Práctica 1: Desarrollo de un sistema de sensado y actuación controlado por USB. Primera parte (Sin puntuación)

1. Objetivos

El principal objetivo del proyecto es el desarrollo de una aplicación de control de salidas y entradas digitales, y adquisición (muestreo, almacenamiento y representación) de señales analógicas y digitales, controlado desde un PC por interfaz USB. El sistema se basará en la placa de bajo coste EK-TM4C123GXL. El sistema implementado tendrá la función de un dispositivo de sensado y actuación (monitorización de entradas analógicas y digitales, adquisición de datos y salidas para control de actuadores), así como algunas funcionalidades adicionales. El proyecto se desarrollará en varias fases, de las cuales en este documento se recogen las especificaciones correspondientes a la primera de ellas, y tienen como objetivo la toma de contacto con la aplicación de partida. En sucesivos documentos se irán proponiendo nuevas especificaciones para completar la funcionalidad del sistema.

El perfil USB utilizado es el perfil estándar CDC, que emula las comunicaciones por puerto serie y es reconocido por diferentes sistemas operativos. La aplicación implementada estará basada en dos módulos: un interfaz gráfico para el PC desarrollado utilizando las bibliotecas QT, y una aplicación para el microcontrolador TIVA TM4C123GH6PM, basada en FreeRTOS y estructurada en varias tareas que permitan realizar el control de las comunicaciones y la adquisición de datos procedentes de los módulos y componentes de la placa *TIVA*. Para desarrollar la aplicación se suministrará un código de partida para cada uno de los dos extremos (Aplicación interfaz basada en QT para el PC y Aplicación de ejemplo basada en FreeRTOS para el microcontrolador). El estudiante deberá ampliar la funcionalidad del software añadiendo nuevas tareas y funciones, de manera que se implementen las especificaciones recogidas en este documento y sucesivos.

2. Herramientas y documentación

2.1 Herramientas software

En una primera fase de la práctica el alumno necesitará utilizar las siguientes herramientas:

- Entorno de desarrollo *Code Composer Studio* versión 11.
- Bibliotecas QT (v6.2.3), Qwt, analogwidgets, ColorWidgets y entorno de desarrollo (IDE) QT Creator.
- Sistema operativo de tiempo real *FreeRTOS v10.4.1,* incluido en los ejemplos de partida.
- Librería de periféricos *TIVAware v2.2.0.295*, incluyendo la librería USBLIB, que gestiona el interfaz USB.

- Drivers USB (versión Windows) del interfaz USB CDC del microcontrolador. Se encuentran en la carpeta *Windows_drivers* de la instalación de *TIVAware*.

2.2 Documentación

La documentación necesaria se encuentra en el campus virtual, e incluye.

- Para la aplicación del microcontrolador:
  - Datasheet del microcontrolador TM4C123GH6PM [1].
  - Manual de usuario de la librería de *drivers* de TIVAware [2].
  - Manual de usuario de la librería USBLIB de TIVAware [3].
  - Manual del programador (API) de FreeRTOS.
  - Manual de desarrollador de firmware de la placa TM4C123GX L[4].
  - Manual de uso de la *TIVA Launchpad* [5].
- Para el PC
  - Manual del programador de la biblioteca QT-5.15 (incluida biblioteca QSerialPort).
  - Manual del programador de la biblioteca Qwt-6.1.

3. Especificaciones

3.1 Arquitectura del Sistema

La aplicación desarrollada debe permitir el establecimiento de una comunicación USB-CDC (emulación de puerto serie) entre un PC y un sistema de adquisición implementado mediante la placa *TIVA Launchpad*, de forma que se puedan enviar diferentes órdenes desde el PC al sistema de adquisición y recibir las respuestas que éste genere (ver figura 1). Además de esta conexión, se dispondrá de una segunda conexión serie emulada, realizada a través del depurador integrado en la placa, que permitirá implementar un interfaz de comandos que se utilizará principalmente para depuración.

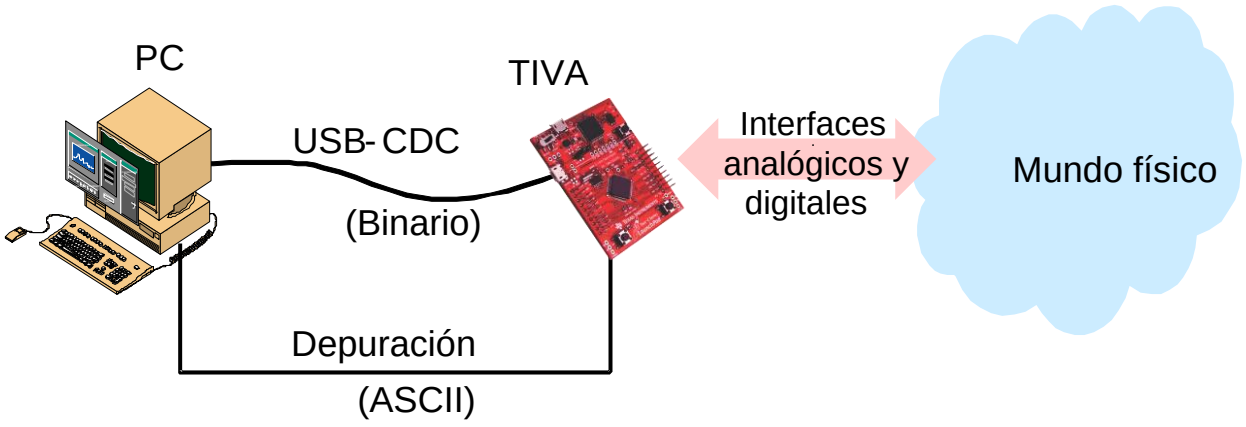

Figura 1. Esquema de conexiones de la aplicación.

El sistema final debe permitir la adquisición de señales tanto analógicas como digitales, y la generación de señales digitales PWM mediante el microcontrolador integrado en la placa. Para implementar de forma completa un sistema de adquisición de datos y control, sería necesario añadir algunos subsistemas adicionales de acondicionamiento de señal y transducción, tal y como se muestra en la figura 2, en la que aparecen recuadrados los módulos que se van a implementar en la presente práctica. Los otros subsistemas son objeto de estudio de otras asignaturas de la titulación, y por tanto no se integrarán en esta actividad.

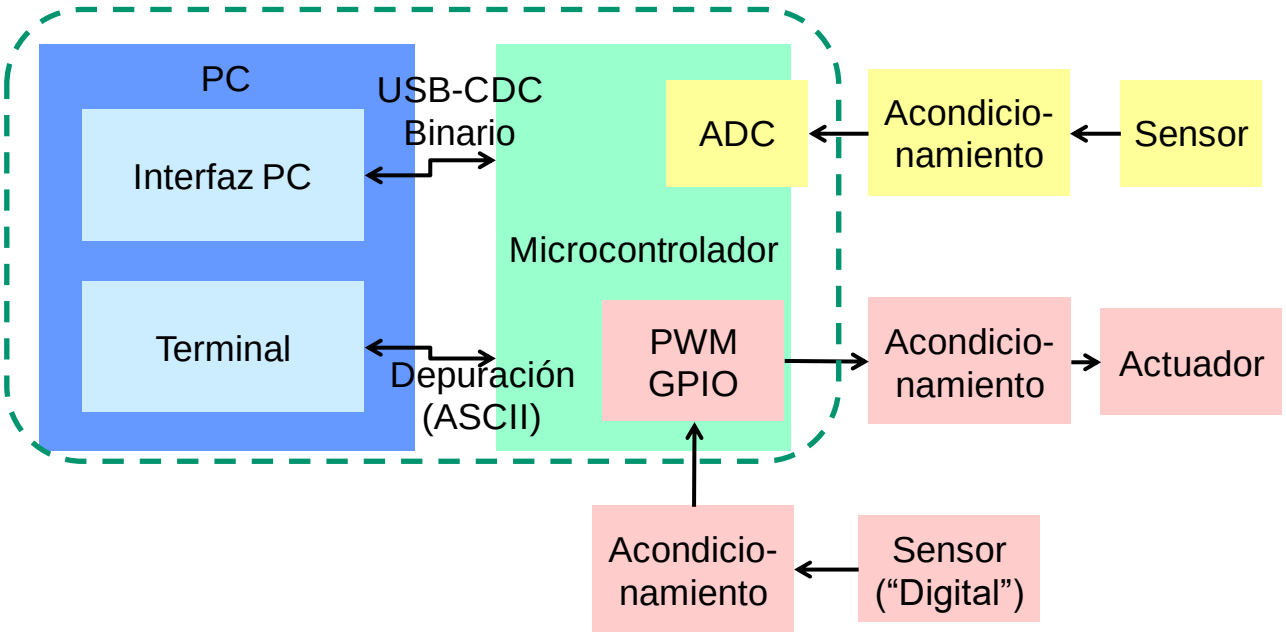

Figura 2. Diagrama de bloques de un sistema de adquisición y control.

La aplicación implementada para el PC, que se comunica con el microcontrolador a través del interfaz directo USB-CDC, es el interfaz de usuario principal del sistema de adquisición y control. Además, tal y como ya se ha comentado, también se podrá establecer una comunicación secundaria con el microcontrolador, a través de una terminal serie, cuya principal misión es la depuración, y que permitirá recoger estadísticas y *logs* del sistema, y realizar algunas acciones. En concreto, el intérprete de comandos accesible por la consola de texto serie permite:

- Visualizar las tareas del sistema y su estado.
- Visualizar la memoria libre de FreeRTOS.
- Visualizar el uso de CPU.
- Visualizar las estadísticas de las tareas.

3.2 Especificaciones funcionales básicas.

Mediante el interfaz de usuario principal, el sistema debe permitir:

- **Control GPIO del led tricolor [Implementado en el ejemplo].** Desde la aplicación se debe poder controlar la activación o desactivación de las señales GPIO conectadas a los pines PF1, PF2 y PF3 (la

de los LEDS verde, rojo y azul de la placa TIVA) de forma **individual,** de manera que sea posible encender o apagar cada color desde el interfaz de usuario.

- **Cambio de modo.** Desde la aplicación se debe poder controlar el modo de los pines de salida que están conectados los LEDs entre modo salida PWM y modo salida GPIO (similar al ejercicio FreeRTOS).

- **Control de color y brillo mediante PWM.** Desde la aplicación se debe poder controlar el ciclo de trabajo de las señales PWM conectadas a los pines PF1, PF2 y PF3 (la de los LEDS verde, rojo y azul de la placa TIVA) de forma **individual,** de manera que sea posible establecer el color RGB del LED tricolor desde el interfaz de usuario (utilizando un widget que permita seleccionar el color deseado). A la vez debe haber un control de "brillo general" que afecte a todos los LEDs.

- **Control mediante sondeo de entradas digitales.** La aplicación del PC debe poder consultar el valor que presentan las entradas digitales PF0 y PF4, que corresponden a los pulsadores SW1 y SW2 de la placa *TIVA*. Para ello se añadirá al interfaz gráfico QT, un botón que permita leer el estado de dichos pines (por sondeo) y mostrarlo en el interfaz de usuario (para mostrar el valor utilice un control de tipo LED (de la biblioteca analogwidgets) para cada pulsador.

- **Control mediante eventos de entradas digitales.** Desde el PC se debe poder activar o desactivar este comportamiento, que consiste en que el microcontrolador avise al PC cuando hay un cambio en el estado de los pulsadores SW1 o SW2 de la *TIVA*, sin que el PC tenga que consultarle. Al recibir los mensajes de notificación, el PC debe mostrar el valor de los pines en el interfaz de usuario. El envío de este evento se debe poder activar y desactivar desde el PC.

3.2 Restricciones

La aplicación debe ser programada considerando las siguientes restricciones o recomendaciones:

- La implementación del software del microcontrolador debe realizarse con FreeRTOS.

- Se programará una tarea de FreeRTOS para interpretar los mensajes enviados por la aplicación interfaz de usuario del PC y recibir y procesar sus posibles parámetros (ver sección 3.3 "Protocolo de comunicación"). Para el intérprete de comandos enviados por la consola serie, se utilizará **otra tarea** diferente.

- El "envío asíncrono de eventos al PC" cuando cambie el estado de los pulsadores debe gestionarse por interrupción en el microcontrolador, y no por sondeo (*polling).* Además, no se permite utilizar variables globales de estado en la implementación de esta funcionalidad. Sí pueden utilizarse variables globales para los manejadores de los mecanismos de IPC y tareas de FreeRTOS.

3.3 Protocolo de comunicación por el interfaz USB-CDC

Para las comunicaciones entre la aplicación del PC y la del microcontrolador se utilizará el perfil CDC (Communication Device Class) del interfaz USB, que emula un puerto serie. Utilizando el puerto serie emulado, se enviarán desde el PC al microcontrolador tramas de longitud variable con las diferentes órdenes que se necesitan para implementar la funcionalidad y sus parámetros. Estas tramas son un conjunto de octetos enviados de forma secuencial por el puerto serie emulado, respetando un determinado formato que se describe a continuación (protocolo de serialización). Cada trama constituye un

mensaje que tiene un determinado efecto en el microcontrolador (provocará la ejecución de una determinada función con los parámetros enviados).

La estructura de las tramas se muestra en la figura 1. Cada trama va precedida por un byte especial de comienzo de trama (con valor 0xFC) y finaliza con un byte de fin de trama (valor 0xFD). Entre el carácter de inicio de trama y el carácter de fin de trama podrá ir un número variable de octetos (con unos límites), que se organizan en campos con longitud de uno o más bytes. Estos campos son un byte que indica el tipo de mensaje, un campo de datos cuya longitud depende del tipo de mensaje y un campo de dos bytes con un CRC para control de errores.

Cada trama enviada por el puerto serie se detecta gracias a los octetos especiales de inicio y de fin, lo que obliga a que ningún otro byte pueda tener los valores 0xFC o 0xFD, ya que de lo contrario se detectaría de forma incorrecta un inicio o un fin de trama. Para ello, a los octetos intermedios, antes de ser transmitidos, se les aplica lo que se conoce como *Byte Stuffing*.

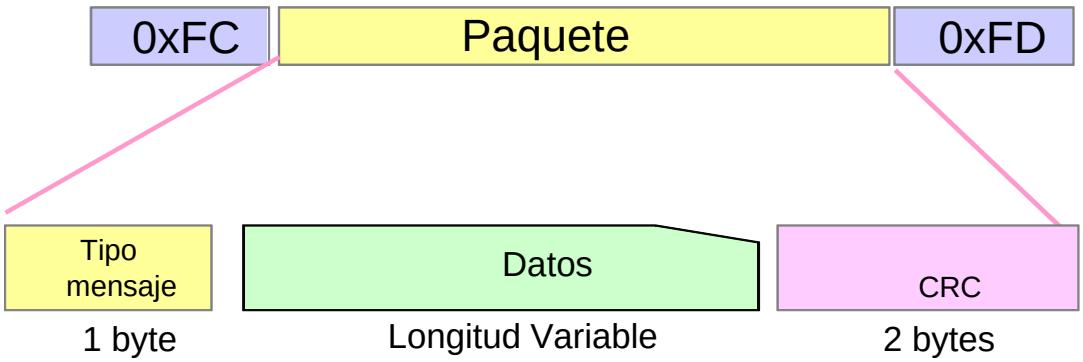

Figura 3. Formato de las tramas utilizadas para enviar mensajes por el puerto serie (sin tener en cuenta el *byte stuffing*)

La técnica de *Byte Stuffing* consiste en introducir un nuevo carácter especial, denominado carácter de escape (con valor 0xEF en este ejemplo), de forma que cada octeto con valor especial (0xCF, 0xDF, o 0xEF) y que vaya a ser enviado entre el carácter de fin y el carácter de inicio, es sustituido por una combinación de dos caracteres. La transformación es la que sigue:

- 0xFD → 0xFE 0xBD
- 0xFC → 0xFE 0xBC
- 0xFE → 0xFE 0xBE

El byte- stuffing es reversible, ya que, al recibir los datos, siempre que se detecte el carácter de escape se examinará el carácter que le sigue, y en función de su valor, ambos se sustituirán por el original:

- 0xEF 0xBD → 0xFD
- 0xEF 0xBC → 0xFC
- 0xEF 0xBE → 0xFE

El byte-stuffing se aplica después de generar el CRC de la información a transmitir. Una vez recibido el paquete en el destino, se deshace el byte-stuffing y se comprueba que el CRC es correcto.

Como ya se ha mencionado, cada mensaje que se envía desde el microcontrolador al PC consta de un código de mensaje (tipo de mensaje) que lo identifica y, generalmente, de unos datos que son los parámetros del mensaje. No todos los mensajes tienen datos asociados.

En el ejemplo de partida se proponen algunos mensajes, aunque deberán programarse algunos más para implementar toda la funcionalidad del sistema. Alguno de los mensajes puede requerir que el microcontrolador responda enviando información de vuelta. En algunos casos, el microcontrolador podrá enviar tramas directamente al PC sin necesidad de recibir un mensaje previo (por ejemplo, cuando se implemente el envío de un evento de cambio de estado de los pulsadores).

Un resumen de los mensajes del ejemplo de partida se muestra en la siguiente tabla:

Mensje	**Código (primer byte)**	**Función**	**Parámetros:**	**Respuesta**
`MESSAGE_PING`	0x01	Pedir un eco al microcontrolador	No tiene	El microcontrolador envía un eco
`MESSAGE_LED_GPIO`	0x02	Apagar o encender los LEDs de forma individual.	Un octeto tratado como un campo de bits que determina de forma individual qué LEDs se encienden.	No tiene
`MESSAGE_LED_PWM_BRIGHTNESS`	0x03	Modificar el valor de brillo (ciclo de trabajo) de los LEDs.	Valor flotante de precisión simple (float) que indica el brillo.	No tiene

El envío de los parámetros se hará en formato Little-Endian, es decir, primero el byte menos significativo y luego el más. Nótese que este orden es el mismo en el que se almacenan los datos tanto en la memoria del PC como del TM4C123GH6PM.

Como ya se ha mencionado, en respuesta a algunos de los anteriores mensajes del PC, el microcontrolador puede también enviar mensajes de respuesta hacia el PC. Algunos de los propuestos se recogen en esta tabla.

Mensaje	**Código (primer byte)**	**Función**	**Parámetros:**	**Respuesta Múltiple**
`MESSAGE_REJECTED`	0x0D	Cuando el microcontrolador recibe un mensaje no implementado o desconocido debe responder con este mensaje	Código del mensaje que no se reconoce.	No
`MESSAGE_PING`	0x01	El microcontrolador responde a una petición de eco	No tiene	No

Tal y como ya se ha comentado, para implementar todas las funcionalidades pedidas, será necesario ampliar el protocolo añadiendo nuevos mensajes. En la documentación a entregar al final de la práctica se deben incluir unas tablas como las aquí presentadas especificando el protocolo finalmente implementado.

4. Aplicación de partida

Para la realización de la práctica se suministra el código de una aplicación de partida que implementa parte de la funcionalidad requerida. Dicha aplicación comprende tanto parte del código que va a ser ejecutado en la placa *TIVA*, como un programa para el PC realizado con las bibliotecas QT. El funcionamiento básico de la aplicación se explicará en clase. Además de dicho código, podrán utilizarse libremente todos los ejemplos que se han suministrado a lo largo de la asignatura, los ejemplos que vienen con la librería TIVAware, y los ejemplos que acompañan a las librerías QT.

Los módulos principales del ejemplo de partida son los siguientes:

Firmware del microcontrolador	
remotelink.c	Implementa el servidor en la TIVA que ejecuta diferentes funciones de C en respuesta a las órdenes enviadas desde Qt en los correspondientes mensajes. Contiene una tarea que se encarga de recibir las peticiones y ejecutarlas.
remotelink_messages.h	Definición de los mensajes que se pueden intercambiar entre el PC y el microcontrolador y de sus parámetros.
serialprotocol.c	Implementa las funciones para crear las tramas que se envían por el interfaz USB-CDC (paquetes de datos serilizados) para indicar los mensajes que se deben procesar y sus parámetros.
usb_dev_serial.c	Implementa las funciones que permiten enviar y recibir datos por el interfaz USB-CDC. Las principales son USBSerialWrite y USBSerialRead.
commands.c	Implementación del intérprete de comandos de texto (consola de texto serie) para depuración.
main.c	Contiene la función main() que inicializa los periféricos y las tareas. También contiene las funciones gancho (“hook”) ejecutadas por FreeRTOS bajo ciertas condiciones.
Carpeta driverlib	Ficheros con las bibliotecas de los periféricos del microcontrolador

Carpeta drivers	Bibliotecas con funciones de más alto nivel (que las contenidas en la driverlib) para controlar los LEDs por PWM, los botones, y el ADC de la placa.
Carpeta “FreeRTOS”	Implementación del sistema operativo que permite trabajar en el microcontrolador con varias tareas
Carpeta USBlib	Implementación de la pila de protocolos (“protocol stack”) USB para la TIVA
Startupccs.c	Tabla de vectores de interrupción
Interfaz de usuario Qt	
main.cpp	Programa principal que muestra la ventana y arranca el interfaz de usuario.
Mainusergui.cpp	Implementa la clase a la que pertenece la ventana principal de la aplicación. Contiene la implementación de los slots que responden a las señales de los elementos del GUI y del objeto “Tiva” que se crea para comunicarse con el microcontolador. En este último slot es donde se procesan los mensajes que llegan desde el microcontrolador.
Tiva_remotelink.cpp	Implementa una clase que permite recibir los mensajes en el microcontrolador, generando una señal cuando ello ocurra.
serialprotocol.c	Este módulo es el equivalente al módulo homónimo del firmware del microcontolador. Implementa el protocolo de serialización, proporcionando las funciones que permiten crear y decodificar las tramas que se envían.
remotelink_messages.h	Definición de los mensajes que se pueden intercambiar entre el PC y el microcontrolador y de sus parámetros. Es idéntico a su equivalente en el microcontrolador.

5. Resto de especificaciones.

Las especificaciones recogidas en este documento constituyen una primera aproximación y toma de contacto con el proyecto práctico que se pretende desarrollar, otorgando una puntuación de 0,6 puntos, siempre y cuando se entregue en plazo.

En sucesivos documentos se irá recogiendo la especificación de funcionalidades más complejas que permitan alcanzar el total de 3 puntos de evaluación continua asignados a esta actividad en la programación docente de la asignatura.

NOTA IMPORTANTE I: En la puntuación, tanto de las especificaciones planteadas en este documento como en las sucesivas, se tendrá en cuenta no sólo si se ha implementado o no la funcionalidad correspondiente, sino que también se considerará cómo de correcta es la implementación (Eficiencia del protocolo, eficiencia en el uso de tareas e IPC, uso de los IPC adecuados en cada caso, correcta gestión de la exclusión mutua, etc.).

NOTA IMPORTANTE II: Aunque se podrán (y deberán) incorporar mensajes al protocolo "remotelink" utilizado para comunicar la aplicación Qt con el microcontrolador, se deberá mantener la compatibilidad con la aplicación de partida. Si tienes alguna duda sobre las implicaciones de esta restricción, por favor no dudes en consultar a los profesores.

6. Bibliografía

[1] Tiva™ TM4C123GH6PM Microcontroller DATA SHEET (tm4c123gh6pm) – Texas Instrument (2013)

[2] TivaWare Peripheral Driver Library User's Guide (spmu298) – Texas Instrument (2013)

[3] TivaWare USB Library User's Guide (spmu297) – Texas Instrument (2013)

[4] EK-TM4C123GXL Firmware Development Package (SW-EK-TM4C123GXL-UG-2.0.1.11577) – Texas Instrument (2013)

[5] Tiva C Series TM4C123G LaunchPad Evaluation Kit User's Manual.(spmu296) – Texas Instruments(2013)