GENERACIÓN DE SEÑALES ANALÓGICAS MEDIANTE PWM EMPLEANDO UN NUCLEO EN TIEMPO REAL SOBRE UNA ARQUITECTURA PC SIN ELEMENTOS ADICIONALES DE ENTRADA/SALIDA

L. MIRÓ, M.A. RODRÍGUEZ, D. CASCADO, S. VICENTE, A. CIVIT
Área de Arquitectura y Tecnología de Computadores – Univ. de Sevilla
E-mail: lourdes@atc.us.es

Se propone la realización de una práctica en la cual se emplea la salida digital del altavoz interno de un PC convencional para generar, empleando un sistema operativo en tiempo real, señales analógicas mediante modulación PWM.

1. Introducción

Cada día es más habitual emplear sistemas de control basados en computadores incrustados. De hecho estos sistemas, si incluimos en ellos desde los microcomputadores más sencillos hasta los basados en los procesadores más potentes, han sustituido a buena parte de los controladores analógicos o digitales cableados.
Los sistemas con procesador incrustado para su conexión con el mundo exterior deben, por una parte, manejar señales analógicas y por otra producir respuestas con unos plazos temporales deterministas. En el presente trabajo se presenta una práctica en la que se generan señales análogicas de salida con temporización absolutamente determinista empleando un computador PC compatible convencional sin ningún elemento adicional de entrada/salida. La practica introduce a los alumnos por una parte en las ideas básicas de programación en Tiempo Real (empleando un núcleo sencillo) y, por otra, permite experimentar de una forma muy flexible con la generación de señales análogicas de baja frecuencia mediante PWM.
Tradicionalmente los sistemas digitales han empleado conversores D/A basados en conmutadores y redes de resistencias. Este tipo de conversores tiene dos limitaciones claras:

- Es poco adecuado para su implementación en circuitos integrados totalmente digitales.
- No permite, de forma sencilla, operar con valores de tensiones e intensidades altos.

2. Conversores PWM

Estas causas han hecho que, en los últimos años se popularicen los conversores PWM en los que se modula el valor de tensión (o intensidad) medio de salida modificando el "*duty cycle*" de una onda "cuadrada" de salida. Así si f es la fracción del ciclo que la salida está a valor lógico alto, V_H es la tensión correspondiente a dicho valor lógico y V_l la correspondiente al valor lógico bajo. La tensión media de salida vendrá dada por:

$$V = fV_h + (1-f)V_l$$

Por tanto modificando el valor de f podemos modificar el valor medio de la salida. Si incorporamos un filtro paso baja que "elimine" la frecuencia correspondiente al periodo de PWM tenemos un autentico convertidor D/A. Evidentemente el ancho de banda de la señal estará comprendido entre DC y, como máximo teórico, la mitad de la frecuencia del generador de PWM. En la práctica tendremos que conformarnos con un valor limitado por las características del filtro paso baja empleado.

Muchos microcontroladores sencillos (ej. Varios miembros de las familias PIC, 8051, 68HCxx, etc) disponen de generadores PWM. En el caso de los sistemas PC compatibles no se dispone de un hardware diseñado específicamente con este fin pero la salida del sistema básico de audio consiste en un transistor de salida controlable desde un temporizador compatible con un i8254. Este temporizador puede programarse como un monoestable digital de forma que genere pulsos a nivel bajo de amplitud programable. Si programamos este monoestable digital de forma periódica tendremos una señal "periódica" en la que en cada ciclo podemos ajustar el tiempo que la señal está a nivel bajo, es decir, tendremos un generador PWM.
El i8253 usado como monoestable permite, en las arquitecturas compatibles PC, generar pulsos a nivel bajo de ancho $n \times 0.88 \mu s$ con valores de n entre 1 y 65535. Dado que no podemos variar la frecuencia de la señal de reloj usada por el contador esto hace que la resolución de la señal de salida decrezca al disminuir el periodo de la señal PWM. Así si tenemos un periodo PWM de $t \in \left[2^k \times 0.88 \mu s, 2^{k+1} \times 0.88 \mu s\right)$ podremos tener, en principio, 2^k valores de salida distintos, es decir, una resolución máxima teórica de k bits. Hay que tener en cuenta que en este caso, por el teorema de muestreo, el ancho de banda de la señal de salida es como máximo (en MHz):

$$BWmax \leq \frac{1.19}{2^{k+1}}$$

Cuando no necesitamos toda la resolución disponible y queremos emplear un filtro paso baja muy sencillo (ej. una red RC básica) podemos emplear técnicas de sobremuestreo. Estas técnicas consisten en producir varios pulsos PWM para cada muestra de la señal a sintetizar. De este modo los efectos de la modulación PWM se desplazan hacia frecuencias más altas del espectro y son, por tanto, más fáciles de eliminar.

3. Núcleos de tiempo real y sistemas rate-monotonic.

Los núcleos en tiempo real son herramientas que nos permiten diseñar de forma sencilla el *software* para sistemas incrustados. Existen multitud de estos núcleos disponibles para arquitectura PC. Varios de ellos son o bien de dominio público o usables libremente con fines educativos. Un ejemplo de núcleo de este tipo es el $\mu C/OS-II$ [Lam98] . Diferentes portes de este núcleo pueden encontrarse en http://www.ucos-ii.com. Otra alternativa clara sería emplear un núcleo comercial como el RTKernel de On Time Systems (puede obtenerse gratuitamente una versión de evaluación de http://www.on-time.com) o incluso el Windows CE [Mur97] cuyas características tiempo real mejoran mucho en la versión 3.0. Todos estos núcleos permiten crear de forma sencilla tareas periódicas y gestionar los mecanismos de comunicación entre ellas. Todos estos núcleos utilizan prioridades asignadas estáticamente a las diferentes tareas. Este tipo de asignación de prioridades no es especialmente adecuado para cargas de propósito general pero, sin embargo, facilita en gran manera el diseño de sistemas de tiempo real duro, es decir, aquellos en que las tareas han de cumplir unos plazos temporales estrictos. Este tipo de sistemas suele implementarse y analizarse empleando técnicas *Rate Monotonic* (vease por ej. [Kri97]). En el caso de tareas periódicas las prioridades deben asignarse de forma decreciente con el periodo de las tareas. En este caso, si llamamos P_i al periodo de la tarea T_i., y asignamos índices a las tareas de forma que:

$$\forall i > j : P_i > P_j$$

El tiempo de respuesta de T_i vendrá dado por:

$$r_i = \sum_{j \le i} \frac{r_i}{T_j} e_j$$

Donde e_j es el tiempo de ejecución de la tarea T_j. Esta ecuación no admite solución analítica pero puede resolverse iterativamente de forma sencilla haciendo:

$$r_i^0 = e_i \quad r_i^k = \sum_{j \le i} \frac{r_i^{k-1}}{T_j} e_j$$

Para sistemas viables, es decir, cuando el tiempo de respuesta de toda tarea periódica es menor o igual que su periodo, esta ecuación siempre convergerá, es decir podemos asegurar que:

$$\exists k : \forall m > k \; r_i^m = r_i^k$$

4. Descripción de la práctica.

En nuestra práctica se emplea uno de los núcleos tiempo real mencionados, en concreto RTKernel, para crear un pequeño sistema con dos tareas:

- Lectora: Lee Muestras del disco y las introduce en una cola de comunicación con la otra tarea. Esta tarea no tiene que cumplir un plazo en cada lectura pues la existencia de la cola de comunicación hace que baste con garantizar el cumplimiento de tiempos medios. Es la tarea menos prioritaria y aprovecha todo el tiempo que deja libre la tarea más prioritaria.
- Reproductora: Se encarga de generar el PWM programando el monoestable de acuerdo con las muestras que extrae de la cola. Debe ejecutarse con el periodo del PWM y, evidentemente, tiene requisitos estrictos de temporización. El periodo de está tarea depende del periodo de muestreo original y del sobremuestreo que queramos emplear.

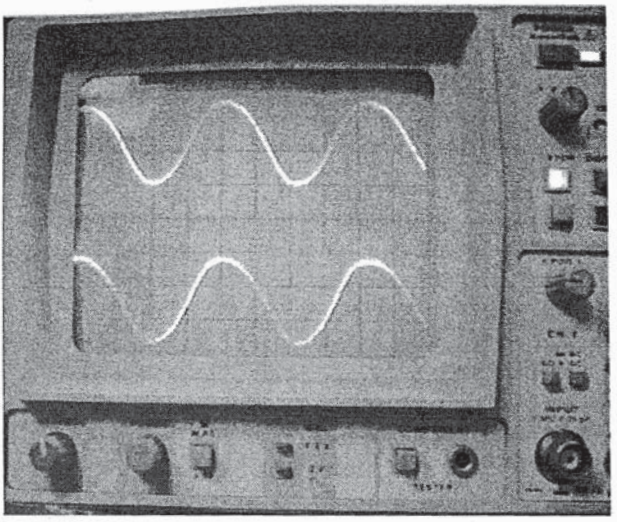
En este caso, la tarea tiempo real duro (reproductora), deberá ser la más prioritaria del sistema y, por las ecuaciones anteriores tendremos que para ella:

$$r_i = e_i$$

Es decir, el tiempo de respuesta coincide con el tiempo de ejecución máximo. Esta es una propiedad general para la tarea de periodo menor en sistemas *Rate Monotonic*.

En un PC basado en procesador Pentium-II a 350MHz la tarea reproductora puede ejecutarse con frecuencias de hasta aproximadamente 100KHz lo que permite la generación de señales cualesquiera de 50KHz de ancho de banda. En el caso concreto de nuestra práctica (correspondiente a la asignatura "Arquitectura de Sistemas en Tiempo Real" de la Titulación de "Ingeniero en Informática" de la Universidad de Sevilla") los alumnos parten de ficheros de audio (.WAV) y además de visualizar la salida obtenida en el osciloscopio pueden escucharla por el altavoz del PC.

En la figura podemos observar la salida generada por el resultado de la práctica tal como la genera el PC sin utilizar ningún tipo de filtro. En estas condiciones parece que debería verse la señal de PWM entre V_h y V_l. Sin embargo, muchos PC's incorporan un condensador en serie con la salida de altavoz. Este condensador elimina la componente de continua de la señal PWM haciendo que ésta aparezca siempre centrada respecto de su valor medio instantáneo. Dado que este valor medio es el valor que codificamos con el PWM, aparece la señal PWM modulada por el valor que hemos codificado.



5. Conclusiones.

La resolución de la señal PWM obtenida en esta práctica es siempre menor a la máxima mencionada anteriormente debido a la introducción de retrasos variables (*jitter*) por el núcleo en tiempo real.

El desarrollo de la práctica, que motiva en gran medida a los alumnos, permite mezclar conceptos de diversas disciplinas que hasta ese momento han sido bastante independientes para el alumno como son los sistemas analógicos, los sistemas digitales y el control en tiempo real.

Referencias

[Lam98] Jean J Lambrose. $\mu C/OS-II$. The Real Time Kernel. R&D Books, 1998. ISBN 0-87930-543-6

[Mur97] John Murray. Real-Time Systems with Microsoft Windows CE. Microsoft Tech. Journal. September 1997

[Kri97] Krishna, C.M., Shin, K. G. "Real Time Systems", Mc Graw Hill, 1997.