COMUNICACIÓN USB ENTRE UN MICROCONTROLADOR Y UN DISPOSITIVO MÓVIL PARA TELEMETRÍA: PRIMER PARTE

Ismael Potolicchio[a], **Lisandro Sabatte**[a], **Dario Mendieta**[a]

[a]*Departamento de Electrotecnia, Facultad de Ingeniería, Universidad Nacional del Comahue, Buenos Aires 1400, Neuquén, Argentina, ismanqn@hotmail.com*

Palabras Clave: USB OTG, Clase CDC, Comunicación USB. Microcontrolador Pic32.

Resumen. El objetivo principal de este trabajo es poder realizar la telemetría de un globo estratosférico para su recuperación en el contexto del proyecto Pehuensat-2. Para esto se emplea el PIC32 Starter Kit y un dispositivo móvil marca Motorola modelo W388 o V3, los cuales comunicaremos por medio del protocolo USB. Con el microcontrolador como host y el móvil como device se puede reportar los datos de los sensores más importantes a un móvil en tierra usando la red GSM-GPRS por medio de mensajes de texto.

1 INTRODUCCIÓN AL PROTOCOLO USB

La comunicación serial USB (Universal Serial Bus) hace su primera aparición en enero de 1996, para terminar con problemas del usuario como la incorporación de placas de expansión, instalación del software y el reinicio de todo el sistema al momento de agregar un nuevo periférico.

En una transmisión de datos por USB intervienen dos actores: el Host Controller (Controlador Anfitrión) y los Devices (dispositivos) conectados físicamente en una topología tipo estrella, permitiendo la conexión simultánea de hasta 127 dispositivos. Un device a su vez puede estar formado por una o más Interfaces, cada una de las cuales se comunica de forma independientemente con las aplicaciones en la PC.

Cada device a nivel de hardware puede contener varios Endpoints (puntos finales) que son un bloque de memoria de datos o registros. Los datos almacenados en un endpoint son datos recibidos o datos esperando a ser enviados. Cada endpoint tiene un sentido que se define desde el punto de vista del Host: un endpoint de entrada (IN) provee datos para ser enviados al host y un endpoint de salida (OUT) guarda datos recibidos desde el host. El host tiene buffers que mantienen los datos recibidos y los que están esperando para ser enviados, pero el host no tiene endpoints (JAN Axelson, 2009).

Finalmente, a nivel lógico la comunicación usb se realiza entre buffers del lado Host controller y los Endponits del lado Device, denominándose a este canal de comunicación Pipe (Tuberías). Gráficamente puede interpretarse el bus USB como indica la fig. 1:

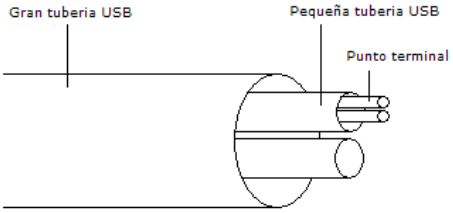


Figura 1: bus USB como esquema de pipes o tuberías.

Dentro de la Gran tubería USB existen pequeñas pipes, las cuales se corresponden uno a uno con el número de dispositivos que existen conectados al bus. Dentro de cada pequeña tubería existen los puntos terminales o endpoints.

En lo que respecta a velocidad, USB ofrece una solución que balancea performance y costo soportando cuatro tipos de velocidades de transmisión(*Estándar y especificaciones técnicas, 2010*). La versión 1.0 (enero 1996) y la
1.1 (septiembre 1998) soportan solo las velocidades 1.5 Mbps (low speed) y 12 Mbps (full speed). La versión 2.0 (abril 2000) de la especificación de USB define una velocidad de 480 Mbps (high speed). En la actualidad la versión 3.0 (noviembre 2008) soporta una velocidad teórica de 4.8Gbps (super speed)

1.1 Host y dispositivos

En USB el Host tienen el control total sobre el bus, es el encargado de iniciar la comunicación y de hacer todo lo posible para asegurar que los dispositivos conectados puedan enviar y recibir información cuando lo necesiten.

Todos los devices conectados al bus “escuchan”, esperando a que el Host les solicite información para iniciar una comunicación.

1.2 Transferencias, transacciones y paquetes

En el estándar USB se define las transferencias como el bloque de más alto nivel por el cual host y device intercambian información. La fig. 2 muestra la arquitectura y jerarquía de una transferencia:

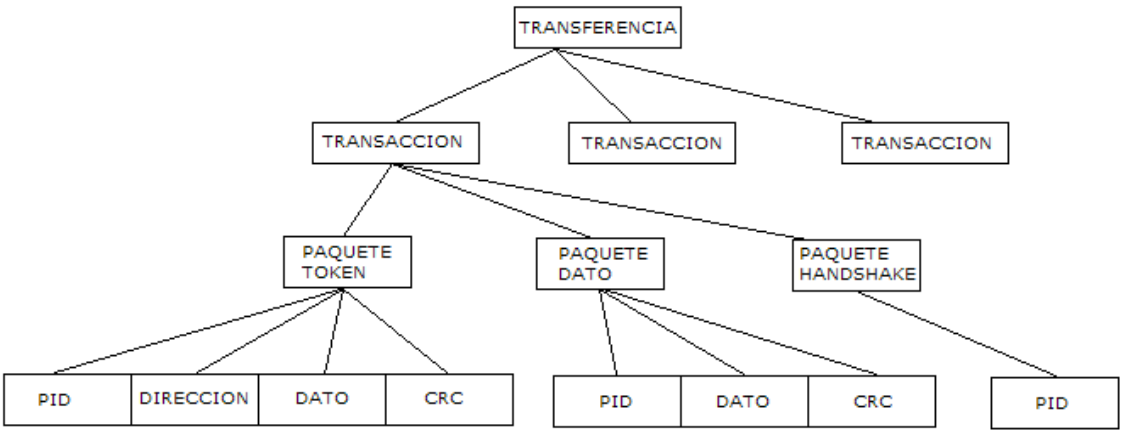


Figura 2: arquitectura y jerarquía de una transferencia

Según la fig. 2, cada transferencia USB consiste de múltiples transacciones, y cada transacción contiene paquetes que contienen información. A su vez los paquetes se componen de campos, que son los bloques de más bajo nivel y básicos en el estándar USB.

El estándar define cuatro tipos de transferencias: control, interrupt, bulk, o isochronous. Cada una tiene un formato y un protocolo para ajustarse a las diferentes necesidades.

- Transferencias de control: para configuración o control del estado de la comunicación entre el host y el dispositivo.
- Transferencias isocrónicas: para transferir información que requiere una tasa constante de transmisión.
- Transferencias de interrupción: en transferencia de datos en pequeñas cantidades y de forma infrecuente.
- Transferencia tipo bulk: se usa para transferencias que requieren grandes cantidades de datos, pero en las cuales el tiempo de llegada de los daros no es crítico.

Todos los tipos de transferencia pueden ser bidireccionales en USB 2.0 y en USB 1.0, salvo las transferencias de interrupción.

1.3 Clases de dispositivos

Dentro de los dispositivos USB existe una distinción llamada clase. Cuando un grupo de periféricos comparten atributos o cuando responden a peticiones similares, se les agrupa en clases.

Clase	Device
0x02	CDC – Comunicaciones
0x03	HID - Interfase humana (mouse, teclado)
0x08	MSD - Almacenamiento masivo
0x01	Audio
0x06	Imágenes (cámaras de video, web o fotográficas)
0x07	Impresoras

Tabla 1: Clases de dispositivos USB

Cada clase posee su protocolo de comunicación pero sin salirse de la arquitectura de transferencia mostrada en la fig. 2 y posee un código que la identifica. Además, dentro de cada clase pueden existir distintas subclases y protocolos de comunicación. Algunas de las clases USB más conocidas se mencionan en la tabla 1.

La clase CDC 0x02 es la utilizada en este trabajo. Se utiliza en comunicaciones y control, como por ejemplo el adaptador Ethernet y el modem. En la Pc, el Host utiliza el driver usbser.sys.

Además del código particular que identifica a cada clase, resulta interesante nombrar que de acuerdo a qué tipo de dispositivo se trate, es el tipo de transacción que se realizara durante la comunicación.

1.4 Enumeración

Es el intercambio de información inicial con el device que le permite al Host identificar la clase, subclase y protocolo que utiliza el device. Una vez que el host detecta un nuevo dispositivo, realiza una serie de transferencias de control conteniendo pedidos estándar de USB para el endpoint 0 del dispositivo. Todos los devices USB deben soportar transferencias de control, los pedidos estándar, y el endpoint 0 bidireccional. Para una enumeración satisfactoria, el dispositivo debe responder a cada pedido retornando la información solicitada y tomando otras acciones requeridas.

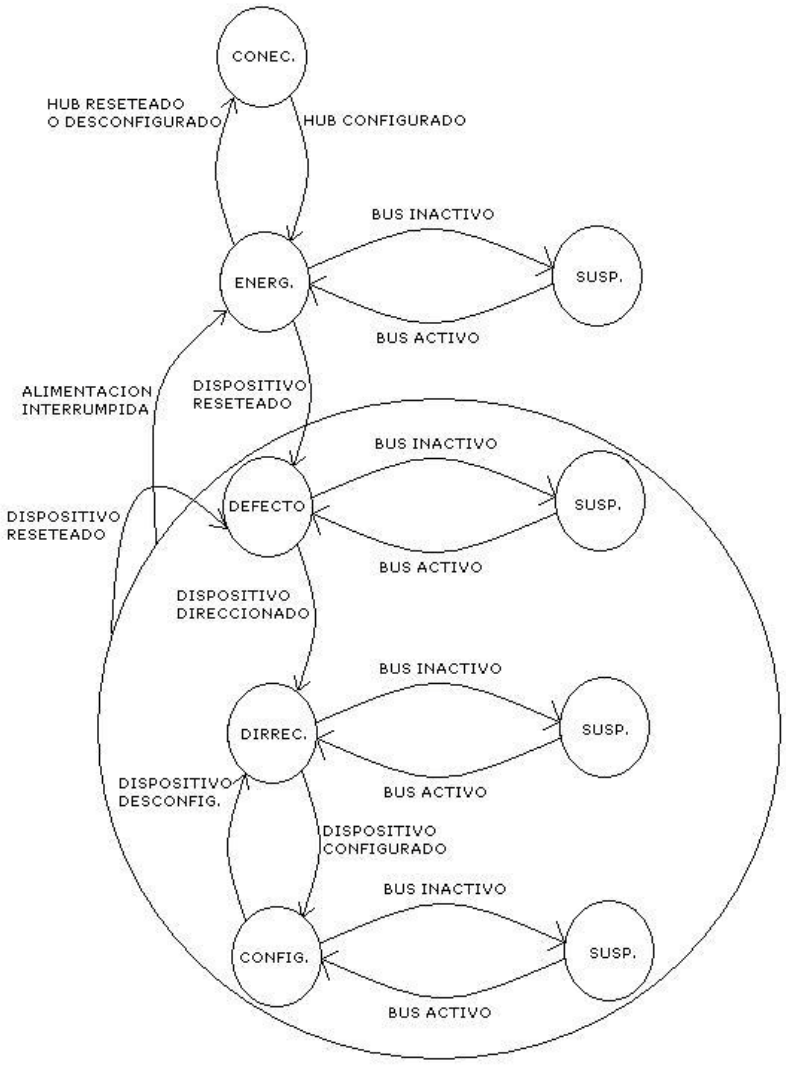


Figura 3: maquina de los seis estados, según el estándar USB.

La norma USB define seis estados en los que puede hallarse un dispositivo con respecto a la comunicación USB. Estos estados son: Energizado, Defecto, Direccionado, Configurado, Conectado y Suspendido como se muestran en la fig. 3. A través del proceso de enumeración, el dispositivo va pasando por los distintos estados hasta llegar al de Configurado, momento en el cual se ha establecido completamente el camino de comunicación y se realiza el intercambio de datos propiamente dicho.

1.5 Descriptores

Los descriptores son bloques de datos almacenados en el device que contienen las características y requerimientos del mismo y están disponibles antes cualquier pedido del Host. Los tipos de descriptores:

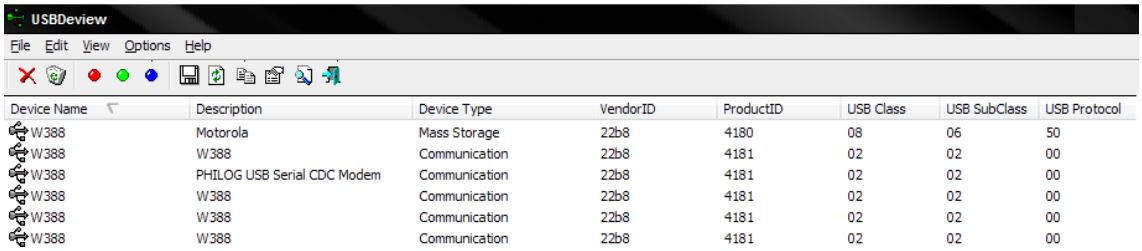
- Descriptor de dispositivo: contiene información básica acerca del dispositivo. Es el primero en leerse al conectarse el dispositivo al bus.
- Descriptor de configuración: provee información acerca de los requerimientos de alimentación y de las interfases soportadas por el dispositivo.
- Descriptor de interfase: detalla el tipo de interfase y el número de endpoints utilizados.
- Descriptor de endpoint: identifica el tipo de transferencia y su dirección.
- Descriptor de string: proveen información adicional y son generalmente opcionales.

2 DESCRIPCIÓN DEL SISTEMA

La clase de dispositivo utilizado para este proyecto es el dispositivo de comunicaciones. Utilizando el articulo “Communication Device Class (CDC) Host” brindado por Microchip (Applicattion Note 1247, 2009), se pudo obtener información técnica acerca de estos dispositivos. Luego, se busco un teléfono celular que presente las características requeridas para entrar dentro de la clase de dispositivos de comunicaciones, además de cumplir con la subclase y el protocolo de comunicaciones requerido. Según (Applicattion Note 1247, 2009), las características que debía presentar el teléfono celular a utilizar debían ser las siguientes:

Clase	Dispositivo de comunicación (0x02)
Subclase	Modelo de control abstracto (0x02)
Interfase	Interfase de comunicaciones (0x02)
Protocolo	Comandos AT (0x01)

Un teléfono que presentaba estas características era el Motorola W388. Utilizando el programa USBDeview pudimos constatar que el teléfono anteriormente nombrado se ajustaba a los requerimientos:



Device Name	Description	Device Type	VendorID	ProductID	USB Class	USB SubClass	USB Protocol
W388	Motorola	Mass Storage	22b8	4180	08	06	50
W388	W388	Communication	22b8	4181	02	02	00
W388	PHILOG USB Serial CDC Modem	Communication	22b8	4181	02	02	00
W388	W388	Communication	22b8	4181	02	02	00
W388	W388	Communication	22b8	4181	02	02	00
W388	W388	Communication	22b8	4181	02	02	00

Figura 4: clase del dispositivo

Al conectar el teléfono a la PC, esta lo reconoce como un modem, lo identifica como un puerto serial, lo enumera y agrega el software necesario para que este pueda funcionar (driver). Puede verse también, que como el teléfono posee una tarjeta SD, el programa nos muestra que el host (la PC en este caso) también detecta al teléfono como un dispositivo de almacenamiento masivo.

2.1 Comandos AT

El teléfono elegido para la comunicación soporta comandos AT. Para probar la comunicación mediante la utilización de estos comandos utilizamos el Hyperterminal de Windows XP. Al conectar el teléfono y abrir el programa debemos configurar el puerto virtual que se genero. Esta configuración deberá coincidir con la que se encuentra en el administrador de dispositivos de Windows XP. Una vez hecho esto, si iniciamos la comunicación, recibiremos el siguiente mensaje por parte del teléfono:

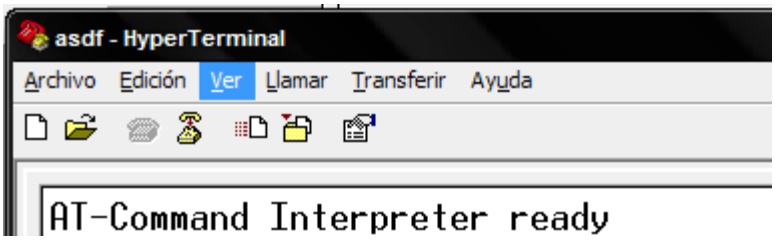


Figura 5: transferencia de comandos AT

Si bien existe una extensa lista de comandos AT(UIT Serie V, 2003), solo nos limitaremos a mostrar la utilización de los comandos que utilizamos para nuestro proyecto. Para asegurarnos que el dispositivo esté disponible, enviamos el comando AT, viendo en pantalla si el celular responde:

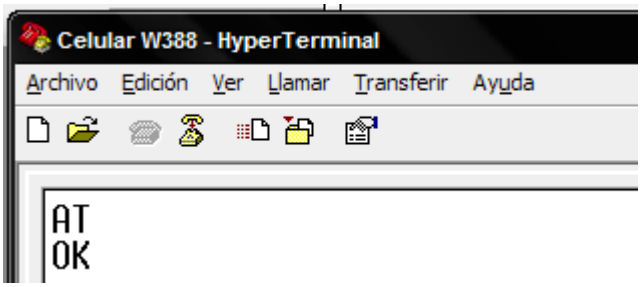


Figura 6: transferencia de comandos AT

Al escribir el comando AT y presionar enter, el celular responde con un OK, lo que nos demuestra que el mismo está disponible para la comunicación. Luego para realizar el envío de mensajes se hace lo siguiente:

Mandar AT+CMGF=1 para entrar en el modo SMS
Mandar AT+CSCA="numero" para establecer el número del centro de mensajes (Personal +541151740055)
Mandar AT+CMGS="numero" para establecer el numero al cual le quiero mandar el mensaje
Escribir el mensaje y presionar CTRL+Z

Si bien lo que uno escribe en el Hyperterminal es lo anteriormente nombrado, los caracteres se envían en formato ASCII por lo que para comprender correctamente que es lo que se envía, debemos utilizar un software de análisis de puertos seriales. En este caso el programa utilizado es

el HDD Free Serial Port Monitor, junto con el programa SIOW.

2.2 El host: microcontrolador Pic32

Como ya conocemos, en toda comunicación USB se necesita un host. El host por lo general es implementado por la PC. Sin embargo en este caso, queremos usar al teléfono para realizar telemetría por lo que no es viable la utilización de una PC. En su lugar se utiliza un microcontrolador de la gama de 32 bits de Microchip. Este microcontrolador viene inserto en una placa de desarrollo llamada PIC 32 Starter Kit. El microcontrolador utilizado posee la propiedad de actuar como Host o como dispositivo según la situación lo amerite. Sin embargo, en este caso actuara solo como Host. Para hacer que esto sea viable debemos utilizar la "USB Embedded Host Stack", una librería que Microchip provee gratuitamente y que puede bajarse del sitio oficial de Microchip. Esta librería trae diferentes demos y diferentes archivos, los cuales son necesarios para realizar la programación del microcontrolador (Applicattion Note 1141, 2008). En (Applicattion Note 1247, 2009) se puede encontrar una lista de todos los archivos que deben ser incluidos en el programa al momento de realizar la programación. Uno de los archivos mas importantes para que el microcontrolador sea capaz de realizar la detección del teléfono celular es el que se denomina usb_config.c (que además tiene asociado un usb_config.h), el cual es creado por medio de una aplicación que viene incluida en la "USB Embedded Host Stack". Esta aplicación se denomina USBConfig (Applicattion Note 1140, 2009), y nos permite crear una TPL (Target Peripherial list), la cual es una lista de especificaciones técnicas de los dispositivos que conectaremos al microntrolador.

3 ENSAYOS CON PIC32 Y EL MOVIL

El primer acercamiento que tuvimos para hallar la solución del problema propuesto fue mediante el driver "Communication Device Class (CDC) Host" descargado de la pagina de Microchip. Dado que este esta hecho para un hardware diferente al utilizado por nosotros, tuvimos que modificarlo para lograr la compatibilidad. El primer logro fue cargar el móvil mediante el dispositivo host, luego se avanzo con la enumeración del device. Actualmente estamos en la etapa de transferencia de datos seriales para poder mediante estos enviar los comandos AT necesarios.

4 CONCLUSIONES

Se pudo analizar el proceso de comunicaciones entre el Host de la Pc y el móvil. Se logro además operar directamente el celular con comandos AT. Se está trabajando en terminar la interface con el Pic32 y el móvil. Hasta el momento se consiguió enumerar el device y se está desarrolando la emulación del puerto serial.

REFERENCiAS

AXELSON, Jan. *USB COMPLETE*. Fourth Edition. Lake View Research LLC, 2009.

USB, *Estándar y especificaciones técnicas*. Disponible vía Web en http://www.usb.org, visitada en mayo 2010.

Amardeep Gupta, Microchip Techonology Inc. Applicattion Note 1247, 2009.

Unión Internacional de Telecomunicaciones, Serie V: Comunicación de datos por la red de datos telefónica, procedimientos de control. Marcación y control automáticos asíncronos en serie,97-100, 07/2003

Bud Caldwell, Microchip Techonology Inc. Applicattion Note 1141, 2008.

Kim Otten, Microchip Techonology Inc. Applicattion Note 1140, 2008.