Interfaz USB genérica para comunicación con dispositivos electrónicos

Avance de Proyecto

Andrés Aguirre

Carlos Grossy

Rafael Fernández

¿Que es el proyecto?

- Una respuesta a la necesidad de comunicar de forma ***sencilla*** y ***genérica*** dispositivos electrónicos no necesariamente pensados para interactuar con un PC.
- La solución se basa en tres puntos:
  - Un componente de hardware.
  - Un medio de comunicación (USB).
  - Una arquitectura (software y firmware).

Motivación

- Utilizar una PC para comunicarse con dispositivos electrónicos, logrando:
  - Aumentar la potencialidad de los dispositivos.
  - Aprovechar las capacidades de procesamiento, y almacenamiento del PC.
  - Aumentar la Interacción con el mundo físico.
  - Simplificar el manejo de los dispositivos.
- Uso de microcontroladores como parte de la solución.

Por Que USB?

- Desuso de puertos paralelos, seriales.
- No hay que abrir la PC.
- Versátil.
- Disponibilidad.
- Tecnología actual.

Descripción del Proyecto

- Objetivo
  - Construcción de hardware y software necesarios para facilitar la comunicación con dispositivos electrónicos por medio del USB.
  - Ocultar la complejidad de la interfaz USB.
  - Arquitectura modularizada y extensible.
    - Firmware, API, protocolo de comunicación, Drivers.
    - Bibliotecas de alto nivel para distintos dispositivos
  - Soporte para Linux y Windows.
  - Reutilización de controladores ya existentes en el sistema operativo.
  - Estudio de la tecnología USB (teórico y aplicado).

Tecnología USB

El paradigma USB

- Un solo tipo de conector para todos los periféricos.
- Habilidad para conectar varios dispositivos periféricos al mismo conector.
- Un método para facilitar los conflictos por recursos.
- Conexión en caliente.
- Detección y configuración automática de los periféricos.

El paradigma USB (cont)

- Bajo precio para la implementación del sistema y los periféricos.
  - La electrónica es mas compleja del lado del host que del periférico.
- Aumento en la capacidad de performance.
- Soporte para hardware y software legado.
- Implementación de bajo consumo energético

Puntos débiles de USB

- Velocidad
  - USB 2.0 480 Mbps / Firewire2 3.2 Gbps
- Distancia
  - 5 metros usando un cable, 30 metros usando cables que unen 5 hubs con el dispositivo
- Comunicación Peer-to-Peer
  - No posee comunicación entre dispositivos, solución parcial USB OTG.
- Ausencia de Broadcasting
  - El host envía un mensaje a cada dispositivo de forma individual
- Complejidad del Protocolo
  - Algunas interfaces viejas permiten conectar circuitos muy simples con protocolos muy básicos.

Velocidades

- USB soporta tres tipos de velocidades
  - 1.5 Mbps (low speed)
  - 12 Mbps (full speed)
  - 480 Mbps (high speed)

Componentes del Host:

- Host
  - Es una PC u otra computadora que contiene un controlador host USB y un hub root.
  - Controlador host
    - Da formato a los datos para trasmitir en el bus.
    - Traduce los datos recibidos a un formato que el SO pueda entender.
  - Hub root
    - Tiene uno o varios conectores para conectar dispositivos.

Las tareas del Host:

- Detección de dispositivos
  - Enumeración: el host asigna una dirección y solicita información adicional de cada dispositivo.
- Es el arbitro del BUS.
- Manejo del flujo de datos
  - Varios dispositivos pueden querer transferir datos al mismo tiempo, el host debe planificar el tiempo para cada dispositivo
- Detección de Errores
- Suministro de Energía
- Intercambio de Datos con Periféricos

Las tareas del Dispositivo:

- Detectar comunicaciones dirigidas hacia el
  - Un periférico no puede iniciar una comunicación por si solo. En cambio este debe esperar y responder a una comunicación del host
  - Cada dispositivo monitorea la dirección de dispositivo contenida en cada comunicación en el bus.
- Detección de errores

Transferencias:

- Las comunicaciones pueden ser divididas en dos categorías:
  - Comunicaciones utilizadas para enumerar dispositivos
  - Comunicaciones utilizadas por aplicaciones que llevan a cabo los propósitos de los dispositivos

Tipos de Transferencias

- Control
  - funciones definidas por la especificación USB.
  - permiten al host leer información acerca del dispositivo
  - asignar una dirección a un dispositivo, etc
- Bulk
  - La velocidad es un factor interesante pero no es crítico.
  - Si el bus esta ocupado, las transferencias son retardadas
- Interrupt
  - Semántica distinta a interrupción clásica
  - Latencia (tiempo max entre transferencias) garantizada.
- Isochronous
  - Latencia y frecuencia garantizadas.
  - No posee control de errores
  - Trasmisión de datos multimedia en tiempo real.

Transferencias: Device endpoints

- Parte única e identificable de un dispositivo USB
- Es un bloque de memoria de datos o un registro en el chip del controlador de dispositivo.
- Los datos almacenados en un endpoint son datos recibidos o datos esperando a ser enviados.
- El host no tiene endpoints.
- El host sirve de comienzo y fin para la comunicaciones con los device endpoints.

Transferencias:
Device endpoints (cont)

- La dirección de un endpoint consiste de un número de endpoint y un sentido.
  - Numero: 0 – 15, Sentido: IN, OUT
- Endpoint de control, transfiere en ambos sentidos.
- Todo dispositivo debe tener configurado el endpoint 0 como endpoint de control.

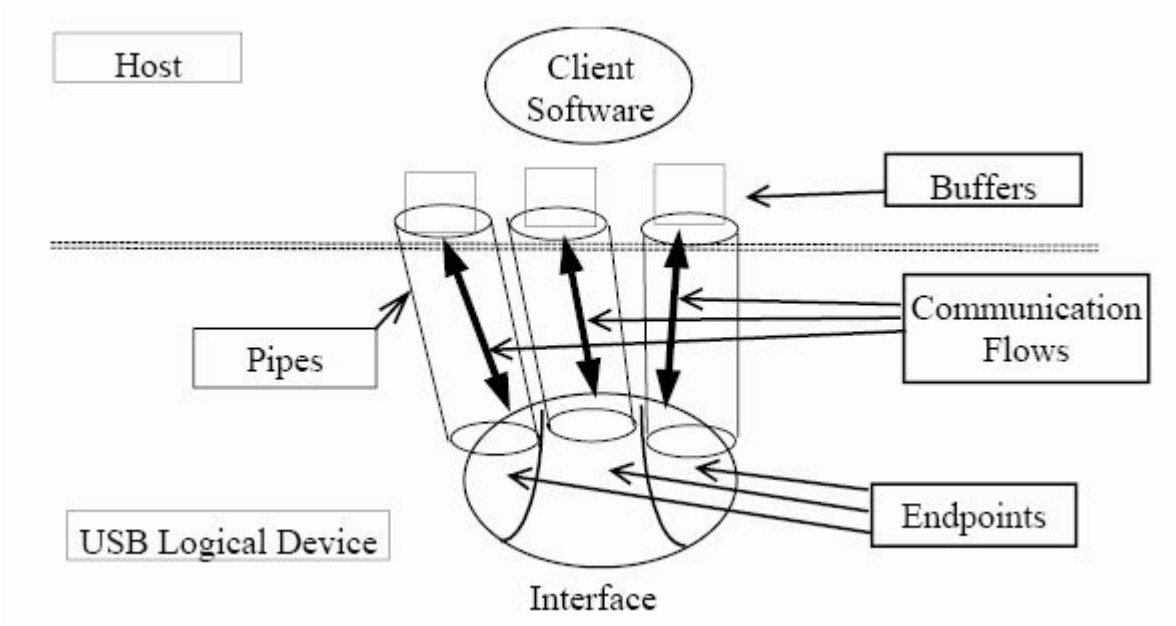
Transferencias:

Pipes

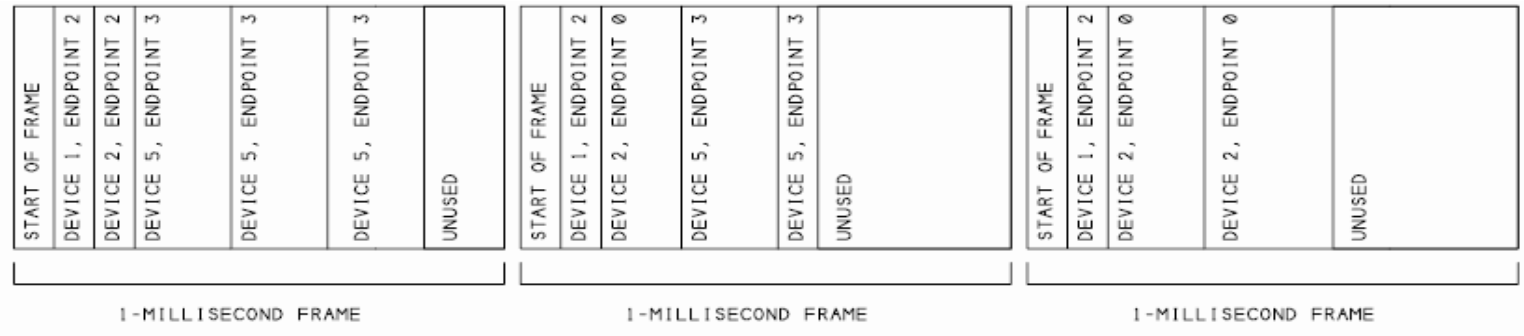
- Antes que una conexión pueda ocurrir el host y el dispositivo deben establecer un pipe.
- Un pipe USB es una asociación entre un endpoint y el software en el host controlador.
- Cada dispositivo tiene un pipe de control por defecto que usa el endpoint cero.

Transferencias:

Flujo de datos



Transferencias: Planificación



- Una transferencia USB consiste de transacciones.
- El host planifica las transacciones dentro de los frames.
- La especificación pone límites acerca de esta planificación.
  - No más del 90% de c/frame puede ser usado para transferencias periodicas (iso, int).
  - Mínimo 10% de c/frame para transferencias de control

Transferencias:
Transacciones

- La especificación USB define una transacción como la entrega de servicios a un endpoint.
- Las transacciones consisten en uno, dos, o tres paquetes
- Existen tres tipos de transacciones que se definen según el sentido del flujo de datos y el propósito:
  - SETUP
  - OUT
  - IN

Fases de una transacción

- Cada transacción tiene hasta tres fases, o partes que ocurren en secuencia
  - token, data, y handshake
- Cada fase consiste en la transmisión de uno o dos paquetes
- Cada paquete es un bloque de información con un formato definido.
  - Comienzan con un Packet ID (PID) que contiene información identificatoria puede ser seguido por una dirección de endpoint, datos, información de estado, o un numero de frame, y bits de chequeo de error.

Fases de una transacción (cont)

- En la fase de token de una transacción, el host inicia la comunicación enviando un paquete de token.
  - El PID indica el tipo de transacción, como Setup, IN, OUT, ó Start-of-Frame.
- En la fase de data, el host o el dispositivo pueden transferir cualquier tipo de información en un paquete de datos.
  - El PID incluye un valor de secuenciamiento
- En la fase de handshake, el host o dispositivo envía información de estado en un paquete de handshake.
  - El PID contiene un código de estado (ACK, NAK, STALL o NYET).

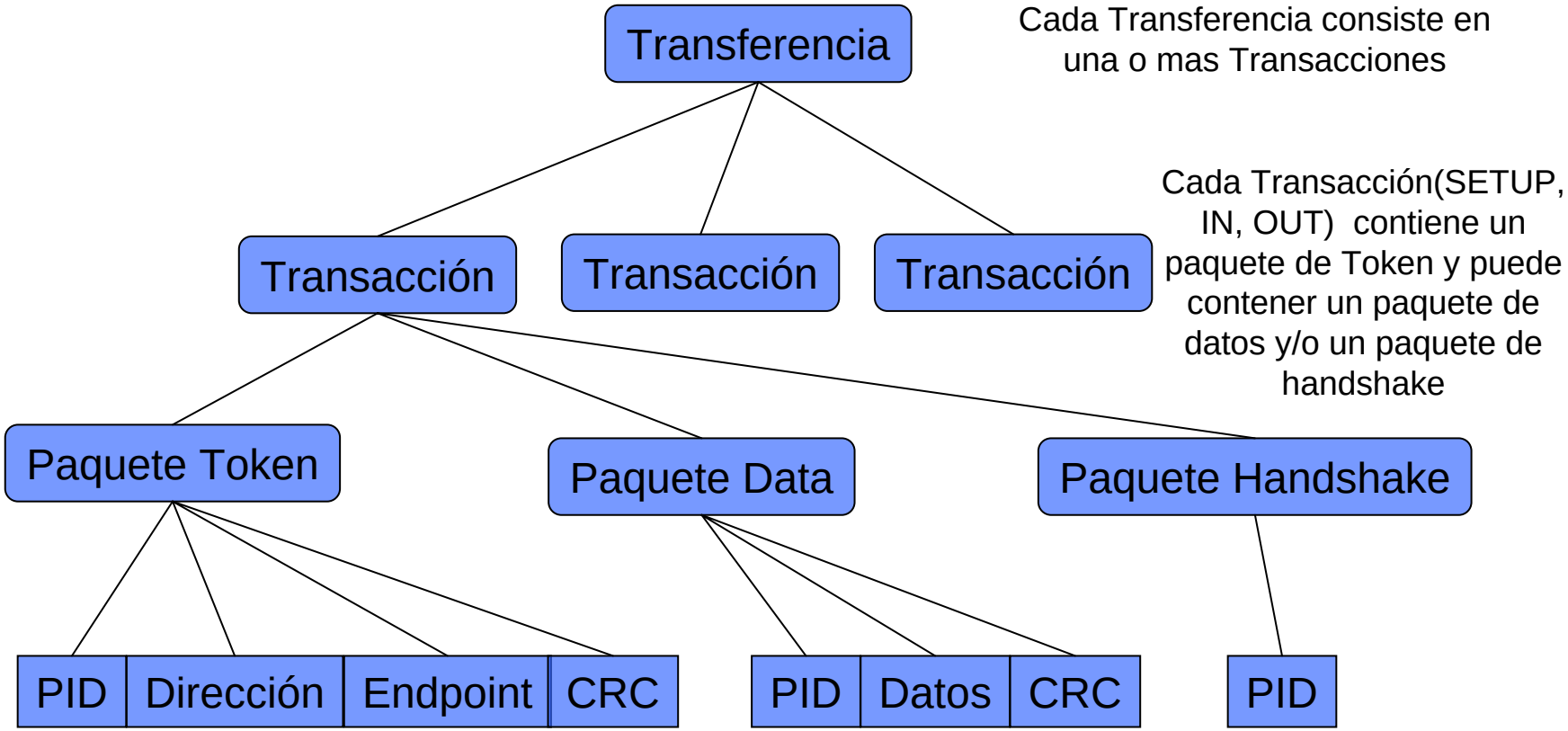
Paquetes

- Toda transacción tiene un paquete de token.
  - El host es siempre la fuente de este paquete
  - Configura la transacción identificando:
    - El dispositivo que lo recibe.
    - El endpoint.
    - El sentido de cualquier dato que la transacción vaya a transferir

Paquetes (cont)

- Dependiendo del tipo de transferencia y si es el host o el dispositivo que tiene información a enviar, un paquete de datos precede al paquete de token.
- El sentido especificado en el paquete de token determina cuando el host o el dispositivo envía el paquete de datos.
- En todos los tipos de transferencias exceptuando las isochronous, el receptor de el paquete de datos retorna un paquete de handshake conteniendo un código indicando si la transacción fue exitosa o falló.

Aclarando ideas …



Enumeración:

- Determina:
  - Que dispositivos han sido conectados al bus.
  - Que parámetros requieren, como ser:
    - Consumo de energía
    - Número y tipo de endpoint
    - Clase de producto
    - Etc.
- El host luego asigna una dirección al dispositivo, para permitirle transferir datos en el bus

Enumeración

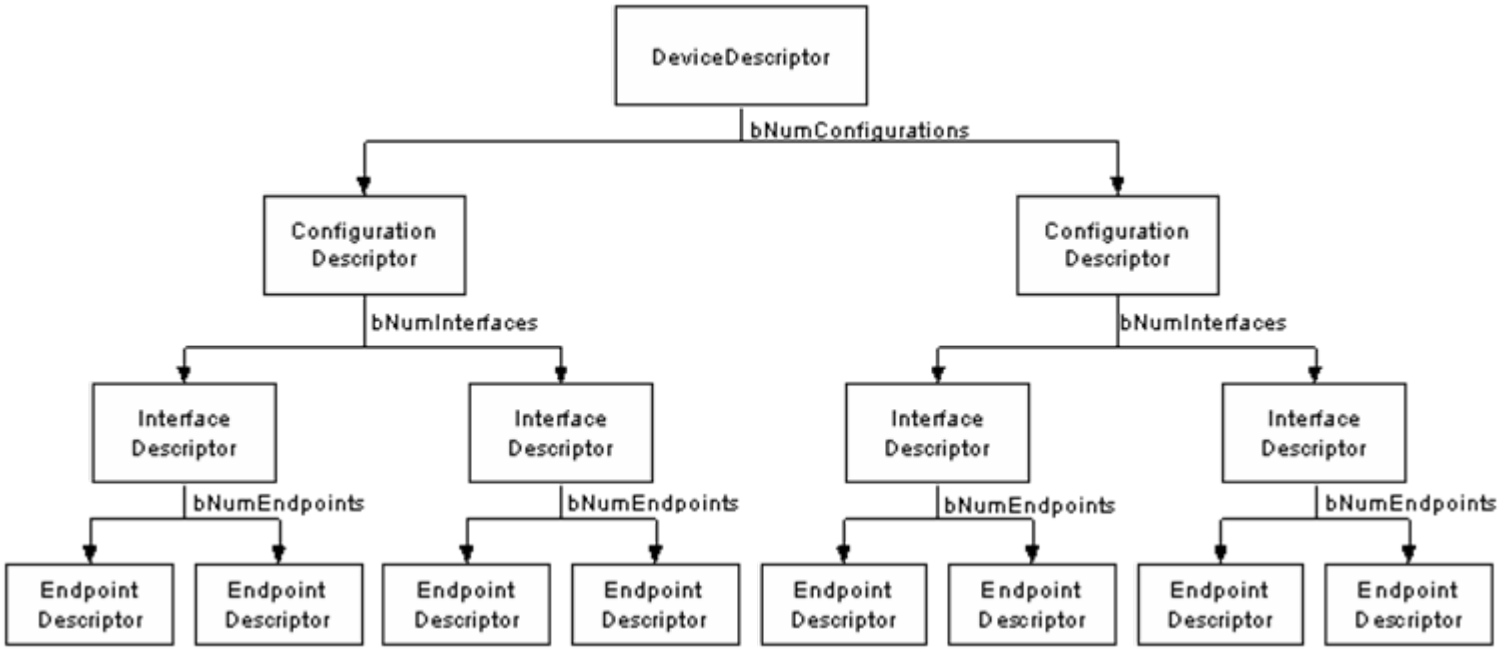
Descriptores:

- Estructuras de datos (información, funcionalidad del dispositivo).
- Perminten al host aprender acerca de un dispositivo.
- Todos los dispositivos USB deben responder a pedidos para los descriptores USB estándar.
- El dispositivo debe guardar información de los descriptores y responder a pedidos por los descriptores.

Enumeración

Tipos de descriptores:

- Durante la enumeración el host usa transferencias de control para solicitar descriptores a un dispositivo.



Enumeración

Tipos de descriptores (cont):

- Device Descriptor
  - Contiene información básica acerca del dispositivo.
    - fabricante, número de producto, número de serie, la clase de dispositivo y el número de configuraciones.
  - Un dispositivo puede tener un único device descriptor.
- Configuration Descriptor
  - Provee información acerca de los requerimientos de alimentación del dispositivo y cuantas interfaces son soportadas.
  - Puede haber mas de una configuración para un dispositivo.

Enumeración

Tipos de descriptores (cont):

- Interface Descriptor
  - Detallan el número de endpoints usados en la interface, como el tipo de interface.
  - Puede haber mas de una interface para una configuración.

Enumeración

Tipos de descriptores (cont):

- Endpoint Descriptor
  - Identifican el tipo de transferencia y su sentido, como otros datos específicos de un endpoint.
  - Puede haber varios endpoints en un dispositivo
  - pueden ser compartidos en distintas configuraciones.
- String Descriptor
  - Varios de los descriptores previos referencian a uno o mas string descriptors.
  - Proveen información amigable acerca de la capa.
  - Generalmente opcionales.

Clases de Dispositivos

- Definen atributos y servicios que son compartidos por muchos dispositivos o interfaces
- Los sistemas operativos pueden proveer driver para las clases en común
- Una especificación de clase define
  - Número y tipo de los endpoints
  - Valores para los ítems en los descriptores estándar
  - Descriptores class-specific, interfaces, usos de endpoints y pedidos de control.

Algunos ejemplos de clases de dispositivos:

- Audio
  - MIDI
- Comunications
  - Modem
- Human Interface (HID)
  - Joystick, mouse, teclado
- Mass Storage
  - Pendrive
- Printer
- Etc.

Soluciones de conectividad USB

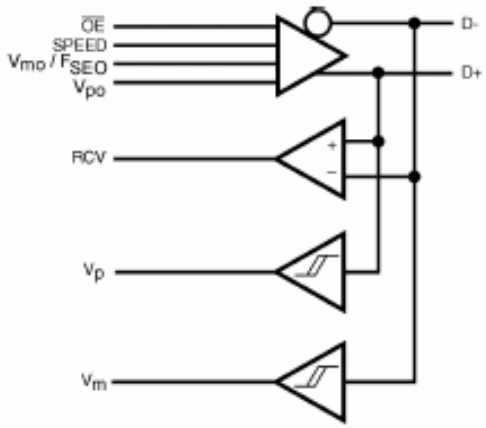
Soluciones de conectividad USB

- Opciones
  - Transceivers USB
  - Conversores USB a serial o paralelo
  - Controladores de periféricos
    - Externas
    - Embebido en un microcontrolador

Transceivers USB

- Realiza interfaz entre dispositivos programables y lógica estandar con la capa física de USB.
- No maneja transacciones ni endpoints.
- Incorporan: Reguladores de voltage, detectores de conexión.
- Ejs: Fairchild USB1T20, Philips ISP110x.

Logic Diagram



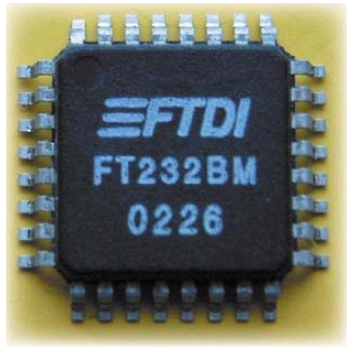
Functional Truth Table

Input					I/O		Outputs			
Mode	V_{po}	V_{mo}/F_{SEO}	$\overline{OE}$	SUSPND	D+	D–	RCV	V_p	V_m	Result
0	0	0	0	0	0	1	0	0	1	Logic 0
0	0	1	0	0	0	0	U	0	0	$\overline{SE0}$
0	1	0	0	0	1	0	1	1	0	Logic 1
0	1	1	0	0	0	0	U	0	0	$\overline{SE0}$
1	0	0	0	0	0	0	U	0	0	$\overline{SE0}$
1	0	1	0	0	0	1	0	0	1	Logic 0
1	1	0	0	0	1	0	1	1	0	Logic 1
1	1	1	0	0	1	1	U	U	U	Illegal Code
X	X	X	1	0	Z	Z	U	U	U	D+/D– Hi-Z
X	X	X	1	1	Z	Z	U	U	U	D+/D– Hi-Z

X = Don't Care Z = 3-STATE U = Undefined State

Conversores USB

- Transforman USB a otra interfaz conocida para realizar la interface con un microcontrolador.
- Ej: FTDI FT232BM
  - USB 2.0 (Full speed)
  - USB <-> USART
    - Velocidad de 300 a 3M bauds
    - Buffer de 384 bytes de recepción y 128 de envio
    - Full Handshaking y señales de modem.
    - Bit Bang mode: Transforma las señales de control en puerto de E/S de 8 bits.
  - Interfaz con EEprom para customizar VID, PID, etc.
  - Drivers de puerto COM virtual para Windows, MacOS y Linux.
  - Reuso de sistema y firmware
  - Solución concreta y poco configurable.



Controladores de periféricos externos

- Proveen la capacidad de comunicación a sistemas basados en microcontroladores/microprocesadores.
- Manejo a nivel de transacciones y endpoints.
- Es visto como un periférico por parte del microcontrolador y un dispositivo USB por parte de la PC.
- No funcionan en forma autónoma.
- Poco impacto en sistemas existentes.
- Ej: Philips ISP1581.

Philips ISP1581

- USB 2.0
- Soporta detección de velocidad automática (high y full)
- 7 Endpoints de entrada, 7 de salida. Soporta double buffer.
- Los endpoints pueden ser interrupt, bulk, o isochronous.
- 8Kb de memoria FIFO integrada.
- Interfaces:
  - Interface de bus independiente para la mayoría de los microcontroladores/microprocesadores (12.5 MByte/s)
  - Interface DMA de alta velocidad (12.8 Mbyes/s)
  - Interface directa con periféricos ATA/ATAPI
- Conexión al bus USB controlada por software (SoftConnect tm)
- Data transceiver y regulador de voltaje de 3.3 V integrados.

Controladores de periféricos embebido en un microcontrolador

- Se incorpora dentro del mismo microcontrolador el hardware necesario para conectarse directamente al USB.
  - La comunicación con dicho modulo se realiza por medio de registros especiales y memoria compartida (dual port RAM) y línea de interrupciones dedicada.
  - Las comunicaciones se manejan a nivel de endpoints.
  - “Dependencia” de una arquitectura
- Ejemplos:
  - TSUB3210 (Texas Instruments)
  - PIC18F4550 (Microchip)
  - AT90USB1287 (Atmel)

Tabla de comparación entre microcontroladores

	TUSB3210	PIC18F4550	AT90USB1287
Arquitectura	CISC (8052)	Harvard RISC 75+8 inst	Harvard RISC 135 inst
Velocidad	12 Mhz	48 Mhz	16 Mhz
Package	TQFP 64	TQFP 44, QFN 44, DIP 40	TQFP 64, QFN 64
Memoria de programa	*6K ROM, 8K RAM (Firmware)	32Kb Flash autoprogramable por software	128Kb Flash autoprogramable por software
Memoria datos	768 bytes	2 Kb	8 Kb (hasta 64 KB externos)
USB 2.0 (full y low speed)	512 Bytes compartida, 3 endp IN, 3 OUT. transferencias interrupt y bulk	1024 Bytes compartida, hasta 32 endp con ping pong buffering, soporta todas las transferencias	832 bytes compartida, 6 endpoints con ping pong buffering, soporta todas las transferencias
Eeprom	no	256 bytes	4 Kbytes
Modo Bajo Consumo	Si	NanoPower, 3 modos Sleep	Si, 6 Modos Sleep

Tabla de comparación entre microcontroladores

	TUSB3210	PIC18F4550	AT90USB1287
Pines de E/S	Hasta 36	Hasta 35	Hasta 48
Timers	3 de 16 bits	1 de 8 bits 3 de 16 bits	2 de 8 bits 2 de 16 bits
I2C	Master	Master/Slave	TWI* Master/Slave
SPI	No	Master/Slave	Master/Slave
USART	No	Si	Si
Canales PWM	No	Hasta 2 de 10 bits de resolucion	Hasta 6 de 2-16 bits de resolucion
A/D	No	13 canales 10 bits	8 canales 10 bits
Otros	Bootloader I2C o USB, niveles de prioridad en interrupciones, soporte multiproducto	Soporte bootloader, prioridad de interrupciones programables, multiplicador por hardware, 2 comparadores analógicos, Streaming Paralel Port. ICSP e ICD Bloqueo de secciones de mem.	Soporte bootloader, vector de interrupciones con prioridad fija, multiplicacion por hardware, mparadores analógicos,modos bajo consumo, USB OTG,Bloqueo de secciones de mem. JTAG.
Documentación	Poca, algunas notas de aplicación.	Mucha, recursos en la web, muchas notas de aplicación, framework USB	Poca, Framework USB, algunas notas de aplicación.
Entornos de desarrollo y compiladores	En general los de 8052, de 3eras partes, algunos gratuitos.	MPLAB, 3ras partes, varios compiladores	AVR Studio 4, 3ras partes

Elección del microcontrolador

- La elección queda entre el PIC18F4550 y el AT90USB1287 y se tienen en cuenta los siguientes criterios:
  - Aspectos Técnicos
    - El AT90USB1287 en general es superior al PIC18F4550.
  - Documentación
    - Mayor documentación y notas de aplicación disponible del PIC18F4550.
  - Infraestructura y Conocimientos Previos
    - Experiencia previa (taller de firmware)
      - Conocimiento de arquitectura y herramientas de desarrollo.
      - Hardware de programación/debugging disponible.
    - Kit de desarrollo PICDEM FS USB.
  - Disponibilidad
    - PIC18F4550 disponible en plaza.
    - PIC18F4550 disponible en package DIP40.
- ***Se tomó la decisión de usar el PIC18F4550 para la implementación en el proyecto de grado.***

Entornos de desarrollo y compiladores del PIC18F4550

- IDEs
  - Microchip MPLAB (Windows)
  - CodeBlocks (Windows, Linux)
- Compiladores C
  - Hi-Tech PICC18
    - Demo funcional (30 días)
  - MPLAB C18
    - Versión de estudiante (deshabilita optimizaciones a los 60 días)
  - SDCC
    - Familias PIC16 y PIC18 en desarrollo

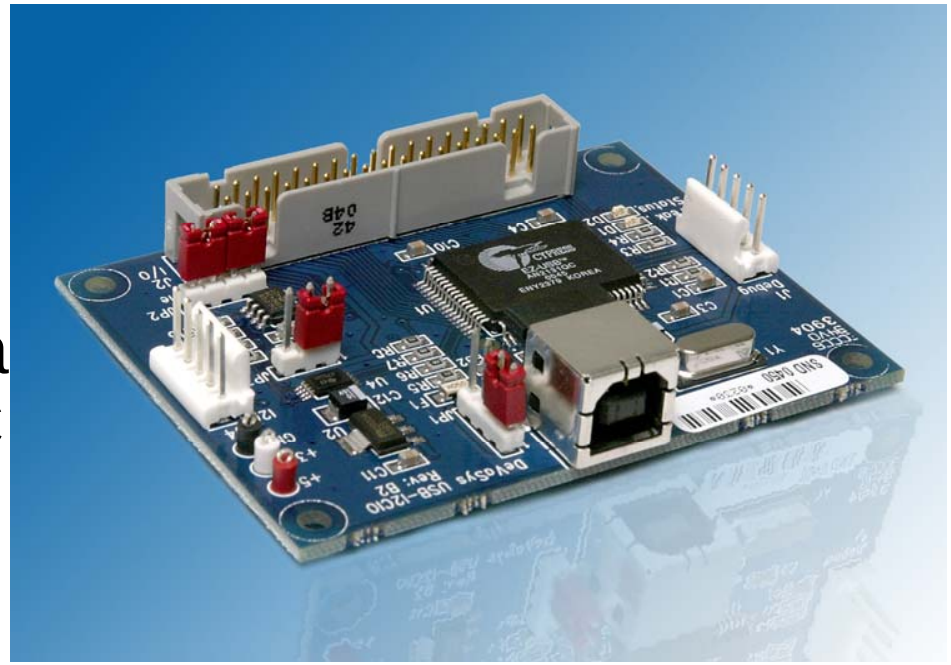
Microchip USB Firmware Framework

- Sistema de archivos para facilitar la creación de aplicaciones USB
- Estructura lógica modular
- Ejemplos de uso
  - Bootloader
  - Clase CDC
  - Clase HID
- Driver genérico de microchip (Windows)
- Aplicaciones de alto nivel
- Desventajas (para nuestra solución)
  - Gran cantidad de elementos configurados en tiempo de compilación
  - Algunas capacidades no utilizadas
  - Utiliza el módulo de USB mediante polling
- Experimentos realizados
  - Bootloader
  - Aplicación en C para interactuar con firmware de demo.

Proyectos relacionados.

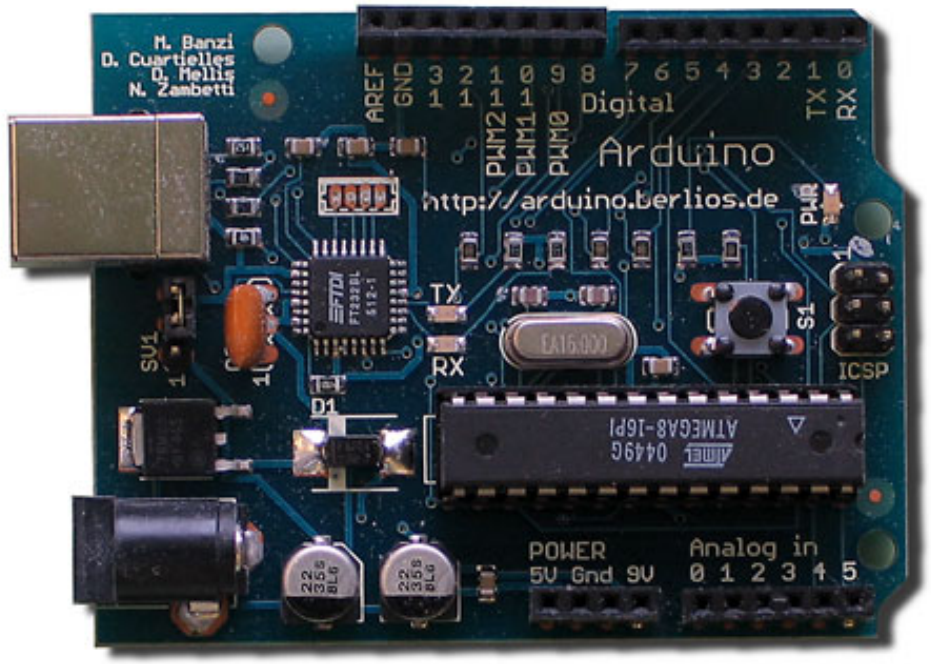
DevaSys - USB I2C/IO

- Cypress AN2131QC.
- 20 bits I/O.
- Interface I2C.
- Onboard 16KB I2C eeprom.
- Conector de 5 pin para conectar hardware I2C
- Bootloader
- Incluye API



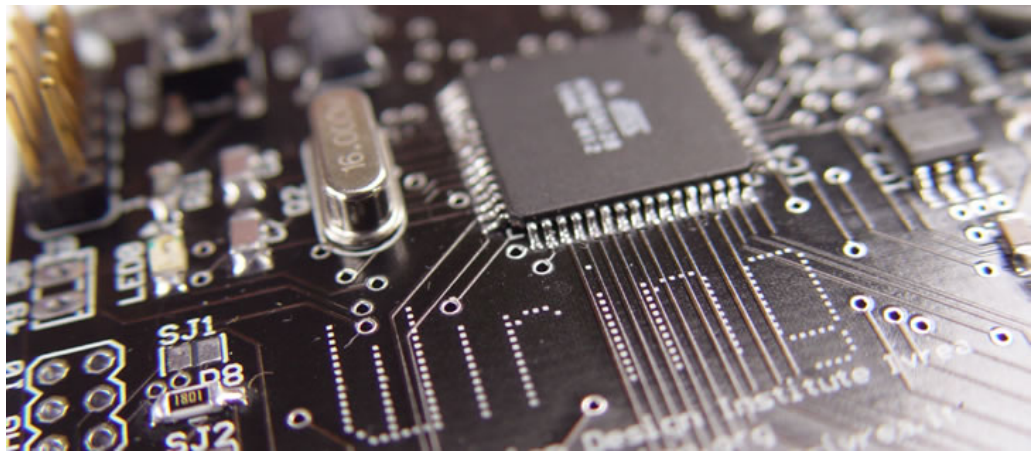
Arduino:

- Atmega8 de Atmel
- Modo stand alone
- Posee IDE propio
- Digital I/O (14 pins)
- Analog I/O (6 pins)
  - A/D
  - PWM
- Comunicación serial
- Lenguaje de programación wiring (C reducido)



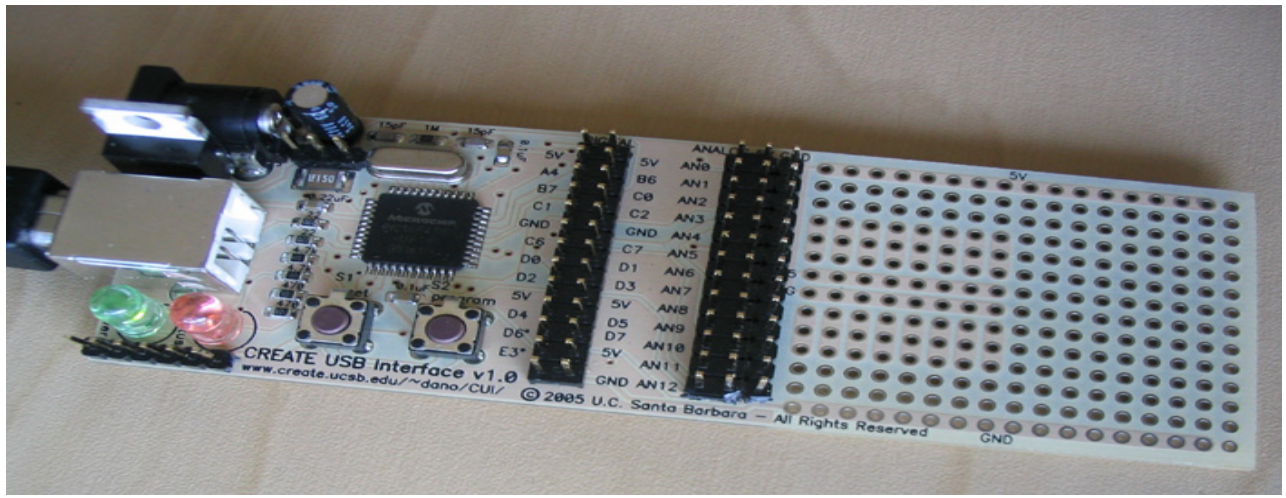
Wiring:

- ATMega128 Microcontroller
- Utiliza lenguaje wiring
- 43 digital pins (I/O)
- 8 entradas analogicas
- 6 salidas PWM
- 2 puertos serial
- I2C
- 8 pins para interrupciones externas
- Posee IDE propio
- 128KB de memoria de programa flash



CUI (Create USB Interface):

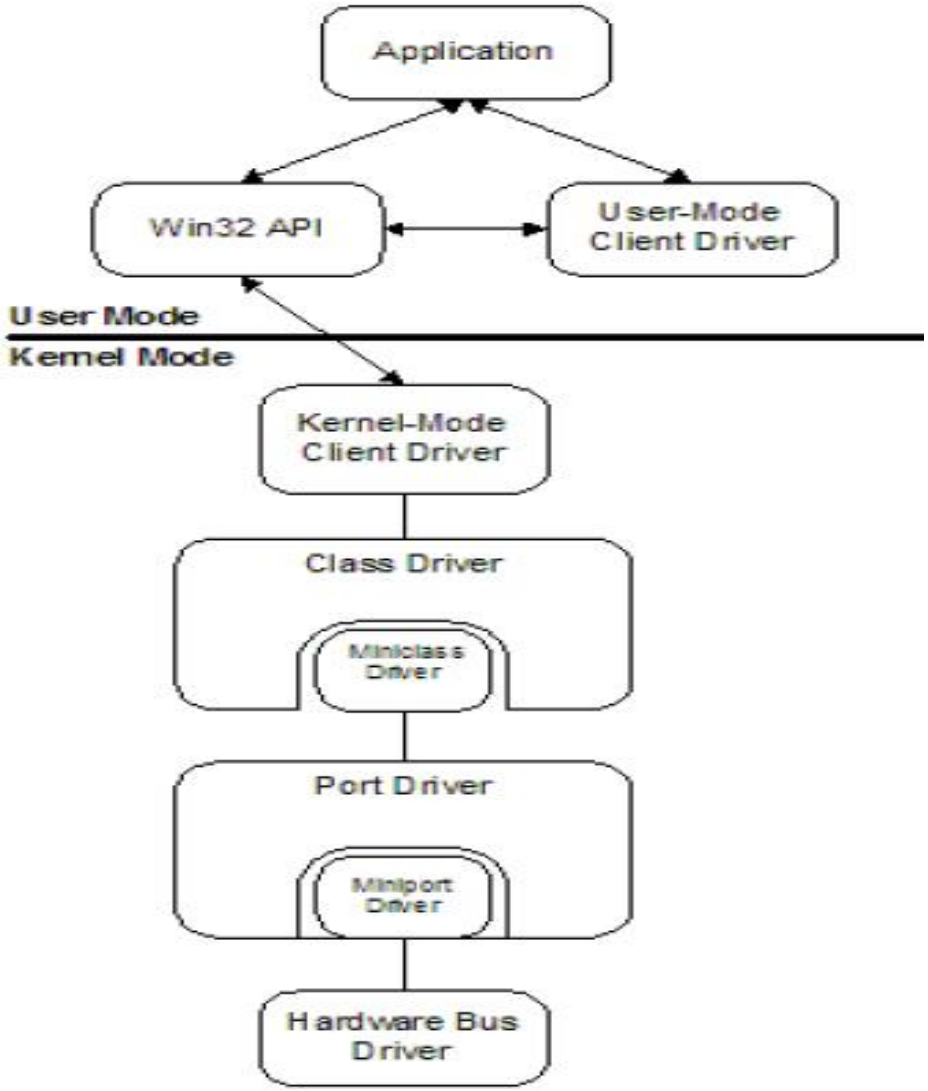
- Microcontrolador PIC18F4550
- 32 KB de memoria
- 17 puertos generales de I/O
- 13 entradas A/D



Modelos de Controladores USB

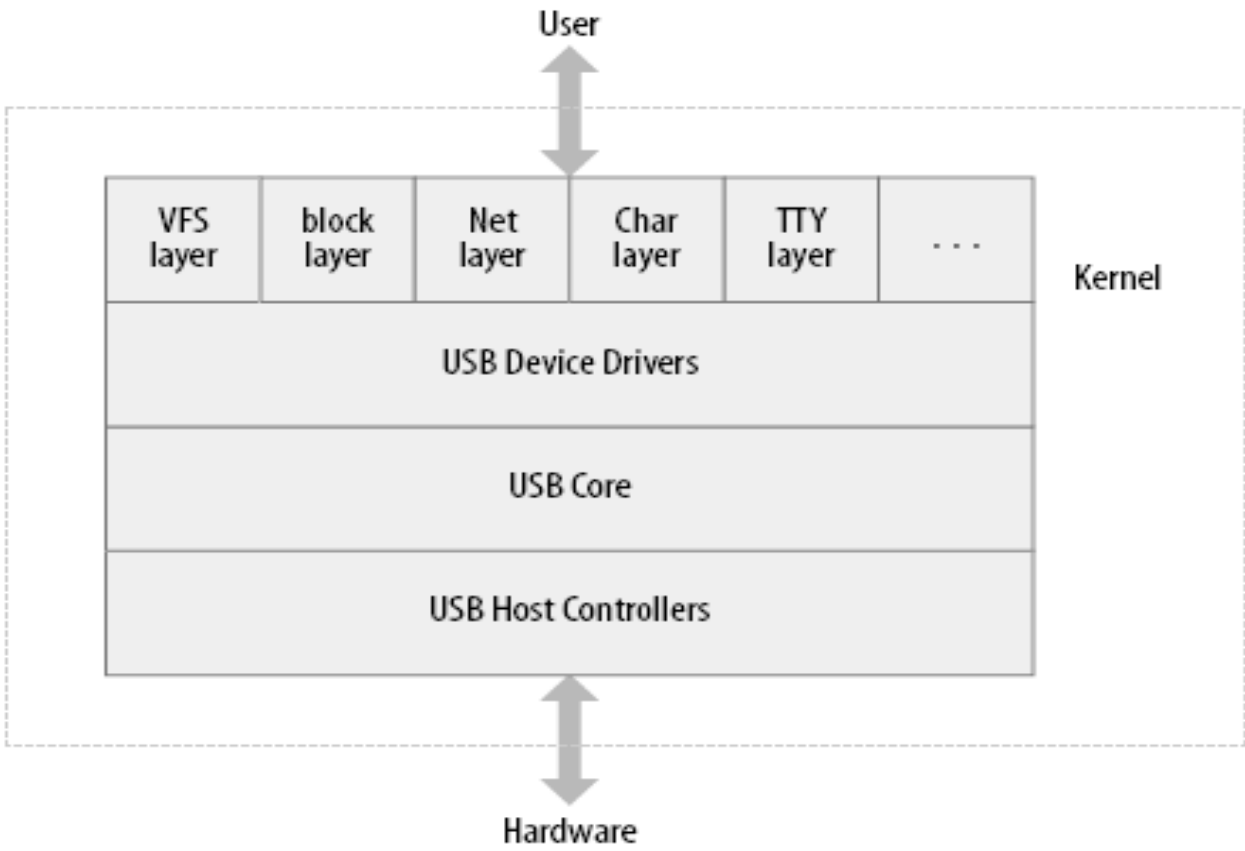
Modelos de controladores USB

- Windows



Modelos de controladores USB (II)

- Linux



Herramientas de Desarrollo y Depuración de Controladores

Herramientas de Desarrollo

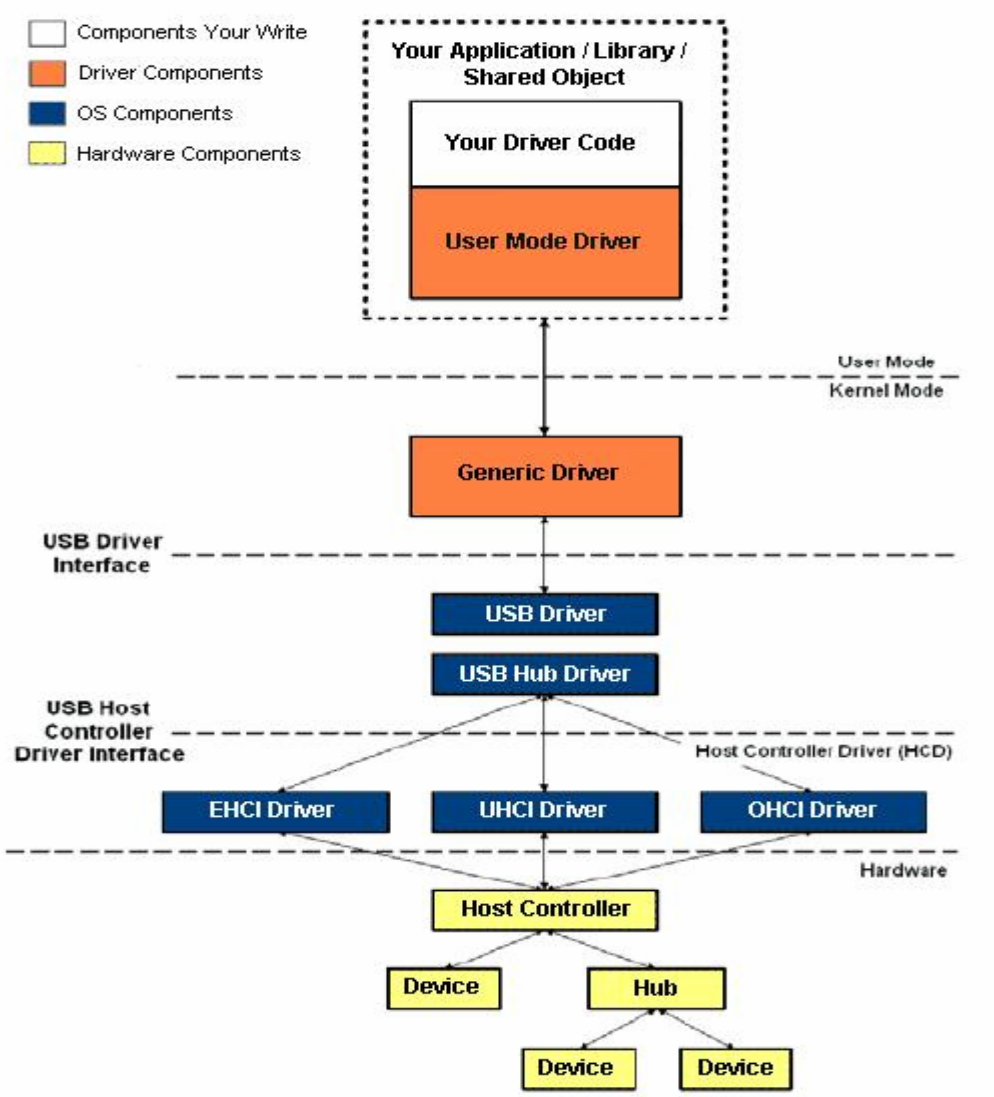
- Herramientas “Básicas”
  - Windows Driver Development Kit (DDK)
    - Es gratuito y esta disponible en la Web. http://www.microsoft.com/whdc/devtools/ddk/default.mspx
    - Es el conjunto mínimo de herramientas que brinda Microsoft para la construcción de un controlador de dispositivo.
      - Compilador C y Link Editor.
      - Utilitarios para la depuración y testeo.
      - Código fuente de ejemplo y documentación.

Herramientas de Desarrollo (II)

- Linux Driver Development Kit (LDDK)
  - Es gratuito y esta disponible en la Web. http://www.kroah.com/log/2006/05/24/
  - Es una recopilación de información de cómo escribir controladores en la plataforma Linux.
    - Fuentes y documentación completas del Kernel 2.6.16.18 de Linux.
    - El libro “Linux Device Drivers (3º Edition)” en formato digital.

Herramientas de Desarrollo (III)

- Controladores Genéricos



Herramientas de Desarrollo (IV)

- ***WinDriver USB 8.02 (Jungo)***

  - Cumple con los estándares USB 1.1 y 2.0.
  - Soporta los 4 tipos de transferencias, todas las velocidades, administración de energía.
  - Detección de toda la información de los dispositivos USB conectados.
  - Generación de código específica para el hardware.
  - Herramientas de depuración gráficas.
  - Windows Vista / Server2003 / XP / 2000 / Me / 98 / NT 4.0 / CE 4.x-5.0 / Mobile 5.0 y Linux 2.4-2.6.
  - Borland Delphi, Gcc, VB 6.0, VS.Net C# y VB.

Herramientas de Desarrollo (V)

- Controladores Personalizados
  - ***KernelDriver 6.11 (Jungo)***
    - Cumple con los estándares USB 1.1 y 2.0.
    - Generación de un esqueleto del controlador.
    - Acceso al hardware por medio de entorno gráfico.
    - API para acceso y control del hardware (modo núcleo).
    - Windows Server2003 / XP / 2000 / NT 4.0 / Me / 98.
    - Cualquier compilador C de 32-bit (gcc, VC++, etc.).

Herramientas de Desarrollo (VI)

- Otras herramientas
  - LibUSB
  - jUSB
  - JSR80 (javax.usb)

Herramientas de Depuración

- Analizadores vía Software

  - SourceUSB 2.0 (SourceQuest)
    - Soporta USB 1.x y 2.0.
    - Centrado en actividad del Host.
    - No usa controladores filtro (menos invasivo).
    - Rastrea IRP’s y URB’s que pasan por el stack de controladores USB.
    - Permite buscar y filtrar la información capturada por varios criterios.
    - Windows 2000 / XP / Server 2003 / Vista.

Herramientas de Depuración (II)

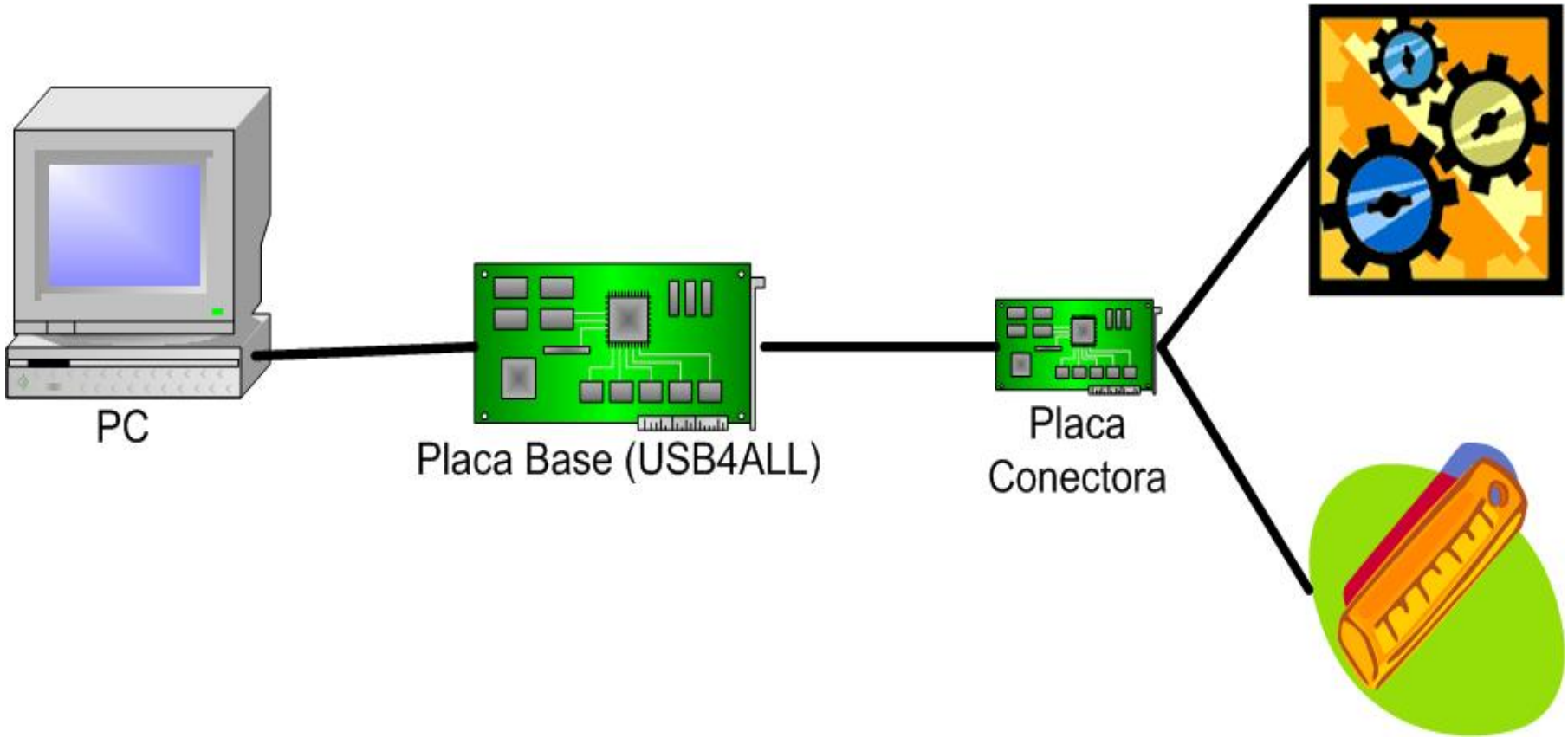
- Analizadores vía Hardware

  - USB Explorer 200 (Ellisys)
    - Análisis continuo y no invasivo.
    - 32 Mbytes memoria FIFO
    - USB 2.0 (Low, Full, High).
    - Información del estado del bus, transacciones, paquetes, descriptores, peticiones estándar y performance de la comunicación.
    - Software para la visualización
      - Registro en tiempo real y exportación de la información.
      - Búsquedas y filtros sobre la información.

Arquitectura

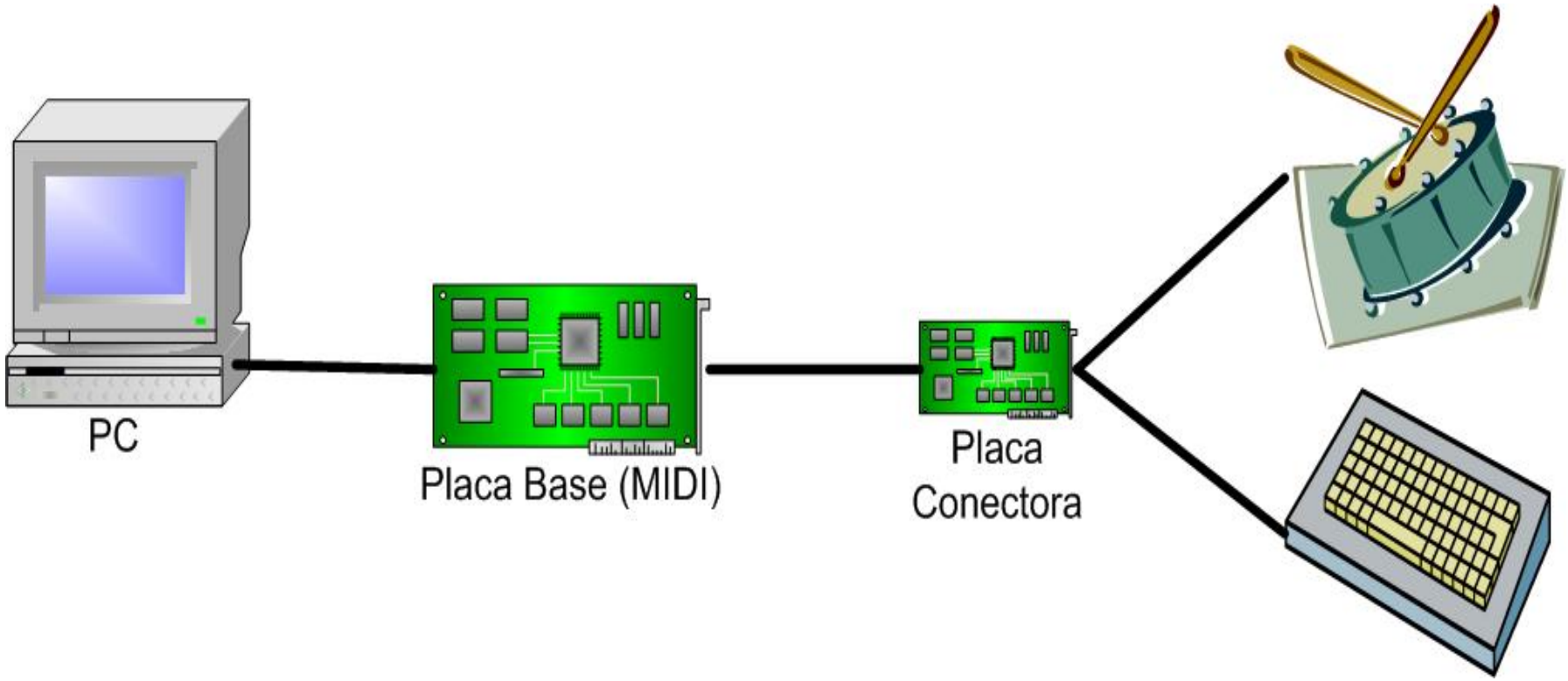
Escenarios

- Modo USB4ALL

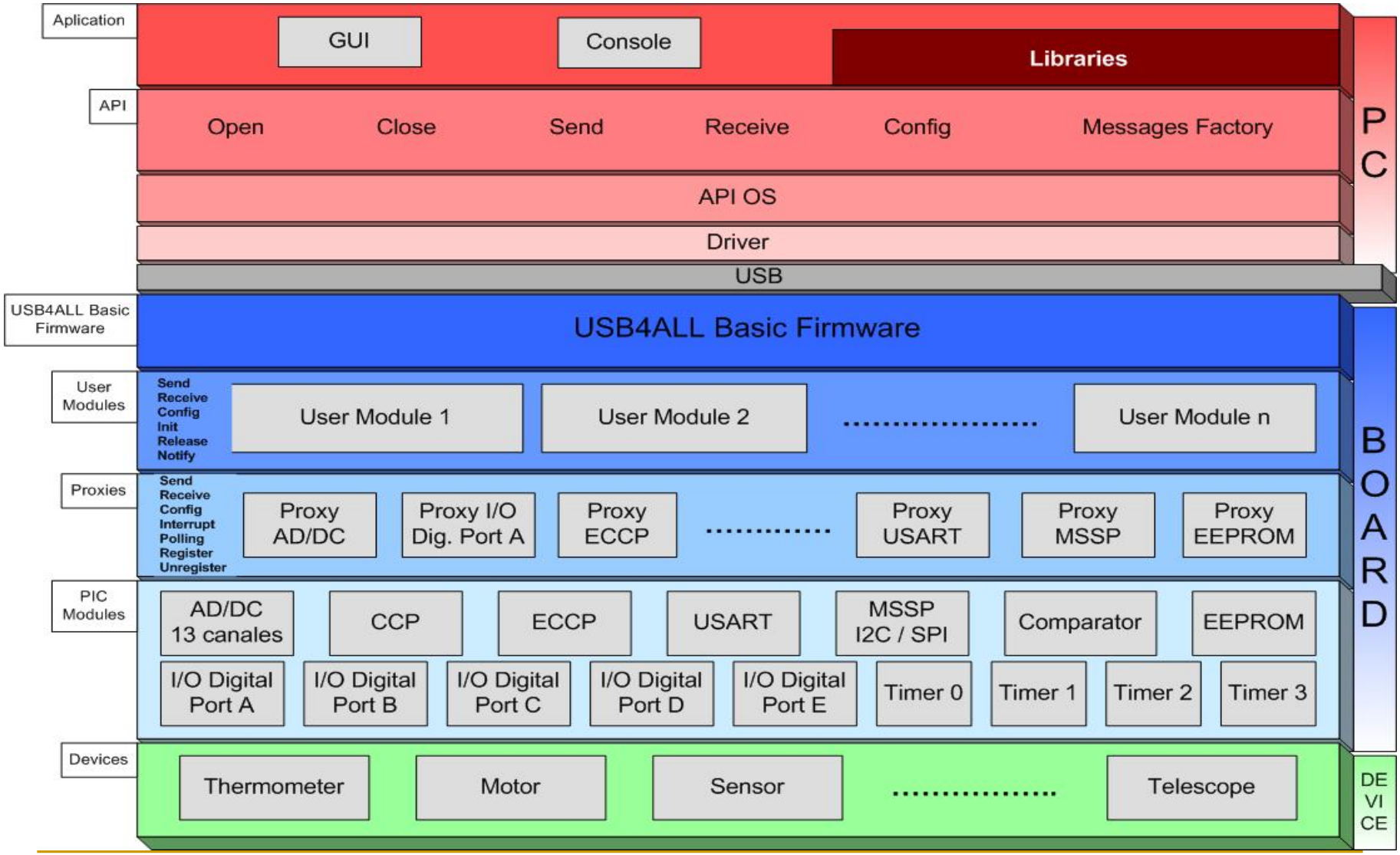


Escenarios

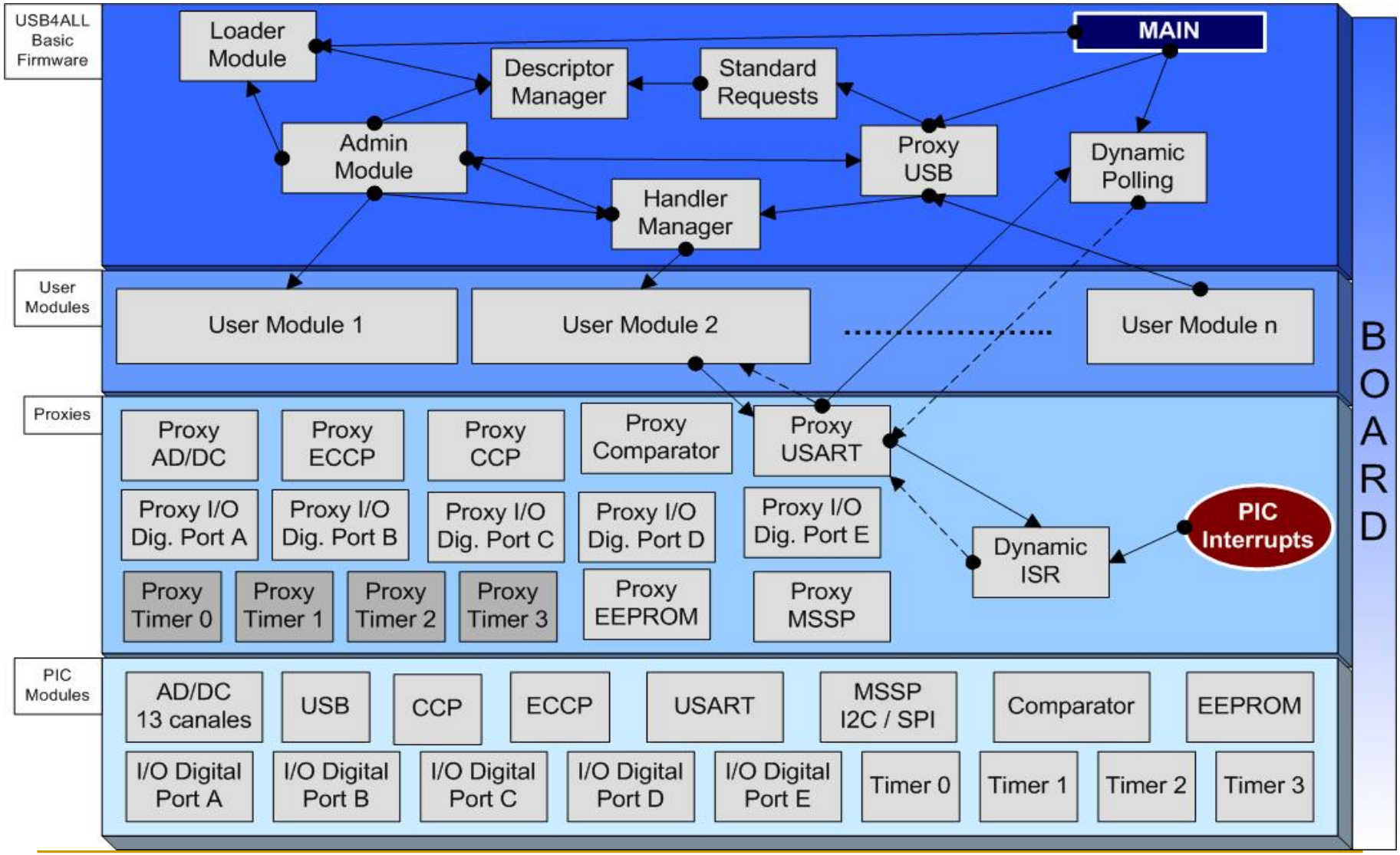
- Modo Clase USB



Vista General



Vista Placa Base



Componentes de la placa base

- PIC18F4550
- Ficha RJ11 (Programador y Debugger)
- Cristal de cuarzo principal y secundario para RTC
- Conector USB
- Conector IDE
- Led de alimentación
- Botón Reset
- Botón Boot

Situación actual y próximos pasos

Situación actual y próximos pasos

- Situación actual
  - Finalizando el estado del arte.
  - Finalizando la primera etapa del diseño (Arquitectura).
  - Experimentación con la placa FS Demo Board.
- Próximos pasos
  - Entrega del documento Estado del Arte.
  - Comienzo de la implementación de la solución (placa base y firmware).
  - Prototipo sencillo que utilice toda la arquitectura.

Preguntas