RIELAC, Vol.XXXIII 1/2012 p.1-15 Enero - Abril ISSN: 1815-5928

RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

Algoritmos para visualización a través de módulos LCD gráficos

Joel Pino Gómez[1], Ailyn M. Hernández Mauri[2], Yosvany Vento Ramos[3], Fidel E. Hernández Montero[4]

1 Universidad de Pinar del Río, Cuba, jpino@tele.upr.edu.cu, Calle Martí, No. 270. Pinar del Río

2 ETECSA, Pinar del Río, Cuba, , jpino@tele.upr.edu.cu

3 ETECSA, Pinar del Río, Cuba, yosvany.vento@etecsa.cu

4 Universidad de Pinar del Río, Cuba, fidel@tele.upr.edu.cu, Teléf. (48)- 755318

RESUMEN / *ABSTRACT*

El desarrollo de sistemas embebidos (como el que se acomete en los laboratorios del Grupo de Investigaciones para el Diagnóstico Avanzado de Maquinarias(GIDAM) de la universidad de Pinar del Río(UPR)) en general presenta limitaciones para realizar de manera autónoma la visualización de datos y resultados de procesamiento, así como para posibilitar una interfaz visual efectiva de interacción con los usuarios. Este problema encuentra solución en el empleo de pantallas de cristal líquido o módulos LCD (Liquid Cristal Display). Por ello, el objetivo de este trabajo consistió en desarrollar y validar un paquete de algoritmos de programación para un módulo gráfico LCD. El sistema desarrollado es una aplicación a microcontrolador para manejar el LCD. Inicialmente se logró la visualización esperada en forma de simulación, y posteriormente se desarrolló su implementación práctica. Se obtuvieron los resultados esperados, presentando las visualizaciones de las señales graficadas una calidad comparable con las de programas profesionales.

Palabras claves: LCD, Algoritmos, Microcontrolador.

Embedded systems development, such as that carried out by GIDAM at the University of Pinar del Rio, in general conveys limitations for, in an autonomous way, displaying data processing outcomes. This problem can be solved by using Liquid Crystal Displays Modules (LCD). Hence, the goal of this work is to develop and test a group of algorithms for displaying on LCD. Firstly, algorithms were tested in a simulation environment, and then they were implemented on a real application. The obtained results were the expected ones and the outcomes signal representations could be compared to those obtained by using professional softwares.
Key words:LCD, Algorithms, Microcontroller.

Displaying Algorithms for Graphic LCD Modules.

1. INTRODUCCION

Las empresas cubanas están limitadas para aplicar estrategias de diagnóstico predictivo, debido al alto costo de la instrumentación requerida para desarrollarlo, pero además tampoco existe el conocimiento difundido de cómo implementarlo. Por otra parte, los sistemas actualmente introducidos no permiten la aplicación de técnicas de procesamiento avanzadas, por lo que no pueden ser utilizados en tareas de diagnóstico complejas.

Una variante para resolver los inconvenientes de la aplicación de técnicas predictivas en el país la constituye el diseño y construcción de un sistema de diagnóstico portable, el cual permita, además del procesamiento de señal necesario, mostrar sus resultados y establecer la interacción adecuada con el usuario. El sistema se basaría en el esquema básico de los procesadores digitales de señal (DSP), partiendo fundamentalmente de sensores del tipo acelerómetro para medir las vibraciones.

El sistema a desarrollar, según el diseño propuesto, no es capaz, de manera independiente, de visualizar los resultados deprocesamiento, ni proporcionar una interfaz visual atractiva de interacción con el usuario.

Se propone establecer un paquete de algoritmos de programación para un módulo gráfico LCD (pantallas de cristal líquido), que permita la visualización de los resultados de procesamiento, así como la interactividad con el usuario. Se pretende además comparar los resultados a obtener de forma simulada y los resultados prácticos con los del software profesional Matlab.

2. DESCRIPCIÓN DEL SISTEMA

El desarrollo del trabajo se dividió en dos partes fundamentales: una de simulación y otra de desarrollo práctico. Cada una de estas partes fue tan necesaria como la otra pues los elementos manejados en la primera parte tributan de manera directa al trabajo en la segunda parte.

2.2ELEMENTOS Y HERRAMIENTAS UTILIZADAS EN EL PROCESO DE DESARROLLO Y PRUEBA DE ALGORITMOS EN ENTORNO DE SIMULACIÓN

2.1.1 BREVE DESCRIPCIÓN DE LOS LCD

Una pantalla LCD es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos portables, ya que consume cantidades muy pequeñas de energía eléctrica [1].

Las memorias internas de las pantallas LCD son la memoria RAM (Random Access Memory), que a su vez se subdivide enDDRAM (Display Data RAM) y CGRAM (CharacterGenerator RAM), y la memoriaROM (ReadOnlyMemory), que posee un generador de caracteres ROM(CG-ROM). La CGRAM es la que contiene los caracteres definibles por el usuario[1].

Asociado a un LCD generalmente existirá el controlador correspondiente debido a que las unidades de microprocesadores no son capaces de manejar el LCD directamente. El microprocesador le transmite al controlador del LCD la información que se visualizará en pantalla.

Uno de los modos usados para mostrar los datos en pantalla es el modo gráfico, en el que cada píxel de la pantalla está caracterizado por una palabra, de tamaño determinado (puede ser de 1 bit, 8 bits, etc.), en la memoria DDRAM. Si se guarda en una dirección de la RAM un dato, por ejemplo, en la dirección 00H el valor 11110000, se mostrará en un LCD monocromático los ‘1’ como píxeles activados y los ‘0’ como píxeles desactivados, tal y como se muestra en la Figura 1.

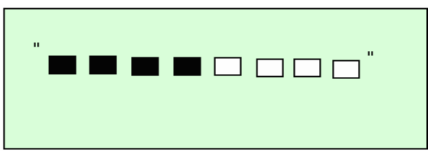

Fig. 1: Representación gráfica de un byte de datos (ocho píxeles)

El módulo LCD utilizado en el entorno simulado de este trabajo es el PG128128-A, el cual presenta características muy similares (el mismo controlador de LCD, por ejemplo) al utilizado para la implementación práctica de la aplicación (ver Figura 2) [2].
El LCD PG128128-A contiene el controlador de TOSHIBA T6963C. Diseñado con el objetivo de controlar pantallas de cristal líquido de pequeño, mediano y gran tamaño. Es este circuito integrado el que recibe e interpreta todos los comandos que se le envían al módulo LCD, ya sea desde una PC, un microcontrolador o desde cualquier otro dispositivo que se desee utilizar para manejar la misma [3].
El controlador, puede servir de interfaz entre la pantalla y cualquier microcontrolador de ocho bits. También permite la comunicación entre el procesador y la memoria VRAM (Video RAM) que poseen las pantallas. Se encarga de generar las señales de tiempo y datos necesarias para el correcto funcionamiento del resto de los circuitos integrados que contribuyen al manejo del LCD. Tiene una memoria ROM que le permite generar 128 caracteres diferentes CG-ROM y posee capacidad para controlar hasta 64 kB de memoria RAM externa VRAM. Por último, cabe destacar que este controlador puede soportar una amplia gama de formatos de pantallas y tiene la capacidad de combinar textos y gráficos dentro del LCD.

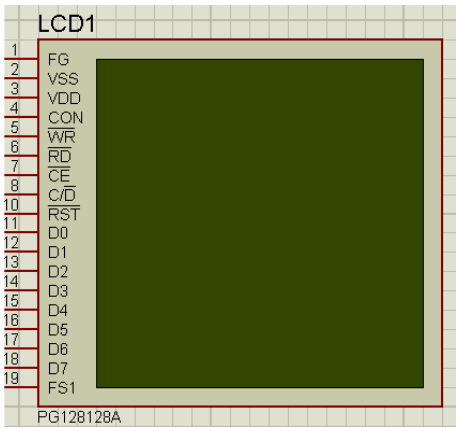

Fig. 2: Módulo LCD PG128128A

2.1.2 PLATAFORMA DE SIMULACIÓN PROTEUS

Proteus es el entorno de simulación utilizado en este trabajo. Este software está diseñado para la realización de proyectos de desarrollo de equipos electrónicos en todas sus etapas: diseño, simulación, depuración y construcción. Específicamente en este trabajo se empleó la herramienta ISIS, la cual permite la elaboración avanzada de esquemas electrónicos con una biblioteca que incorpora más de 6000 modelos de dispositivos electrónicos digitales y analógicos.
Proteus es capaz de leer los ficheros con el código ensamblado para los microprocesadores de las familias PIC, AVR, 8051, HC11, ARM/LPC200 y BASIC STAMP y simular perfectamente su comportamiento. Incluso puede mostrar su código e interactuar en tiempo real con su hardware, permitiendo usar modelos de periféricos animados tales como diodos emisores de luz(LED), LCD, teclados, terminales RS232, simuladores de protocolos I2C, etc.

2.1.3 SOFTWARE PROFESIONAL KEIL µVISION

Para implementar y verificar los algoritmos de programación se empleó el software profesional Keil uVision3, pues este posibilita la creación y compilación de programas para microprocesadores en lenguaje ensamblador y en lenguaje C/C++. Además, crea un fichero ejecutable con extensión .hex que posteriormente puede ser cargado por el microcontrolador en el entorno simulado de Proteus, o descargado en la etapa de desarrollo práctico en la memoria de programa del microcontrolador. El lenguaje de programación utilizado es C.

2.1.4 MICROCONTROLADOR UTILIZADO

Un microcontrolador es un circuito integrado que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y unidades de entrada/salida [1]. El microcontrolador utilizado en el trabajo con el módulo LCD de forma simulada es el MCS8051 (ver Figura 3). El modelo MCS8051 contiene un código especial de soporte que proporciona una

depuración de niveles de fuente con el ambiente de desarrollo del software Keil µVision. Al mismo tiempo, Proteus permite que uno o más ficheros de programa sean cargados hacia la memoria de código interna del microcontrolador; los ficheros pueden ser ficheros Intel hex o ficheros OMF51.

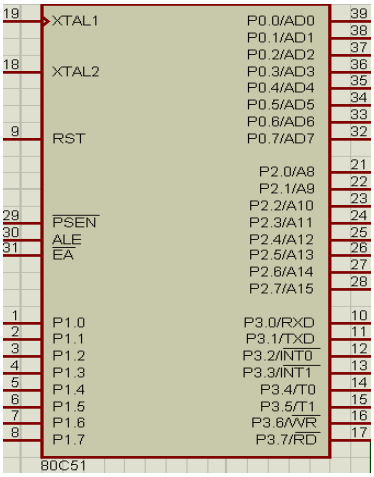

Fig. 3: Microcontrolador MCS8051

2.2 ELEMENTOS Y HERRAMIENTAS UTILIZADAS EN EL PROCESO DE IMPLEMENTACIÓN Y PRUEBA DE ALGORITMOS EN ENTORNO PRÁCTICO REAL

2.2.1 LCD UTILIZADO

El módulo LCD utilizado es el correspondiente a la serie 128240-A de DISPALYTECH. Este módulo LCD está constituido por un circuito impreso en el que se encuentran integrados los drivers del LCD y los pines para su interconexión (ver Figura 4) [4].
En total se pueden visualizar 16 líneas de 30 caracteres cada una, es decir, 16 x 30 = 480 caracteres en modo texto y 240 x 128 píxeles en modo gráfico.

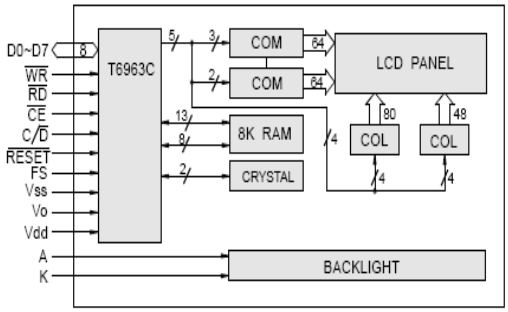

Fig. 4: Diagrama en bloques del sistema

Como se puede apreciar en la Figura 4, este módulo LCD presenta el controlador T6963C, el cual es el que recibe e interpreta todos los comandos que se le envían al módulo LCD, ya sea desde una computadora personal (PC), un microcontrolador o desde cualquier otro dispositivo que se desee utilizar para manejarlo. Este controlador de LCD fue caracterizado en la Sección 2.1.1.

2.2.2 MICROCONTROLADOR UTILIZADO

Para desarrollar el sistema práctico se empleó un microcontrolador de 8 bits de la familia 8051, fabricado por Dallas Semiconductors, el DS5000. Algunas de las características de este microcontrolador son: 4 puertos paralelos de 8 bits bidireccionales, cinco vectores de interrupción, un reloj externo de 12 MHz (11.0592 MHz) y se alimenta con 5 V de DC. Este

dispositivo en particular tiene 32 kB de RAM no volátil (NV RAM), la cual se comparte en memoria de programa y de datos, según convenga [5].Además, se puede programar por puerto serie, lo que facilita su utilización. Presenta un encapsulado de 40 pines como se muestra en la Figura 5.

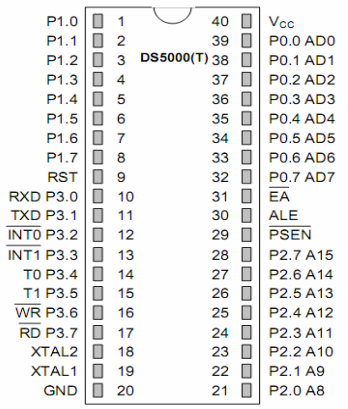

Fig. 5: Circuito integrado, microcontrolador DS5000

3. DESARROLLO DE SISTEMA Y PRUEBA DE ALGORITMOS EN ENTORNO DE SIMULACIÓN

Como se mencionó antes el trabajo se dividió en dos partes fundamentales: una de simulación y otra de trabajo práctico. La idea rectora de todo el trabajo fue establecer un paquete de algoritmos de programación que permitiera la visualización de señales y otras utilidades características a los desarrollos de sistemas electrónicos autónomos. Primero se trabajó en el montaje del sistema Microcontrolador-LCD en el entorno simulado de Proteus 7.2 y desde ese ambiente se desarrollaron los algoritmos y programas correspondientes en C para inicializar el LCD, escribir píxeles, escribir líneas, y toda una estrategia para visualizar señales y para que el usuario pueda interactuar ampliamente con tales visualizaciones. Una vez alcanzados los resultados deseados de forma simulada se trabajó en el desarrollo práctico del sistema, que con características similares al anterior, sería el encargado de mostrar en un LCD real los resultados obtenidos en la etapa de simulación.

3.1 DISEÑO DE LA APLICACIÓN

El trabajo de simulación desde la plataforma Proteus 7.2 se desarrolló en la herramienta ISIS 7 Professional. El diseño consistió en un módulo LCD, el PG128128A, y un microcontrolador, el 80C51, conectados entre sí de forma tal que el puerto 1 del microcontrolador se conectó a los 8 pines del bus de datos del LCD, mientras que y los cuatro pines menos significativos del puerto 3 del microcontrolador a los pines de control del LCD, en este caso, WR, RD, CE y C/D. Para lograr la interacción del sistema con un usuario se agregaron 5 teclas, las cuales se conectaron directamente a 5 pines del puerto 2 del microcontrolador. Como los 128 bytes disponibles de memoria RAM interna del microcontrolador son insuficientes para manejar los datos y la visualización se hizo necesario conectar memoria de datos externa, en particular el circuito integrado 6116 de 2 kB de RAM estática, empleándose además un dispositivo digital Latch, el 74HC373, para multiplexarpor el puerto 0 direcciones y datos. Se puede observar este diseño en la Figura 6.

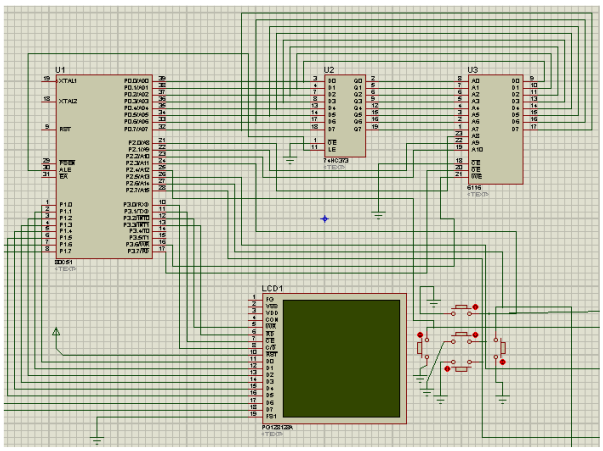

Fig. 6: Diseño de la aplicación en Proteus 7.2

3.2 ESTRATEGIA DE PROGRAMACIÓN

En el programa implementado, después de realizar la inicialización del LCD, se establece un esquema de manipulador de eventos (con la correspondiente definición de 'objeto'), en este sentido se debe destacar que esto fue realizado sobre la programación de un microcontrolador. Para el hardware específico establecido, son 5 los posibles eventos a atender, los cuales están directamente relacionados con las 5 teclas acopladas al sistema (la idea del manipulador de eventos es esencial si se tiene en cuenta la utilización de otros medios de interacción con el sistema, tales como joystick, panel táctil, etc.)[1]. Cuando se oprime una tecla, se considera que ha ocurrido un evento específico a atender, y se ejecuta la subrutina de atención correspondiente a la tecla específica oprimida. Al terminar su ejecución, se retorna al manipulador de eventos como se puede observar en la Figura 7.

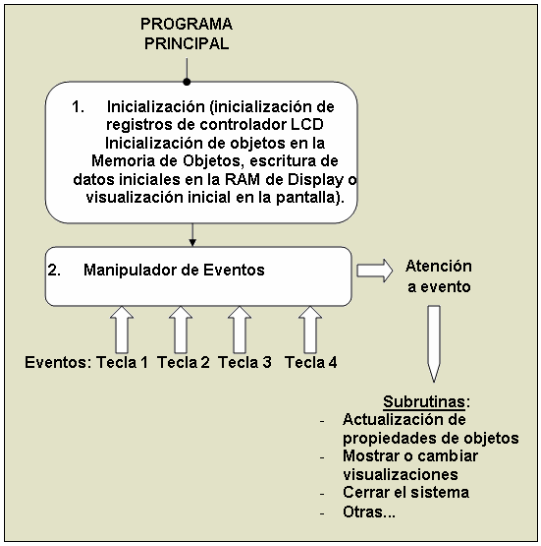

Fig. 7: Estrategia de programación

Cualquier evento generado modificaría algún elemento que se esté visualizando en pantalla, por ejemplo, mover el cursor, activar/desactivar objetos visuales, cambiar hacia una nueva visualización, etc. Para realizar esto, a cada elemento significativo a modificar, por ejemplo, botones de comandos, teclas visuales, cajas de texto, etc., se le asignó una instancia de objeto con propiedades. Por tanto, a cada uno de ellos se le llamó "objeto visual" (ver Figura 8).

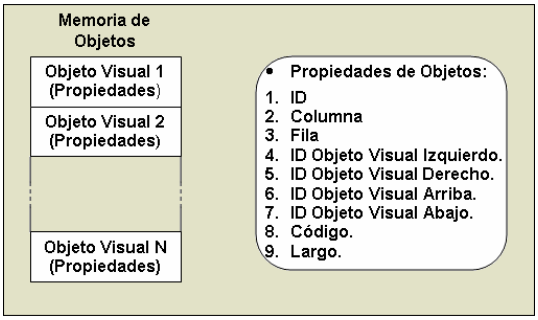

Fig. 8: Características de los objetos

3.3 EJEMPLOS DE ALGUNOS ALGORITMOS

A continuación se presenta una descripción de algunos de los algoritmos desarrollados.

- Buscar Dirección

Esta función se utiliza para determinar qué bit de la memoria RAM de display se corresponde con un píxel determinado de la pantalla. Una vez dados la fila y la columna correspondientes, el algoritmo devuelve la dirección de memoria que contiene el píxel, así como el bit correspondiente, dentro de esta localización. En la Figura 9 se muestra el algoritmo correspondiente a dicha función.
La esencia de este algoritmo consiste en calcular la cantidad de píxeles existentes desde el primer píxel de la pantalla (superior izquierdo), hasta el píxel en cuestión, cuya posición se define como (Columna, fila). Posteriormente, se divide esa cantidad de píxeles entre el tamaño de una palabra en memoria (8 bits), para saber la cantidad de palabras (cantidad de posiciones o bytes en la RAM de display) que existen antes del byte en memoria que "contiene" el píxel buscado (este valor se corresponde con el cociente de la división). El resto de la división estaría indicando el bit que ocupa el píxel en el byte que lo contiene[1].

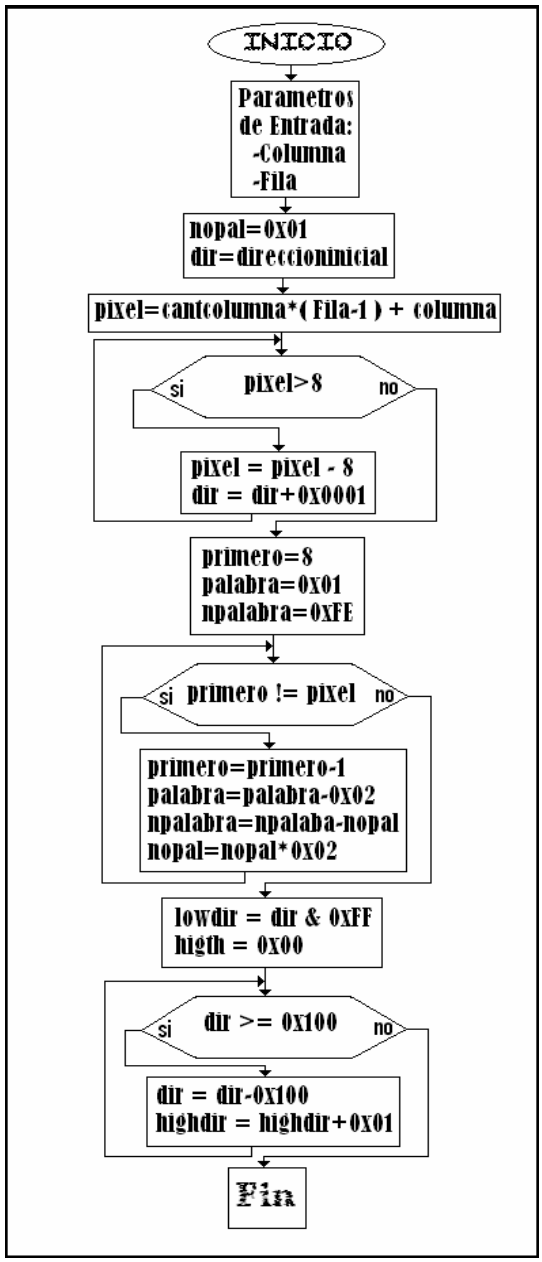

Fig. 9: Algoritmo para buscar la dirección de un píxel

- Dibujar Línea:

Esta función permite dibujar o borrar una línea, siempre de izquierda a derecha. En la Figura 10 se muestra el algoritmo correspondiente a dicha función. Este parte de las condiciones iniciales, que serían las coordenadas del píxel extremo izquierdo de la línea, con su par (columna, fila) característico, y las coordenadas del píxel final extremo derecho. Según estos puntos y la ecuación lineal de la recta, se calculan la pendiente (m) y el intercepto con el eje "y" (n). A partir de estos valores y teniendo en cuenta si la recta crece más en dirección horizontal o vertical, se recorre la misma mediante el incremento unitario de la "x" o de la "y" respectivamente, y finalmente se calcula el valor ideal de "y" o de "x" en cada caso.

Posteriormente se determina cuál es el píxel más próximo al ideal calculado (a cada píxel le corresponde un par de números enteros correspondientes a su coordenada: fila y columna, mientras que los resultados ideales calculados constituyen valores reales). Ante los pares de valores reales (x, y) calculados a partir de la ecuación de la recta, las coordenadas correspondientes de los píxeles se asumen como los valores enteros que más se aproximen a tales valores reales calculados[1].

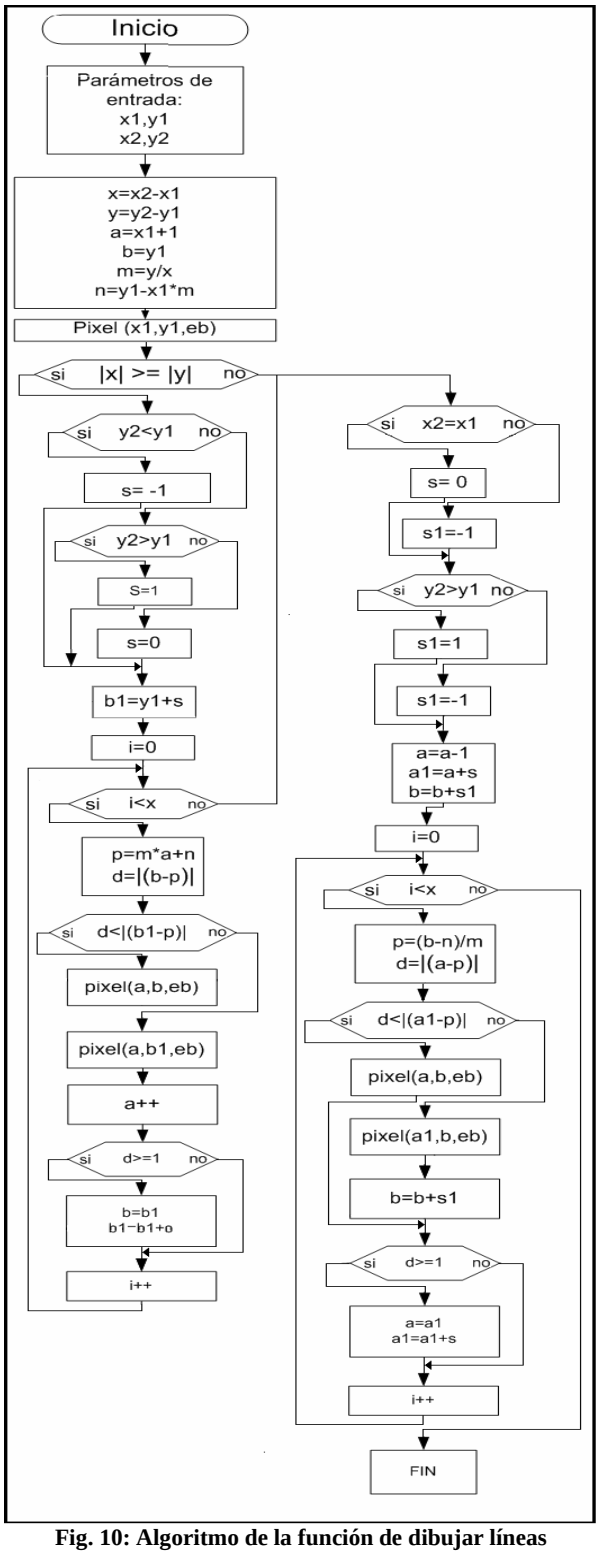

Fig. 10: Algoritmo de la función de dibujar líneas

3.4 VISUALIZACIONES DESARROLLADAS

Específicamente se diseñaron dos aplicaciones de visualización: una para simular el funcionamiento de un instrumento virtual (se le llamó “pantalla de gráfico”) y otra, con funcionalidades de un editor de texto (se le llamó “pantalla de texto”). Para acceder a estas aplicaciones se implementó una tercera pantalla, a la que se le llamó “pantalla de menú”.
En la Figura 11 se muestra la pantalla de menú obtenida. Como se puede observar, presenta dos objetos visuales: TEXTO y GRAFICO que dan acceso a las pantallas de texto y gráfico respectivamente. Una vez que se oprima la tecla “Enter”, se cambia a la pantalla que haya sido seleccionada.

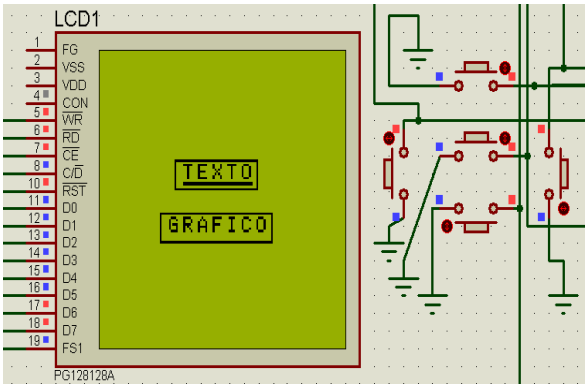

Fig. 11: Interfaz de la pantalla de menú

En la Figura 12 se muestra la pantalla de texto. Se puede observar que la misma está conformada por un área diseñada para escribir texto (el rectángulo de la parte superior de la pantalla), una emulación de teclado con caracteres y dígitos, y otras teclas con funciones especiales tales como espacio (SPACE), borrar (DEL), volver a la pantalla de menú (VOLVER) y cuatro flechas (arriba, abajo, derecha e izquierda) para desplazar el cursor por el área de edición de texto. Se muestra además un ejemplo de texto editado.

Fig. 12: Ejemplo de Pantalla de texto

En la Figura 13 se muestra un ejemplo de visualización de la pantalla de gráfico. Como se puede observar esta está conformada por un área para graficar (se estarán graficando las muestras de señal almacenadas en un bloque de memoria), cuatro cuadros de texto para que el usuario pueda seleccionar los extremos en los que se quiere visualizar la señal, un teclado con dígitos del 0 al 9 y la coma para poder entrar los valores decimales, dos flechas (arriba y abajo) para cambiar de cuadro de texto, un botón OK para ejecutar la visualización y un vínculo para volver a la pantalla de menú (VOLVER). Se ha mostrado en la Figura 13 la visualización de un coseno de amplitud 5 V y frecuencia de 2 Hz, en el intervalo de 0 s a 1 s (la frecuencia de muestreo se simuló a 110 Hz).

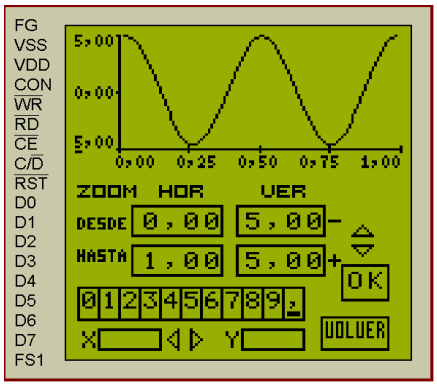

Fig. 13: Interfaz de la pantalla de gráfico

4. DESARROLLO DEL TRABAJO PRÁCTICO REAL

4.1DISEÑO DE LA APLICACIÓN

Primeramente fue necesario construir una interfaz con el empleo del circuito integrado MAX232[6] (ver Figura 14) para lograr la comunicación entre el microprocesador DS5000 y la PC por el puerto serie RS-232 de esta última[7]. En la Figura 15 se muestra la tarjeta construida al efecto.

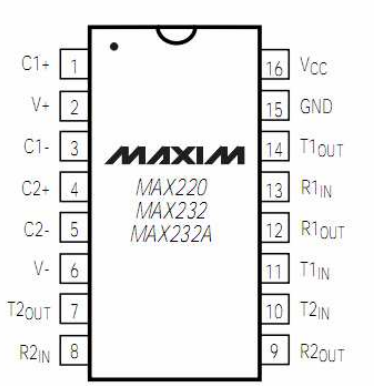

Fig. 14: Circuito integrado, Max 232

Las muestras de señales a visualizar fueron generadas en Matlab y enviadas vía RS232 desde la computadora al microcontrolador. Para esto se empleó la misma interfaz que se usó para cargarle los programas al DS5000. Se construyó además una tarjeta que contuviera las teclas en el mismo modo en que se utilizaron durante la etapa previa de simulación.
Se construyó una placa de desarrollo que consistió en cuatro buses, de 8 bits cada uno, colocados alrededor del microcontrolador de forma que todos quedaran alineados. Los pines P3.0 y P3.1 están conectados además a un conector de dos pines, que va la interfaz RS232 para la comunicación serie.
En esta placa se incluyó la configuración de cargado de programa en serie. La cual consistió en un interruptor doble donde en una posición se pone a Vcc el pin RST y a tierra PSEN para cargar el programa y en la segunda posición se invierte. Para energizar el circuito se colocó un interruptor al pin Vcc del DS5000. Se colocaron además dos LEDs para indicar cuándo el circuito está energizado (LED rojo) y para indicar cuándo está en modo de cargado de programa (LED verde). Para resetear el microcontrolador solo basta con cambiar el interruptor un momento a la posición de cargado de programa y luego regresarlo a su posición normal. Otra forma de resetearlo sería desconectar el circuito pero esto resetearía también el LCD que está conectado en paralelo a la placa.

En este diseño se tuvo en cuenta que el DS5000 trabaja con 5 V por lo que se decidió alimentarlo por el puerto USB (Universal Serial Bus) de la máquina. Por otra parte el LCD trabaja de igual modo con 5 V y recibe entonces alimentación desde esta placa que posee dos conectores extra de alimentación para posibles periféricos a conectar.

Para la interfaz de las teclas también se hizo una placa que consta de un bloque de resistencias Pull-up, pues van conectadas a Vcc, y los botones están conectados a tierra de modo que al presionar sobre alguno de ellos se impone un cero lógico en el pin de puerto al que están conectados.

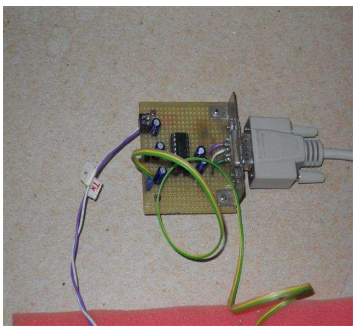

Fig. 15: Módulo de interfaz RS232

El sistema completo construido se puede observar en la Figura 16.

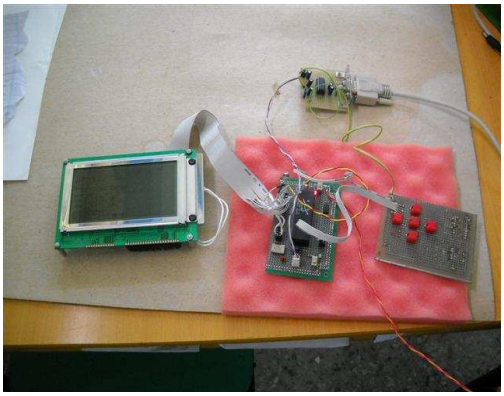

Fig. 16: Sistema LCD, microcontrolador, interfaz RS232, teclado

4.2 SOBRE LA PROGRAMACIÓN E IMPLEMENTACIÓN PRÁCTICA DE LOS ALGORITMOS

La programación de los algoritmos implementados en la práctica coincide con la de los algoritmos utilizados en la fase de simulación. Es decir, de manera general se utilizaron los mismos algoritmos y solo en algunos casos estos sufrieron pequeños cambios para optimizar el funcionamiento del sistema.

Para poder efectuar las visualizaciones se trabajó con arreglos de variables con punto flotante, que ocupaban mucho espacio en memoria, por lo que se almacenaban en la memoria externa. Trabajando la programación en C del microcontrolador, utilizando µVision, estos datos solo pueden ser accesados a través de variables tipo punteros por lo que se hizo necesario analizar el espacio de memoria disponible y realizar los cálculos para no solapar arreglos de datos. De este modo todas las variables de tipo flotante fueron almacenadas en memoria externa. Para el trabajo con la memoria se declararon los punteros apuntando a una dirección fija. Como los incrementos de los punteros van asociados generalmente a ciclos, y los ciclos a su vez tienen una variable que se va incrementando, entonces si se apunta a la suma del puntero y la variable que se incrementa se obtiene el mismo resultado que si se incrementa el puntero en sí, pero sin modificarlo. De esta forma se evita cargarle la dirección al puntero cada vez que se reinicie el ciclo. A continuación se explica mediante un ejemplo la idea anterior:

```
floatxdata *ptr = 0x7388;  // Arreglo de datos       desde 7388H-7544H
unsigned char a;
for(a=0;a<111;a++)
{    {
          scanf("%f",&*(ptr+a)); //Aquí se leen los datos y se van guardando
          printf(" %f\n",*(ptr+a)); //Aquí se envían desde la memoria externa
}
   }
```

4.3 RESULTADOS OBTENIDOS

En la figura 17 se muestra el resultado práctico de visualizar en el LCD la misma señal visualizada de forma simulada mostrada en la Figura 13. Las muestras de la señal fueron generadas desde Matlab y enviadas desde la PC al Microcontrolador. Se puede comparar además el resultado alcanzado con la visualización de la misma señal desde Matlab, mostrada en la Figura 18.

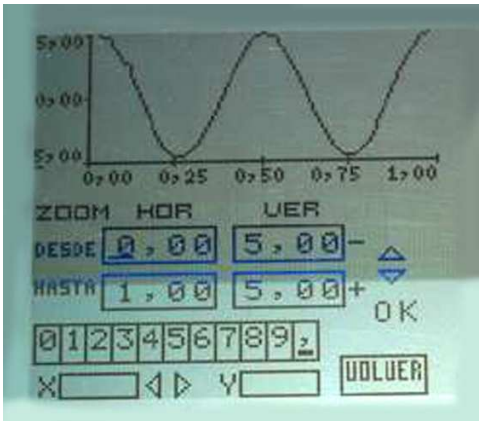

Fig. 17: Gráfico de coseno. Foto del LCD

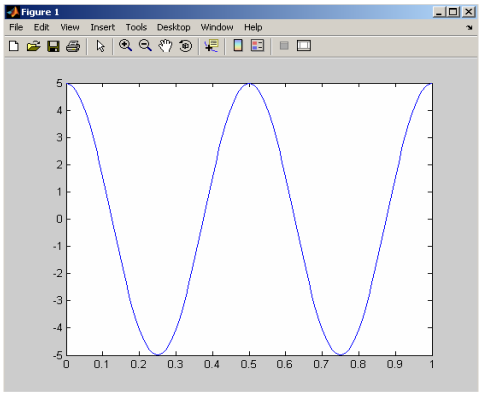

Fig. 18: Gráfico del coseno desde Matlab

En la Figura 19 se muestra un tren de pulsos cuadrados bipolar de 2 V pico a pico, en el que se puede observar la proporción de la amplitud respecto a la escala que va desde -5 V hasta 5 V.
En la Figura 20 se muestra el resultado que provoca realizar un acercamiento (zoom) al tren de pulsos cuadrados mostrado en la figura anterior entre los valores de amplitud de -1 V hasta 1 V.

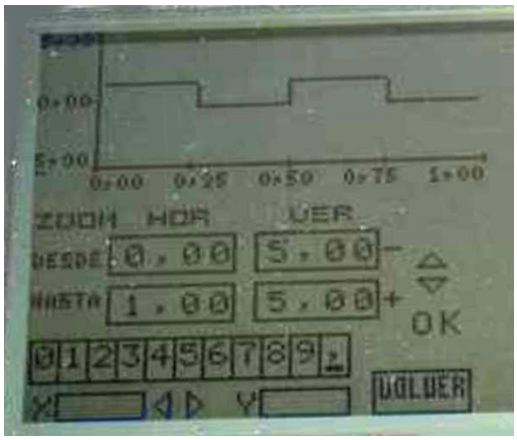

Fig. 19: Gráfico de un tren de pulsos cuadrados

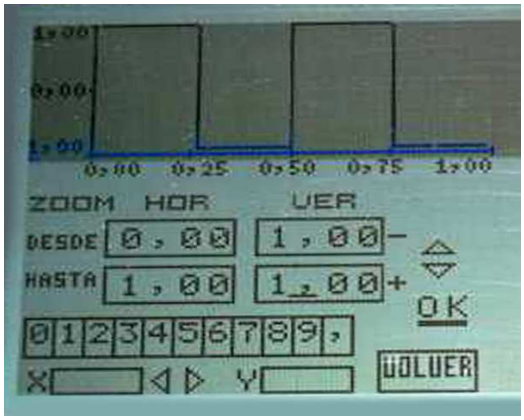

Fig. 20: Acercamiento gráfico del tren de pulsos cuadrados

Como se ha podido apreciar, los resultados prácticos alcanzados guardan una estrecha vinculación con los resultados simulados expuestos, y ambos gran similitud con las visualizaciones de las mismas señales en el software Matlab.

CONCLUSIONES

Mediante el empleo del módulo PG128240-A y el paquete de algoritmos de programación, se logró implementar un sistema, capaz de visualizar, de manera independiente, los resultados del procesamiento en Sistemas Digitales Autónomos, creándose las bases para el establecimiento de una plataforma de interacción con el usuario, que puede ser nombrada: Interfaz gráfica para sistemas electrónicos portátiles.

Las funciones gráficas más importantes que se lograron implementar de forma simulada fueron:

- Emulador de Editor de Texto: En una pantalla se visualizó teclado y área de edición, y a través de 5 teclas externas, se permitía editar cualquier texto en la pantalla.
- Visualizador de señales: En una pantalla se visualizaron las muestras, almacenadas en memoria, de una señal. Se permitió ejecutar la función de “zoom”.

De forma práctica se logró visualizar una pantalla que funciona como un osciloscopio digital en la que se visualizaron las muestras de una señal entradas por el puerto serie. Se logró además ejecutar la función de “zoom”.

REFERENCIAS

1. Pino Gómez, J. y Hernández Mauri, AM., "Interfaz gráfica para sistemas portátiles de medición" Tesis de Diploma (Ingeniero en Telecomunicaciones y Electrónica). Departamento de Telecomunicaciones. Facultad de Informática y Telecomunicaciones. Universidad de Pinar del Río Hermanos Saíz Montes de Oca, Pinar del Río, Cuba, 2008.
2. DatasheetPG 128128-A. POWERTIP. Disponible en: <http://www.powertipusa.com/pdf/pg128128a.pdf>. Fecha de consulta diciembre de 2011.
3. DatasheetT6963C. FINDCHIPS. Disponible en: <http://www.datasheetarchive.com>. Fecha de consultadiciembre de 2011.
4. DatasheetPG 128240-A. POWERTIP. Disponible en: < http://www.powertipusa.com >. Fecha de consulta diciembre de 2011.
5. DatasheetDS5000. FINDCHIPS. Disponible en: <http://www.datasheetarchive.com>. Fecha de consultadiciembre de 2011.
6. DatasheetMAX232. FINDCHIPS. Disponible en: <http://www.datasheetarchive.com>. Fecha de consultadiciembre de 2011.
7. Vento Ramos, Y., "Implementación práctica de una interfaz gráfica para sistemas portátiles de medición" Tesis de Diploma (Ingeniero en Telecomunicaciones y Electrónica). Departamento de Telecomunicaciones. Facultad de Informática y Telecomunicaciones. Universidad de Pinar del Río Hermanos Saíz Montes de Oca, Pinar del Río, Cuba, 2010.

AUTORES

Joel Pino Gómez, Ingeniero en Telecomunicaciones y Electrónica, Universidad de Pinar del Río, Pinar Del Río, Cuba, jpino@tele.upr.edu.cu.

Ailyn M. Hernández Mauri, Ingeniera en Telecomunicaciones y Electrónica,ETECSA, Pinar del Río, Cuba,jpino@tele.upr.edu.cu.

Yosvany Vento Ramos, Ingeniero en Telecomunicaciones y Electrónica,ETECSA, Pinar del Río, Cuba, yosvany.vento@etecsa.cu.

Fidel Ernesto Hernández Montero, Ingeniero en Telecomunicaciones y Electrónica, Doctor en Ciencias Técnicas, Universidad de Pinar del Río, Pinar del Río, Cuba, fidel@tele.upr.edu.cu.

Recibido: Diciembre 2011
Aprobado: Febrero 2012