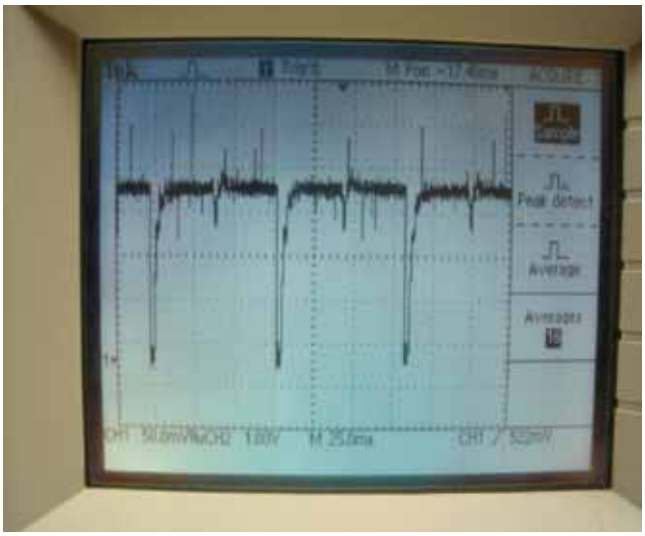
Con el objetivo de comparar las señales de corriente presentes en las bobinas del motor, se obtiene las señales provenientes de un instrumento de medición bien calibrado, y las señales obtenidas mediante el sistema de captura. Al final se comparan ambas señales y se observa su alta similitud.

Como primer paso se verifica la señal presente en las resistencias de sensado del circuito de control.

La figura 3 muestra la señal de corriente tomada en las resistencias de sensado del motor de pasos explicado en la primera parte del artículo. La misma fue registrada con un osciloscopio Tektronix.

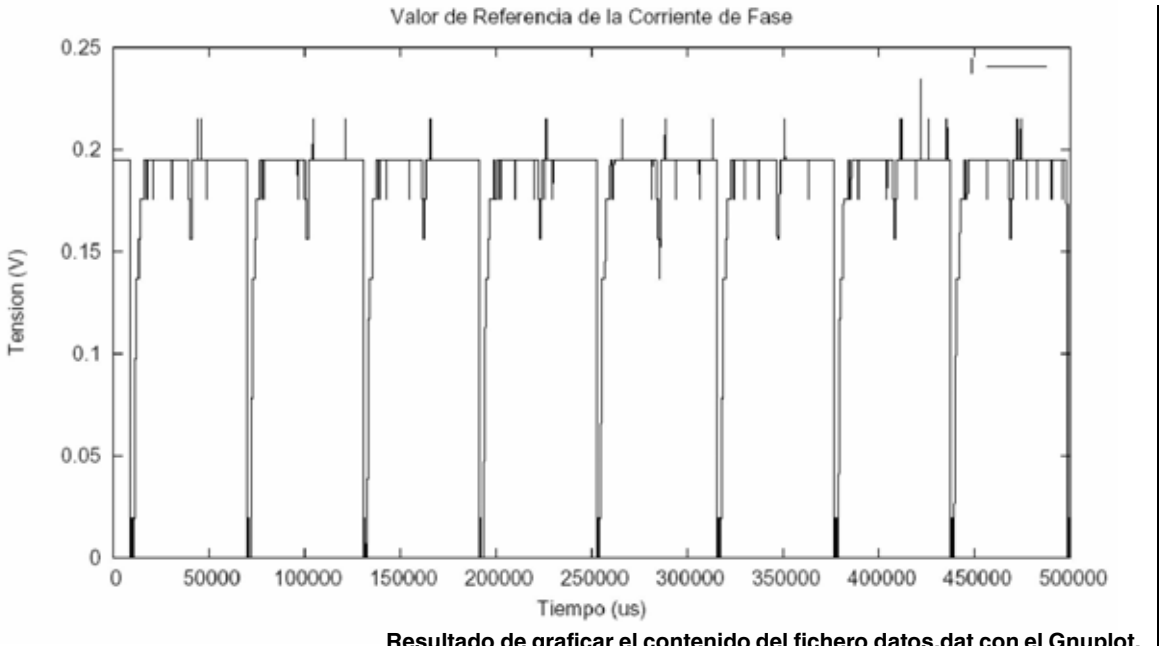
Empleando el hardware descrito anteriormente, una microcomputadora con el software en C y trabajando con el sistema operativo Linux modificado para ejecutar tareas en tiempo real, se obtuvo un fichero con los valores de corrientes circulantes por las resistencias de sensado.

La figura 4 muestra la resultante de graficar los valores



Señal de la corriente circulante por la resistencia de sensado.

adquiridos con la tarea de tiempo real, con el motor en funcionamiento. Para este caso se muestreó la señal cada 100µs durante un tiempo de 500 ms, tal y como se puede observar. Es posible consultar en la bibliografía las diferentes gráficas con distintos períodos de muestreo.[4]



Resultado de graficar el contenido del fichero datos.dat con el Gnuplot.

4

CONCLUSIONES

1. En el presente artículo queda expuesto el trabajo que ha sido realizado para la obtención de la tarea de adquisición de datos en tiempo real estricto, sobre RT-Linux.
2. La tarea de adquisición de datos es utilizada en la captura de la forma de onda de la señal de corriente de fase del motor de pasos y es validada su efectividad, mediante la comparación visual del resultado que ofrece, con la imagen de un osciloscopio Tektronix digital.
3. Con este trabajo se demuestra la factibilidad de la utilización del microkernel de RT-Linux para realizar tareas en tiempo real estricto, tales como la captura de datos inherentes a señales de variación rápida, como lo son las señales de naturaleza eléctrica.
4. El RT-Linux es una alternativa basada en la utilización de software libre, válida para la experimentación con accionamientos eléctricos y el desarrollo de sus esquemas de control.

REFERENCIAS

1. "Ficha técnica del circuito integrado ADC0804LCN".
2. Duany Renté, Ernesto y otros: "Aplicación de RT-Linux en el control de motores de pasos. Primera parte." *Revista Ingeniería Energética*,Vol. XXVIII, No. 2, Ciudad de La Habana, 2007.
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. Barabanov, Michael and Victor Yodaiken: *Getting Started with RTLinux*, FSM Labs, Inc, April, 2001.
Sitios web:www.gnuplot.org, www.rtlinux.org.

AUTORES

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

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

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