Capítulo 1

El protocolo USB

El protocolo USB es un sistema de comunicación diseñado durante los años 90 por seis fabricantes vinculados a la industria informáticas, Compaq, Intel, Microsoft, Hewlett-Packard, Lucent, NEC y Philips, con la idea de proveer a su negocio de un sistema que permita la conexión entre las PC's y los periféricos con un formato estándar, de forma tal que permita la compatibilidad entre los distintos fabricantes.

Hasta ese momento, el gran ecosistema de periféricos, sumado a los nuevos avances y desarrollos, hacia muy compleja la interoperatividad de todos ellos. Cada uno de los fabricantes desarrollaba componentes con fichas, niveles de tensión, velocidades, drivers y un sinnúmero de etc diferentes, lo cuál dificultaba al usuario estar al día y poder utilizar cada componente que compraba. Lo más probable era encontrar que cuando se comparaba una PC, se requería cambiar el teclado, el mouse y/o algún periférico específico. Esto también complicaba a las mismas empresas productoras, por que la introducción de un nuevo sistema requería de mucho soporte extra para poder conectar todo lo ya existente.

Todo esto, quedó saldado con el aparición de la norma USB, que debido a la gran cuota de mercado de sus desarrolladores, fue adoptado en forma rápida y se transformó en la especificación por defecto a la hora de seleccionar un protocolo. Al punto tal esto se cumplió que hoy, más de 20 años después, es muy difícil encontrar PC's con otro tipo de puertos, salvo que en el momento de su compra uno solicite especialmente un puerto determinado. Así, cualquier PC nueva disponible en el mercado debe poseer puertos USB para la conexión de los periféricos.

Desde el punto de vista técnico, el protocolo USB es un sistema del tipo maestro-esclavo, donde el maestro, denominado *HOST*, debe ser necesariamente una PC (o un dispositivo con software y hardware capaces de incorporar los drivers necesarios) y cualquier periférico a ella acoplada será un esclavo[23].

Para describirlo es conveniente diferenciar tres partes. Una capa física, en donde se definen los componentes que intervienen, una capa de protocolo, en donde se define el formato, el marco en el que son enviados los paquetes, como se direccionan y como se comunican entre sí, y una parte lógica, en donde cada componente es visto solamente como un extremo y define como fluyen los datos desde un extremo hacia la PC y viceversa.

1.0.1. Capa física

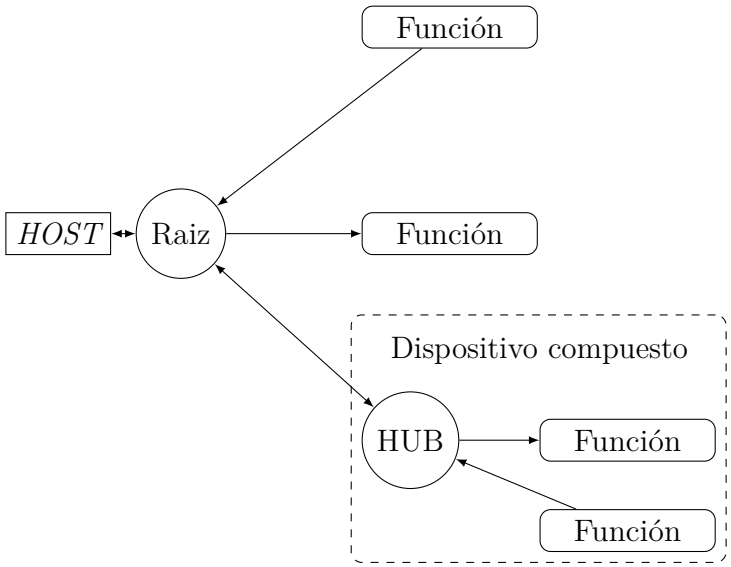

Figura 1.1: Topología de un sistema USB

En esta sección no se describirán los detalles de las conexiones eléctricas ni mecánicas a las que se refieren las especificaciones de la norma USB debido a dos cuestiones fundamentales. Una de ellas es que toda esta sección de la norma está resuelta ya por los fabricantes de la interfaz que se utiliza en este trabajo. A su vez, maneja todas las señales, arma y desarma los paquetes que salen hacia la PC y que llegan de ella respectivamente. Por otro lado, no es el objetivo de este trabajo adentrarse en esos detalles. Gracias a la extensión de este tipo de comunicación existen una gran cantidad de fabricantes en el mercado que fabrican cada uno de los componentes, ya sean, cables, conectores en todas sus versiones, adaptadores de un tipo de estos, su costo es despreciable con respecto a cualquier tipo de desarrollo en ese sentido y son de una muy buena calidad, es decir que todos cumplen con lo que la norma establece. Sí, se describen los dispositivos físicos y su categoría, según la norma, en función del rol que cumplen.

La comunicación USB posee una topología maestro-esclavo. Es decir, existe un dispositivo que dirige todas las transferencias de datos y otros que responden a sus directivas. Por esto, el elemento central de cualquier comunicación USB es el *HOST* (director o anfitrión, por su traducción de la voz inglesa). Él es el que posee el *Host USB Controller*[23]. Esto quiere decir que tiene la capacidad de registrar los dispositivos acoplados, asignarles dirección, colocar los paquetes de salida y/o llegada en sus respectivas listas y servilos a los procesos que utilizan esta comunicación. Además, el *HOST* se encarga de enviar los tokens a todos los periféricos, con la dirección del dispositivo, el sentido de la comunicación, el tipo de transferencia que se espera y todas las acciones de control que el sistema requiera. En la mayoría de los casos, el *HOST* es una PC, auqnue también puede ser cualquier dispositivos "inteligente" como un smartphone.

En el otro extremo de la comunicación, se encuentran lo que la norma denomina *funciones*[23]. Las *funciones* son todos los periféricos que actúan como fuente o sumidero de información. Es decir, en caso de periféricos de entrada, serán una fuente de datos hacia el *HOST*. Si los

periféricos son de salida, serán un sumidero de la información que proporciona la PC. Los casos de periféricos de entrada/salida, se denominan *dispositivos compuestos*.

Existe también, a los fines de la norma,un elemento intermedio, denominado *HUB* (concentrador o distribuidor, según la traducción del inglés). Este dispositivo se encarga de conectar dos o más *funciones*, ya sea de entrada o salida, de recibir y enviar las direcciones y de regenerar las señales que el *HOST* envía y deben ser recibidas por las *funciones*.

La Figura 1.1 muestra la topología típica de un sistema USB. En ella, se observa el *HOST* como un rectángulo, las *funciones* como rectángulos con los bordes redondeados y los distribuidores como círculos. Se puede notar que el *HOST* posee un distribuidor propio llamado *Raiz* en el cual se conectan todos las *funciones* y distribuidores. Cada *Función*posee una única dirección. Pueden existir dispositivos que posean funciones diversas con un mismo encapsulado, como por ejemplo un auricular que tenga micrófono incorporado. Este dispositivo, tendrá un *HUB* que concatena dos *funiones* diferentes.

1.0.2. Capa lógica

Desde el punto de vista lógico, cada dispositivo es visto por el *HOST* como un extremo (EP, del inglés, *endpoint*) independiente, que posee solo un modo de comunicación, es decir, el protocolo se comunicara solo por un tipo de transferencia y en un único sentido con cada *EP*. En otras palabras, USB registra un periférico de entrada/salida como un *EP* de entrada y otro de salida en forma independiente.

Esta independencia brinda la posibilidad de configurar cada extremo de forma diferente y obtener el ancho de banda necesario para la subida y bajada de datos, los tiempos de acceso al bus, la dirección y todo lo relacionado a los modos de comunicación conforme a los requerimientos.

El protocolo entiende que entre le *HOST* y cada uno de los extremos existe un tubo (la norma en ingles habla de *pipes*) en donde la información es colocada y transferida. Luego, cada tubo posee la configuración establecida por el controlador del *HOST* y se comunica con cada *EP* por medio de estos tubos. A los fines del usuario, esto es lo importante, por cuanto se solicita acceso al bus y define en que buffer va a contener los datos a enviar o transmitir y el protocolo se encarga de el empaquetado, el armado de los cuadros, el acceso el bus y el posterior envío de datos.

1.0.3. Capa de protocolo

En la capa de protocolo, la especificación de la norma detalla cómo se compone un cuadro y cómo deben ser estructurados los paquetes para que sean efectivamente enviados a través del sistema. Cada mensaje que se intercambia por el bus se denomina paquete. Cada paquete puede poseer hasta cinco campos, dependiendo del tipo de paquete que sea enviado a través del sistema y de quien sea el remitente. A su vez, cada paquete comienza con una señal de sincronismo (*SYNC*) y un Comienzo de Paquete (SOP de *Start-of-Packet*), y terminan con una

señal de Fin de Paquete (EOP por *End-of-Packet*).

Por otra parte, los paquetes está temporalmente encapsulados en cuadros. Cada cuadro posee un Comienzo de Cuadro (SOF, *Start-of-Frame*) y posee una duración de 1 ms, hasta el próximo SOF. En las comunicaciones de alta velocidad, es decir, aquellas que poseen una tasa de bit de 480 $\mathrm{Mbit\,s^{-1}}$. Se subdivide un cuadro en 8 micro-cuadros de 125 µs cada uno.

Campos de Paquetes

Cada paquete contiene un campo denominado identificador de paquete (PID del inglés *Packet Identifier*). El PID indica el tipo de paquete que se está enviando y, como consecuencia, el formato de cada uno, es decir, que campos acarrea y que control de datos utiliza.

A su vez, cuando el host solicita algo al sistema, lo realiza a través del denominado campo de dirección. Este campo, se compone de dos partes, la primera es el campo de dirección de la función y el segundo es la dirección de extremo.

Los mensajes de datos, poseen un campo dedicado de forma específica a los datos. Puede poseer un numero entero de bytes, desde 0 a 1024.

Para corroborar el envío de datos, USB utiliza verificación de redundancia cíclica (CRC o *Cyclic Redundancy Checks*). Los paquetes especiales y los de token poseen un verificador CRC5, es decir, de 5 bits, cuyo polinomio generador es:

$$G(X) = X^5 + X^2 + 1$$

Por su parte, los paquetes de datos, poseen CRC16, ya que pueden llegar a ser bastante extensos. En su caso, el polinomio generador está dado por:

$$G(X) = X^{16} + X^{15} + X^2 + 1$$

Existe un campo relativo a los cuadros temporales, que se denomina campo de número de cuadro. Este es enviado por el *HOST* en cada SOF y es incrementado a cada cuadro. Los micro-cuadros también poseen un número de cuadro, sin embargo, este es aumentado solamente cada 8 micro-cuadros, es decir, el número se incrementa cada 1 ms y se repite durante los 7 micro-cuadros de 125 µs, en comunicaciones de alta velocidad.

Formato de paquetes

- **Paquetes Token:** A través de este tipo de paquetes el host envía las directivas a los distintos periféricos. Estas directivas pueden ser IN, cuando solicita datos de un periférico; OUT, cuando se dispone a enviar datos hacia una *función*; SOF, que indica el inicio de cada cuadro, para que cada función se sincronice y SETUP, cuando va a enviar un paquete de configuración a algún extremo.
- **Paquete de Datos:** Este tipo de PID puede ser emitido por un dispositivo, si es que envía datos al host o bien por el mismo host cuando el flujo de datos es a la inversa.

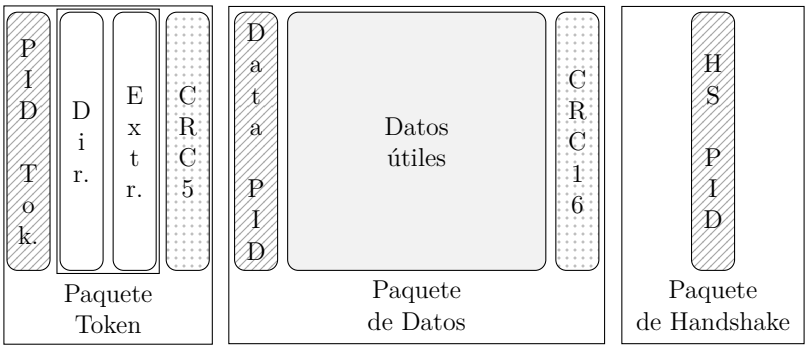

Figura 1.2: Distintos tipos de paquetes USB

- **Paquete de *Handshake*:** Es enviado por el receptor del mensaje y le informa al emisor el estado de la transferencia. ACK significa que el paquete fue recibido sin errores; NAK, los datos poseen error o el emisor no puede enviar; la señal STALL quiere decir que la solicitud no es soportada o que el extremo está detenido; NYET implica que no hay respuesta aún por parte del receptor.

- **Paquetes Especiales:** Son paquetes con propositos específicos. Con ellos se señalan preambulos emitidos por el *HOST*, se informan errores, se solicitan mensajes divididos en diferentes paquetes y se intercambia señales de ping para conocer el estado de los componentes del sistema.

Cada uno de los tipos de paquetes posee un formato específico, tales como se muestran en la Figura 1.2. En ella se observa que los paquetes de token envían un PID, una dirección y un CRC5; los paquetes de datos, se componen de un PID, los datos transmitidos y un CRC16; en el caso de los paquetes de Hanshake, solo el PID indica que tipo de mensaje se envía. Los paquetes especiales no se detallan ya que el formato es muy variable, en función del paquete.

1.0.4. Flujo de datos

Como se menciono anteriormente, el host envía un toquen SOF que sirve para sincronizar los dispositivos al bus. En un sistema USB, el host provee la base de tiempo y envía cada 1 ms un SOF (Start of frame, o su traducción, inicio de cuadro) seguido de un numero de 11 bits que sirve para contar cada uno de los marcos. Además, en sistemas de alta velocidad, cada cuadro se divide en ocho microcuadros de 125 µs, que también son marcados por un SOF, sin embargo, el contador no se actualiza por cada microcuadro.

Luego de esto, el sistema puede comenzar con la transferencia de datos. USB dispone 4 tipos de posibles transferencias, que se detallan un poco más adelante, y que pueden ser usadas conforme a los diferentes requerimientos del sistema.

Cada transferencia de datos está compuesta por un primer paquete de token, emitido por el host, que posee el tipo de transferencia que se espera, sea de entrada, de salida, de control o especial; la dirección de la función que debe responder o recibir el mensaje enviado por el bus y

los verificadores CRC5.

Luego, el siguiente paquete posee los datos que se transfieren, precedido por un PID de datos, y verificadores CRC16. Este paquete es transmitido por el emisor de los datos. Finalmente, el receptor devuelve un paquete de *handshake*, indicandole al emisor si el transferencia fue efectiva o no.

Transferencias por paquetes (Bulk transfers)

Este tipo de transferencias puede ser dispuesta para trasmitir un gran flujo de datos. No posee perdida de datos gracias a un sistema de chequeo y retransmisión de datos. El inconveniente que presenta este tipo de transferencias es que en un nivel de prioridades se presenta en el final del sistema. Es decir, el bus solo va a ser usado para transferir este tipo de datos siempre que se encuentre desocupado, o bien, se le asignará una proporción ínfima de ancho de banda para poder trasnmitir con este modo. Es comunmente usado para trasmitir datos que no son críticos en tiempo, por ejemplo para scanners e impresoras.

Transferencias de interrupción

Este tipo de transferencias sirve para enviar y recibir paquetes de datos que requieren un buen sistema de control de errores, pero que, son más restrictivos en tiempos. El sistema siempre destinará un intervalo fijo de tiempo para transmitir los datos que estén pendientes para trasnferencias de interrupción.

Transferencias Isocrónicas

Este tipo de transferencias está destinado a datos que son críticos en tiempo. Es usado, principalmente para enviar datos "a chorro", como ser el caso de *streaming* de audio o video, en donde los datos producidos deben ser rápidamente llevados al usuario.

No posee un control de errores muy sofisticado, más que un simple código CRC, pero no existe mecanismo de retransmisión de datos ni handshake entre los *EP* y el *HOST*.

Como el tiempo es el requerimiento crítico en este tipo de datos, el controlador le asigna una determinada cantidad de tiempo de bus, o en otras palabras, una determinada cuota de ancho de banda.

Transferencias de control

Este tipo de transferencias solo las emite el host y el sistema las utiliza para configurar cada dispositivo. Debido a su criticidad, el controlador dispondra encada cuadro de una fracción de ancho de banda para las trasnmisiones de control. Es el tipo de transferencias que posee el sistema de

detección de errores más sofisticado, de forma tal de asegurar la integridad de los datos de control.

A cambio de esto, solo muy poca información puede ser trasmitida por cada cuadro, de hasta 64 bytes en sistemas de alta velocidad.

Bibliografía

[1] R. Pallàs-Areny and J. G. Webster, *Sensors and signal conditioning*. Wiley-Interscience, 2001.

[2] D. M. Considine, *Encyclopedia of instrumentation and control*. McGraw-Hill, Inc., 1971.

[3] A. Perez Garcia, "Curso de instrumentación," p. 261, 2008.

[4] J. Fraden, *Handbook of modern sensors: physics, designs, and applications*. New York, NY: Springer New York, 2010.

[5] E. Slawiñski and V. Mut, *Humanos y máquinas inteligentes: conocimiento educativo sobre el comportamiento interno de robots que actúan juno y para el hombre*. Saarbrücken, Alemania: Editorial Académica Española, 2011.

[6] K. Ogata, *Modern control engineering*. Aeeizh, 2002.

[7] G. Binnig and H. Rohrer, "Scanning tunneling microscopy," *Surface Science*, vol. 126, pp. 236–244, mar 1983.

[8] S. Mendis, S. Kemeny, and E. Fossum, "CMOS active pixel image sensor," *IEEE Transactions on Electron Devices*, vol. 41, pp. 452–453, mar 1994.

[9] C. Hu-Guo, J. Baudot, G. Bertolone, A. Besson, A. S. Brogna, C. Colledani, G. Claus, R. D. Masi, Y. Degerli, A. Dorokhov, G. Doziere, W. Dulinski, X. Fang, M. Gelin, M. Goffe, F. Guilloux, A. Himmi, K. Jaaskelainen, M. Koziel, F. Morel, F. Orsini, M. Specht, Q. Sun, I. Valin, and M. Winter, "CMOS pixel sensor development: a fast read-out architecture with integrated zero suppression," *Journal of Instrumentation*, vol. 4, pp. P04012–P04012, apr 2009.

[10] J. Baudot, G. Bertolone, A. Brogna, G. Claus, C. Colledani, Y. Değerli, R. De Masi, A. Dorokhov, G. Dozière, W. Dulinski, M. Gelin, M. Goffe, A. Himmi, F. Guilloux, C. Hu-Guo, K. Jaaskelainen, M. Koziel, F. Morel, F. Orsini, M. Specht, I. Valin, G. Voutsinas, and M. Winter, "First test results of MIMOSA-26, a fast CMOS sensor with integrated zero suppression and digitized output," *IEEE Nuclear Science Symposium Conference Record*, pp. 1169–1173, 2009.

[11] C. L. Galimberti, F. Alcalde Bessia, M. Perez, M. G. Berisso, M. Sofo Haro, I. Sidelnik, J. Blostein, H. Asorey, and J. Lipovetzky, "A Low Cost Environmental Ionizing Radiation Detector Based on COTS CMOS Image Sensors," in *2018 IEEE Biennial Congress of Argentina (ARGENCON)*, pp. 1–6, IEEE, jun 2018.

[12] M. Pérez, J. Lipovetzky, M. Sofo Haro, I. Sidelnik, J. J. Blostein, F. Alcalde Bessia, and M. G. Berisso, "Particle detection and classification using commercial off the shelf CMOS image sensors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 827, pp. 171–180, aug 2016.

[13] M. Pérez, J. J. Blostein, F. A. Bessia, A. Tartaglione, I. Sidelnik, M. S. Haro, S. Suárez, M. L. Gimenez, M. G. Berisso, and J. Lipovetzky, "Thermal neutron detector based on COTS CMOS imagers and a conversion layer containing Gadolinium," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 893, pp. 157–163, jun 2018.

[14] T. Hizawa, J. Matsuo, T. Ishida, H. Takao, H. Abe, K. Sawada, and M. Ishida, "32 × 32 pH image sensors for real time observation of biochemical phenomena," *TRANSDUCERS and EUROSENSORS '07 - 4th International Conference on Solid-State Sensors, Actuators and Microsystems*, pp. 1311–1312, 2007.

[15] H. H. Goldstine and A. Goldstine, "The Electronic Numerical Integrator and Computer (ENIAC)," *Mathematical Tables and Other Aids to Computation*, vol. 2, p. 97, jul 1946.

[16] M. Perez, F. Alcalde, M. S. Haro, I. Sidelnik, J. J. Blostein, M. G. Berisso, and J. Lipovetzky, "Implementation of an ionizing radiation detector based on a FPGA-controlled COTS CMOS image sensor," in *2017 XVII Workshop on Information Processing and Control (RPIC)*, pp. 1–6, IEEE, sep 2017.

[17] R. Biswas, *An Embedded Solution for JPEG 2000 Image Compression Based Back-end for Ultrasonography System*. PhD thesis, IIT, Kharagpur, 2018.

[18] T. Yanagisawa, T. Ikenaga, Y. Sugimoto, K. Kawatsu, M. Yoshikawa, S.-i. Okumura, and T. Ito, "New NEO search technology using small telescopes and FPGA," in *2018 IEEE Aerospace Conference*, vol. 2018-March, pp. 1–7, IEEE, mar 2018.

[19] M. Perez, *DETECCIÓN DE RADIACIÓN IONIZANTE UTILIZANDO SENSORES DE IMAGEN CMOS COMERCIALES*. PhD thesis, 2018.

[20] I. Micron Technology, "1 / 2-Inch Megapixel CMOS Digital Image Sensor MT9M001C12STM (Monochrome)," pp. 1–35, 2004.

[21] IEEE Computer Society, *IEEE Standard for Ethernet*, vol. 2018. 2018.

[22] IEEE Computer Society, *Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications IEEE Computer Society Specific requirements Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, vol. 2012. 2016.

[23] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, and Philips, *Universal Seral Bus Specification*, vol. Revision 2.0. 2000.