ISSN: 1692-7257 Volumen 2 Número 6 año - 2005

Revista Colombiana de
Tecnologías de Avanzada

SYSTEM OF FAST PROTOTYPING BASED ON A DSP AND FREE SOFTWARE

SISTEMA DE PROTOTIPADO RÁPIDO BASADO EN UN DSP Y SOFTWARE LIBRE

Ing. Carlos Andrés Ramos Paja, MSc. Edinson Franco Mejía, Ing. Ángela María Maya Castro

Universidad del Valle
caramos@univalle.edu.co , efm@univalle.edu.co, *anmaya@yahoo.com*
Cali - Colombia

Abstract: : In this article the design and implementation of an autonomous system of control based on a digital processor of signs is presented, where it is possible to download and to execute diagrams of blocks designed in a scientific tool of easy handling called Scicos ©Inria.

Resumen: : En este artículo se presenta el diseño e implementación de un sistema autónomo de control basado en un procesador digital de señales, donde es posible descargar y ejecutar diagramas de bloques diseñados en una herramienta científica de fácil manejo llamada Scicos ©Inria..

Keywords: Sistemas Embebidos, Instrumentación Virtual, Diseño digital, Ingeniería de Control, Prototipado Rápido, Procesamiento digital de Señales, Scilab y Scicos.

1. INTRODUCCIÓN

En el proceso de diseño e implementación de un sistema de control o de sistemas de procesamiento en general, se utiliza un modelo matemático que refleja el comportamiento del sistema a controlar ante condiciones específicas. Al obtener un comportamiento satisfactorio (simulación), se prosigue a implementar los modelos matemáticos del controlador en hardware para evaluar su comportamiento. En este artículo se presenta una solución para la implementación final de controladores, permitiéndole al desarrollador utilizar un entorno fácil de manejar combinado con un potente procesador de manejo matemático.

El artículo esta desarrollado de la siguiente forma: en la segunda sección se presenta una mirada a las diferentes opciones que dispone un desarrollador para implementar un controlador o sistema de procesamiento, se muestran sus ventajas y desventajas. En la sección tres se presenta la arquitectura desarrollada para obtener un sistema autónomo de control, se presentan los modelos de memoria y ejecución que permiten desarrollar grandes diagramas de bloques utilizando muy poca memoria RAM en el sistema embebido. En la sección cuatro se reporta un caso de aplicación del sistema y finalmente se entregan conclusiones y expectativas de trabajos futuros.

2. SISTEMAS PARA IMPLEMENTACIÓN DE CONTROLADORES

Como se mencionó anteriormente, en el proceso de diseño de un sistema de procesamiento de datos, y para el control de sistema de altas prestaciones, es necesario verificar el comportamiento del controlador frente al sistema real de forma rápida, permitiendo cambios con un esfuerzo mínimo.

Para realizar este proceso de prototipado existen múltiples opciones con diferentes grados de eficiencia y costos. De acuerdo a la naturaleza de la solución, en el desarrollo del proyecto se propone la siguiente clasificación: 1. Programación directa en un PC(*Personal Computer o Computador Personal)* e interacción utilizando un sistema de adquisición de datos, 2. Programación directa de un sistema embebido y finalmente 3. Utilización de un sistema de prototipado rápido.

Estas tres clasificaciones permiten ubicar fácilmente cualquier producto de prototipado que se encuentre en el mercado o que se desarrolle con fines específicos. A continuación se describen claramente cada una de las clases de sistemas de prototipado y se ilustran con ejemplos reales.

2.1 Programación directa de un PC

Una de las formas más utilizadas para implementar sistemas de procesamiento de información, es utilizando tarjetas de adquisición de datos para interactuar con sistemas físicos, implementando los algoritmos en un programa discreto desarrollado en algún lenguaje de programación.

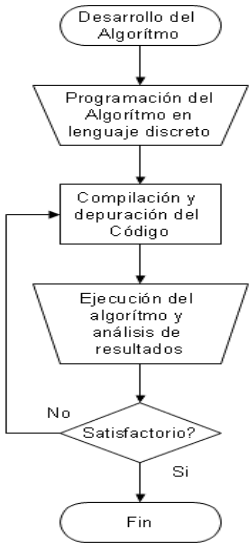

Fig.1 : Programación directa de un sistema de procesamiento (PC)

En la Figura 1 se presenta el diagrama de flujo para la programación directa de un sistema de procesamiento utilizando un PC: es necesario definir el algoritmo a implementar (lo cual es el objetivo primario), luego se debe codificar el algoritmo en un lenguaje de programación, y por lo tanto es necesario invertir tiempo en la depuración del código y corrección de errores en tiempo de compilación. Además, esta tarea supone un cierto nivel de programación que muchos desarrolladores no tienen. Luego de evaluar el comportamiento del sistema, se realizan los ajustes a nivel de código.

Para este tipo de implementación es necesario requerir de un sistema de adquisición de datos que satisfaga las condiciones de la planta a controlar. El costo de un sistema convencional de adquisición de datos oscila entre los \$US 600 a \$US 1200, existiendo sistemas con valores mucho mayores (Instruments, 2005).

Es importante notar que los controladores implementados con sistemas digitales deben ser discretos y es necesario respetar el tiempo de muestreo, lo cual es un requerimiento difícil de garantizar en los sistemas operativos más populares de hoy.

Para solucionar este inconveniente se han desarrollado los *sistemas operativos de tiempo real* y las soluciones de tiempo real, las cuales se dividen en dos grupos (Maya, 2002):

- *Soft Real-Time:* Son sistemas que respetan el tiempo de ejecución de una tarea con un error impredecible, el cual se restringe a un valor por debajo del mínimo tiempo requerido en cada una de las tareas a ejecutar.
- *Hard Real-Time:* Son sistemas que respetan exactamente el tiempo de ejecución de cada tarea, garantizando los tiempos definidos por el usuario.

La implementación de sistemas *Soft Real-Time* se han desarrollado utilizando el sistema operativo de Microsoft DOS, donde la única tarea es el sistema de control. También se han logrado implementar utilizando Microsoft Windows, anulando los procesos del sistema; así como en el sistema operativo Linux, el cual permite una resolución efectiva de milisegundos (FSMLabs, 2005).

La implementación de sistemas *Hard Real-Time* es posible en dos casos: realizando una programación a nivel de hardware o utilizando un sistema operativo de tiempo real. En la primera opción el programador debe realizar el código necesario para manejar los dispositivos del PC, ya que al no contar con sistema operativo el desarrollo es difícil y largo. La segunda opción es una de las más utilizadas en el sector científico, ya que los sistemas operativos de tiempo real ofrecen grandes utilerías y ayudas para implementar tareas cíclicas respetando el tiempo de ejecución de cada una. Un ejemplo de estos sistemas operativos es **QNX** (Qnx, 2005), el cual se ha convertido en un estándar en el sector industrial e investigativo, pero sus costos impiden que centros de investigación con recursos limitados puedan adquirirlo. Una alternativa es el sistema operativo de libre distribución **RT-Linux** (FSMLabs, 2005), el cual es una extensión de tiempo real para Linux.

La implementación de controladores (lineales y no lineales) utilizando RT-Linux es una tarea difícil porque es necesario programar los algoritmos, la interfase de usuario y cualquier otra utilidad que se requiera, lo cual incrementa el tiempo de prototipado y depuración, añadiendo una carga extra al trabajo inicial.

2.2 Programación directa de un sistema embebido

Los sistemas embebidos son arquitecturas reducidas que disponen de los recursos necesarios para cumplir con una tarea específica de forma autónoma e independiente, permitiendo en algunos casos la supervisión remota o ajuste de parámetros en línea.

La masiva utilización de este tipo de sistemas radica en tres características principales: primero, son sistemas de hardware reducido y por lo tanto más simple de programar respecto un PC; segundo, al ser reducidos no requieren de un sistema operativo, siendo posible implementar algoritmos *Hard Real-Time* con un costo bajo; tercero, es relativamente fácil expandir un sistema embebido adecuándolo a nuevas condiciones.

La mayor dificultad de implementación en un sistema embebido es la programación, ya que un ingeniero de control puede conocer perfectamente el algoritmo de control, pero tener dificultades en la codificación e implementación final. Para solventar este inconveniente, se desarrollaron los sistemas de prototipado rápido.

2.3 Sistema de prototipado rápido

La implementación de algoritmos en sistemas de tiempo real, que permitan una rápida modificación y un conocimiento mínimo de la plataforma sobre la cual se esta desarrollando, es una de las soluciones comerciales más explotadas en los últimos años.\par

Fig. 2: Sistema de Prototipado rápido

En la Figura 2 se presenta el diagrama de flujo para implementar un algoritmo utilizando un sistema de prototipado rápido. Estos sistemas normalmente disponen de una interfase gráfica en la cual se

define mediante bloques los algoritmos a implementar, se realizan simulaciones y finalmente se descarga el algoritmo al sistema final de procesamiento, permitiendo la inspección de señales y variables. Este proceso hace transparente la generación del código ejecutable, permitiendo al usuario enfocarse en los algoritmos que se desean implementar y no en la programación para la implementación (Ramos, 2002).

Estos sistemas generan código ejecutable para múltiples plataformas las cuales se pueden dividir en tres grupos: primero, los sistemas operativos convencionales (p.e. Microsoft Windows o Linux, donde el software de manipulación matemática Matlab y sus ToolBox's *Real Time Workshop* y *Real Time Windows/Linux Target* generan aplicaciones a partir de diagramas de bloques generados con Simulink (MathWorks, 2005)); segundo, la generación de código para sistemas operativos de tiempo real tales como QNX o RT-Linux. Matlab dispone de dos ToolBox's para generación de código para estos sistemas operativos, siendo de libre distribución el correspondiente a RT-Linux, llamado ST-RTL (Garcia et al., 2002); como tercera opción esta la generación de código para sistemas embebidos, los cuales se pueden encontrar en todas las gamas, desde sistemas de bajo costo hasta sistemas de grandes prestaciones, multiprocesamiento, paralelismo, etc. Un ejemplo de sistema de bajo costo es el utilizado en el desarrollo del proyecto, consta de un DSP en punto flotante Texas Instruments TMS320C31, un conversor Análogo a Digital de 14 bits y un conversor Digital a Análogo de igual resolución. Como ejemplo de sistema robusto y potente, se tiene el sistema dSPACE, el cual consta con un procesador RISC(*Procesador con número de instrucciones reducido)* conversores PWM trifásicos, entradas y salidas análogas con 16 bits de resolución y muchas otras prestaciones (dSPACE, 2005).

Las ventajas de estos últimos sistemas son notables ya que disponen de gran potencia de cálculo con procesadores dedicados, garantizando el tiempo de ejecución de los algoritmos a un costo razonable, donde el tiempo de desarrollo se reduce al diseño del algoritmo y su puesta a punto, sin preocuparse por la implementación o la plataforma.

3. SISTEMA DE PROTOTIPADO RÁPIDO BASADO EN DSP Y SCICOS

Debido a la necesidad de un sistema de prototipado rápido que permita implementar algoritmos de procesamientos de datos con una interfase amigable y gran potencia de cálculo a bajo costo, se diseño un sistema basado en un DSP TMS320C31 y el software libre(*Bajo licencia de uso libre GPL)* Scilab (utilizando la interfase gráfica basada en bloques Scicos).

Se escogió el software Scilab, el cual consiste en un motor de manipulación matricial, ToolBox de sistemas de control, procesamiento digital de señales y una interfase de simulación basada en bloques llamada Scicos. El sistema permite diseñar un diagrama de bloques, realizar simulaciones del mismo, ejecutarlo en el DSP y leer señales desde el sistema en tiempo de ejecución.

El sistema se divide en tres partes: el generador de código, el sistema de conexión entre el Scicos y el DSP y el sistema de ejecución dentro del DSP.

3.1 Generador de Código y Sistema de Conexión

El generador de código se encarga de construir las estructuras básicas donde
la información del diagrama de bloques se almacena para luego ser descargada al DSP. En la Figura 3 se presenta el diagrama de bloques del generador de código.

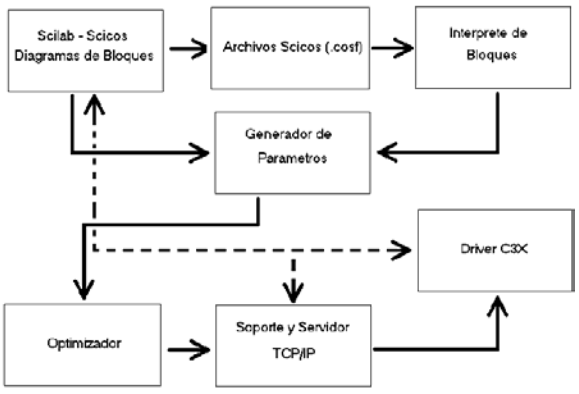

Fig.3: Generador de Código

Inicialmente el usuario diseña un diagrama de bloques utilizando la herramienta Scicos, donde es posible simular el sistema con los bloques

funcionales disponibles los cuales tienen el mismo algoritmo que su equivalente implementado en el DSP. Scicos permite guardar dichos diagramas en dos formatos: el formato binario llamado *.cos* y el formato texto llamado *.cosf*, este último es el utilizado para obtener la información sobre el diagrama de bloques a descargar.

Posteriormente, el interprete de bloques busca en el archivo (.cosf) los bloques y sus respectivas conexiones, creando así una serie de estructuras que almacenan la información de cada bloque. Con estos datos, el generador de parámetros crea una estructura a descargar en la memoria del DSP para almacenar de forma eficiente la información, y junto con el optimizador crean un vector de referencia, el cual indica al DSP como ejecutar los diferentes bloques y accesar a los parámetros sin requerir de posiciones de memoria definidas, ahorrando de esta forma cerca de un 30\% de memoria RAM, el modelo de memoria se presenta en detalle más adelante.
Para transmitir el diagrama de bloques, actualizar parámetros y accesar variables en tiempo de ejecución se escribió un driver(*Controlador de dispositivo, el cual permite al sistema operativo manipular un periférico)* que controla el DSP desde el sistema operativo Linux. El DSP se conecta utilizando el puerto paralelo del PC en modo SPP(*Puerto paralelo en modo estándar*) permitiendo accesar la memoria RAM del DSP para lectura y escritura. Adicional a esto, se desarrollo un servidor basado en Sockets(*Estructura básica para la comunicación* utilizando)que se comunica con dos bloques especialmente desarrollados para Scicos, encargados de leer y escribir el servidor para accesar remotamente el DSP. Este sistema permite a un usuario accesar el DSP para descargar diagramas, leer variables y modificar parámetros desde cualquier punto del hiperespacio, lo cual permite diseñar e implementar controladores sobre una planta física en el laboratorio desde un PC con acceso a la red.

3.1.1 Modelo de memoria circular: El modelo de memoria circular es ampliamente conocido en los sistemas embebidos, siendo muy utilizado para la implementación de buclas de repetición en lenguaje ensamblador.

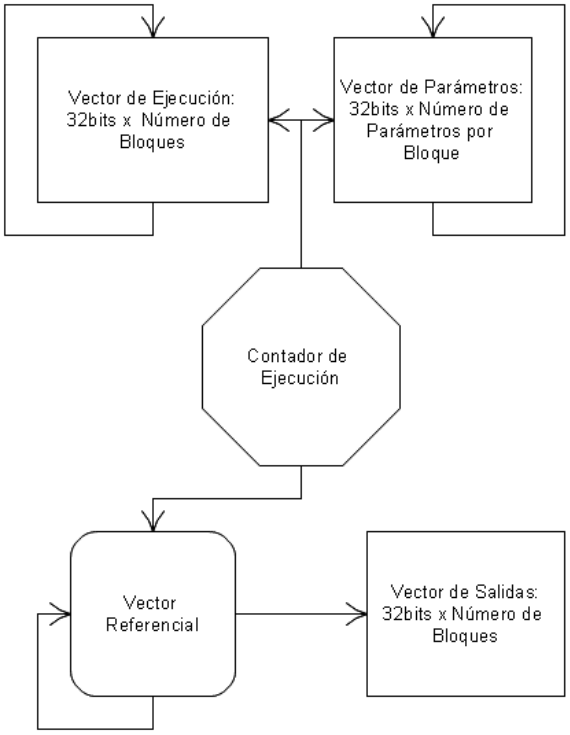

Fig. 4 : Modelo de Memoria Circular

El objetivo del nuevo modelo de memoria es almacenar la mayor cantidad de datos utilizando unas pocas referencias para accesarlos. Como se observa en la Figura 4, todo diagrama de bloques queda completamente descrito por cuatro vectores de datos, una variable principal y tres punteros a memoria.
El *contador de ejecución* describe el bloque en ejecución realizando una llamada por referencia (al vector de ejecución). Utilizando esta misma variable, se referencian los parámetros del bloque y se accesa el vector de referencia que indica las posiciones de las salidas de los bloques que sirven como entradas. Todo esto es posible debido a que el optimizador organiza los valores de ejecución y los parámetros en el orden en que se ejecutan los bloques, por ende el apuntador de cada uno de estos vectores se incrementa cuando se incrementa el contador de ejecución; al finalizar la ejecución los punteros se llevan a sus valores iniciales reiniciando el sistema para un nuevo ciclo, es por esto que reciben el nombre de *punteros circulares* y el modelo de memoria se denomina *circular*.

3.2 Sistema de ejecución del DSP

Para la ejecución de los bloques, se desarrollo un entorno de ejecución(Conocido comúnmente como *Run Time System)*en lenguaje C, utilizando el sistema de desarrollo *Code Composer C3X/C4X*, el cual permite escribir, compilar, simular y depurar programas para el TM320C3X/4X en un PC.

Para garantizar *Hard Real-Time* se utiliza uno de los temporizadores del TMS320C31, con el cual se sincroniza el tiempo de espera entre la adquisición de un dato desde el bus y la salida por el mismo hacia una salida DAC o PWM.

Este tipo de ejecución *interpretada* permite una fácil actualización del sistema, pero más importante, permite definir la cantidad de memoria libre para el desarrollo del diagrama de bloques, y mediante la implementación del modelo de memoria circular, optimizar su utilización.

3.2.1 Arquitectura de Hardware:

El sistema fue desarrollado utilizando el Kit de desarrollo *TMS320C3X DSK(*Starter Kit *)*, el cual permite el acceso al bus de datos y al bus de memoria del DSP. Utilizando el dispositivo lógico programable Flex 10K, se diseño un banco de expansión de memoria para el DSP, el cual permite al DSP manejar 2Mb de memoria externa, además se adicionaron diez conversores ADC de 12 bits y 333 Khz de frecuencia de conversión, así mismo se adicionaron diez conversores DAC de 12 bits con tiempo de conversión de 12 ns (Jaramillo, 2002).

4. CASO DE IMPLEMENTACIÓN: CONTROL DE UNA PLANTA DE PRESIÓN

Se implementó un controlador digital para la planta de presión *FeedBack PCT14*, la cual dispone de un sensor de presión y una válvula de acción lineal. Utilizando un sistema de adecuación de señal se obtiene una entrada y una salida de voltaje entre 0 y 2 volts que representan el rango de apertura de la válvula y la señal de presión en un tanque respectivamente.

El objetivo es regular la presión de aire en un tanque mediante la variación del flujo de entrada y un flujo de salida fijo. El modelo matemático de la presión en el tanque respecto la apertura de la válvula, normalizado a señales de voltaje entre 0 y 2 volts es el siguiente:

$$G(s) = \frac{1.13}{1+1.26s}$$

Utilizando un tiempo de muestreo de 0.1 segundos, se diseñó un controlador digital con la siguiente función de transferencia:

$$H(z) = 1.54\frac{z-0.91}{z-1}$$

Fig. 5: Sistema Autónomo de Control.

La implementación en el sistema autónomo de control (Figura 5) se realizó mediante el diagrama de bloques de la Figura 6. Esta implementación utiliza los bloques de entrada y salida de datos (para acceder los conversores ADC y DAC de la tarjeta TMS320C3X DSK), un bloque de función de transferencia digital, un bloque de referencia y un bloque de tiempo de muestreo.

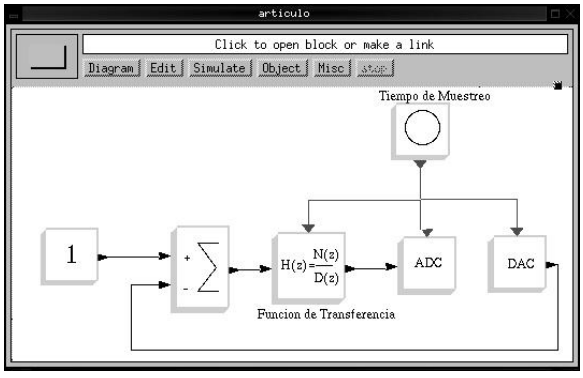

Fig. 6: Diagrama de bloques del controlador digital.

El controlador se configura en el bloque función de transferencia, el tiempo de muestreo se asigna a voluntad y el *Set Point* se define en el bloque constante.

Para descargar el diagrama de bloques basta con guardarlo en formato *.cosf* y ejecutar la aplicación *Scicos2dsp*, la cual interpreta el archivo y lo descarga al DSP donde se ejecuta al instante.

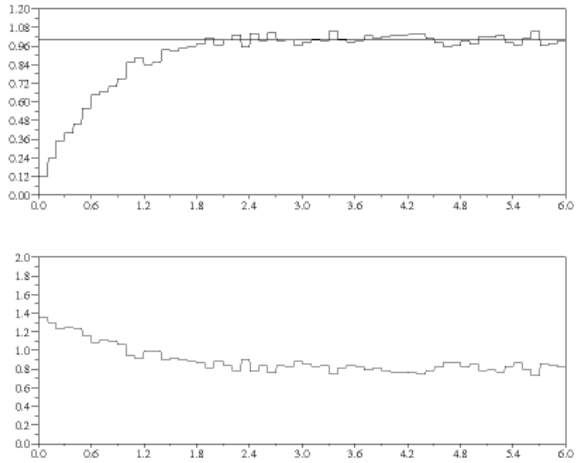

Fig. 7: Señales de Presión(Arriba) y control de apertura de la válvula (Abajo).

La lectura de las señales y variación de parámetros se realiza mediante el acceso al servidor de conexiones que gestiona el intercambio de información con el DSP (puerto 5555); en la Figura 7 se observa las señales de presión (salida del conversor ADC) y control (salida del controlador), las cuales son accesadas desde un PC remoto a través del protocolo TCP/IP.

5. CONCLUSIONES Y TRABAJOS FUTUROS

Los sistemas de prototipado rápido permiten la implementación de algoritmos complejos sin preocuparse por la codificación de los mismos, y junto con la utilización de sistemas embebidos es posible realizar sistemas *Hard Real-Time* a un costo moderado.

La reducción de memoria de direccionamiento en un sistema microprocesado es posible mediante la utilización de punteros circulares y referencias directas a memoria, pero es necesario realizar una organización de la información para permitir tanto ahorro de memoria como disminución en el número de instrucciones para accesarla.

Un desarrollo posterior es el diseño y construcción de un sistema de multiprocesamiento utilizando dos o mas procesadores (DSP, microprocesador, etc.) que permitan un aumento real de velocidad compartiendo las tareas a realizar.

REFERENCIAS

dSPACE (2005). dspace homepage.

FSMLabs (2005). Rt-linux homepage.

Garcia, R. Murillo, F.Wornle, B.G. Stewart and D.KHarrison (2002). Real-time remote network control of an inverted pendulum using st-rtl.

Instruments, National (2005). National instrumentshomepage.

Jaramillo, Adolfo Andrés (2002). *Sistema deAdquisición de datos en Tiempo Real y altas prestaciones*. Universidad del Valle - Proyecto de Grado- Meritorio.

MathWorks (2005). Mathworks homepage.Maya, Ángela María (2002). *Desarrollo de bloquesfuncionales para el DSP TMS320X3X*. Universidad del Valle - Proyecto de Grado.

Qnx (2005). Qnx homepage.

Ramos, Carlos Andrés (2002). *Sistema de Comunicación e intercambio de información entre un DSP y Scicos*. Universidad del Valle – Proyecto de Grado - Laureado.