

# Capítulo 1

## Introducción

El presente informe busca dar a conocer al lector las tareas y actividades desarrolladas por el autor, en el marco del Trabajo Final de la carrera Ingeniería Electrónica, dictada en la Facultad de Ingeniería de la Universidad Nacional de San Juan. El objetivo del trabajo es diseñar e implementar una interfaz para la transmisión de datos hacia una computadora personal (PC), adquiridos por sistemas desarrollados en arreglos de compuertas de campo programables (FPGA) para aplicaciones científicas, a través del Bus Serial Universal (USB). A lo largo de este documento, se comprenderá la problemática que se resuelve y la configuración, fundamentos y modo de uso del sistema propuesto.

En la sección 1.1 se presentan las motivaciones de este trabajo y se detalla la problemática a resolver. Luego, se detallan los objetivos que persigue este trabajo. Seguido a esto, se otorga un esquema que describe la solución planteada y se justifica el protocolo elegido. Finalmente, se repasan algunos conceptos importantes de la norma USB que luego se utilizan en el trabajo desarrollado.

### 1.1. Motivación

El grado de avance que han experimentado la electrónica y la tecnología en general, gracias a la industria de los semiconductores, permite que la producción científica pueda adquirir una gran cantidad de datos. Para llevar a cabo la producción del conocimiento, es necesario el relevamiento y registro de diferentes tipos de magnitudes físicas y/o químicas sobre el objeto o proceso a investigar. En muchas ocasiones, estas magnitudes resultan difíciles de observar y cuantificar, por lo que es conveniente transformar las variables a conocer en otras más sencillas de medir. Para este propósito, se utilizan transductores.

Se conoce como transductor a cualquier dispositivo que recibe estímulos energéticos de una condición, situación o fenómeno físico y/o químico y los convierte en una señal asociada y definida de otra forma de energía[1, 2]. En otras palabras, los transductores son conversores de energías[1, 2, 3]. Se denomina sensor a una clase particular de transductor que genera, como variable de salida, una señal eléctrica que está especialmente adaptada para ser ingresada en un circuito electrónico, o adecuada al sistema de medida que se utilice [4, 5, 6].

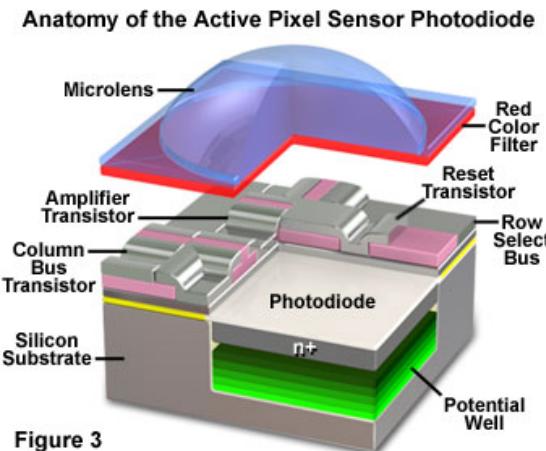


Figura 1.1: Esquema físico de un APS[8]

Las altas escalas de integración de circuitos alcanzadas en la actualidad posibilitan el diseño de sistemas sensoriales cada vez más complejos, en los cuales se logra agrupar miles de sensores en áreas reducidas, obteniendo medidas simultáneas y flujos crecientes de datos. Este trabajo se centrará en la transmisión de datos provenientes de sensores de imagen, uno de los desarrollos que se encuentra en boga.

Una imagen, desde un punto de vista digital, es un arreglo bidimensional de números, los cuales pueden ser exhibidos en una pantalla en forma de intensidad y colores de luz. Cada punto del arreglo que se muestra en pantalla se denomina pixel, acrónimo del inglés *PIcture EElement*, o elemento de imagen. Por esto, un sensor de imagen puede estar compuesto, bien por un arreglo bidimensional de sensores lumínicos (como la cámara de un teléfono celular), como por un transductor que es simultáneamente desplazado y medido, (método utilizado, entre otras, para la microscopía de fuerza atómica [7]), o por una combinación de ambos métodos. Por ejemplo, un scanner posee un arreglo lineal de transductores que son desplazados a través de la hoja para generar una imagen digital. En cualquiera de los casos, es de suma utilidad que la lectura de imágenes sea realizada en el menor tiempo posible, ya que cada imagen conlleva una cantidad no menor de datos.

Uno de los trabajos que más aportó al desarrollo de sensores de imágenes modernos, fue la introducción de los APS (*Active Pixel Sensor*, o sensor de píxeles activos) [9]. Este sensor integra en un proceso CMOS (acrónimo inglés de Metal-Óxido-Semiconductor Complementario, que es el método actualmente más económico para integrar transistores en una única pastilla de silicio), un fotodiodo, un transistor de reset (utilizado para controlar el tiempo de integración, es decir, de exposición a la luz) transistores de selección (utilizados para conectar un pixel determinado dentro del arreglo) y un amplificador seguidor de fuente en cada pixel[8]. El fotodiodo, previamente cargado, transduce la luz en una descarga eléctrica y el amplificador convierte la carga remanente en tensión para facilitar su lectura. La Figura 1.1 muestra el dibujo de un APS. Se observa el área sensible a la luz y los diferentes transistores que intervienen en su funcionamiento. Además se incorpora una micro-lente cuya función es la de enfocar los fotones sobre el área sensible y un filtro utilizado para identificar colores. En el caso de sensores

monocromáticos, se omite la colocación del filtro de color durante la fabricación.

A partir del desarrollo de los APS, se fue perfeccionando el método hasta obtener circuitos integrados con mayor cantidad de pixeles y que pueden tener diversas aplicaciones. Por ejemplo, en los trabajos [10] y [11] se presentan sensores CMOS basados en la arquitectura MIMOSA (de *Minimum Ionizing particule MOS Active pixel Sensor*). Estos sensores se desarrollaron con el objetivo específico de detección de radiación ionizante.

También existen desarrollos de sensores de radiación a través de APS comerciales. Perez *et al.* identificaron eventos producidos por partículas alfa en campos de radiación mixtos mediante el procesamiento de imágenes adquiridas con sensores comerciales CMOS[12] y desarrollaron detectores de neutrones térmicos con sensores APS cubiertos con una capa de Gd<sub>2</sub>O<sub>3</sub>[13]. Galimberti *et al.* utilizaron un sensor de imágenes comercial para realizar un detector de gas Rn en el ambiente[14]. En otro trabajo, Hizawa, *et al.* fabricaron un sensor que adquiere imágenes midiendo el pH con cada uno de los pixeles[15], pudiendo observar de fenómenos químicos en tiempo real.

Como se mencionó antes, una imagen digital es un arreglo de datos. Esto quiere decir que un sensor de imágenes con  $n$  pixeles de largo y  $m$  de ancho, captura  $n \times m$  datos en cada lectura. A su vez, para digitalizar valores, un circuito debe poseer, al menos, un conversor analógico-digital (ADC) de  $x$  cantidad de bits, lo que implica que cada dato estará compuesto por  $x$  dígitos binarios, es decir, un volumen importante de datos por cada lectura. Como ejemplo, un sensor comercial VGA, en su configuración más básica, posee 640 líneas horizontales y 480 verticales, con una resolución de 8 bits por cada pixel, lo que otorga 2.457.600 bits por cada lectura del sensor.[16] Si además se incorpora la cantidad de imágenes que se toman en función del tiempo (cuadros por segundo o fps), nos otorga un flujo de datos para nada despreciable.

Desde el punto de vista de la electrónica digital, para poder adquirir y transmitir grandes volúmenes de datos, se requiere de circuitos que sean capaces de operar a altas frecuencias de conmutación. El diseño de dichos circuitos no es trivial, ya que cuando las longitudes de onda de las señales presentes son comparables con las dimensiones físicas de dichos circuitos, debe considerarse el uso de líneas de transmisión[17]. Esto implica que no se puede diseñar utilizando un criterio de uniformidad en los parámetros y exige un análisis mas detallado y preciso.

Otro problema que presentan los circuitos electrónicos digitales tiene que ver con los tiempos de propagación de las corrientes y tensiones que circulan a través de ellos. Cuando se aplica un impulso en un conductor, debido a las capacidades propias de los materiales utilizados, las tensiones pueden demorar unos instantes en establecerse. Puede suceder que varias señales lleguen a los puertos de un dispositivo por conductores con distintas longitudes y generen retardos diferentes. Esto puede ocasionar un comportamiento indeseado si no se toman los recaudos adecuados.

Aún suponiendo un perfecto diseño, los circuitos digitales de alta velocidad se encuentran limitados en la frecuencia de conmutación por la temperatura que se necesita disipar. La potencia consumida por estos dispositivos es proporcional a la frecuencia de funcionamiento[18]. Parte de esta potencia se transforma en calor y produce un aumento en la temperatura. Si el incremento

es indiscriminado, puede destruir los circuitos.

Una posible solución para disminuir la frecuencia de las señales sin perjudicar la tasa de transferencia es la incorporación de varios conductores para enviar datos en paralelo. La cantidad de conductores a través de los cuales circula la información, se denomina ancho de bus. Idealmente, para lograr una tasa de transferencia determinada, se podría disminuir la frecuencia tantas veces como conductores se agreguen. Por ejemplo, transmitiendo por cuatro filamentos, se podría enviar la misma información a un cuarto de la frecuencia que se necesitaría con uno solo de iguales características.

Existen distintas tecnologías para efectuar la lectura de los datos generados por los sensores y su posterior transmisión. La incorporación y evolución de microcontroladores permite capturar y procesar volúmenes crecientes de datos. Sin embargo, este tipo de dispositivos posee una estructura rígida: su capacidad de procesamiento se encuentra limitada a una instrucción por ciclo de reloj y a un ancho de bus definido. Para aumentar los volúmenes de datos que circulan a través de ellos, no es posible aumentar el ancho de bus, sino que se torna necesario incrementar la frecuencia de funcionamiento, generando los problemas anteriormente detallados.

Una solución óptima, sin considerar los costos asociados a esto, sería el desarrollo de un circuito integrado de aplicación específica (ASIC del inglés *Application Specific Integrated Circuit*). En este tipo de circuitos, el diseñador elabora un circuito que puede operar a altas velocidades y, a su vez, obtener un ancho de bus sin restricciones, más que las dimensiones físicas del área donde será realizado el circuito. Sin embargo, cuando sí se considera el costo asociado a este enfoque, se vuelve una solución ineficiente en bajas cantidades. La manufactura de este tipo de dispositivos puede tener un costo de miles hasta cientos de miles de dólares, dependiendo del proceso de fabricación utilizado. Gran parte de estos costos son no recurrentes, es decir, solo se pagan una vez por proyecto. En grandes cantidades de dispositivos, este tipo de soluciones se vuelven más convenientes.

Otro enfoque, es la utilización de Arreglos de Compuertas Programables por Campo (FPGA, acrónimo del inglés *Field-Programmable Gate Array*). Un FPGA es un dispositivo electrónico que posee la capacidad de sintetizar casi cualquier circuito digital. En esencia, es una matriz de bloques lógicos (también llamadas *slices* o celdas lógicas, dependiendo del fabricante), que contienen Tablas de Verdad(LUTs o *Look-Up-Table*) y flip-flops (ff), entre otras cosas, y pueden ser interconectadas entre sí, según el criterio del usuario. Así, permite implementar una solución digital en un circuito físico, a diferencia de los microcontroladores, lo realiza a través de un algoritmo almacenado en una memoria, incorporando la ventaja de definir el ancho de bus necesario para relevar una gran cantidad de datos y transmitirlos a frecuencias de trabajo menores, además de ejecutar tareas en paralelo, disminuyendo los tiempos de procesamiento. A su vez, al ser implementado en un área muy pequeña, debido a la integración del sistema, este tipo de sistemas puede trabajar a frecuencias muy elevadas, lo que implica una mayor tasa de datos aún. A pesar de la gran diversidad de precios existentes en el mercado, una FPGA de costos menores a la centena de dólares suele tener muy buenas prestaciones para la mayor parte de las aplicaciones.

Existen diversas publicaciones en donde se observa el uso de FPGAs para la implementación de sistemas que producen imágenes. Por ejemplo, el desarrollo de un detector de radiación

ionizante utilizando una sensor de imagen CMOS comercial. Para ello, los autores utilizaron una FPGA para configurar diversos parámetros del sensor con el fin de generar estrategias para la identificación de partículas alfa en campos de radiación mixtos y transmitir imágenes a una computadora personal (PC) a través de un puerto UART[19]. Se denomina ultrasonografía a la técnica de adquirir imágenes basándose en reflexiones de ultrasonido. Sus aplicaciones son múltiples, en las que se destaca el diagnóstico médico debido. Un trabajo reciente desarrolló un sistema que mejora la obtención de ecografías médicas con bajo costo utilizando una FPGA[20]. El autor presentó un algoritmo para la supresión de ruido de impulso en tiempo real para imágenes codificadas como JPEG 2000 realizado y probado en Matlab e implementado en una FPGA. Yanagisawa *et al.*, desarrollaron un sistema con telescopios pequeños para explorar objetos de campo cercano con la finalidad de monitorear cuerpos celestes que puedan colisionar con el planeta[21]. En este trabajo, se aprovechó la velocidad de los circuitos implementados en FPGA para minimizar el tiempo de adquisición.

El desarrollo de nuevos sensores brinda a los investigadores un gran volumen de datos. En muchos casos, la obtención de datos por si misma no otorga información, sino que es necesario procesar y analizar los mismos. La invención y evolución de las computadoras, como así también el desarrollo de nuevos algoritmos, dan lugar a procesamiento de datos cada vez más complejos en tiempos mucho menores. Las primeras ENIAC, computadoras de propósito general desarrollada en el año 1946 para el cálculo de tablas balísticas de las fuerzas armadas estadounidenses, podía ejecutar 20 operaciones cada  $10\ \mu s$  [22], es decir, ejecutaba instrucciones con una frecuencia máxima de  $200\ kHz$ . A su vez, tuvo un costo aproximado de U\$S 500.000, pesaba 5 t y consumía 175 kW. En contraste con aquello, es posible conseguir en el mercado actual, computadoras con tamaño y peso reducido, que ejecutan instrucciones en cuestión de nanosegundos, (5 órdenes de magnitud menos), consumen menos de 1 kW y cuestan algunos cientos de U\$S. A tal punto ha evolucionado esta tecnología, que se cuenta con computadoras muy potentes en casi cualquier laboratorio, oficina u hogar. La capacidad de cálculo que exhiben estos dispositivos, sumada al desarrollo de nuevos métodos y algoritmos de cálculo, permite a los investigadores procesar datos en tiempo reducido, facilitando el análisis y la generación de nueva información.

En todos los casos que se consideran en este trabajo, la generación de datos y el procesamiento de lo mismos se da en sistemas diferentes. Es decir, los datos son relevados por los sensores y adquiridos luego por los FPGAs. Finalmente llegan a una PC para su posterior procesamiento y análisis. Se requiere, por tanto, de una conexión a través de la cual los datos puedan ser transferidos del sistema de adquisición, la FPGA, a la PC y viceversa. Se torna de suma utilidad, entonces, proveer una comunicación efectiva y robusta que permita transmitir grandes volúmenes de datos en poco tiempo, y de esta forma facilitar los tiempos de desarrollo, pruebas, depuración, procesamiento y análisis.

La implementación de un sistema de comunicación en una FPGA puede ser resuelta de muchas maneras, quedando a criterio del desarrollador utilizar algún protocolo estándar, o bien diseñar uno propio. Sin embargo, en una computadora, las formas de comunicar datos se vuelven un poco más restrictivas y acotadas a los puertos y señales que puede manejar el equipo, conforme el fabricante haya establecido. Este trabajo busca implementar una comunicación entre una computadora personal y una FPGA, utilizando un protocolo estándar, que esté disponible

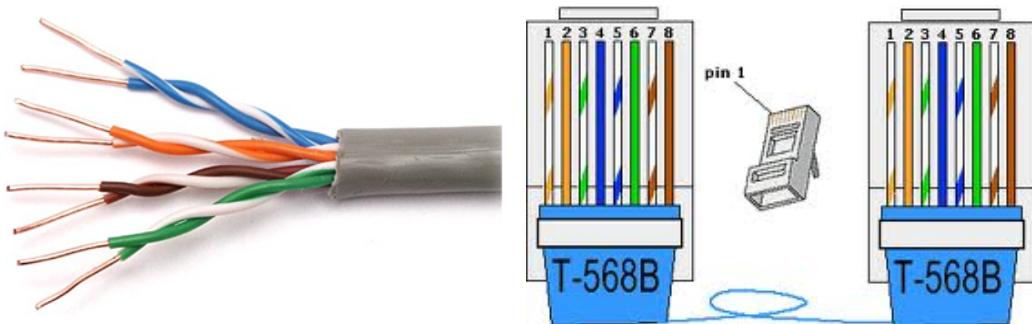


Figura 1.2: Par Trenzado y un dibujo de su ficha de conexión.

en cualquier computadora comercial y que posea una tasa de bit suficiente para poder transmitir imágenes.

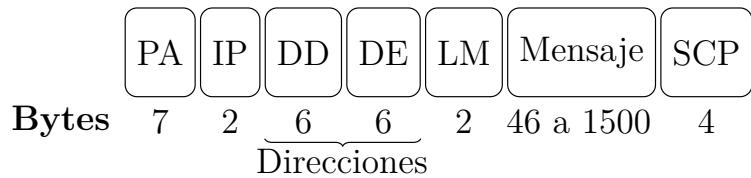
## 1.2. Protocolos disponibles para la transmisión de datos entre PC y FPGA

El estándar más exigente de la norma americana de la SCTE (Sociedad de Ingenieros de Comunicación por Cable) utilizada Televisión Digital, posee una tasa de  $38.8 \text{ Mbit s}^{-1}$ [23]. Por su parte, la serie de sensores para adquirir imágenes monocromáticas MT9M001, comercializado por ON Semiconductors posee  $1280 \times 1024$  pixeles, con profundidad de 10 bits y puede operar hasta a 30 cuadros por segundo[24]. La tasa de transmisión necesaria es, por tanto, de  $393.2 \text{ Mbit s}^{-1}$ .

Un requerimiento que posee cualquier periférico informático es el de compatibilidad. No es conveniente utilizar puertos que requieran acceso a la placa madre, como el caso de tarjetas de tipo PCI o PCI express, debido a que no todos los equipos los tienen accesible, como ser computadoras portátiles, y en algunos casos estos pueden estar todos ocupados. Se opta, entonces, por alguno de los tres puertos de moda: Ethernet, dedicado principalmente a conexión de redes mediante cables; Wi-Fi, utilizado para el accesos a la red de forma inalámbrica; y USB, dirigido a la comunicación de periféricos con la PC.

Al hablar de Ethernet o Wi-Fi, se hace referencia a dos formas diferentes de conectarse a una red de computadoras. En otras palabras, se habla de dos o más nodos, compuestos por PCs o cualquier dispositivo electrónico con capacidad de realizar cálculo binario, que pueden intercambiar datos a través de una trama bastante compleja de componentes diferentes. Ambos protocolos hacen referencia solo a la conexión física de los dispositivos y el control de acceso de cada uno de ellos a la conexión. Quedando a cargo de otros sistemas, con sus protocolos, que los datos enviados puedan ser correctamente recibidos por el usuario de la PC. La gran diferencia entre ellos radica en el medio físico que utilizan: Wi-Fi emplea ondas electromagnéticas emitidas mediante radiofrecuencia, mientras que en Ethernet, estas ondas son acarreadas por uno o más conductores, como ser cable coaxial, cables de par trenzado o fibra óptica.

Ethernet, también conocido como IEEE 802.3, es una norma que define cómo se deben



### Referencias

**PA:**Preámbulo

**IP:** Inicio de Paquete

**DD:** Dirección de Destino

**DE:** Dirección de Emisión

**LM:** Longitud del Mensaje

**SCP:** Secuencia de chequeo del paquete

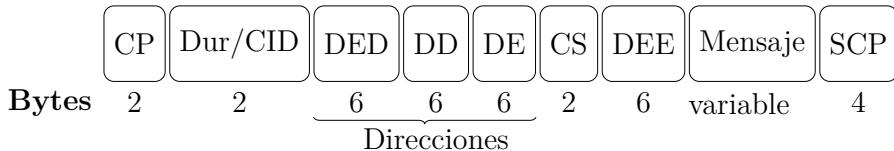
Figura 1.3: Estructura de un paquete Ethernet

conectar nodos a través de conductores para conformar redes de área local (LAN o *Local Area Network*), es decir, redes pequeñas, como ser domésticas, de oficinas o de pequeñas empresas, de forma que puedan transmitir información a velocidades seleccionables entre 1 Mbit/s y 400 Gbit/s [25]. Utiliza una tecnología denominada Acceso Múltiple Sensando la Portadora con Detección de Colisiones (CSMA/CD del inglés *Carrier Sense Multiple Access with Collision Detection*). En una red con CSMA/CD, cada dispositivo debe sensar en forma permanente la conexión a la red, es decir, no existe un dispositivo que dirija el uso del bus, sino que cada uno debe identificar el estado de la red. Los mensajes se envían modulados. Cuando una señal portadora es detectada, todos chequean la dirección del paquete de información que viaja y el mensaje es recibido solamente por el dispositivo que se corresponda con esa dirección. Siempre que exista una señal portadora en el bus, los dispositivos que deseen transmitir información deberán esperar a fin de evitar colisiones, o sea, que dos dispositivos envíen mensajes a la vez y estos se interfieran.

Dependiendo de la frecuencia de la portadora y la tasa de transferencia a la que transporta el mensaje, la norma especifica el conector y la distancia máxima a la que debe conectarse una repetidora, es decir, un dispositivo que reciba, reconstruya y emita la señal recibida. Estos conectores pueden ser cable coaxial, fibra optica o cable de par trenzado. Este último es el más usual en las PCs comerciales y se muestra, junto a su ficha característica en la Figura 1.2.

La información se estructura en paquetes para permitir la comunicación entre muchos nodos de la red. Un paquete, como se observa en la Figura 1.3, se compone de un preámbulo con 7B que sirve para sincronizar los dispositivos en cada extremo de la conexión, 1 B de inicio, 12 B de direcciones, que corresponden 6 al nodo destinatario y 6 al emisor respectivamente, 2 B que indican la longitud del mensaje, entre 46 y 1500 B de datos y 4 B para la verificación de la transmisión. Otra definición importante de la norma, son las características eléctricas de las señales, pero no se detallan en este trabajo porque varían en función de la velocidad del puerto.

Por su parte Wi-Fi, perteneciente a la asociación de compañías denominada Wi-Fi Alliance, se rige por la norma que estableció esta última. Existe una norma equivalente, encuadrada en la especificación IEEE 802.11, referida a las redes de area local inalámbrica, o WLAN (siglas del



### Referencias

**CP:** Control de Paquete

**Dur/CID:** Duración del paquete/Identificación de conexión

**DED\*:** Dirección de Enrutador de Destino

**DD\*:** Dirección de Destino

**DE:** Dirección de Emisión

**CS\*:** Control de Secuencia

**DEE\*:** Dirección de Enrutador de Emisión

**SCP:** Secuencia de chequeo del paquete

\*Pueden no estar dependiendo del tipo de mensaje

Figura 1.4: Estructura de un paquete Wi-Fi

ingles *Wireless Local Area Network*). Wi-Fi se enfoca en las que se refieren a las comunicaciones de radiofrecuencia con portadora de 2.4 GHz, que se incorporan en las revisiones b, g y n de la norma IEEE. IEEE 802.11 está pensado especialmente para dispositivos portátiles y móviles. La norma define a los dispositivos portátiles como aquellos que pueden ser trasladados con facilidad pero operan estáticos y los móviles se identifican por poder trabajar en movimiento [26]. La principal característica que posee este tipo de comunicación es la falta de conductores para la elaboración de la red, sin contar las conexiones entre los transceptores que emiten y reciben las señales de radiofrecuencias y los nodos, en donde la información es producida y/o consumida. En cuanto al formato del paquete de datos, el cuál se muestra en la Figura 1.4, es bastante similar al de Ethernet. En primer lugar, se envían dos bytes de control que indican el tipo de paquete a enviar. Luego siguen dos bytes que, dependiendo de la etapa de la comunicación puede indicar la duración del mensaje a transmitir o un identificador de una conexión establecida previamente. Siguen entre 6 y 18 bytes de direcciones del enrutador que recibe los datos, el nodo emisor y el destinatario. Continúan, dos bytes de control de secuencia se utilizan para fragmentar transmisiones largas. Continua un campo más para dirección que corresponde a la red emisora de 6 bytes. Todos los campos de dirección pueden variar en función del tipo de mensaje que se envía. Los últimos dos campos de la trama corresponden a la información que se quiere comunicar (hasta 2312 bytes) y un código de chequeo por redundancia cíclica de 32 bits (4 bytes).

Existen múltiples ventajas de utilizar radiofrecuencias para conectarse a la red, tales como la libertad de mover el punto de trabajo y la economía a la hora de armar redes con muchos nodos. Sin embargo, posee algunas desventajas notorias, propias del medio de propagación, que lo hacen no tan óptimo para los fines del presente trabajo. Las redes inalámbricas tiene la característica de que no son del todo confiables: posee múltiples fuentes de interferencia, ya que varias tecnologías que utilizan la misma frecuencia (Bluetooth, Zig-Bee, WUSB, microondas). A su vez, suele presentar variaciones temporales y asimetrías en las propiedades de propagación, lo que puede provocar interrupciones en la comunicación.

Ambos protocolos proporcionan una solución de conexión de redes de nivel físico y ejecutan

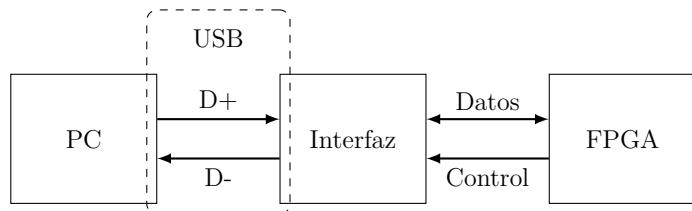


Figura 1.5: Esquema propuesto para implementar la comunicación

tareas de control de acceso al medio (MAC) a fin de evitar colisión en los datos, es decir, que dos dispositivos transmitan en forma simultánea e interfieran la comunicación. Sin embargo, para establecer una red, faltan componentes físicos y lógicos tales como un sistema de control enlace lógico (Logic Link Control), un sistema de direccionamiento, como el Protocolo de Internet (IP), una capa de transporte de datos, (como el protocolo TCP) y las capas de software que permiten acceder a los protocolos anteriormente mencionados.

A pesar de lo anterior, es posible establecer comunicaciones punto a punto con ambos protocolos, simplificando mucho el sistema de transmisión de datos. Sin embargo esta solución presenta un inconveniente no menor: se le quita a la PC un acceso a la red, que en la mayoría de los casos es el único. Esto no es deseable ya que la conectividad es un requisito fundamental en cualquier hogar u organización, ya sea empresarial, gubernamental, científica o de cualquier tipo.

Por su parte el protocolo USB (acrónimo de *Universal Serial Bus*), es una norma desarrollada por seis de las empresas más grandes de la industria informática, pensada y desarrollada para la conexión de teléfonos y periféricos a PCs [27]. En la versión original, USB posee conectores cableados de 4 conductores y presenta una topología de bus, es decir todos los dispositivos se conectan a un mismo circuito conductor. La conexión es manejada por una PC y solo transmite o recibe un dispositivo a la vez. Tal fue la penetración de USB en el mercado, que se transformó en una norma de facto. Actualmente es incorporada casi por defecto en casi todas las computadoras disponibles en el mercado y es necesaria a la hora de comprar e instalar periféricos.

USB presenta diferentes versiones de su norma, cada cual con una o más tasas de transmisión y señalización. La versión 1 posee dos revisiones, 1.0 fue lanzada al mercado en el año 1996 y 1.1 que se presentó en Agosto de 1998. La primera alcanza una tasa máxima de  $1.5 \text{ Mbit s}^{-1}$  y la segunda hasta  $12 \text{ Mbit s}^{-1}$ . USB 2.0 fue presentado en Septiembre del 2000 y es capaz de transmitir a  $480 \text{ Mbit s}^{-1}$ . La tercera versión, USB 3.0, fue lanzada al mercado en 2011 y transmite a una tasa de  $5 \text{ Gbit s}^{-1}$ . Esta última versión fue revisada en julio de 2013 y en septiembre de 2017, ofreciendo  $10 \text{ Gbit s}^{-1}$  y  $20 \text{ Gbit s}^{-1}$  respectivamente.

Se elige para el desarrollo de este trabajo la norma ya que USB 2.0 presenta una tasa de transferencia de datos suficiente para la transmisión de imágenes. A su vez, resulta ideal para los objetivos buscados debido a encontrarse presente en la mayoría de las computadoras y no interferir en la conexión a internet de las mismas. En el Capítulo 2 se profundizarán en conceptos específicos de la norma USB.

Es posible implementar una comunicación USB completa a través de una FPGA. Sin embargo, esto sería demasiado oneroso en términos de tiempos de desarrollo y de recursos de FPGA disponibles para la implementación de otros sistemas, los cuales son el objetivo de la comunicación. Se plantea, entonces, un esquema como el que se observa en la Figura 1.5 en la cual se utiliza una interfaz externa al FPGA. La comunicación USB propiamente dicha será efectuada entre la interfaz y la PC, mientras que se plantea una comunicación diferente entre la interfaz y el FPGA. Este último, por su parte, tendrá la tarea de realizar el control de esta comunicación.

## 1.3. Objetivos

### 1.3.1. Objetivo Principal

El objetivo del presente trabajo es obtener una comunicación USB 2.0 de alta velocidad entre una PC y un FPGA.

Esta comunicación debe realizarse y documentarse de forma tal que pueda ser usado posteriormente en aplicaciones científicas desarrolladas con FPGA's.

### 1.3.2. Objetivos Particulares

Para la consecución del objetivo general, se deben cumplir los siguientes objetivos particulares:

- Comprender el funcionamiento del protocolo USB.
- Seleccionar los componentes a utilizar.
- Configurar los componentes seleccionados.
- Desarrollar un núcleo en VHDL que sirva de interfaz.
- Diseñar e implementar la interconexión de los componentes seleccionados.
- Verificar el sistema desarrollado.
- Desarrollar un documento que explique el modo de uso del código VHDL utilizado.

## 1.4. Estructura del Informe

El presente informe se divide en 2 bloques principales: uno referido al desarrollo del sistema y el siguiente a su forma de uso y verificación.

Dentro del bloque referido al desarrollo del sistema, se encuentran los primeros 5 capítulos:

1. **Introducción:** En este capítulo se intenta exponer lo que motiva el presente trabajo, la propuesta que da solución a la motivación, el objetivo y alcance que el trabajo busca y la estructura del mismo. Se brindan, además, conceptos importantes de la norma USB que son significativos para los objetivos de este trabajo.
2. **Elección de las herramientas para la realización de la interfaz:** Se describe aquí todas las herramientas de las que se vale este trabajo para cumplir con los objetivos propuestos.
3. **??:** Se presenta la arquitectura, configuración y código desarrollado para el presente trabajo, como así también las herramientas específicas provistas por el fabricante, que facilitan el desarrollo.
4. **??:** Este capítulo detalla lo desarrollado para implementar la comunicación entre la FPGA y la interfaz. Se expone una maquina de estados descrita en VHDL y sintetizada en FPGA. También se describe un circuito impreso realizado para conectar ambas partes.
5. **??:** Se desarrolla las tareas desarrolladas a fin de realizar las depuraciones del sistema y la verificación del cumplimiento de las especificaciones.

## 1.5. Sumario del capítulo

En el presente capítulo se expuso la necesidad de la elaboración de un sistema de comunicación que permita la transferencia de datos entre una PC y un FPGA para ser utilizados por sistemas implementados con este último dispositivo. Se planteó una solución utilizando una interfaz comercial que sirve de intermediario entre estas herramientas y se brindó una justificación del empleo del protocolo USB 2.0 de alta velocidad como la implementación óptima del sistema. Se presentó también la estructura del presente informe y se dieron algunos detalles relevantes para este trabajo de la norma USB.



## Capítulo 2

# Bus Serial Universal 2.0

El Bus Serial Universal, o USB por sus siglas en inglés, es un sistema de comunicación diseñado durante los años 90 por seis fabricantes vinculados a la industria informática, Compaq, Intel, Microsoft, Hewlett-Packard, Lucent, NEC y Philips, con la idea de proveer a su negocio de un sistema que permita la conexión de PCs con teléfonos y periféricos con un formato estándar, fácil de usar y 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 características, tales como fichas, niveles de tensión, velocidades, drivers, lo cuál dificultaba al usuario estar al día y poder utilizar cada componente que compraba. Esto también complicaba a las mismas empresas productoras, por que la introducción de un nuevo sistema requería de mucho soporte extra para poder conectar lo existente en forma previa.

Todo esto, quedó saldado con el aparición de la norma USB que, gracias a la gran cuota de mercado de sus desarrolladores, fue adoptada en forma rápida y se transformó en la especificación casi por defecto a la hora de seleccionar un protocolo para periféricos. La penetración en el mercado fue tal que hoy, más de 20 años después, es difícil encontrar PC con otro tipo de puertos, salvo que en el momento de compra se solicite de manera especial. No obstante, cualquier PC nueva disponible en el mercado debe poseer puertos USB para la conexión de, al menos, los periféricos.

El diseño de la norma USB busca resolver tres problemáticas interrelacionadas, que son: La conexión de teléfonos con las PC, la facilidad de uso, es decir, que el usuario solo conecte su dispositivo y pueda utilizarlo, y la expansión en la cantidad de puertos disponibles para conectar periféricos[27]. Para satisfacer estas tres demandas, la norma USB 2.0 busca alcanzar un conjunto de metas que apuntan a la facilidad del uso, la compatibilidad entre versiones diferentes de la misma tecnología, la robustez en el flujo de datos, y la convivencia de diferentes configuraciones temporales en único bus, provistos de una interfaz estándar, ancho de banda que soporte comunicaciones audiovisuales de calidad aceptable y un bajo coste.

El presente capítulo intenta ser un breve resumen con los aspectos más relevantes de la norma en cuanto a su composición física, su topología, los dispositivos que intervienen, la importancia de los mismos y como los datos son transmitidos desde y hacia una PC.

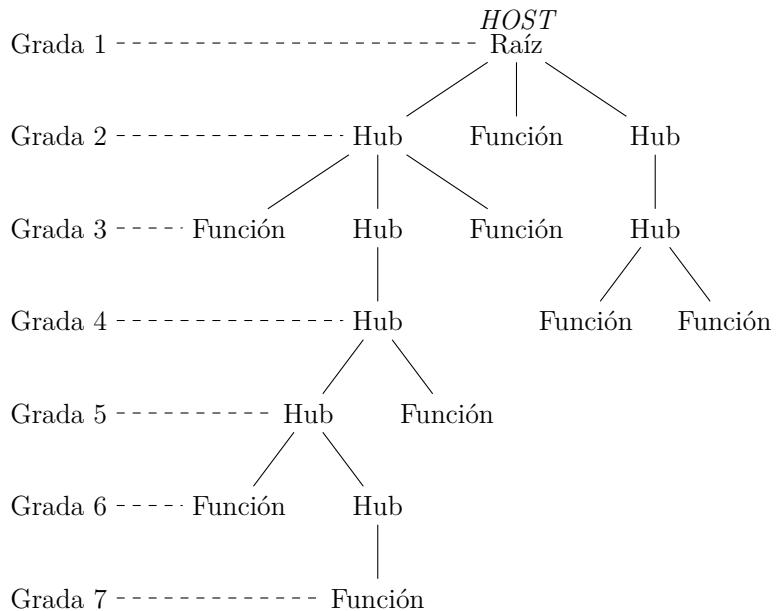


Figura 2.1: Topología de un sistema USB

## 2.1. Descripción general de un sistema USB

Un sistema USB posee un esquema en forma de árbol cuyo nodo principal es el host. Es decir, la comunicación se realiza siempre a través de una única línea a la que se conectan todos los dispositivos (bus). Dado el campo de direcciones provisto por la norma, un sistema USB puede conectar hasta 128 dispositivos. El acceso al bus es administrado por un maestro. El maestro se encarga de solicitar a cada uno de los dispositivos su intervención. Posteriormente, el dispositivo debe responder al pedido del maestro. Este esquema es lo que se conoce como maestro-esclavo. De esta forma, el sistema se asegura que el bus sea utilizado por un dispositivo a la vez para enviar o recibir datos.

En un sistema USB no cualquier dispositivo puede ser maestro. Este rol lo cumple solo uno: una PC, o cualquier dispositivo con capacidad de llevar a cabo las tareas asignadas (que se detallan más adelante); denominado Host por la norma. La palabra *HOST* proviene del habla inglesa y se traduce como anfitrión, aunque en la jerga se conoce comúnmente por su nombre en inglés.

La topología del bus, la que se observa en la Figura 2.1, posee forma de árbol, es decir, puede ser pensada como una comunicación vertical, donde en el punto más alto se encuentra el Host. Siguiendo hacia abajo, el bus puede encontrar dos tipos diferentes de dispositivos: Funciones, cuyo rol es el de proveer una utilidad al sistema, como ser la de captura de imagen, reproducción de audio o el ingreso de comandos; y Hubs (concentradores o distribuidores), que se encargan de conectar una o más funciones al sistema. La norma USB establece gradas, en donde cada Hub introduce una nueva grada que contiene a las Funciones conectadas. Por cuestiones de

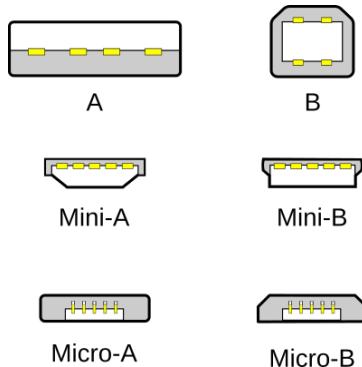


Figura 2.2: Tipos de conectores USB. Los tipo A deben ser usados en el extremo del Host y los tipo B hacia los periféricos[28]

restricciones temporales y tiempos de propagación en los cables, no se permiten más de 7 gradas, incluyendo al Host en la primera. Es decir, no se puede conectar más de 5 Hubs en cascada. La grada 7 sólo puede contener Funciones[27].

Cada uno de estos dispositivos diferentes, se interconectan entre sí a través de cables y conductores específicos, diseñados en forma tal que no sea posible conectarlos en forma equivocada. Para cumplir con la norma, el Host debe tener siempre un zócalo compatible con conectores tipo A y los periféricos para enchufes de tipo B. Se observan las diferencias entre uno y otro en la Figura 2.2. Los cables de conexión poseen dos pares de conductores: uno para la señal de alimentación de 5 V ( $V_{BUS}$  y  $GND$ ) y otro para el flujo de datos ( $D+$  y  $D-$ ).

A nivel eléctrico, la interconexión de datos en los dispositivos se lleva a cabo a través de una codificación de inversión sin retorno a cero, es decir, el cambio de nivel de tensión representa un '0' y el no cambio representa un '1'. Además, la señal de datos es diferencial. Esto implica que cuando  $D+$  es positivo,  $D-$  debe ser negativo y viceversa.

Cabe destacar que, al tener una señal diferencial, la norma USB es half-duplex, es decir, puede transmitir en los dos sentidos (desde Host hacia Funciones y viceversa), pero no puede hacerlo en simultáneo[29], sino que primero debe transmitir un dispositivo y, al finalizar este, el otro es el que tiene acceso al bus.

La velocidad de conmutación en los niveles de tensión de la señal de comunicación puede darse a diferentes valores, dando lugar a tres tipos de tasa de bit: Alta-velocidad (*High-Speed*) implica una tasa de bit de  $480 \text{ Mbit s}^{-1}$ , Velocidad-completa (*Full-Speed*) posee una tasa de bit de  $12 \text{ Mbit s}^{-1}$  y baja-velocidad (*Low-Speed*) transmite a una tasa de bit de  $1.5 \text{ Mbit s}^{-1}$ .

Cuando el Host se comunica con las diferentes Funciones, lo realiza a través de paquetes. Los paquetes implican que la información que se transmite a través del bus está encapsulada en un formato establecido. Cada vez que un dispositivo accede al bus, lo debe hacer de una manera particular definida por el tipo de transferencia, por su rol (Host, Hub o Función) y por el estado de la transmisión dentro del protocolo establecido.

## 2.2. Dispositivos que componen un sistema USB

Dentro de un sistema USB existen tres tipos diferentes de dispositivos: Host, Hubs y Funciones. Cada uno de ellos tiene asignado un rol específico dentro de la comunicación. Se detallan a continuación las tareas pertinentes a cada uno de ellos.

### 2.2.1. Host USB

El Host es quien comanda las comunicaciones. Este dispositivo debe tener capacidades de memoria y procesamiento necesarias para almacenar y ejecutar el software de control. A su vez, necesita de hardware que le permita llevar un monitoreo y control de los eventos que suceden en el bus. Entre las tareas que debe llevar a cabo, se encuentran:

- Detectar la conexión y desconexión de dispositivos.
- Administrar el flujo de los comandos de control con los diferentes dispositivos.
- Administrar el flujo de la información entre él (Host) y los diferentes dispositivos.
- Llevar estadísticas de actividad y estado del bus.
- Proveer potencia a los dispositivos conectados, cuando estos así lo requieran.

Debido a que las tareas que ejecuta el Host requiere una cantidad de recursos de almacenamiento y procesamiento, es usual que el sea una PC la que lleve el rol. El Host es quien inicia la comunicación con las Funciones. Las Funciones, a su vez, responden a lo que fue solicitado por el Host, cuando él lo indique.

### 2.2.2. Hubs USB

Un Hub USB tiene la función de proveer puertos al bus. El primer Hub esta incorporado en el Host y cada vez que se requiere más puertos a los cuales incorporar periféricos, se puede ir agregando a través de Hubs. Otra función importante es la de servir como interfaz entre dispositivos con diferentes velocidades, optimizando así el ancho de banda disponible para la comunicación.

### 2.2.3. Funciones USB

La norma define como Función a todo aquel dispositivo que se conecta al bus y brinda al Host la capacidad de realizar una nueva tarea. Por ejemplo, un teclado otorga un método de entrada adicional, un mouse permite manejar un puntero de la interfaz gráfica, un parlante y un micrófono posibilitan la emisión y recepción de sonidos, respectivamente. Cada una de estas utilidades, compone una Función USB. A su vez, un dispositivo que brinda más de una capacidad es visto por el Host como Funciones separadas conectadas a través de un Hub. Por ejemplo, si se piensa en unos auriculares con micrófono, como los usados por una operadora en

un centro de llamados, aunque se presenten integrados en un mismo producto y tengan un único puerto de conexión al bus, el Host los considera como dos Funciones separadas. Las Funciones, desde un punto de vista de software, son independientes unas de otras, por lo que cuando un programa, llamado cliente, necesita utilizar una de ellas, puede acceder a ésta directamente sin conocer cuantas y cuales funciones diferentes existen en el bus.

Cada Función se compone de un conjunto de extremos. Un extremo es una porción de dispositivo identificable en forma única[27]. Cada extremo tiene características definidas por el diseñador del sistema que deben estar adecuadas a los requerimientos de cada dispositivo. Los extremos tienen un solo sentido de comunicación y un tamaño máximo de mensaje a transmitir o recibir. Cuando se conecta al bus, un dispositivo debe enviar una descripción en donde consten sus extremos y las diferentes formas de configuración de cada uno, con el tipo de mensajes que soporta, el sentido de la comunicación, el tamaño, entre otros parámetros. Esta descripción se lleva a cabo través de lo que la norma llama descriptores.

Todo dispositivo debe contener un extremo con dirección cero dedicado exclusivamente al control de la Función por parte del Host. Debe, como mínimo, poder comunicarse a velocidad completa, es decir, con una señal de  $12 \text{ Mbit s}^{-1}$  y, a su vez, responder a los comandos de control básicos cómo adquirir la dirección, recibir la configuración y enviar los descriptores del dispositivo y sus diferentes configuraciones. Dependiendo de las diferentes requerimientos, el dispositivo puede incorporar otros extremos (15 de entrada y 15 de salida como máximo). Cada extremo no-cero tiene diferente latencia, acceso al bus, ancho de banda, manejo de errores, tamaño máximo de paquete soportado y dirección.

## 2.3. Paquetes USB

Los dispositivo transmiten información a través del bus con formato particular, establecido por el protocolo que dicta la norma USB. Los fragmentos de información que envía un dispositivo se denominan paquetes. Un paquete está compuesto por diferentes campos. El sistema reconoce cada campo, decodifica su información e identificar cada paquete, su emisor, el tipo de datos que envía, el sentido de circulación. Luego, corrobora que los datos transmitidos llegaron a destino en forma satisfactoria. No existe un número infinito de campos y todos pueden resumirse en el presente documento. Sin embargo, solo se detallan a continuación los que el autor considera más relevantes para el objetivo de este trabajo, quedando de lado algunos comandos, por ejemplo, inherentes a los hubs que conectan dispositivos de diferentes velocidades.

### 2.3.1. Campos de paquetes

#### Identificador de paquete

El campo identificador de paquete (PID del inglés *Packet Identifier*) le da a conocer a los distintos dispositivos el tipo de información que contiene el paquete. Por ejemplo, indica si el Host solicita envío o recibo de datos, si envía un comando o si un dispositivo está transmitiendo los datos. Se compone de un campo de 8 bits, de los cuales 4 corresponden al identificador propiamente dicho y los otros cuatro es el complemento a uno de los mismos datos, permitiendo

corroborar que no hubo una pérdida de información.

Existen 4 tipos de PID: Token, que antecede a cualquier transmisión y es emitido por el host; Data, indica paquetes que contienen datos transmitidos; Handshake, a través del cual los componentes del sistema se enteran si la comunicación fue efectiva o no y Special, cuya función no es de interés para este trabajo.

A su vez, los PID Token se dividen en 4 tipos: IN, para indicar que se va a realizar una envío de datos desde un extremo al Host; OUT, antecede a una transmisión de datos en el sentido contrario, es decir del Host a un extremo; SETUP, que señala una secuencia de comandos y SOF (del inglés Start of Frame) que da una señal de sincronismo y control.

Dentro de los PID Data, solo existen diferentes etiquetas que se usan dependiendo del tipo de transmisión. Los PID de Handshake contienen 4 mensajes diferentes: ACK para indicar que el mensaje fue recibido satisfactoriamente y NAK señala que no se pudo enviar o recibir, STALL significa que el extremo se detuvo y NYET de cuenta sobre demoras en la respuesta del receptor.

## Dirección

El campo de Dirección señala cuál es la Función que debe responder o recibir alguna directiva emitida por el host. A su vez, se divide en dos subcampos: uno que indica un dispositivo y la segunda que señala el extremo específico con el cual desea comunicarse.

## Datos

Es el campo que contiene la información útil transferida. Puede tener un largo de hasta 1024 bytes. Cada byte enviado se ordena con el bit menos significativo (LSb del inglés *Less Significative bit*) primero y el bit mas significativo (MSb por sus siglas en inglés) al final.

## Chequeos de redundancia cíclica

El campo de chequeo de redundancia cíclica (CRC) contiene verificadores para corroborar que no hubo pérdida de información. Dependiendo de que tipo de paquete se esté transmitiendo, el CRC puede tener 5 o 16 bits. Los códigos generadores son representados por las ecuaciones 2.1 y 2.2 respectivamente:

$$G(X) = X^5 + X^2 + 1 \quad (2.1)$$

$$G(X) = X^{16} + X^{15} + X^2 + 1 \quad (2.2)$$

### 2.3.2. Formato de paquetes

Cada uno de los paquetes que intervienen en la comunicación USB utilizan diferentes tipos de campos, dando lugar a distintos tipos de paquetes. La figura 2.3 muestra como se conforman algunos de ellos.

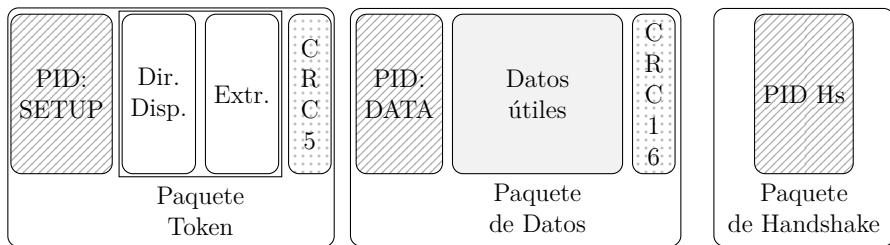


Figura 2.3: Formatos de paquetes

Un paquete de tipo Token está conformado por los campos PID, Dirección y CRC-5 (CRC de 5 bits). Un paquete Token que indica SOF en su campo PID, lleva un formato un poco diferente. En lugar de la dirección, se envía un contador de 11 bits que señala la cantidad de cuadros que han transcurrido desde la puesta en marcha del sistema, seguido de un código CRC-5.

Cada 1 ms el host transmite un SOF e incrementar el contador de cuadros. En sistemas USB 2.0 de Alta velocidad, además, se transmiten 8 subcuadros de 125 µs por cada cuadro. Cada uno de estos subcuadros inicia con un paquete SOF. Sin embargo, el host no actualizará el número de cuadros hasta pasado 1 ms.

El paquete de datos iniciará con un PID que indique que es un paquete de este tipo, luego enviará los datos desde el LSb hasta el MSb y, finalmente, enviará un código CRC-16 (CRC de 16 bits de longitud).

Los paquetes de tipo Handshake (Hs) solo envía un PID con información sobre si el mensaje fue recibido de forma correcta o no.

## 2.4. Tipos de Transferencias

Cada extremo presente en un dispositivo USB, puede estar configurado, en simultaneo, con un solo tipo de transferencias. Es importante, para el diseñador del dispositivo, entender y seleccionar el tipo de transferencia adecuada para cada uso debido a que, de ello depende las características que poseerán las comunicaciones que se efectúen.

Existen cuatro tipos de transferencias definidas por la norma USB: Transferencias de Control, transferencias en masa, transferencias isocrónicas y transferencias de interrupción. Cada una de ellas tiene un propósito y características diferentes, las que se detallan a continuación.

### 2.4.1. Transferencias de control

Las transferencias de control son utilizadas por el Host para configurar, emitir comandos y conocer el estado de los distintos dispositivos acoplados al bus. Se caracteriza por ser una comunicación de ráfagas, es decir, de corta duración, tener alta prioridad y ser no periódica. Habitualmente, son utilizadas para emitir comandos hacia los dispositivos, o bien, para conocer su estado. Sin embargo, esto no quiere decir que pueda ser empleada para transmitir mensajes que no sean específicamente de comando. Debido a la sensibilidad que los mensajes de control

poseen para el sistema USB, estos están dotados del protocolo más estricto de chequeo, corrección y/o retransmisión de datos.

Las transferencias de control poseen dos o tres etapas en su ejecución. En la primera de ellas, el Host debe enviar un Paquete Token que indique SETUP, luego envía un paquete Data con 8 B y esto es respondido por el dispositivo con un paquete Handshake indicando la recepción. Si hiciese falta enviar información extra, en una segunda etapa, el Host transmitirá un paquete Token indicando la necesidad de información. Luego, dependiendo del sentido de los datos solicitado, se enviará un paquete Data con hasta 64 B más y el receptor responderá con un paquete Handshake. Finalmente, en la última etapa, se le permite al dispositivo informar su estado. Para ello, el Host le envía un paquete Token de solicitud de datos, luego la Función responderá con un paquete Data y el Host emitirá con un paquete Handshake, indicando si recibió o no la información.

#### **2.4.2. Transferencias en masa**

Las transferencias en masa son usadas para transferir paquetes grandes en forma de ráfagas, en forma no periódica. Su utilidad consiste en que aprovecha al máximo cualquier espacio de ancho de banda disponible. Gracias al sistema de chequeo de errores, es posible solicitar retransmisiones, asegurando la integridad de la comunicación. Esta transferencia es ideal para comunicar cantidades relativamente grandes de datos que requieren una comunicación fidedigna a costa de sacrificar velocidad en los tiempos de entrega, por ejemplo, una impresora. En un bus que no posee un gran uso, los mensajes alcanzarán el destino en tiempos cortos. Sin embargo, cuando exista una gran cantidad de dispositivos conectados y el ancho del bus se encuentre congestionado, un mensaje largo puede verse demorado.

Cuando se lleva a cabo una operación de este tipo, el Host envía un paquete Token hacia el bus un PID OUT cuando desea transmitir datos o IN si desea recibirlos, la dirección de la Función y su extremo. Luego, el emisor comunica un paquete Data, y finalmente, el receptor de la transferencia responde con un paquete Handshake. Una transferencia en masa (*bulk transfer*) puede poseer un tamaño máximo de 512 B de datos por paquete transmitido.

#### **2.4.3. Transferencias isocrónicas**

El término isócrono o isocrónico está referido a sistemas digitales sincrónicos con la particularidad de que se supone que sucede una cantidad determinada de sucesos en intervalos regulares de tiempo. Esto puede ser logrado compartiendo la misma fuente de sincronismo, o bien, sincronizando los relojes de cada componente. USB envía una señal SOF cada 1 ms (o cada 125  $\mu$ s en los sistemas de alta velocidad), es posible sincronizar por frecuencia o fase, lo que permite tener este tipo de comunicaciones, aún en sistemas con señales de reloj provenientes de fuentes diferentes.

La principal característica de las transferencias isocrónicas es que son periódicas y continuas entre el Host y las Funciones. Se utiliza este tipo de transferencias para comunicar datos que pierde validez cuando no son entregados en un tiempo establecido. Para lograr esto, el Host le asigna una porción fija de ancho de banda por cada cuadro (1 ms). Debido a que los datos pierden validez a lo largo del tiempo, también lo hacen los errores, por lo que no se prevé una

retransmisión de los datos enviados por este sistema.

La ejecución de una transferencia isocrónica se da cuando el host envía un paquete Token con la dirección de un extremo de este tipo de transferencias. Luego, el emisor envía un paquete Data cuyo campo de datos puede poseer hasta 1024 B y un CRC-16. Finalmente el receptor envía un paquete Handshake. Si, dado el caso, el receptor envía un Handshake indicando que el paquete no pudo ser recibido en forma correcta, el mensaje es descartado, sin existir una retransmisión posterior del mismo paquete.

#### **2.4.4. Transferencias de interrupción**

Cuando se requiere de una comunicación con latencia asegurada pero con baja probabilidad de eventos, el tipo de transferencia óptimo para utilizar, son las transferencias de interrupción. En este tipo de trasferencias, el Host consulta cada un periodo de tiempo determinado el estado del extremo que posee este tipo de transferencias. Para ello, envía un paquete Token, luego el emisor transmite un paquete de datos con hasta 64 B, si se trata de dispositivos de velocidad completa, y 1024 B, en el caso de una comunicación de alta velocidad. Finalmente, el receptor responderá con un paquete Handshake.

### **2.5. Descriptores**

Cuando un dispositivo es conectado al bus, debe informar sus características al Host a través de descriptores. Un descriptor es un estructura de datos con formato definido. De esta forma, el sistema conoce las diferentes configuraciones que puede tener cada una de las Funciones conectadas. El conocimiento detallado de estos descriptores por parte de los diseñadores de dispositivos, facilita luego la tarea de selección de cada uno de los atributos que tendrá, como así también, la elaboracion de software de control en la PC.

Cada uno de los descriptores comienzan con su longitud en bytes y el tipo de descriptor que se está enviando. En orden jerárquico, se utilizan categorías de descriptores que van desde los atributos generales a los particulares. En primer lugar, se envía el descriptor DEVICE que informa la versión de la norma USB que cumple el dispositivo, un numero que identifica al fabricante y otro que corresponde al producto, es decir al dispositivo. Esto le permite saber al host que software de control debe utilizar para comunicarse con el dispositivo. A su vez, comunica la cantidad de posibles configuraciones. Luego, si el dispositivo cumple con la norma 2.0 (o más moderna) envía un descriptor de tipo DEVICE\_QUALIFIER con información sobre otras velocidades de comunicación soportadas.

USB diferencia una configuración de otra dependiendo de las necesidades de energía. Un dispositivo podría operar conectado a una fuente de energía externa, o bien, ser potenciado por el mismo bus. Si las potencias de la fuente y del bus son diferentes, podrían verse limitadas las utilidades que ejecutaría la Función. Entonces, cuando el dispositivo funcione con la fuente podría tener una configuración pero cuando se desconecta, deberá informar esta situación al Host, indicando que se debe cambiar la configuración. Esta comunicación se lleva a cabo a través del descriptor de tipo CONFIGURATION. Debe haber tantos descriptores de este tipo como se

indicó en el descriptor DEVICE.

Debido a que cada configuración puede tener diferentes limitaciones en sus funciones dependiendo de la potencia que consuma, se establece que cada configuración tenga a su vez diferentes interfaces. La cantidad de interfaces que tiene una configuración, también debe estar informada en el descriptor CONFIGURATION.

Una interfaz puede verse como el conjunto de extremos que son utilizados por un dispositivo para realizar una función específica. Por ejemplo, se podría pensar en una impresora multifunción. Se puede tener una interfaz para la parte que imprime y otra para el scanner. A su vez, cada interfaz puede variar el ancho de banda requerido a través de los denominados AlternateSettings. Las interfaces y sus diferentes alternativas, se comunican al Host a través del descriptor de tipo INTERFACE.

A su vez, un extremo define la dirección de la comunicación, es decir, si es desde o hacia el Host, un tipo de transferencia, si la comunicación es sincrónica o no, el tamaño máximo de paquete y el ancho de banda necesario. Los extremos se describen a través del descriptor ENDPOINT.

En resumen, la comunicación entre los dispositivos y el Host se efectúa a través de los extremos. Los extremos, a su vez, se agrupan en interfaces y un grupo de interfaces conforman una configuración. Una característica a tener en cuenta es que un dispositivo puede tener diferentes interfaces activas a la vez y las interfaces pueden cambiar durante la operación de características alternativas (AlternateSettings). Sin embargo, para cambiar de configuración, todos los extremos y las interfaces se desactivan.

También existe un tipo de descriptores, denominados STRING, que sirven para colocar a cada uno de los atributos una forma legible por el usuario, aunque puede no ser utilizada.

## 2.6. Sumario del Capítulo

En el presente capítulo se abordaron las partes más relevantes a los fines de este trabajo que corresponden a la norma USB. Resulta importante comprender la norma a fin de lograr una correcta configuración de las partes involucradas en cada etapa del desarrollo.

A partir de la descripción global de un sistema, se pudo comprender cómo identificar los diferentes puertos que intervienen en la norma, las velocidades de operación de un sistema USB, las tensiones que intervienen, la codificación usada y la topología del bus. Se identificaron a su vez, los dispositivos que intervienen y el rol que ocupan dentro del sistema.

Se desarrollaron conceptos sobre los componentes más importantes del protocolo, la forma en que los mensajes se empaquetan, los campos y etiquetas que componen un paquete. Se detallaron los tipos de transferencias USB y sus características. También, se abordaron los descriptores a través de los cuales los dispositivos comunican sus características al Host.

## Capítulo 3

# Elección de las herramientas para la realización de la interfaz

La implementación de la comunicación entre el FPGA y la PC, en este trabajo, se efectúa a través de USB 2.0. Para establecer esta comunicación, como se menciona en la Sección 1.2, se utiliza una interfaz que ejecute las tareas relativas al protocolo, la transmisión y recepción de los datos.

La figura 3.1 muestra en líneas de trazo tres partes principales en las que este trabajo descompone el problema: La comunicación entre el FPGA y la interfaz, la configuración de la interfaz misma y la conexión entre la PC y la interfaz. El desarrollo de cada etapa cuenta con herramientas específicas que facilitan en gran medida la tarea que se realiza. En este capítulo se detalla por separado las características de cada una de estas herramientas.

### 3.1. Interfaz

Un punto importante del presente trabajo está constituido por el módulo de interfaz entre el FPGA y la PC, ya que este es el que efectivamente logra cumplir el objetivo de proveerle al sistema la comunicación entre ambos y que cualquier dispositivo que tenga cierto grado de implementación en el FPGA, pueda transmitir y recibir datos a través del protocolo USB.

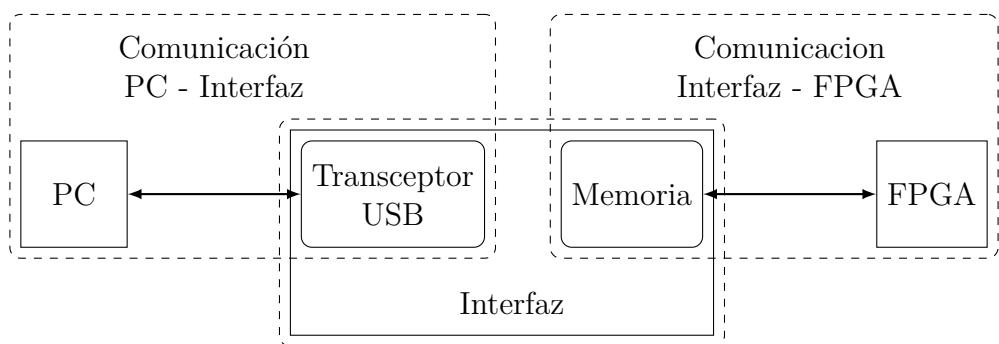


Figura 3.1: Partes en que se desglosa el trabajo

La interfaz está constituida por el controlador USB FX2LP fabricado por Cypress Semiconductor. Dicho controlador, es una circuito integrado que posee en su interior un microcontrolador 8051, a través del cual efectua las tareas que requiere la comunicación USB, sumado a un transceptor USB, el cual codifica y decodifica los paquetes USB que se transmiten a través del bus. A su vez, posee ciertos periféricos e interfaces que otorgan una flexibilidad que permite adecuar el chip a los requerimientos de un desarrollo determinado.

El controlador viene montado en un circuito impreso que posee una serie de componentes adicionales que facilitan la interacción del desarrollador, tales como pulsadores, display de 7 segmentos, módulos de memoria adicional, etc. Este tipo de circuitos impresos armados con la intención de favorecer desarrollo de otros sistemas, se denomina placa de desarrollo. Una placa de desarrollo que, además, incorpora algunas herramientas extra como software, cables de conexión, fuentes, etc. toma el nombre de kit de desarrollo.

Para este trabajo, se utiliza el kit de desarrollo CY 3684 FX2LP EZ-USB, también producto elaborado por Cypress Semiconductor. Este kit posee una placa de desarrollo como la que se observa en la Figura 3.2. Esta placa de desarrollo, posee como componente principal al controlador USB FX2LP e incorpora un display de 7 segmentos, 4 luces led multipropósito, 6 pulsadores, de los cuales 4 son de propósito general, uno de reinicio y otro que envía una señal especial para salir de un modo de bajo consumo. También tiene dos bloques de memorias EEPROM destinadas al almacenamiento del firmware (programa que ejecuta un microcontrolador), lo que otorga la posibilidad de realizar una carga no volátil de la configuración del controlador, memoria flash de 64 kB utilizados como RAM por el programa del controlador, un puerto USB y dos puertos UART con zócalos DE-9. Adicionalmente, cuenta con 6 puertos de 20 pines que permiten comunicarse con el controlador y 1 puerto de 40 pines, compatible con el protocolo ATA.

Como interfaz entre la FPGA y la PC se utilizó kit de desarrollo CY3684 FX2LP EZ-USB Development Kit de Cypress Semiconductor, la que se observa en la Figura ???. Esta placa posee como núcleo el controlador USB CY7C68013A, circuito integrado que posee todas las herramientas necesarias para realizar la interfaz, como así también un buen número de periféricos que permiten al desarrollador realizar pruebas y depuración.

Entre estas, se destacan 6 pulsadores, de los cuales cuatro se utilizan para propósito general, uno para reestablecer los valores por defecto de la placa y uno para enviar señales de suspensión y re establecimiento del programa actualmente cargado en el microcontrolador, lo que coloca al sistema en modo bajo consumo de energía. A su vez, posee dos memorias EEPROM que sirven para cargar firmware y archivos de configuración del sistema, un display de 8 segmentos, 4 leds de múltiple propósito, dos puertos UART, una salida de pines compatible con puertos ATA y 6 puertos de 20 pines que se utilizan para la conexión hacia el chip núcleo. Como soporte para el firmware, posee también un bloque con 64 kB de memoria SRAM.

Se selecciona este controlador como interfaz con el objetivo de utilizar la menor cantidad de los recursos configurables de la FPGA, de forma tal que estos queden disponibles para el desarrollo de sistemas científicos complejos que sean necesarios a posteriori.



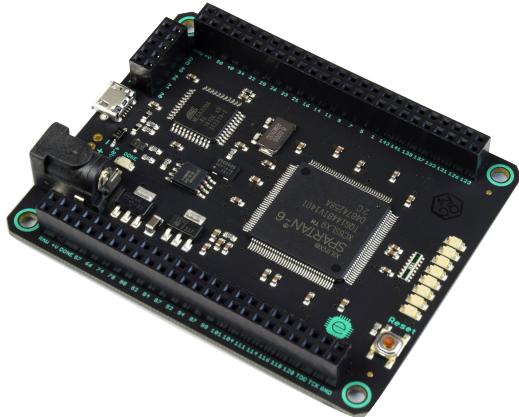


Figura 3.3: Placa de prototipado rápido MOJO v3, diseñada por Embedded Micro

también, con el fin de satisfacer requerimientos especiales, pueden ser diseñados por uno mismo, o bien es posible adaptar con algunos cables las entradas a un dispositivo particular.

Además de estos shields, los diseñadores pensaron en que no sea necesario ninguna herramienta extra a la hora de programar la FPGA. Para ello, dotaron al sistema de un microcontrolador ATmega32U4 de Atmel y cargaron un programa bootloader, que se encarga de transferir la configuración del FPGA cargada desde una memoria flash incorporada al sistema con ese propósito particular, o trasmisida por el usuario desde una PC, a través de un transceptor USB que contiene el microcontrolador. Luego, este último es colocado en modo esclavo y se configura de forma tal que dota al sistema de una comunicación entre la FPGA y una PC, vía USB y se utiliza su ADC para leer los puertos analógicos.

Una vez llegado a este punto, el lector podría preguntar con toda razón ¿por qué es necesario realizar un sistema de comunicación USB extra, si ya cuenta con un microcontrolador que se encarga de dicho asunto? La respuesta se basa en el ancho de banda del sistema de comunicación que dispone la placa. La línea de controladores ATmega incorpora puertos USB 2.0 full-speed. Esto quiere decir que puede enviar datos a una tasa de 12 Mbps. Además, la comunicación entre ambos chips se realiza via SPI (*Serial Peripheral Interface*, o en español Interfaz Serie de Periféricos), comandada por un cristal de cuarzo de 50 MHz, ofreciendo una velocidad de salida que puede resultar lenta a los fines de este trabajo. Se pretende dotar al sistema del mayor ancho de banda posible, utilizando la capacidad de USB 2.0 High-Speed, de hasta 480 Mbps.

### 3.3. Elección de la biblioteca libusb-1.0

libusb es una biblioteca de código abierto, muy bien documentada, escrita en C, que brinda acceso genérico a dispositivos USB. Las características de diseño que persigue el equipo de desarrollo que mantiene la biblioteca es que sea multiplataforma, modo usuario y agnótico de versión.

En el sitio web disponible se explica lo que significa cada uno de estos conceptos:

### *3. Elección de las herramientas para la realización de la interfaz*

---

- Multiplataforma: Se apunta a que cualquier software que contenga esta biblioteca pueda ser compilado y ejecutado en la mayor cantidad de plataformas posibles, dotando al software de portabilidad, es decir, la biblioteca puede ser ejecutada en Windows, Linux, OS X, Android y otras plataformas sin necesidad de realizar cambios en el código.
- Modo usuario: No se requiere acceso privilegiado de ningún tipo para poder ejecutar programas escritos con esta biblioteca.
- Agnóstico de versión: Sin importar la versión de la norma USB que se utilice, el programa se podrá comunicar siempre con el dispositivo USB que se requiera.

Otra ventaja que posee la biblioteca libusb es que, al ser de código abierto, posee una gran comunidad que contribuye al crecimiento del proyecto, como así también otros proyectos que utilizan esta biblioteca. Así, existe una gran variedad de ejemplos que facilitan el aprendizaje en su utilización y adaptaciones para diferentes lenguajes de programación, que se adapte a los conocimientos previos de la persona que desarrolla programas.



# 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. Slawiński 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] R. Turchetta, K. R. Spring, and M. W. Davidson, “Digital Imaging in Optical Microscopy - Introduction to CMOS Image Sensors,” (accessed in July 2019).
- [9] S. Mendis, S. Kemeny, and E. Fossum, “CMOS active pixel image sensor,” *IEEE Transactions on Electron Devices*, vol. 41, pp. 452–453, mar 1994.
- [10] 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.
- [11] 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.

---

**BIBLIOGRAFÍA**

---

- [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] 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.
- [15] T. Hizawa, J. Matsuo, T. Ishida, H. Takao, H. Abe, K. Sawada, and M. Ishida, “ $32 \times 32$  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.
- [16] ON Semiconductor, “NOIP1SN0300A Global Shutter CMOS Image Sensors,” 2014.
- [17] N. Ida, *Engineering Electromagnetics*. Cham: Springer International Publishing, 3th ed., 2015.
- [18] J. F. Wakerly, *Digital Design: principles and practices*, vol. 1. Pearson, 1999.
- [19] 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.
- [20] R. Biswas, *An Embedded Solution for JPEG 2000 Image Compression Based Back-end for Ultrasonography System*. PhD thesis, IIT, Kharagpur, 2018.
- [21] 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.
- [22] 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.
- [23] S. of Cable Telecommuniocations Engineers, *American National Standard ANSI/SCTE 07 2006. Digital Tansmission Standard for Cable Television*. Society of Cable Telecommunications Engineers, Inc., 2006.
- [24] I. Micron Technology, “1 / 2-Inch Megapixel CMOS Digital Image Sensor MT9M001C12STM (Monochrome),” pp. 1–35, 2004.

## BIBLIOGRAFÍA

---

- [25] IEEE Computer Society, *IEEE Standard for Ethernet*, vol. 2018. 2018.
- [26] 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.
- [27] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, and Philips, *Universal Serial Bus Specification*, vol. Revision 2.0. 2000.
- [28] “Usb hardware.” [https://en.wikipedia.org/wiki/USB\\_hardware](https://en.wikipedia.org/wiki/USB_hardware). Ingreso: 8 de agosto del 2019.
- [29] T. Riihonen, *Desing and analysis of duplexing Modes and Forwarding Protocols for OFDM(A) Relay Links*. PhD thesis, 2015.