ET1032 Informática Industrial
5 de junio de 2013
Soluciones a las preguntas

7 de junio de 2013

1. Un usuario desea copiar en el disco duro de su ordenador un archivo que ha guardado en un dispositivo de almacenamiento USB. Describid con todo el detalle posible las componentes de un sistema informático, tanto físicas (*hardware*) como lógicas (*software*) que intervienen y qué función desempeña cada una de ellas, desde que se inserta el dispositivo USB hasta que el archivo se termina de copiar en el disco duro. **(2 puntos)**.

   Inicialmente, mediante un mecanismo eléctrico, el subsitema físico USB –previsiblemente una entrada de un HUB conectado al controlador raíz– detecta que se ha introducido un dispositivo en el conector. A partir de este momento, el gestor de entrada/salida del sistema operativo, respondiendo a una interrupción del subsistema USB, se encarga de negociar con el dispositivo para identificarlo y asignarle recursos, a través del bus USB. Al detectar que se trata de un dispositivo de almacenamiento entraría en funcionamiento el gestor de sistemas de ficheros, parte también del sistema operativo, que lo analizaría y leería cierta información.

   Una vez se ha integrado este dispositivo en el sistema, la interfaz de comandos del usuario –relacionada con el sistema operativo aunque se ejecute a nivel de usuario– posiblemente avisaría al usuario de la presencia de un nuevo dispositivo de almacenamiento. Entonces el usuario, mediante esta interfaz de comandos, seleccionaría el archivo origen y el directorio destino –para esta navegación se realizarían llamadas al sistema operativo relacionadas con la gestión de sistemas de ficheros– y solicitaría al sistema la copia. El código encargado de la copia suele ser parte de la interfaz de comandos de la que ya se ha hablado.

   Durante el proceso de copia se realizarían peticiones frecuentes al gestor de sistemas de ficheros del sistema operativo para leer datos de un medio –propiamente del sistema de ficheros implantado en el medio–, copiarlos en memoria y escribirlos en el otro medio. En la lectura intervendría el gestor de entrada/salida que se comunicaría con el bus USB y este con el dispositivo USB; para la escritura, de nuevo el sistema gestor de entrada/salida se comunicaría con la controladora del disco duro para que ésta realizara la copia de la información sobre el medio magnético.

   Una vez la controladora acabara de actualizar la información en el medio –no sólo datos sino también enlaces, bloques y toda la metainformación propia del sistema de ficheros– informaría al gestor de sistemas de ficheros que a su vez informaría al programa de copia que a su vez informaría al usuario.

   Por supuesto, para la ejecución de todo el código, tanto del sistema operativo como del usuario, intervendrían el procesador y la memoria del sistema aunque no se haya indicado explícitamente.

2. Se desea diseñar un sencillo sistema empotrado con un microcontrolador PIC18 que regule la intensidad lumínica de una bombilla de incandescencia de 220V en alterna y unos 100W de potencia. El sistema dispondrá de dos pulsadores, uno para aumentar y otro para disminuir la intensidad lumínica. Además estos pulsadores servirán, respectivamente, para encender la bombilla con intensidad máxima y para apagarla totalmente. Una pulsación sostenida sobre uno de ellos provocará el efecto de incremento o decremento, mientras que una pulsación breve el de brillo máximo o apagado. El sistema dispondrá además de una entrada que permita verificar el estado de la bombilla y de un led que se encenderá en caso de que se detecte que está fundida.

   Proponed un esbozo de la electrónica del sistema y especificad con suficiente detalle un programa que se encargara de realizar las funciones indicadas. **(2 puntos)**.

   El esbozo de la electrónica aparece en otro documento. Nos referiremos a él a lo largo de las discusiones siguientes.

   Las conexiones de los pulsadores y el led son sencillas. Tal y como aparece en la figura 1, ambos pulsadores se unen mediante uno de sus contactos a un pin de entrada y a través de una resistencia de *pull-up* a Vcc; el otro contacto del pulsador va a masa. De esta forma, en ausencia de pulsación el pin detectará un 1 lógico y al pulsar el valor leído pasará a 0. El ánodo del led se conecta a un pin de salida a través de una resistencia para limitar la corriente y el cátodo va a tierra. De esta forma el led se enciende escribiendo un 1 en la salida.

   La conexión general para el control de la bombilla aparece en la figura 3.a. El control de intensidad se realizará mediante un triac gobernado por la señal `ACT`. El extremo libre de la bombilla se conectará a la fase de la alimentación alterna y entre el otro extremo y el triac se pondrá el detector de bombilla fundida. El triac cerrará el circuito conectando su extremo libre al neutro de la red. Como se puede ver, hay dos versiones del circuito de activación del triac y otras dos del detector de bombilla fundida. La decisión estriba en si se quiere aislar electricamente el circuito de 220V del circuito del microcontrolador o no. Analizaremos primero este segundo caso, por ser más sencillo y ofrecer algunas ventajas.

   Si no queremos aislar ambos circuitos, como aparece en la figura 2 el neutro de alterna se conectará a la masa del circuito de control. Eso nos permitirá, como se ve en la figura, mediante una resistencia de elevada impedancia y dos diodos de protección detectar mediante el micro los pasos por cero de la señal alterna –serán, con cierto error, aquéllos instantes en que el pin detecte un cambio de valor lógico a su entrada. En estas condiciones la activación del triac se haría mediante una simple resistencia (figura 3.b) y la detección de bombilla fundida mediante otra (figura 4.b), de baja resistencia y capaz de soportar corrientes elevadas, que permitiría detectar si circula corriente a su través –en ese caso, la caída de tensión en su extremo `A` con respecto al neutro podría ser leída como un valor analógico en el pin.

   Si ambos circuitos desean aislarse completamente a nivel eléctrico, no se conectaría la masa al neutro y los circuitos de activación del triac y detección de bombilla fundida cambiarían. El primero podría ser como el que aparece en la figura 3.c, utilizando un optotriac que atacara la entrada `ACT` del triac, mientras que el segundo podría utilizar un transformador o simplemente una bobina y un detector magnético –de efecto hall, por ejemplo– como aparece en la figura 4.a. Aunque no aparece en la figura, si se desea tener referencia de los pasos por cero de la señal alterna –algo necesario para controlar un triac– habría que poner de nuevo un dispositivo similar a este último, no incluido en la figura.

   De las dos opciones descritas se recomienda la primera, es decir aquélla en que existe contacto eléctrico entre el neutro y masa, por su sencillez.

   El programa que gestionara el sistema no sería muy diferente para ambos casos. Sin entrar en detalles de implementación –se podrían usar interrupciones o prueba de estado, por ejemplo,

cambiando drásticamente la implementación– la especificación sería la siguiente:

- **Pulsadores:** cuando se detecte una pulsación en alguno de los dos pulsadores se espera un pequeño lapso -algunos ms. Si al acabar este lapso el pulsador sigue activo, se incrementa o decrementa una variable que guarda la intensidad actual de la luz. El valor de dicha variable se mantendrá siempre entre 0 –luz apagada– y un cierto valor máximo. Por otra parte, si se detecta que el pulsador ya no está activo, directamente se lleva la variable a 0 o al valor máximo. No se ha especificado qué ocurre si se pulsan a la vez ambos pulsadores. Se podrían ignorar ambas pulsaciones o dar prioridad a uno de ellos –es decir, si uno está pulsado se ignora lo que ocurre con el otro.
- **Control de la iluminación:** en general el control de iluminación se suele hacer mediante PWM, adaptando el ciclo de trabajo al valor de la variable que guarda la intensidad actual de la luz. Sin embargo, el trabajar con corriente alterna plantea algunas dificultades para ello. Si los ciclos de PWM no se sincronizan con los de la corriente alterna, el brillo puede variar de forma extraña. En general, sin entrar en detalles de implementación, se debería actuar sobre el triac sincronizádose con los pasos por cero –dado que el triac se corta en ellos. A partir de aquí la variable de intensidad regula el ciclo de trabajo de nuestra señal de activación, cuya frecuencia deberá ser de 100 Hz para adaptarse a los semiciclos de la red eléctrica. Es decir, tras un paso por cero se activa el triac durante un tiempo que será de 10 ms en caso de intensidad máxima y el porcentaje correspondiente de este tiempo en otro caso.
- **Detección de bombilla fundida:** siempre que la bombilla no esté apagada se puede monitorizar la señal que indica si circula o no corriente a su través –entre los terminales A y B del sensor, como dijimos antes. Si tras algunos ciclos se ve que que el sensor no indica actividad, se encenderá el led de bombilla fundida.

3. Un ordenador sólo es capaz de ejecutar programas que contengan instrucciones máquina del repertorio propio de su unidad central de proceso. Sin embargo, a lo largo del curso los ejercicios de programación se han realizado escribiendo programas en un fichero de texto que se adecuaba a las especificaciones del lenguaje de programación C. Explicad con el mayor detalle posible todos los pasos, programas y ficheros intermedios que invervienen en el proceso de generación del archivo ejecutable que entiende un ordenador a partir del fichero de texto que contiene las fuentes del programa creado por el programador de la aplicación. **(2 puntos)**.

   Efectivamente, para poder utilizar lenguajes de alto nivel y no tener que aprender el lenguaje ensamblador propio de cada procesador, se necesita un proceso de compilación y montaje –*link*– para acabar generando un programa ejecutable a partir de su código fuente. En detalle, los pasos y programas necesarios serían los siguientes:

   - Una vez diseñado el programa y habiendo esbozado al menos su código, se utilizaría un editor de textos para crear los ficheros con el código fuente. Si se está programando en lenguaje C estos ficheros serían uno o varios archivos de código, con la extensión `.c` y uno o varios de cabezera con la extensión `.h`. En uno de los ficheros de código estaría la función de inicio de la ejecución, `main()`.
   - Una vez editados los ficheros fuente se invocaría al compilador. En el caso del lenguaje C aquél llamaría en primer lugar al preprocesador del lenguaje, que analizaría los ficheros de cabecera –tanto los que hemos escrito nosotros dentro del conjunto de archivos fuente, como los propios de las bibliotecas que vamos a utilizar y hemos incluido en nuestros fuentes mediante `#include <xxx.h>`– y realizaría las sustituciones de símbolos y macros,

analizaría las deficiones de tipos y prototipos, etcétera. Posteriormente se ejecutaría el compilador propiamente dicho que generaría el código objeto a partir de nuestros ficheros fuente. Este proceso generaría, para cada uno de los módulos fuente `.c` independientes, un fichero objeto –`.o`, `.obj`– que contrendía el código máquina generado a partir de las fuentes y tablas de síbolos con nombres de funciones y variables que no existen en el módulo y referencias a direcciones de memoria para completar más adelante.

- Por último el montador o *linker* se encargaría de crear el fichero ejecutable a partir de los diversos módulos objeto y el código de las bibliotecas estándar que hemos utilizado en nuestro programa. Este programa analiza las tablas de símbolos de los módulos objeto y extrae el código de las funciones invocadas bien de los ficheros objeto en que se encuentran –en el caso de utilizar varios ficheros fuente– bien de las bibliotecas utilizadas. Lo mismo hace con las variables, que en alguno de estos archivos –objeto o bibliotecas– deben estar definidas y con su espacio de memoria reservado. El montador además se ocupa de resolver las direcciones de memoria de todos los síbolos del sistema, bien mediante valores predefinidos, bien mediante referencias que, posteriormente, resolverá el cargador del sistema operativo cuando el archivo ejecutable sea llevado a memoria para crear un proceso.

Cada una de estas etapa analiza los ficheros a partir de los cuales trabaja e indica los errores o advertencias –`warnings`– oportunos en caso de econtrar alguna circunstancia señalable. Si el código del programa es correcto en lo que al lenguaje de programación respecta, pero es incorrecto en cuanto a su ejecución, se podrán producir errores con posterioridad en tiempo de ejecución que, normalmente, tratará el sistema operativo –cuando se trate de errores que puedan afectar al sistema y no de meras incorrecciones en el tratamiento de datos.

4. Responded de forma concisa y objetiva, justificando brevemente las respuestas, a las preguntas que aparecen a continuación. **(4 puntos, 0.8 por pregunta)**.

   a) Un sistema de ficheros basado en inodos utiliza para todas sus estructuras bloques de 4KB. Sabiendo que el inodo de un fichero contiene, además de otra información, 1KB de datos, 500 punteros directos y 12 punteros indirectos y que el tamaño de los punteros es de 4 bytes, indicad el tamaño máximo de los archivos que se pueden almacenar en dicho sistema.

   Dada la estructura propuesta sabemos que el inodo de por sí ya contendrá 1KB de datos. Como el tamaño de los bloques es de 4KB, cada puntero directo apuntará a esa cantidad de datos; como tenemos 500 el tamaño máximo de los datos apuntados por punteros directos será de 2000KB –no de 2MB pues un mega es 1024×1024 no 1000×1024.

   Por otra parte cada puntero indirecto apunta a un bloque de punteros; como el tamaño de cada puntero es de 4 bytes, en un bloque de 4KB tenemos 1024. De nuevo cada puntero apunta a un bloque de 4KB de datos, por lo que los 1024 punteros del bloque apuntado por uno indirecto permiten encontrar 4096KB de datos. Como tenemos 12 punteros indirectos el total serán 49152KB de datos.

   Sumando las tres partes, $49152KB + 2000KB + 1KB$ tenemos un tamaño máximo de fichero de $51153KB = 52380672B$.

   b) Poned un ejemplo de sistema para el cual el atributo más importante de la garantía de funcionamiento sea la fiabilidad y otro para el que lo sea la disponibilidad.

   La fiabilidad es propia de los sistemas sin reparación, que deben funcionar largos periodos de tiempo sin averías. Así pues, ejemplos clásicos son los sistemas embarcados en misiones espaciales no tripuladas. Por otra parte, un dispositivo tan sencillo como una bombilla, que se sustituye y no se repara, también tiene la fiabilidad como atributo más importante.

La disponibilidad, por otra parte, es el atributo más importante de los sistemas que deben estar casi siempre en disposición de prestar su función. Es decir, dispositivos que, si sufren una avería, puedan recuperarse –repararse o reconfigurarse– muy rapidamente para volver a prestar su servicio de inmediato. Esta característica es propia de los servidores informátios en general –servidores web, por ejemplo– de entre los que son ejemplos paradigmáticos los servidores de líneas aéreas o bancarios. Otros sistemas que participan de este atributo son ciertas máquinas autónomas, como las expendedoras automáticas, cajeros automáticos, etcétera.

c) El sistema informático que gestiona una máquina fotocopiadora se adapta perfectamente a la definición de sistema empotrado. Indicad las características de este tipo de sistemas señalando de qué manera participa de ellas el sistema propuesto.

Si revisamos las características de los sistemas empotrados que aparecen en la documentación de la asignatura, podemos ver que en cuanto a su función, el sistema informático está incorporado a la fotocopiadora, sin ser visible para el usuario, y contribuye a la función de aquélla. Su tarea es repetitiva y consiste en automatizar el proceso de realización de fotocopias: una vez configurados los parámetros de la copia y presionado el botón de inicio, desplaza el tubo de luz recorriendo el papel y genera la copia en la hoja destino. Adicionalmente detecta posibles errores.

En cuanto a su estructura, el conjunto de sensores –presencia de papel, estado de la cubierta, niveles de tinta, detección de atascos . . . – y de actuadores –luz, motores y mecanismos, salida de tinta o toner . . . – es complejo y específico; la interfaz de usuario es reducida –algunos pulsadores y botones y tal vez una pequeña pantalla– y, como se ha dicho antes, el programa que rige su funcionamiento es repetitivo y está contenido en el sistema.

d) A la vista de las siguientes declaraciones en lenguaje C, indicad, en el supuesto de que sean correctas, el tipo de datos de las siguientes expresiones: `v_mat`, `&v_mat` y `*v_mat`.

```
struct mat {
  int f, c;
  double *data;
};

struct mat v_mat[20];
```

Dado que `v_mat` es un puntero constante, es fácil darse cuenta de que `&v_mat` es una expresión incorrecta, ya que las constantes del lenguaje no residen en direcciones de memoria –no existe &1, por ejemplo. Una vez analizado esto, a la vista del código anterior `v_mat` es de tipo `struct mat *`, es decir, puntero a `struct mat` y `*v_mat` de tipo `struct mat`.

e) Comentad las ventajas e inconvenientes de los buses cuyo nivel físico se basa en conexiones punto a punto.

La principal ventaja es que los buses punto a punto aprovechan de manera óptima las características del medio de transmisión pues todos los parámetros eléctricos están perfectamente definidos, por la que la velocidad de transmisión puede ser máxima para el medio dado –incluyendo por supuesto los dispositivos a ambos extremos del medio.

La desventaja es la poca conectividad, dado que sólo se pueden comunicar dos dispositivos. Por ello, los buses de estas características necesitan dispositivos activos de encaminamiento de los mensajes –a un nivel superior– para ampliar esta conectividad.