TERMINAL PROGRAMABLE PARA CAPTURA DE DATOS

Rafael Rico López
Julio Cesar Jiménez García-Baquero
Juan Ignacio Pérez Sanz

Departamento de Automática
Universidad de Alcalá

*Se presenta aquí un terminal inteligente para captura de datos basado en el procesador 8088 que, además, es programable. Ha sido diseñado para trabajar en bus de campo y puede cargar un programa desde el computador anfitrión (*host*) que gobierna dicho bus.*

Es un terminal de bajo coste ya que se ha proyectado de forma modular y respetando la arquitectura software del PC con lo que las aplicaciones desarrolladas en este corren igualmente sobre nuestro diseño. Es decir, no son necesarias herramientas de desarrollo especiales.

1. SITUACIÓN ACTUAL

En la actualidad, se ha generalizado la utilización de sistemas de captura de datos extendidos en grandes áreas. Es lo que se conoce como buses de campo. En estos buses de campo, cuyo fin es interconectar puntos distantes, los nodos son inteligentes: por un lado se encargan de relacionarse con los demás nodos y con el computador anfitrión (*host*) mediante un protocolo adecuado, y por otro, realizan la adquisición de datos, almacenan temporalmente los mismos y poseen una cierta autonomía de trabajo (control de alarmas, aprendizaje, etc.).

El bus propiamente dicho, como medio de comunicación, puede ser una línea telefónica, una red *ethernet* o una línea serie. Este último ha sido el caso más habitual hasta hace muy poco, debido a la robustez, comprobada en el ambiente industrial, de la norma RS-485. Ahora, también encontramos en el mercado, módulos de adquisición remota para otro tipo de estándares de comunicaciones con el objeto de incorporar esos terminales inteligentes de captura de datos a redes corporativas de ordenadores. Se pretende que todo el sistema de comunicaciones de una organización (redes de gestión, redes de control de planta, *intranet* e *internet*) esté sustentado por la misma red.

Como venimos explicando en los párrafos precedentes, en muchas aplicaciones en las que se desea realizar la captura de datos en puntos

remotos conectados mediante lineas serie (bien RS-232 o bien RS-485) con un computador anfitrión, es conveniente disponer de nodos inteligentes capaces de hacer una mínima gestión de las transferencias de E/S. Una característica muy importante de estos nodos es que sean de bajo coste. Si el coste de estos terminales de captura de datos fuera excesivo podría convenir más su sustitución por un ordenador tipo PC u otro dispositivo más simple (aunque fuera menos inteligente).

El coste de estos terminales vendrá dado por una serie de factores: en primer lugar el coste de diseño, en segundo lugar el coste del desarrollo de aplicaciones concretas y en tercer lugar el coste de los componentes físicos y de su montaje.

La inteligencia de estos nodos suele restringirse a unas pocas opciones de trabajo (tipo de sensor, tipo de alarma, tipo de transferencia, etc.) que son configuradas por el computador anfitrión. En otras ocasiones, los nodos tienen una mayor capacidad de proceso y, en ese caso, los programas suelen ser más extensos y residir en memorias no volátiles. Este es el caso de terminales basados en microcontroladores convencionales o PIC. En ambos modelos la flexibilidad es muy pequeña.

2. OBJETIVOS

Nuestro objetivo es diseñar un terminal inteligente para captura de datos de bajo coste y que tenga una mayor flexibilidad de operación, es decir, que pueda correr un programa cargado remotamente desde el computador anfitrión.

En lo que al coste se refiere, de los tres factores enumerados anteriormente, nos quedaremos tan sólo con los dos primeros, ya que el último -coste de los componentes- es cada día más bajo para el común de los dispositivos electrónicos.

El coste de diseño viene fuertemente determinado por el tiempo invertido en depurar el proyecto, hasta conseguir el funcionamiento deseado, y por la cuantía de las herramientas necesarias para llevar a cabo dicha depuración. En este sentido, hemos procurado facilitar la depuración y simplificar las herramientas involucradas en ella ideando un diseño modular en el que cada uno de los módulos se puede probar por separado.

El coste de desarrollo de aplicaciones está íntimamente relacionado con la complejidad de las herramientas necesarias para generar los programas que serán cargados en el terminal y su disponibilidad y portabilidad. Para abaratar precios en este apartado se ha intentado que el terminal trabaje tal y como lo haría un PC desde el punto de vista de la arquitectura *software*. De esta manera un programa que corra en el PC lo hará en el terminal de la misma manera y, por tanto, las herramientas a utilizar serán las ya conocidas para un ordenador PC.

Para aumentar la versatilidad del terminal y mejorar sus prestaciones, el programa a ejecutar será cargado desde el computador anfitrión en el momento del arranque. Esta característica hace que nuestro terminal no sólo sea inteligente y capaz de trabajar en áreas extendidas, sino que puede adaptarse a diferentes entornos o situaciones en función de las necesidades de cada coyuntura.

Dado que buscamos una arquitectura *software* de PC, hemos elegido como procesador el micro 8088 y los dispositivos controladores de periféricos de su familia (8255, 8259, etc.).

3. ARQUITECTURA *HARDWARE* DEL TERMINAL

Como ya se ha indicado más arriba, se ha ideado un diseño modular fácil de depurar. Consta de tres partes: unidad central con el microprocesador, el reloj, la memoria y el controlador de interrupciones; controladores programables de E/S paralelo y serie; y finalmente, periféricos. Cada uno de los módulos está conectado con los otros a través de un bus de 40 líneas. Estos buses permiten la conexión de cada módulo entre si y de manera independiente con las señales del bus de expansión ISA de un PC a través de una placa de adaptación (placa de *buffers* PCL-750 de PC-LabCards). Así podemos probar el correcto funcionamiento de ellas, separadamente, desde el ordenador.

Figura 1. Estructura modular del terminal programable.

3.1. Unidad central

Esta placa esta formada por el microprocesador 8088 de Intel, la memoria, el reloj (8284) y el controlador de interrupciones programable (8259).

El procesador trabaja en modo mínimo de manera que genera todas las señales de control de bus y, por tanto, simplifica el diseño.

El mapa de memoria está compuesto de 64Kbytes de RAM y 64Kbytes de EPROM distribuidas tal y como lo están en un PC, es decir, la memoria no volátil a partir de la posición más alta, para que incluya la posición de arranque (FFFF0h), y la volátil a partir de la posición más baja.

La velocidad de reloj es de 3Mhz y es generada gracias al integrado 8284.

El controlador de interrupciones se ha situado en el mismo puerto que ocupa en el mapa de E/S del PC, es decir, en el 20h con lo que podemos programarlo exactamente con las mismas rutinas escritas para el PC.

En la figura 2 se muestra el esquema de conexiones de esta placa.

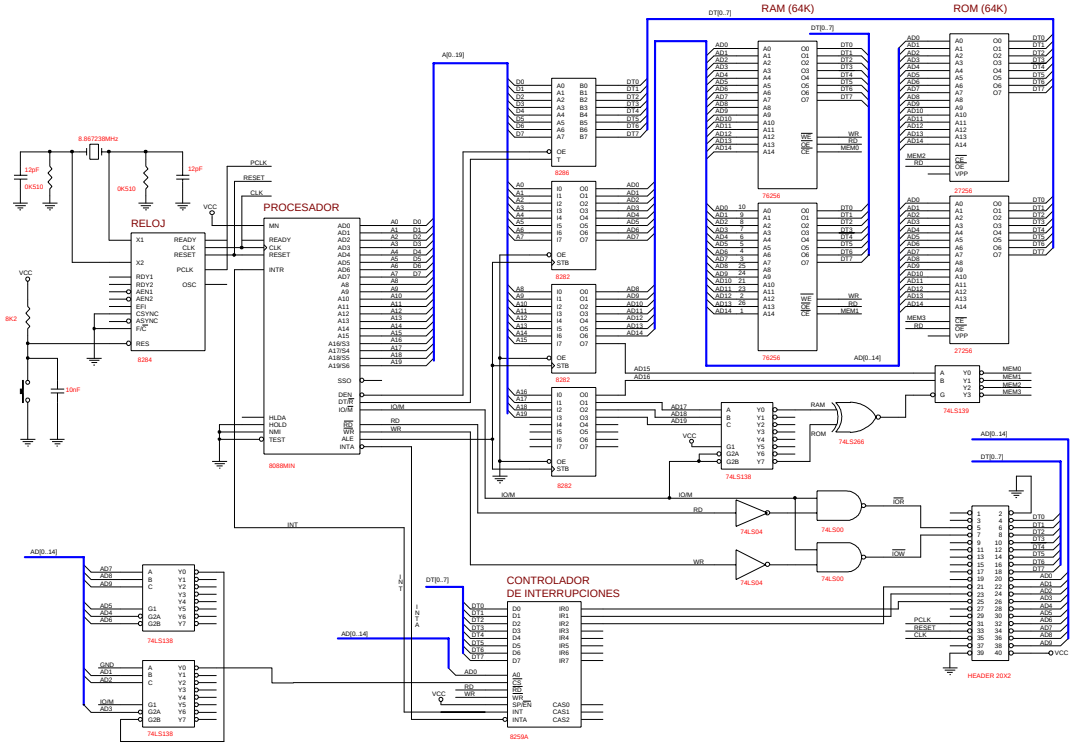

Figura 2. Esquema de la placa de la unidad central.

3.2. Placa de controladores

Esta placa contiene los controladores de periféricos tradicionales en la familia Intel. Tenemos dos USART 8251 para trabajar con dos canales serie, dos interfaces RS-232 para ambos canales serie, basados en los circuitos 1488 y 1489, y un 8255 para trabajar con entradas/salidas paralelas.

Los controladores 8251 se han seleccionado, en lugar de los 8250, ya que dan la posibilidad de realizar comunicaciones síncronas. En nuestro caso, la distribución es la siguiente: un canal serie sirve para comunicar el terminal con el computador anfitrión y el otro para captura de datos mediante un dispositivo que cuente con esta conexión (lector de tarjetas magnéticas, sonda o instrumento). El primero se denomina COM y se encuentra en el puerto 0200h mientras que al segundo se le ha asignado el nombre AUX y reside en 0304h. El COM cuenta con una serie de servicios de la BIOS y del sistema que permiten un fácil manejo.

El controlador de interface paralela programable (PPI 8255) reside en el puerto 0300h y se va a encargar de gobernar los dispositivos estándar de entrada y salida, es decir, el teclado y la pantalla LCD.

En total disponemos de cinco canales de entrada/salida, dos serie y tres paralelos, que podemos configurar de acuerdo a las especificaciones de nuestro sistema. Aquí, se le dan a los canales unos periféricos que sirven de ejemplo pero que pueden ser cambiados en función de las necesidades. Así pues, el interface con el computador anfitrión podría ser una línea RS-485, un bus *ethernet* o un enlace por radio y así sucesivamente con el resto de periféricos que podamos imaginar.

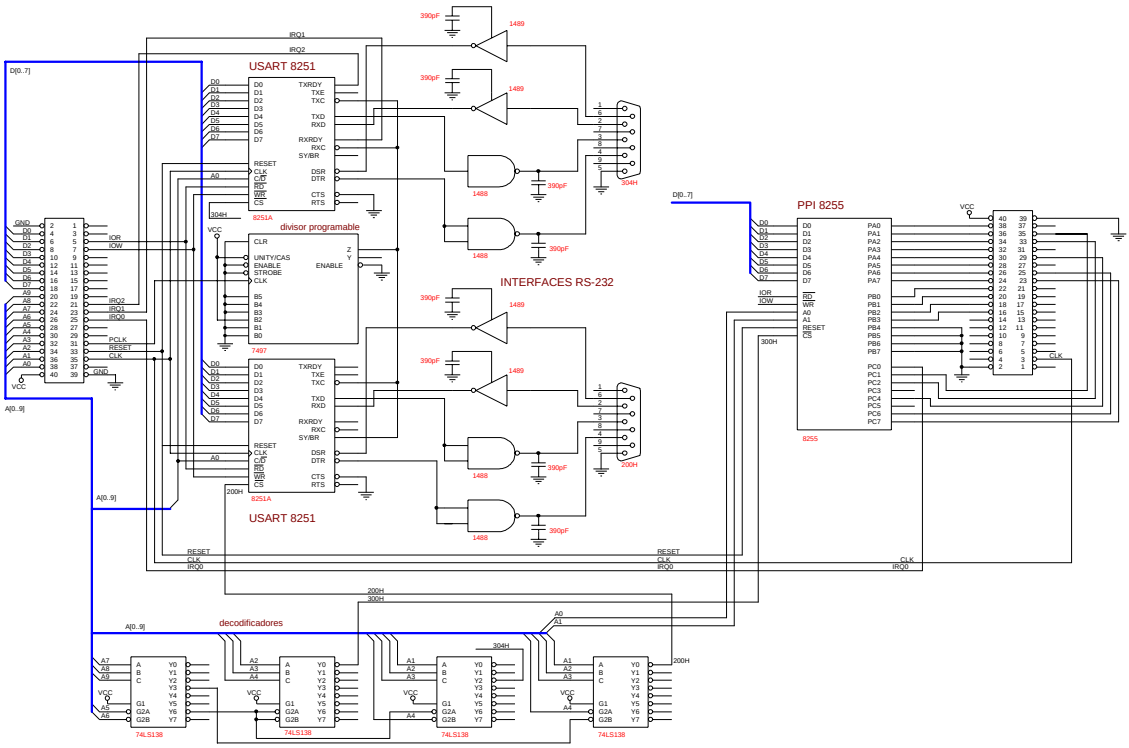

Figura 3. Esquema de la placa de controladores.

En la figura 3 se presenta el esquema de esta placa. Como se puede ver, se ha incluido también un divisor programable de frecuencia de 6 bits, el integrado 7497.

3.3. Placa de periféricos

La placa de periféricos contendrá todos aquellos que sean necesarios para nuestro sistema. En el caso que usamos de ejemplo, cuenta con una pantalla LCD de 2 líneas por 16 caracteres y un teclado de 16 teclas. Ambos dispositivos se relacionan con el controlador PPI 8255 de la placa anterior. En el caso de la pantalla directamente y en el caso del teclado mediante un integrado 74922, que actua de codificador y sirve para disparar una interrupción cada vez que se pulsa una tecla.

La pantalla y el teclado van a permitir que el usuario pueda determinar que tipo de programa se va a cargar en el terminal y la configuración de

algunas de sus funciones. En un terminal de uso normal no sería necesario incluir estos dispositivos y si fuera necesario algún tipo de configuración esta se podría realizar desde un conmutador *microswitch*. Nosotros lo hemos incluido de cara a comprobar las posibilidades del diseño.

En la figura 4 podemos observar cómo se ha realizado el conexionado de esta parte.

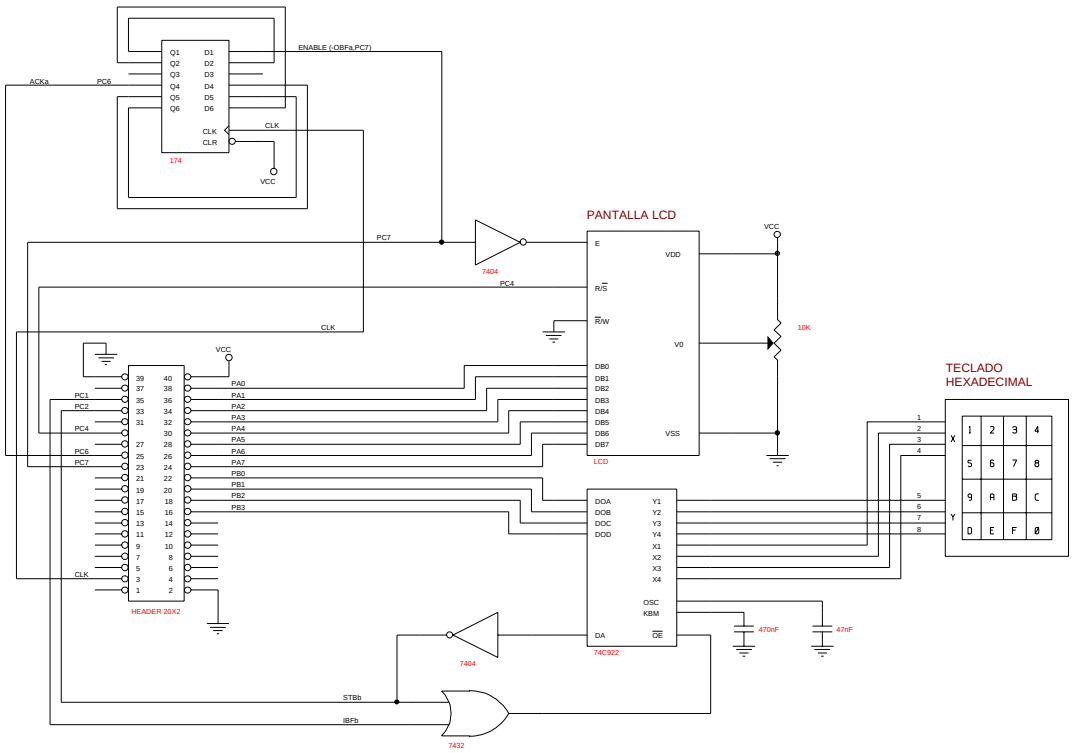

Figura 4. Esquema de la placa de periféricos.

4. ARQUITECTURA *SOFTWARE* DEL TERMINAL

Como ya hemos señalado, se ha procurado que la arquitectura *software* del terminal programable sea la misma que la del PC con objeto de simplificar el desarrollo de aplicaciones concretas para el mismo. A la hora de referirnos a la arquitectura *software* podemos hacer una distinción entre el entorno de E/S y los servicio del sistema operativo.

4.1. Entorno de E/S

Por entorno de E/S queremos significar el mapa de puertos de E/S que utilizamos y los controladores que manejamos. Para que el terminal que hemos construido reproduzca el entono del PC ha de utilizar un mapa similar y el mismo modo de programación de los controladores. Esto último lo conseguimos haciendo que sean los de la familia Intel. En lo que al mapa de E/S se refiere, el controlador de interrupciones se ha situado en el mismo

puerto de la placa base (20h) que se le asigna en el PC y el resto de controladores se ha situado en el rango de puertos reservado a los periféricos que se conectan a través del bus de expansión.

4.2. Servicios del sistema operativo

En el PC, como en cualquier otro computador, muchas de las E/S se realizan gracias a los servicios del sistema operativo en lugar de mediante accesos directos a los puertos. La misión del sistema operativo es generar una máquina virtual que permita al usuario manejar el *hardware* sin conocer en profundidad las características del mismo. En el caso de nuestro terminal, esta función del sistema operativo no es fundamental puesto que es muy sencillo y podemos trabajar directamente con las peculiaridades de su arquitectura, sin embargo, hemos optado por seguir utilizando los servicios del sistema operativo D.O.S. y de la B.I.O.S. para hacer portables las aplicaciones desarrolladas en el entono PC.

En este sentido, se trabaja con la misma lista de interrupciones que tiene un PC y se construye en memoria la tabla de vectores de interrupción a partir de la posición más baja de memoria. Las rutinas de los servicios han sido escritas para la peculiar arquitectura de nuestro terminal pero se llaman de la misma manera que en el PC. En la tabla adjunta se da la lista de interrupciones y servicios (parámetro AH) que constituye el reducido sistema operativo del terminal. Las rutinas de servicio a estas interrupciones se encuantran grabadas en memoria ROM.

interrupción	tipo	servicios	descripción
0h	CPU	-	- división por cero; devuelve el control al programa montador
8h	IRQ 0	-	- se almacena el valor de la tecla pulsada en la posición de memoria reservada al efecto
9h	IRQ 1	-	- recibe un byte por COM y lo escribe en la posición de memoria reservada; también almacena el estado
Ah	IRQ 2	-	- escribe un byte en COM desde la posición de memoria reservada al efecto
10h	BIOS	06h 07h 09h y 0Ah 0Eh	- borra la pantalla LCD mediante *scroll* hacia arriba - borra la pantalla LCD mediante *scroll* hacia abajo - escribe caracter de AL las veces que indique CX en la posición del cursor - escribe caracter de AL una sola vez y avanza el cursor
11h	BIOS	-	- devuelve información (en AX) acerca de los dispositivos instalados en el terminal
12h	BIOS	-	- devuelve el tamaño (en AX) de la memoria en unidades de 1Kbyte

14h	BIOS	00h 01h 02h 03h	- inicializa el puerto serie COM - escribe un byte por COM desde AL - recibe un byte por COM en AL y el estado en AH - obtiene el estado de puerto (AH) y módem (AL)
16h	BIOS	00h y 10h 01h y 11h 05h	- espera pulsación y la escribe sobre AL - devuelve en AL la tecla pulsada y vacia el *buffer* - escribe en el *buffer* del teclado como si se hubiera pulsado una tecla
19h	BIOS	-	- *reboot* saltando a la posición de arranque
21h	sistema	01h 02h 08h 09h 0Bh 0Ch 25h 35h 4Ch	- entrada desde el teclado con eco - muestra un caracter por pantalla - entrada desde el teclado sin eco - imprime una cadena por pantalla - comprueba el estado de la entrada estándar - borra el *buffer* del teclado - activa un vector de interrupción - obtiene la dirección de una rutina de tratamiento - devuelve el control al programa montador cuando finaliza la ejecución del programa cargado
29h	sistema	-	- escribe el caracter AL en pantalla llamando a los servicios de la interrupción 10h, 9h
78h	BIOS	-	- enmascara y desenmascara interrupciones
resto	-	-	- STI e IRET

Tabla 1. Listado de interrupciones y servicios del terminal programable.

El programa de arranque, tras finalizar la inicialización y configuración del terminal, sería el equivalente al montador de cualquier sistema operativo de características estándar, ya que se ocupa de cargar el programa en memoria desde el computador anfitrión y transferirle el control.

Dadas las características *hardware*, ya descritas, del terminal, especialmente su memoria RAM de 64Kbytes, los programas que puede ejecutar son del tipo COM. Estos programas no soportan la división de memoria en segmentos y, por tanto, solo ocupan un espacio máximo de 64K en el cual reside el código y la pila. Por este motivo, el programador ha de tener cuidado a la hora del dimensionamiento de la pila en razón del espacio necesario para el código.

5. MODO DE OPERACIÓN

Después de un *reset* o en el momento de alimentar el terminal, el microprocesador empieza a ejecutar el programa escrito en ROM a partir de la posición de arranque (FFFF0h). Este programa se encarga de escribir la tabla de vectores de interrupción, las tablas de variables y mensajes, de configurar los dispositivos programables (controladores de periféricos), de comprobar el correcto funcionamiento del canal de comunicación con el computador anfitrión y de solicitarle un ejecutable para ser cargado en el área de memoria RAM. La

solicitud de ejecutable puede hacerse directamente o ser seleccionada por menú desde el teclado. En cualquier caso, el terminal pasará a correr un programa proveniente del exterior y que puede ser distinto en función de las necesidades de cada momento.

El ejecutable es cargado remotamente bajo el control de un programa de comunicaciones con control de errores, etc., que recibe dicho ejecutable tipo COM sin la cabecera asignada por el D.O.S. Una vez cargado el programa en la memoria volátil del terminal inteligente a partir de la posición 0600h, todo el control del mismo lo tiene esa aplicación. Las características de funcionamiento serán aquellas que resulten de las especificaciones con las que se diseñó el *software*.

Dado que tenemos un computador completo -aunque de recursos limitados- las posibles aplicaciones pueden abarcar innumerables aspectos: adquisición de datos, alarmas, tratamiento de datos, formateado de datos, volcado de datos en el computador anfitrión por sondeo, por temporización o por interrupción, control de máquinas, etc. Y todo ello sobre la base de cualquier protocolo de comunicaciones que se desee implementar.

6. CONECTIVIDAD

Este terminal programable admite, como ya se ha señalado, cualquier interface con el computador anfitrión: buses serie RS-232 o RS-485, buses *ethernet*, líneas paralelo, radioenlaces, etc. Solo es necesario utilizar el hardware adecuado como E/S para disponer del nivel físico requerido.

En cuanto a los protocolos, podemos utilizar igualmente aquel que queramos ya que es tan solo un problema de programación de la memoria ROM del terminal. En el dispositivo que nosotros hemos fabricado, la memoria no volátil está dividida en dos pastillas de 32Kbytes cada una. Pues bien, una de ellas contiene el programa de arranque con las rutinas de inicialización y las de configuración del equipo (la de posiciones más altas) y la otra contiene el programa de comunicaciones con el computador anfitrión y los servicios de atención a las interrupciones. De esta manera, sustituir un programa de comunicaciones por otro con protocolos nuevos será cambiar una sola de las pastillas de memoria ROM.

En resumidas cuentas, el terminal admite cualquier tipo de enlace tanto a nivel físico como a nivel de protocolos.

7. CONCLUSIONES

Una de las grandes cualidades de este terminal es su versatilidad dado que puede resolver problemas de conectividad de cualquier tipo y correr aplicaciones de lo más variado.

Otra de sus ventajas es el bajo coste tanto de diseño como de desarrollo de aplicaciones debido a su arquitectura *hardware* y *software*.

Estas dos características son las que lo hacen interesante. No podemos, sin embargo, recomendarlo cuando el requerimiento más importante es el reducido tamaño. Para este caso, sería mejor utilizar microcontroladores PIC u otros dispositivos.

8. BIBLIOGRAFIA

- JIMÉNEZ GARCÍA-BAQUERO, JULIO CESAR. *TFC: Terminal inteligente basado en microprocesador*. Escuela Politécnica de la Universidad de Alcalá de Henares, septiembre 1997.
- RODRIGUEZ ROSELLÓ, MIGUEL ÁNGEL. *8088-8086-8087 Programación ensamblador en entorno MS DOS*. Anaya Multimedia, 1987.
- LIU, YU-CHENG Y GIBSON, GLENN A. *Arquitectura, programación y diseño de sistemas en microprocesadores 8086/80186/80286*. Anaya Multimedia, 1990.
- MESSMER, HANS-PETER. *The indispensable PC hardware book*. Addison-Wesley, 1995.
- EGGEBRECHT, LEWIS C. *Interfacing to the IBM personal computer*. Howard W. Sams & Co., 1983.
- FRUTOS REDONDO, JOSÉ A. DE Y RICO LÓPEZ, RAFAEL. *Arquitectura de computadores*. Servicio de Publicaciones de la Universidad de Alcalá, 1995.