PRACTICA 2

Planificador con prioridades

Tutor: Javier Fernández Muñoz

Objetivo de la práctica

Esta práctica permitirá al alumno familiarizarse con la utilización de planificadores con prioridades y con el uso de sistemas operativos de tiempo real como VxWorks y con micro controladores como Arduino

Descripción

Se trata de construir una tarjeta de sonido que reproduzca ficheros de audio modulados por impulsos codificados (PCM) utilizando un micro controlador Arduino que recibe los datos del fichero desde un PC con VxWorks que esta conectado por un puerto serie. El programa resultante deberá asegurar los tiempos de respuesta de las melodías.

Se deben implementar dos modelos de la practica. El primero permitirá la reproducción de ficheros PCM codificados con 4.000 muestras por segundo donde cada muestra será de 1 bit (0 ó 1). El segundo permitirá la reproducción de ficheros PCM codificados con 4.000 muestras por segundo donde cada muestra será de 8 bit (entre 0 y 255).

Cada modulo de la practica se divide en dos partes: la implementación del hardware con del programa que controla el Arduino para reproducir la canción y el programa de Vxworks que se encarga de enviar el fichero de audio en los plazos de respuesta correspondientes:

Descripción de la modulación por impulsos codificados.

La modulación por impulsos codificados (MIC o PCM por sus siglas inglesas de Pulse Code Modulation) es un procedimiento de modulación utilizado para transformar una señal analógica en una secuencia de bits (señal digital), este método fue inventado por Alec Reeves en 1937. Una trama o stream PCM es una representación digital de una señal analógica en donde la magnitud de la onda analógica es tomada en intervalos uniformes (muestras), cada muestra puede tomar un conjunto finito de valores, los cuales se encuentran codificados, como se ve en la siguiente figura:

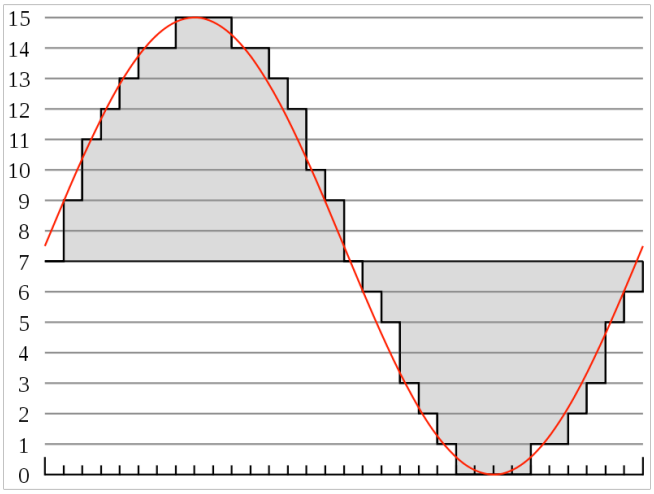


Ilustración 1: Onda sonora codificada con PCM

La creación de dichos ficheros se realiza mediante muestreo digital. Este consiste en tomar muestras (medidas) del valor de la señal n veces por segundo, con lo que tendrán n niveles de tensión en un segundo. Después del muestreo obtenemos la forma de onda.
Para un canal telefónico de voz, donde la frecuencia más alta transmitida es de 3,4 kHz (redondeando 4 KHz) es suficiente con tomar 8.000 muestras por segundo, o, lo que es lo mismo, una muestra cada 125 µseg. Esto es así porque, de acuerdo con el teorema de muestreo, si se toman muestras de una señal eléctrica continua a intervalos regulares y con una frecuencia doble a la frecuencia máxima que se quiera muestrear, dichas muestras contendrán toda la información necesaria para reconstruir la señal original.

Sin embargo para la practica utilizaremos ficheros con 4.000 muestras por segundos (una muestra cada 250 µseg) por problemas con la velocidad de emisión del puerto serie y otros temas similares. La reproducción resultante es suficientemente buena para los propósitos de esta práctica.

Otros tema importante es el rango de valores que puede alcanzar cada muestra. En sistemas más profesionales se suele utilizar valores de 16 bits (entre 0 y 2^16) pero es posible utilizar valores de 8 bits (entre 0 y 2^8) aunque la calidad de la reproducción no sea la mejor. Como ejemplo extremo utilizando el rango mas pequeño posible de 1 bit (entre 0 y 1) es posible reconocer la canción reproducida.

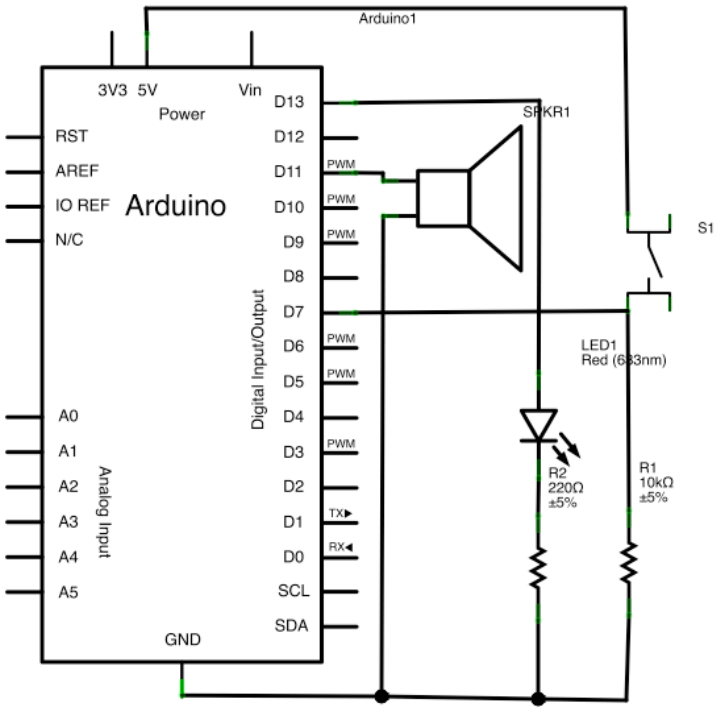
PARTE 1: Fichero codificado con 4.000 muestras de 1 bit cada una.

Descripción del modulo de Arduino

El modulo de Arduino tendrá que cumplir los siguientes requisitos

- Reproducir con un altavoz y muestra a muestra, un fichero PCM enviado desde el PC por el puerto serie a la máxima velocidad (115200 baudios). Para reproducir cada una de las muestras de 1 bit se utilizara la función displayWrite para escribir un 1 o un 0 en el pin de salida correspondiente. Este proceso se repetirá 4.000 veces por segundo.
- Activar o desactivar la funcionalidad de "mute". Para ello se dispondrá de un botón. Cuando se pulse el botón el sistema pasara de reproducción normal a "mute" (descartando la muestra que toca por un valor de 0, pero sin parar la reproducción) o viceversa, según corresponda.
- Mostrar con un LED cual es el modo de reproducción actual (reproducción normal o "mute". El LED encendido indicara que el "mute" esta activado.

El esquema del circuito necesario es el siguiente:



NOTA: Para poder reproducir a la velocidad requerida es necesario aumentar el tamaño del buffer del puerto serie en Arduino. Para ello en el fichero "HardwareSerial.cpp" que se encuentra en el directorio "arduino-1.0/hardware/arduino/cores/arduino" hay que cambiar el valor del la constante SERIAL_BUFFER_SIZE a 512.

El Arduino deberá realizar las siguientes tareas:

Tarea 1: Reproducir las canciones que se le suministren:

El formato de la canción será el siguiente:

- Formato PCM
- 1 canal (Mono)
- Muestreo: 4000 Hz (muestras por segundo)
- 1 bit por muestra (cada byte incluirá 8 muestras)

Para reproducir una muestra es necesario obtener un byte del fichero (leyendo el puerto serie) y separar las 8 muestras para reproducirlas por separado. Para ellos se utiliza la siguiente línea:

```
digitalWrite(soundPin, (data & mask) );
```

Donde la mascara es un byte con todos 0 menos un 1 en la posición de la muestra (mask = 2^pos). Al aplicar la operación AND a nivel de bit nos queda un valor igual a cero si la muestra era 0 y un valor mayor que cero si la muestra era 1.

El algoritmo a implementar es el siguiente (debe repetirse cada 250 μseg):

- Si la mascara corresponde al 1º bit.
  - Leer un byte del puerto serie
- Reproducir el bit indicado del último byte leído usando la mascara.
- Cambiar la mascara por la que corresponde al siguiente bit.

Tarea 2: Leer el botón y cambiar el modo de reproducción:

Esta tarea deberá muestra N veces por segundo si el botón esta pulsado o no (el alumno deberá elegir el valor de N, pero se recomienda al menos 10 veces por segundo para evitar los ruidos debidos a la pulsación).

En caso que el estado del botón pase de pulsado a no pulsado, el modo de reproducción del sistema deberá cambiar de reproducción normal a "mute" o viceversa. (Recordar que "mute" implica cambiar el valor de la muestra actual por cero pero se siguen leyendo los bytes del puerto serie para que no se sature.)

Tarea 2: Mostrar con un LED el modo de reproducción:

Esta tarea deberá encender el LED si el sistema esta en modo "mute" o apagarlo en caso contrario. El alumno deberá elegir cuantas veces por segundo se ejecuta esta tarea.

Se pide:

1. Diseñar un planificador (cíclico o por prioridades) que asegure que los plazos de las tareas se cumplirán, incluyendo las interacciones entre ellas.
2. Construir un programa para Arduino que implemente el diseño anterior y que realice las tareas antes especificadas junto con el hardware especificado.

CONSEJO: En lugar de utilizar un planificador cíclico (que será muy largo por la diferencia de periodos), se recomienda implementar la tarea 1 con una interrupción hecha con el Timer 1 y juntar las tareas 2 y 3 en una (eligiendo el mismo periodo) para así poder utilizar el mismo tipo de análisis que se aplica a un planificador con prioridades.

Descripción del modulo de Vxworks

El PC con Vxworks deberá realizar las siguientes tareas:

Tarea 1: Leer el fichero de audio y mandarlo al puerto serie:

Esta tarea deberá leer 256 bytes del fichero de audio correspondiente (que ha sido abierto antes) y enviarlos al Arduino escribiéndolos por el puerto serie (que también ha sido abierto antes y configurado a 115200 baudios)

Esta tarea debe repetirse cada 512 milisegundos (aprox. dos veces por segundo).

Tarea 2: Recibir órdenes del usuario para parar o continuar la reproducción.

La tarea deberá recibir órdenes a través de la función `unblockRead` (que es capaz de detectar si se ha pulsado o no una tecla sin bloquearse) para evitar bloquea el thread. Este recogerá del teclado los comandos Pause y Reanudar

- Comando Pause: Se enviará mediante el carácter '0'. Cuando se envié, la tarea de reproducción deberá cesar la reproducción de la canción en curso (en lugar de la música se envían ceros y se deja de leer el fichero original).

- Comando Reanudar: Se enviará mediante el carácter '1'. Cuando se envié, la tarea de reproducción deberá continuar la reproducción de la canción en curso.

La tarea deberá comprobar la existencia de una orden cada 2 segundos y ejecutarla en caso de que exista.

Tarea 3: Mostrar el estado del reproductor.

La tarea deberá mostrar por la consola el estado del reproductor, para ello deberá escribir los siguientes mensajes:

- "**Reproducción en pausa**" Cuando la reproducción se encuentre detenida.
- "**Reproducción en marcha**" Cuando la reproducción prosiga.

La tarea deberá mostrar el estado cada 5 segundos.

Se pide:

3. Diseñar un planificador por prioridades que asegure que los plazos de las tareas se cumplirán, incluyendo las interacciones entre ellas.
4. Construir un programa para VX-Works que implemente el diseño anterior y que realice las tareas antes especificadas.

PARTE 2: Fichero codificado con 4.000 muestras de 8 bit cada una.

Descripción del modulo de Arduino

La implementación del modulo de Arduino es idéntica a la de la parte 1 salvo la tarea 1 que debe modificarse de la siguiente forma:

Tarea 1: Reproducir las canciones que se le suministren:

El formato de la canción será el siguiente:

- Formato PCM
- 1 canal (Mono)
- Muestreo: 4000 Hz (muestras por segundo)
- 8 bit por muestra (cada byte incluirá 1 muestra)

Dado el funcionamiento del altavoz y que solo tenemos señales de 0 o 5 voltios para reproducir muestras de 256 niveles de potencia es necesario emitir una señal cuya frecuencia sean lo mas altas posible variando el ciclo de actividad, según la intensidad que queramos dar. Para ello es necesario:

- Configurar el Timer 2 para general una señal (PWM) por el pin 11. La configuración necesario será la siguiente:
  - Modo: Fast PWM (bits WGM2 = 011)
  - Preescalado: el mas pequeño (bits CS2 = 001)
  - Configuración salida A: Salida no invertida.
    (bits COM2A = 10)
  - Ciclo de actividad: Al registro OCR2A se asignará el valor de cada muestra en cada momento.

El algoritmo a implementar es el siguiente (debe repetirse cada 250 μseg):

- Leer un byte del puerto serie
- Asignar a OCR2A el byte

Descripción del modulo de Vxworks

La implementación del modulo de Vxworks es idéntica a la de la parte 1 salvo la tarea 1 que debe modificarse de la siguiente forma:

Tarea 1: Leer el fichero de audio y mandarlo al puerto serie:

Esta tarea deberá leer 256 bytes del fichero de audio correspondiente (que ha sido abierto antes) y enviarlos al Arduino escribiéndolos por el puerto serie (que también ha sido abierto antes y configurado a 115200 baudios)

Esta tarea debe repetirse cada 64 milisegundos (aprox. 15 veces por segundo).

Recomendaciones Generales

Realice primero el diseño de los planificador incluyendo la forma de compartir el estado del sistema mediante variables compartidas. Utilice las técnicas que suministra VxWorks (RMS, herencia de prioridades, techo de prioridades, etc.) y Arduino (interrupciones, mascara global de interrupciones, etc.)

Para realizar el programa se suministrará un código de apoyo en el fichero ***practica_2_2020-v1*** el cual consta de los siguientes ficheros:

`music_console.c`	-> FICHERO A MODIFICAR DE VxWORKS. Incluye un código de ejemplo para enviar los datos del fichero al arduino.
`Sketch/sketch.ino`	-> FICHERO A MODIFICAR DE ARDUINO. Incluye un código de ejemplo para reproducir audio a 4000Hz con muestras de 1 bit.
`Let_it_be_1bit.raw`	-> Fichero de sonido de muestra (4000 hz, 1 bit por muestra).
`Let_it_be.raw`	-> Fichero de sonido de muestra (4000 hz, 8 bits por muestra).
`extras`	-> Directorio con los ficheros necesarios para probar cada parte por separado. Mirar Apendices.

NOTA: Para poder configurar el sistema, es necesario copiar el fichero de musica (let_it_be_1bit.raw) al directorio C:\Documents and Settings\str\Escritorio (o modificar la constante FILE_NAME en el fichero music_console.c).

Documentación a Entregar .

Para entregar la práctica deberá ejecutarse el entregador Web situado en la página de la signatura (http://www.arcos.inf.uc3m.es/~ii_str) en el apartado entrega de practicas. En caso de que se entregue la práctica varias veces sólo será válida la última entrega.

Los ficheros a entregar son los siguientes:

`memoria.txt | memoria.pdf | memoria.doc`

Memoria de la práctica. Debe incluir el diseño del planificador.

`music_console_parte1.c`

Código fuente del modulo VxWorks para la parte 1.

`music_console_parte2.c`

Código fuente del modulo VxWorks para la parte 2.

`sketch_parte1.ino | sketch_parte1.pde`

Código fuente del modulo de Arduino para la parte 1.

`sketch_parte2.ino | sketch_parte2.pde`

Código fuente del modulo de Arduino para la parte 2.

Plazo de Entrega.

Viernes 13 de Diciembre de 2019.