Práctica 1: Desarrollo de un sistema de sensado y actuación controlado por USB. Segunda parte

1. Objetivos

En este documento se recogen las especificaciones correspondientes a la segunda fase del proyecto práctico. En sucesivos documentos se irán proponiendo nuevas especificaciones para completar la funcionalidad del sistema.

En esta segunda parte se propone trabajar con un periférico, el conversor analógico-digital (ADC), que permite medir señales analógicas de entrada, y de esta forma poder tomar medidas de diferentes sensores analógicos que se conecten a la placa. La TIVA cuenta con 2 de estos módulos y con varios pines que se pueden configurar como entradas analógicas.

2. Aplicación de partida

El código de partida será el mismo que la fase anterior, sobre el que el estudiante ya ha debido implementar algunas modificaciones. El estudiante deberá modificar del código para incorporar las nuevas funcionalidades que se solicitan en este documento, manteniendo en funcionamiento las funcionalidades añadidas en las etapas anteriores.

El ejemplo de partida ya traía integrada parte de la funcionalidad necesaria para esta segunda parte, permitiendo capturar la señal analógica de 4 pines de entrada al pulsar un botón del interfaz de usuario. Los valores capturados se envían hacia el PC y se representan en 4 controles numéricos en el mismo interfaz.

2. Especificaciones adicionales. [1,5 puntos]

2.1 Ampliación del número de canales.

Se modificará el código de partida de forma que se capturen y representen las muestras de dos canales adicionales.

- El muestreo y posterior conversión de señales analógicas se realizará para un total de **6 canales** utilizando **los canales analógicos de entrada AIN0, AIN1, AIN2, AIN3, AIN4 y AIN11**, multiplexados respectivamente con pines de los puertos PE (**PE3, PE2, PE1 y PE0**), PD (**PD3**) y PB (**PB5**). Estos canales son accesibles a través de los conectores de los puertos GPIO correspondientes en la placa.
- La **resolución empleada en la conversión será de 12 bits** y la **velocidad de conversión** será de 1MS/s.

2.2 Adquisición periódica continua de datos analógicos del micrófono KY-038 para su representación en tiempo real

Para implementar esta funcionalidad, se debe poder configurar desde el interfaz de usuario del PC la frecuencia de muestro para el muestreo y la conversión A/D de señal analógica del micrófono KY-038 (terminal A0 del KY-038[1]) que se introduce en la TIVA a través del canal analógico AIN11 (PB5). Los datos digitales correspondientes a las muestras obtenidas del conversor A/D de la TIVA se deben transmitir hacia el interfaz de usuario del PC, que conforme los reciba los irá añadiendo a una gráfica que irá actualizando en tiempo real. También debe ser posible detener la adquisición de datos periódica. **Esta nueva especificación debe funcionar simultáneamente junto con la anterior** (cuando el usuario pulse el botón correspondiente de la interfaz Qt, se seguirá tomando una muestra de cada uno de los 6 canales, enviando al PC y representando en los indicadores de tipo numérico). También funcionará simultáneamente con las funcionalidades de la primera parte de las especificaciones.

- En esta especificación, se utilizará un ADC con el modo **disparo por timer**, y se configurará uno de los *timers hardware* del microcontrolador (que el estudiante deberá escoger, teniendo cuidado de no utilizar uno que esté ocupado[2]) para realizar disparos del ADC con una frecuencia determinada. De esta forma, las muestras del canal analógico se capturarán de forma periódica y la frecuencia de muestreo vendrá marcada por el timer hardware del microcontrolador que se haya configurado. En este modo, por cada disparo provocado por el timer, se tomará una muestra del canal al que está conectado el micrófono (PB5). El timer hardware debe ser configurado de forma que dispare las conversiones del ADC por hardware, sin producir interrupciones asociadas al timer. La finalización del muestreo y conversión A/D de cada muestra del canal de entrada sí se controlará mediante interrupción. La frecuencia de muestreo del canal analógico al que está conectado se podrá configurar desde el interfaz de usuario y el usuario la podrá variar entre **1 Hz y 16Khz**.

- Las muestras convertidas del canal al que está conectado el micrófono se irán acumulando en el microcontrolador y se mandarán hacia el PC agrupadas (más adelante se proporcionan más detalles). Al ser recibidas en la aplicación Qt, las muestras se añadirán a una gráfica del interfaz de usuario que se irá actualizando en tiempo real conforme vayan llegando las muestras. Las muestras recibidas se pintarán en tiempo real en la gráfica. En dicha figura **se representarán al menos 4096 puntos** con el valor de las últimas 4096 muestras recibidas. Los valores de la gráfica se inicializarán a 0 y se irán actualizando conforme se vayan recibiendo nuevas muestras.

- Al incrementar la velocidad de muestreo (conversión y envío de las muestras) es posible que las muestras no puedan enviarse suficientemente rápido hacia el PC por la conexión USB. Para reducir la sobrecarga (*overhead*) y mejorar la eficiencia del protocolo, tal y como se ha indicado en el punto anterior, no se enviarán las muestras inmediatamente hacia el PC cada vez que se complete una conversión, sino que irán acumulando antes de proceder a su envío hacia el PC en un único

[1] Además de conectar el terminal A0 del KY-038 al canal AIN11 (PB5) de la TIVA, es necesario conectar el terminal G del KY-038 con el pin de masa GND de la placa TIVA y el terminal + del KY-038 con el pin de 3.3V de la placa TIVA. El terminal D0 del KY-038 no lo utilizaremos en esta práctica por lo que no será necesaria su conexión.

[2] La librería RGB usa los TIMERS 0 y 1 para el control PWM de los LEDs. Además, hay que tener en cuenta el timer configurado para la medida del uso de la CPU (*CPUUsage*), que se puede cambiar.

mensaje que agrupe varias muestras. Se **acumularán 64 muestras** del canal al que está conectado el micrófono.

- La adquisición de las muestras de forma periódica deberá poderse arrancar y detener desde el PC, además de ser posible cambiar la frecuencia de muestreo, tal y como ya se ha indicado previamente. Para esta última especificación utiliza un widget o varios widget que permitan visualizar la frecuencia de muestro que se está enviando.

- Como ya se ha mencionado, esta funcionalidad debe funcionar en paralelo con la funcionalidad del anterior (apartado 2.1), es decir, si el usuario pulsa en el botón del interfaz de usuario, se capturará una muestra de cada uno de los 6 canales, se enviarán inmediatamente hacia el PC y se mostrarán en los indicadores numéricos correspondientes. Para que la implementación de esta funcionalidad no se complique excesivamente, se recomienda que se **utilicen los dos ADC del microcontrolador**, uno para cada funcionalidad.

2.3 Reproducción de sonido mediante la interfaz de usuario Qt

Los datos digitales procedentes del muestro y la conversión A/D del micrófono serán reproducidos en tiempo real mediante la interfaz de usuario Qt, siempre y cuando la frecuencia de muestro sea superior a 4000Hz. Para ello, la interfaz dispondrá de dos botones que permita la reproducción y la pausa en tiempo real de las muestras que llegan del conversor de la TIVA. Para facilitar esta especificación, se proporcionará un ejemplo sobre cómo utilizar la biblioteca multimedia de Qt para reproducir sonidos.

2.4 Consideraciones finales

Una vez implementadas las funcionalidades anteriores, presta atención y ten en cuenta los siguientes aspectos:

- Cada tarea añadida a la aplicación gasta recursos de memoria del RTOS porque es necesario crear las estructuras para su gestión y reservar un espacio para la memoria de pila de la tarea. Modifica el código, de forma que **una única tarea se encargue de enviar hacia el PC los mensajes de la "notificación asíncrona del estado de los botones", y de los mensajes de las funcionalidades 2.1 y 2.2.** Para ello deberás combinar de forma adecuada mecanismos de IPC que permitan una implementación eficiente de esta optimización (por ejemplo, una combinación de flags de eventos con varias colas de mensajes, o bien un conjunto de colas que sea capaz de gestionar varias colas). Repasa la operación básica de estos mecanismos que se explicó en los vídeos y el texto del tutorial de FreeRTOS.

- Se deberá procurar y comprobar que no se produzcan **problemas en la comunicación** (envío de mensajes hacia el PC) debido a la coexistencia de diversas tareas que participen en el envío de estos mensajes por el puerto serie. Cuando varias tareas comparten un mismo recurso compartido pueden producirse problemas que **deben solucionarse empleando mecanismos de exclusión mutua**. Esto también deberá tenerse en cuenta en sucesivas ampliaciones de la funcionalidad.

- En lo relativo a la frecuencia de muestro del canal asociado al micrófono, se ha especificado que se **alcance una frecuencia máxima de 16KHz**, lo que supondría 16000 interrupciones por segundo. Si con esta frecuencia se ejecutan dentro de la ISR todas las funciones necesarias para llevar a cabo la operación a realizar por la interrupción, el tiempo de ejecución que conllevan estas operaciones no

hace posible que se pueda alcanzar esta frecuencia de muestro. Por este motivo, se hace necesario **disminuir el código a ejecutar para cada interrupción o para gran parte de ellas**. Para ello, se propone utilizar el secuenciador de más pasos del conversor y realizar el tratamiento de los datos para el envío a la cola cuando se haya llenado la FIFO del secuenciador correspondiente. El secuenciador con mayor número de pasos es el secuenciador 0 que tiene una FIFO de 8 pasos. La ISR lo que tiene que hacer es obtener las muestras del secuenciador, enviarlas a la cola, y realizar el cambio de contexto, comprobando que no se ha despertado una tarea de mayor prioridad que la que estaba en curso cuando se produjo la interrupción. Todos estos pasos, conllevan un tiempo de ejecución que hace que no sea posible que se produzcan 16000 interrupciones por segundo, y por ello se propone que el **envío de las muestras a la cola y el cambio de contexto no se haga con cada interrupción, sino que se haga cuando la FIFO del secuenciador se haya llenado (cada 8 interrupciones), disminuyendo así la sobrecarga en media que se introduce**.

- Por limitaciones hardware del reproductor de sonidos del PC es **necesario que las muestras se reproduzcan con una frecuencia mínima de 11KHz, por lo que será indispensable que cuando la frecuencia sea menor hacer algún tratamiento con los datos que permita obtener esta frecuencia mínima**. En las especificaciones 2.3 se ha indicado que la frecuencia de muestro del canal PB5 mínima reproducible son los 4KHz, por lo que es obvio que en este caso en la interfaz Qt habrá que crear más muestras a partir de las recibidas que permitan alcanzar la frecuencia de reproducción mínima del PC que son los 11KHz. Una solución sencilla es triplicar cada una de las muestra, alcanzando así un frecuencia de 12KHz, que supera la frecuencia mínima reproducible. Esto mismo también se podrá hacer para frecuencias superiores a los 11Khz, lo que sí será importante es indicar en todos los casos la frecuencia de reproducción resultante. Por ejemplo, si hemos programado una frecuencia de muestro de 12KHz y la opción de triplicar muestras habrá que indicar al reproductor que la frecuencia de reproducción son 36KHz.

2.5 Puntuación de las especificaciones [1,5 puntos]

- EP2.1 Añadir la captura de dos nuevos canales PD (PD3) y PB (PB5), además de los 4 canales ya incluidos en el código de partida. Realizar el envío de las 6 muestras y la representación en los indicadores de la interfaz Qt, manteniendo el resto de las especificaciones. **0,15 puntos**.

- EP2.2. Adquisición periódica continua de los datos analógicos correspondientes al micrófono KY-038 conectado al canal AIN11 (PB5) y representación en tiempo real del valor analógico del canal. (**0,55 puntos.), que se desglosan en:**

  - EP2.2.A. Configurar el segundo ADC del microcontrolador para disparo por timer y configuración de un timer para controlar la adquisición de muestras por parte del ADC, permitiendo controlar la frecuencia de adquisición y arrancar y parar el proceso de captura periódica, enviando las muestras capturadas hacia la interfaz Qt. (**0,3 puntos**).

  - EP2.2.B Envío de las muestras agrupadas hacia la interfaz Qt en bloques de 64 muestras de y representación en tiempo real de las muestras convertidas en voltios en una gráfica en la aplicación Qt con las muestras recibidas del ADC al capturarlas en modo de muestreo periódico. En la gráfica se representará una curva para el canal PB5 con al menos 4096 puntos/muestras. (**0,25 puntos)**

- EP2.3 Reproducción de sonido mediante interfaz Qt (**0,4puntos**)
- EP2.4. Optimización final del código (**0,4 puntos**)
  - EP2.4. A Implementar un mecanismo de exclusión mutua para evitar la existencia de errores en la comunicación provocados por la multitarea (varias tareas accediendo al USB para enviar mensajes hacia el PC). El envío de los datos a través de USB debe ser protegido con un mutex de la forma más simple posible (**0,1 punto**)
  - EP2.4.B. La fusión de varias tareas en una única tarea que se encargue del envío de los mensajes correspondientes a las funcionalidades 2.1, 2.2 y la "notificación asíncrona del estado de los botones" de la especificación 1 mediante la utilización del mecanismo flag de eventos o grupos de colas. (**0,2 puntos**)
  - EP2.4.C. Las variables de los mecanismos IPC que comunican las interrupciones (botones puerto F, especificación 2.1 ADC0 y especificación 2.2 ADC1) con la tarea unificada deben seguir la misma filosofía que la del código de partida, es decir, deben ser variables globales de tipo static. Esto significará que será necesario crear funciones que permitan el acceso a las funcionalidades de FreeRTOS. Por ejemplo, en el código de partida se ha definido para la especificación 2.1 la variable global cola_adc que permite el envío de las muestras desde la interrupción del ADC0 a la tarea que se encuentra en el fichero main.c. La creación de esta cola y acceso a la misma en el fichero main.c se hace a través de las funciones configADC_IniciaADC0 y configADC_LeeADC0 que están definidas en la biblioteca del ADC definida por los ficheros configADC.c y configADC.h (**0,1punto**)