

Proyecto Fin de Máster

**SENSOR INTELIGENTE PARA
PROCESAMIENTO DE
IMÁGENES SOBRE LINUX
EMBEBIDO EN
PROCESADORES ARM**

Autor: Francisco Javier Cortés Martínez

Tutor: Sergio Toral Marín

Índice de contenido

INTRODUCCIÓN.....	7
Objetivos del proyecto.....	8
SISTEMAS EMPOTRADOS.....	11
Soluciones Hardware.....	11
Fixed Function Engines.....	12
ASSPs (Application-specific Standard Products).....	13
Media Processors.....	14
DSPs (Digital Signal Processors).....	15
Procesadores RISC empotrados.....	16
FPGAs (Field Programmable Gate Array).....	17
Arquitecturas procesadoras SoC.....	18
ARM.....	18
PowerPC.....	19
MIPS.....	19
SS.OO. en sistemas empotrados.....	20
Microsoft Windows CE.....	21
Symbian.....	22
VxWorks.....	23
Distribuciones GNU/Linux.....	23
SISTEMAS DE VISIÓN ARTIFICIAL CON APLICACIÓN AL TRÁFICO RODADO..	27
Aplicaciones de visión artificial.....	27
Clasificación por funcionalidad.....	27
Sistemas de obtención de datos de tráfico.....	27
Sistemas de detección de incidentes.....	28
Sistemas urbanos.....	28
Sistemas para detección de exceso de velocidad.....	28
Sistemas para detección de paso en rojo.....	29
Sistemas de control de acceso.....	29
Clasificación por arquitectura.....	29
Basada en Rack.....	29
Basada en PC.....	30
Arquitectura Compacta.....	31
Sensores de visión.....	33
Formatos de vídeo.....	36
Formatos analógicos.....	36

VGA.....	36
Component Video.....	36
S-Video.....	37
Vídeo Compuesto (Composite Video).....	37
Formatos digitales.....	39
Sin compresión.....	39
Con compresión.....	40
Transmisión de vídeo digital.....	43
Protocolos para videoconferencia.....	43
Protocolo de Video.	44
Protocolos de audio.....	44
CIF (Common Interface Format o Formato de Interfaz común).....	45
Frecuencia de Frames y Motion Compensation.....	45
H.323 – Videoconferencia sobre redes TCP/IP.....	45
H.320 – Videoconferencia sobre RDSI.....	46
H.321 – Videoconferencia sobre ATM.....	47
H.324 – Videoconferencia sobre POTS.....	48
H.310 – Videoconferencia sobre ATM – MPEG2.....	49
RTP.....	49
IMPLEMENTACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL IP.....	53
Diseño del sistema.....	53
Especificaciones iniciales.....	53
Sistema desarrollado.....	54
Tecnología empleada.....	55
Formato de la tarjeta.....	56
Alimentación.....	56
Memorias	57
Conexión de red.....	57
Entradas / Salidas de vídeo.....	60
Encoder MPEG-4.....	61
Dispositivos de almacenamiento masivo.....	61
Conectividad BlueTooth.....	62
Entradas / Salidas digitales.....	63
Puertos Serie.....	63
Instalación de GNU/Linux.....	64
Fase 1: Arranque del microprocesador.....	66
Fase 2: El gestor de arranque.....	67
Fase 3: El Kernel de Linux.....	69
Sistemas de Ficheros.....	71

Herramientas de desarrollo.....	72
Desarrollo de aplicaciones	75
Drivers.....	75
PRP.....	76
GPIO.....	77
Aplicaciones.....	78
CARACTERIZACIÓN DEL SENSOR IP.....	80
Libpcap.....	80
Inicialización del programa.....	82
Filtros de captura.....	83
Bucle de captura.....	86
Extracción de datos.....	87
Procesado de datos.....	89
Transmisión de vídeo y redes.....	90
Unicast.....	90
Multicast.....	91
Unicast + Servidor.....	92
Desarrollo de las pruebas.....	93
Motivación de las pruebas.....	93
Prueba 1: Robustez del sistema en redes sucias.....	93
Prueba 2: Medida del tiempo entre paquetes.....	95
CONCLUSIONES.....	98
Aportaciones del proyecto.....	98
Resultado del proyecto.....	98
Ampliaciones y futuros trabajos.....	99
ANEXO I. MULTICAST.....	102
Usos de Multicast IP.....	102
Direccionamiento.....	102
Tipos de direcciones.....	102
Unicast.....	103
Difusión (Broadcast).....	103
Multicast.....	103
Anycast.....	103
Asignación de direcciones multicast.....	104
Protocolos y aplicaciones.....	104
Rutado.....	105
Distribución a nivel de enlace.....	105
Protocolos usados en multicast IP.....	106
BIBLIOGRAFÍA.....	108

Agradecimientos.....109

La Organización Mundial de la Salud, organismo fundado en 1948 como agencia de Naciones Unidas especializada en cuestiones de salud pública a nivel internacional, ha publicado recientemente informes sobre la prevención de accidentes de tráfico [1]. Estos informes subrayan que las deficiencias en los sistemas de tráfico están causando serios daños sobre la salud pública global. Estos informes también señalan que el nivel de daños por accidentes de tráfico es inaceptable y, en la mayoría de los casos, evitable. Por ejemplo, el coste económico de los accidentes de tráfico se estima en un 1% del PIB (Producto Interior Bruto) en países con bajos ingresos, un 1,5% en países con ingresos medios y un 2% en los países desarrollados. El coste global estimado por año de los accidentes de tráfico a nivel mundial ronda los 518000 millones de dólares. Sin embargo, estos mismos informes hacen hincapié en el escaso dinero que se invierte en investigación y desarrollo para la prevención de los accidentes de tráfico y concluyen que se debería prestar más atención a la investigación y desarrollo de sistemas que permitan aumentar la seguridad del tráfico rodado.

Los objetivos de la investigación en sistemas de seguridad para tráfico son bastante extensos, y cubren, entre otros aspectos, el desarrollo de nuevos sistemas de monitorización de tráfico. Las técnicas actuales para monitorización del tráfico se basan en sensores que tienen capacidades limitadas, y a menudo son costosos y complicados de instalar [2]. El uso de cámaras de vídeo (muchas de las cuales ya están instaladas para la vigilancia de carreteras) junto con técnicas de visión artificial ofrecen una alternativa atractiva a los sensores actuales. El análisis de vídeo en aplicaciones de tráfico y métodos similares para medir, analizar e interpretar parámetros de tráfico es un área de investigación relativamente nueva en el ámbito de la seguridad en el tráfico rodado.

Estos equipos electrónicos dotados de capacidad para procesar señales de vídeo, también conocidos como sensores inteligentes basados en visión artificial, tienen un gran potencial para medir diversos tipos de parámetros de tráfico [3][4]. Las aplicaciones de un sistema de análisis de vídeo completamente desarrollado son numerosas, desde un simple sistema de observación de tráfico (contaje de vehículos) hasta sistemas más complejos (medición de velocidad, detección de incidentes, detección de infracciones, reconocimiento de matrículas, etc.). Si la información

obtenida con este tipo de sensores se procesa de forma distribuida se pueden obtener otros datos de interés a largo plazo, como por ejemplo el comportamiento de usuarios, estadísticas de entradas/salidas o tiempos de viaje. Las características de este tipo de sistemas hacen posible una disminución de los costes de instalación y mantenimiento, pudiendo llevarse a cabo estas operaciones sin interrumpir ni afectar al tráfico.

Objetivos del proyecto

El presente proyecto se enmarca dentro de una apuesta innovadora realizada por la empresa ACISA (Aeronaval de Construcciones e Instalaciones). ACISA, empresa de servicio adquirida a finales del año 2006 por ALDESA, con delegaciones por todo el territorio nacional y una plantilla de casi 600 trabajadores, tradicionalmente se ha dedicado a la instalación y mantenimiento de sistemas de gestión de tráfico tanto urbano como interurbano siendo algunas de las referencias más importantes en Andalucía a los trabajos realizados las siguientes:

- Tráfico Urbano: Contrato de Conservación de las instalaciones de control de tráfico y movilidad de la ciudad de Granada, incluyendo control semafórico, control de uso carriles bus por visión artificial y los controles de acceso a áreas restringidas mediante pilonas automáticas o lectura automática de matrículas.
- Tráfico Interurbano: Sistemas de señalización, vigilancia y medición en los accesos a la ciudad de Almería, instalados previamente a los Juegos del Mediterráneo.
- Alumbrado Público: Conservación de instalaciones de alumbrado público en Sevilla, Granada y Algeciras.
- Infraestructura de comunicaciones: Puerto de Huelva, Parque Metropolitano Industrial y Tecnológico de Granada.

La actividad de I+D de ACISA, cuyo Centro de I+D+i se localiza en el Parque Tecnológico Cartuja 93, se ha ido vertebrando con proyectos, en una actividad continua desde 1998. En la actualidad, y en el periodo Julio 2005-Julio 2007 la empresa ha realizado una apuesta innovadora en el proyecto denominado **“Visión Artificial Aplicada al Tráfico y Transporte”**. El objetivo del proyecto ha sido el desarrollo de equipos para el control del tráfico y transporte, basados en el tratamiento de vídeo e imágenes (visión por computador). El proyecto ha sido respaldado por la Administración Andaluza dentro del Plan de Investigación y Modernización de Andalucía, y en el mismo han colaborado investigadores de los grupos de “Aplicaciones Cibernéticas de la Electrónica a las Tecnologías de la Información (ACETI)”,

“Ingeniería de Automatización, Control y Robótica (IACR)”, ambos de la Universidad de Sevilla, y el Departamento de Óptica de la Universidad de Granada. El proyecto ha sido certificado por AENOR como proyecto de I+D según la normativa UNE 166001 EX.

La visión por computador como disciplina ha desarrollado durante la última década cantidad de técnicas y algoritmos que necesitan ser acomodados en los sistemas reales. Así mismo, la aplicación de la visión por computador en aplicaciones de tráfico y transporte todavía está en estado de fuerte crecimiento y por lo tanto con capacidad de albergar nuevas propuestas novedosas, como el proyecto comentado, por lo cual la empresa está gestionando patentes sobre algunos aspectos concretos del desarrollo.

Los objetivos de este tipo de sistemas son muy amplios: obtener información relativa al tráfico (número de vehículos que circulan en un cierto punto, densidad de tráfico ...), identificación de incidentes (formación de colas, circulación lenta, vehículo parado en arcén ...), detección de infracciones (circulación por zonas o carriles restringidos, paso con semáforo en rojo ...), que pueden incluir la identificación del vehículo causante del mismo (reconocimiento automático de matrículas) .

Los primeros equipos desarrollados dentro del proyecto genérico, están destinados a la medición de datos de tráfico. Estos equipos, de aspecto similar a una cámara, son capaces de analizar el tráfico en varios carriles, obteniendo para cada uno de los mismos el número de vehículos que pasan, la densidad de circulación e indicar la formación de colas.

Los datos obtenidos pueden ser retransmitidos a un centro de control de tráfico con fines muy variados: Elaborar información para el ciudadano, adaptar los tiempos de funcionamiento de los semáforos en las intersecciones, base de datos para la toma de decisiones relativas a movilidad, etc.

Las técnicas actuales de monitorización de tráfico rodado se basan normalmente en sensores que ofrecen pocas o limitadas características y son costosos de instalar y mantener, ya que normalmente se instalan en el pavimento. La instalación y mantenimiento de los equipos basados en visión por computador, a diferencia de los convencionales, puede hacerse sin interrumpir el tráfico en la vía, con el beneficio que esto implica. Además los sensores no se ven afectados por obras, como es el caso de los que se instalan en el pavimento. Estas razones hacen que los responsables municipales encuentren en este tipo de sistemas una alternativa interesante.

Los sistemas empotrados son sistemas con capacidad de procesamiento con una integración firme de hardware y software diseñados para implementar una aplicación específica. Este tipo de sistemas están presentes en equipos de electrónica de consumo (hogar, ocio, comunicaciones), sistemas de seguridad, equipos de instrumentación, sistemas aeroespaciales y de navegación, equipos de transporte, etc.

Para dar una idea de la importancia de los sistemas empotrados se pueden citar que en 2002, sólo el 2% de los 6.2 billones de procesadores fabricados fueron para PCs, Macs y estaciones de trabajo. Tal es la importancia, que la Unión Europea promueve y guía el desarrollo de sistemas empotrados a través de las plataformas ARTEMIS[5] y PROMETEO[6]. Estas iniciativas marcan una tendencia a desplazar la inteligencia de los sistemas desde el servidor hacia dispositivos cada vez más cercanos al usuario. También advierten que es necesario generar tecnología propia sobre la cual basar los desarrollos, para evitar dependencia tecnológica.

Existen múltiples soluciones para el procesamiento de vídeo con sistemas empotrados. A lo largo de este capítulo se presentarán las soluciones tanto hardware como software disponibles, poniendo especial atención en las tecnologías que finalmente se han utilizado en la implementación del prototipo.

Soluciones Hardware

Como punto de partida sería conveniente identificar las características exigibles a un sistema de procesamiento de vídeo y determinar cuáles de ellas son más importantes de cara al algoritmo que se vaya a implementar. Esto nos ayudará a seleccionar la tecnología que mejor se ajuste a nuestras necesidades.

El conjunto de características básicas que se deben tener en cuenta a la hora de seleccionar la tecnología es el siguiente:

- **Velocidad.** Las tareas en procesamiento de video digital, como ocurre con el procesamiento de otros muchos tipos de señal, suponen una alta carga computacional para los procesadores. Por este motivo, es muy importante

seleccionar cuidadosamente si un procesador tiene suficiente velocidad para desarrollar la aplicación.

- **Precio.** El precio del chip es importante, pero el coste por canal o el precio total del sistema (incluyendo librerías, cores propietarios, herramientas de desarrollo, etc.) puede ser más importante.

- **Eficiencia energética.** En el caso de aplicaciones portátiles este aspecto puede ser muy importante dado que la duración de las baterías dependerá de lo eficiente que sea nuestro sistema en el consumo de energía.

- **Flexibilidad.** Algunas clases de procesadores son más flexibles que otras y pueden asumir cambios en las características del producto o actualizaciones con posterioridad a la fabricación del sistema. Sin embargo, por regla general, mientras más flexible sea el procesador, menos eficiente es en coste y energía.

- **Compatibilidad con procesadores anteriores.** Este aspecto es importante si se espera reutilizar software de productos anteriores.

- **Evolución de los productos del fabricante (Vendor Roadmap).** Es importante conocer los planes que el fabricante tiene con respecto al producto, si va a continuar en esa misma línea o si tiene previsto retirar el producto. En este aspecto también influye la vida útil que hayamos previsto para nuestro sistema.

- **Disponibilidad como chip o como núcleo con licencia.** Algunos procesadores se venden como chips ya encapsulados y otros en forma de licencias de propiedad intelectual, para su uso en la construcción de chips a medida.

Una vez señaladas las características deseables de un sistema de procesamiento de vídeo se analizarán las diferentes soluciones que ofrece el mercado:

Fixed Function Engines

Los dispositivos denominados “Fixed-function engines” [7] usan estructuras de procesamiento hardware para una máxima eficiencia. No usan flujos de instrucciones y no son programables. Este planteamiento sacrifica la flexibilidad a favor de una excepcional velocidad de procesamiento, eficiencia energética.

Usar fixed-function engines puede simplificar el sistema de diseño y testeado. Dado que estos dispositivos no son programables, los desarrolladores no tienen que aprender nuevas herramientas de programación o integrar múltiples módulos software y, por supuesto, no tienen que preocuparse por problemas inherentes al software como por ejemplo las interacciones entre las tareas que se procesan en paralelo en los

procesadores y que pueden dar lugar a efectos indeseables como los interbloqueos, las condiciones de carrera, etc.

Los fixed-function engines se comercializan normalmente como propiedad intelectual bajo licencia para su integración en chips. De esta forma, un fixed-function engine se ajusta mejor a tiradas de gran volumen como teléfonos móviles. A veces estos dispositivos se comercializan directamente como chips. Así, por ejemplo, un dispositivo con función fija como un decoder MPEG-2 puede ser la solución más rentable para añadir funcionalidad a un producto existente, particularmente cuando el producto tiene un procesador que puede manejar las funciones de control e interfaz con el dispositivo.

Un ejemplo de este tipo de dispositivos puede ser el decoder 5150 de Hantro. Este dispositivo se vende como propiedad intelectual y se puede encontrar integrado como periférico interno de algunos procesadores. El módulo está pensado para ser usado como coprocesador específico para tareas de decodificación MPEG.

La gran desventaja de los fixed-function engines es su falta de flexibilidad. Dado que no son programables, los desarrolladores no pueden modificar con facilidad el hardware para readaptarlo a los nuevos estándares o para añadirle funcionalidades. Este problema es si cabe más crítico en el ámbito de las aplicaciones de vídeo puesto que muchas de ellas no están aún lo suficientemente maduras y los estándares en los que se apoyan evolucionan rápidamente.

ASSPs (Application-specific Standard Products)

Los ASSPs son chips específicos para una aplicación altamente integrados. A diferencia de los ASICs (Application-specific Integrated Circuits), que normalmente se diseñan para su uso dentro de los propios productos que comercializa la empresa, los ASSPs son diseñados por fabricantes de chips y ofrecidos como chips convencionales a múltiples desarrolladores de sistemas. Puesto que desarrollar un chip complejo es caro y requiere mucho tiempo, los ASSPs normalmente están disponibles sólo para aplicaciones muy determinadas donde existe una gran demanda o se prevé que la haya.

Un ejemplo de ASSPs es el Vaddis 5R de Zoran. Este chip está especializado en el procesamiento de audio y vídeo específico utilizado en grabadores de DVD. Los algoritmos clave que se requieren están bien definidos (compresión / descompresión MPEG-2). Aunque el Vaddis 5R incluye dos procesadores RISC, usa aceleradores

hardware con función fija para las tareas que requieren una carga computacional más elevada como la compresión MPEG-2 o la conversión del espacio de color. Por este motivo, el Vaddis 5R (y otros ASSPs como él) comparten las ventajas y desventajas asociadas a los fixed-function engines: buen rendimiento y eficiencia energética pero tienen una flexibilidad limitada. Esta flexibilidad limitada se traduce en que los diseñadores de sistemas tienen limitadas las oportunidades de marcar una diferencia con otros productos que usen el mismo ASSP. También significa que los diseñadores quedan de alguna manera “atados” a la evolución que establezca el fabricante para el chip, puesto que el fabricante normalmente sólo sacará al mercado un nuevo chip para añadir un conjunto significativo de nuevas funcionalidades o incluso un producto con una funcionalidad bastante diferente.

Los ASSPs que están basados principalmente en núcleos procesadores para tareas de computación intensiva sacrifican el coste y la eficiencia energética para ganar flexibilidad. Los ASSPs de este tipo normalmente se comercializan junto con software de apoyo como drivers, APIs, etc., liberando al diseñador del sistema de gran parte del trabajo a bajo nivel. Sin embargo, el desarrollo de software y la integración pueden requerir un esfuerzo significativo en comparación con el que se requiere cuando se usan ASSPs basados en hardware de función fija.

Media Processors

Los Media Processors [8] se encuentran a medio camino entre los ASSPs y los DSPs en la relación especialización frente a flexibilidad. Estos procesadores están especializados en tareas asociadas con procesamiento de audio y video más que en tareas de procesamiento de señal de carácter general, como es el caso de los DSPs. Los media processors suelen ser multiprocesadores heterogéneos que incorporan un procesador principal similar al de un DSP más dos o tres coprocesadores especializados y periféricos específicos para audio y video.

Un ejemplo de media processor es el Philips PNX1500. Este chip está basado en un potente núcleo procesador eficiente en tareas de procesamiento de video. También incluye una serie de aceleradores hardware de función fija y periféricos especializados. El procesador principal, que es programable por el diseñador, maneja tareas complejas de procesamiento de video como por ejemplo la compresión MPEG-2. A diferencia de los ASSPs como el Zoran Vaddis, el PNX1500 es suficientemente flexible como para ser usado en otro tipo de compresión como por ejemplo H.264. Esta flexibilidad repercute en el precio y en la eficiencia energética.

La naturaleza de los media processors (estructura multiprocesadora heterogénea) hace que el software sea más difícil de elaborar que en otros procesadores programables. Por ejemplo, para implementar una tarea de video determinada, normalmente es necesario programar dos o más elementos procesadores y coordinar su interacción. Para solucionar este problema los fabricantes suelen proporcionar componentes de librería optimizados. Los fabricantes de media processors hacen hincapié en el uso de C ó C++ para el desarrollo de software sobre sus procesadores y no recomiendan o no soportan el uso de lenguaje ensamblador. Esta tendencia por enfocar el desarrollo sobre lenguajes de alto nivel está pensado para abstraer al diseñador de la complejidad que subyace en la arquitectura del procesador. Por otra parte, el diseñador deja en manos del compilador la responsabilidad de generar código eficiente, y esto no siempre es posible. Los desarrolladores de las herramientas de compilación deben invertir un considerable esfuerzo afinando sus herramientas para código de alto nivel para conseguir un buen rendimiento.

DSPs (Digital Signal Processors)

Los DSPs están diseñados para ser usados en un amplio rango de aplicaciones de procesamiento de señal. Típicamente los DSPs emplean menos funcionalidades específicas para vídeo y menos paralelismo que los media processors. Para compensar su falta de paralelismo, los DSPs están diseñados para trabajar a mayores tasas de instrucciones por segundo. Este hecho puede complicar el diseño del sistema e incrementar el consumo de energía. Por otra parte, los DSPs necesitan menos frecuencia de reloj que los procesadores RISC para realizar tareas de video. Para poder alcanzar altas tasas de instrucciones por segundo con frecuencias de reloj no demasiado elevadas, los fabricantes de DSPs han diseñado arquitecturas internas que mejoran el rendimiento en cada ciclo de reloj y aumentan el grado de paralelismo. Un ejemplo muy interesante de esta característica es la arquitectura VLIW (Very Long Instruction Word) que utilizan los DSPs de la familia C6000 de Texas Instruments. A grandes rasgos, este método consiste en empaquetar las instrucciones en grupos de hasta 8 instrucciones que se ejecutan en paralelo gracias a la replicación de las estructuras que componen la ALU (sumadores, multiplicadores, etc.). De esta forma, si por ejemplo tenemos un DSP funcionando a 600 MHz se pueden alcanzar hasta 4800 MMACS (4800 millones de instrucciones de multiplicación/acumulación por segundo). Hay que decir que no siempre se alcanzan estas velocidades puesto que la velocidad depende del código que se esté procesando y de la eficiencia del compilador para empaquetar las instrucciones. En los casos en que haya instrucciones de salto o instrucciones que esperan el resultado de las instrucciones inmediatamente anteriores es posible que no sea posible empaquetar las 8 instrucciones y en su lugar sólo se

puedan empaquetar unas pocas, reduciéndose así la velocidad de procesamiento. Este hecho también complica la programación en ensamblador, por lo que se recomienda trabajar con lenguajes de alto nivel y dejar la tarea del empaquetado de instrucciones al compilador.

Las principales ventajas de los DSPs son su flexibilidad y la potencia de sus herramientas de desarrollo. Históricamente los DSPs han tenido herramientas de compilación un poco pobres e ineficientes, pero en los últimos años la tendencia ha cambiado notablemente y las herramientas de desarrollo han evolucionado hacia entornos más amigables.

Una diferencia importante entre los DSPs y los procesadores RISC es el soporte para sistemas operativos. Los DSPs normalmente soportan un conjunto reducido de sistemas operativos en tiempo real con una funcionalidad muy básica, pero no suelen soportar sistemas operativos convencionales como Linux o Windows CE. En consecuencia, muchos diseñadores usan un DSP para llevar a cabo el procesamiento de vídeo y un procesador RISC para ejecutar un sistema operativo que se encargue de las tareas no relacionadas con el vídeo. Sin embargo, recientemente, algunos fabricantes de DSPs han creado sofisticados sistemas operativos o han hecho adaptaciones para que puedan ser utilizados en sus procesadores, como en el caso de Linux, aunque estas distribuciones no son gratuitas.

Históricamente, los fabricantes de DSPs no han considerado prioritario mantener la compatibilidad de generación en generación y esto ha hecho más difícil la portabilidad y reutilización de código de un procesador a la generación siguiente. Esto también está cambiando, y en la actualidad el código puede ser reutilizado en distintos DSPs, como es el caso de la familia C6000 de Texas Instruments.

Procesadores RISC empotrados

Estos procesadores son bastante populares en aplicaciones empotradas. Históricamente han sido máquinas de propósito general con algunas funcionalidades añadidas no específicas por lo que se han usado en aplicaciones de vídeo para la coordinación de las tareas y para todas aquellas tareas que rodean estas aplicaciones pero que no son específicamente de vídeo.

Hasta ahora, los procesadores RISC sólo eran suficientemente rápidos para llevar a cabo tareas de procesamiento de vídeo muy simples. Hoy, sin embargo, el aumento de la frecuencia de trabajo permite a los procesadores RISC asumir mayores cargas

computacionales. Además, los procesadores RISC están ganando paralelismo y se les están añadiendo características específicas para video. Mientras que las tareas que demandan muchos recursos como la compresión de vídeo en alta resolución están por encima de la capacidad de los procesadores RISC empotrados, estos procesadores se muestran adecuados para tareas de vídeo más ligeras como por ejemplo para aplicaciones de videoconferencia. Un ejemplo de procesadores RISC empotrados son el los procesadores de la familia XScale de Intel, basados en núcleos ARM.

A pesar de la menor eficiencia en el tratamiento de video que tienen estos procesadores, los RISC se benefician de otras ventajas en el campo del desarrollo de aplicaciones. Los procesadores RISC empotrados suelen estar respaldados por una infraestructura sofisticada de desarrollo de software y soportados por comunidades enteras de programadores. Por regla general, los procesadores RISC son más fáciles de programar que el resto de procesadores. La desventaja está en que no hay tanto soporte específico para aplicaciones de video dado el carácter más general de estos procesadores.

La evolución de los procesadores RISC es más clara que la de los otros chips expuestos en los apartados anteriores de modo que la planificación resulta más fácil para los diseñadores que contemplan la creación de múltiples generaciones de productos. Además, la compatibilidad con procesadores más antiguos casi siempre se mantiene. Otra de las ventajas de muchos procesadores RISC es que el mismo núcleo puede ser utilizado por varios fabricantes, como es el caso de los núcleos ARM.

FPGAs (Field Programmable Gate Array)

Aunque las FPGAs no es lo primero que nos viene a la cabeza a la hora de abordar un problema de procesamiento de video, lo cierto es que su flexibilidad y su alto grado de paralelismo pueden ser muy adecuadas para aplicaciones de procesamiento de vídeo que supongan una carga elevada. Las FPGAs contienen un array de bloques lógicos configurables, interconexiones programables, bloques de entrada/salida y, en algunos casos, bloques especializados de función fija.

Las FPGAs pueden ser configuradas para ajustarse a los requerimientos de una aplicación de video y pueden proporcionar potencia de cálculo para computación masiva y un alto ancho de banda en las interfaces con la memoria. Algunas FPGAs disponen de bloques específicos como multiplicadores, PLLs y bloques de memoria que pueden mejorar sustancialmente el rendimiento en algoritmos de procesamiento de video. Las FPGAs constituyen el grupo más flexible entre las soluciones expuestas y

pueden ser reprogramadas para implementar nuevas funcionalidades o adaptarlas a los estándares emergentes. Desafortunadamente, esta alta flexibilidad se paga en el coste por unidad y en la eficiencia energética. Por ejemplo, una FPGA es bastante menos eficiente energéticamente que un ASIC o un ASSP y pueden llegar a costar hasta 100 ó incluso 1000 dólares por unidad. Sin embargo, los fabricantes de FPGAs han introducido recientemente dispositivos más baratos poniendo las FPGAs al alcance de un mayor rango de aplicaciones. El avance de la tecnología ha posibilitado que las FPGAs tengan cada vez más capacidad de procesamiento, así, una FPGA actual puede contener alrededor de 180000 elementos lógicos y 384 multiplicadores de 18 bits. Para hacernos una idea de la evolución en número de elementos y precio de las FPGAs citaremos algunos datos:

En 1996 con 25\$ se podía comprar el equivalente a 1000 elementos lógicos mientras que en 2005, con la misma cantidad, se podía comprar el equivalente a 33000 elementos lógicos, 500KBytes de RAM y 35 multiplicadores 18x18 bits.

Una previsión del año 2000 indicaba que en ese mismo año, los modelos más potentes de FPGAs tenían alrededor de 2 millones de puertas lógicas y estimaban su potencia de cálculo en 1600 MMACS. Esa misma previsión indicaba que para el año 2002 el número de puertas se elevaría hasta los 10 millones y la potencia de cálculo hasta las 500000 MMACS.

Arquitecturas procesadoras SoC

Entre las tecnologías presentadas anteriormente, nos centraremos en los procesadores RISC empotrados. Se escogió esta tecnología fundamentalmente por tres motivos: el gran soporte para el desarrollo de aplicaciones, la posibilidad de instalar un sistema operativo y el bajo coste de los dispositivos.

Dentro del ámbito de los procesadores RISC, existe una gran variedad de arquitecturas de sistemas empotrados. Diversas empresas se dedican a fabricar microprocesadores y chips para estos sistemas. Entre las arquitecturas disponibles destacan tres: ARM, PowerPC y MIPS.

ARM

ARM (Advanced RISC Machine) es una familia de procesadores producida por ARM Holdings Ltd. Fue creada en 1985 por el Acorn Computer Group, como el primer procesador RISC con gran impacto comercial en el mundo.

Hay que resaltar que ARM no fabrica sus propios procesadores, sino que vende bajo licencia el core de la CPU, que luego integran otros fabricantes en sus dispositivos. A pesar de este hecho, existe compatibilidad entre los procesadores basados en núcleos ARM aunque sean de distinto fabricante puesto que utilizan el mismo juego de instrucciones y el código binario generado por las herramientas de compilación es válido para todos los núcleos ARM. Existen muchos fabricantes que comercializan procesadores basados en núcleos ARM, como Intel, Freescale, Toshiba, Samsung, Texas Instruments, etc.

Gracias a su diseño, el ARM tiene relativamente pocos componentes en el chip, por lo que no alcanza altas temperaturas y tiene bajos requerimientos de energía. Esto lo ha hecho perfecto para el mercado de aplicaciones empotradas (embedded applications). Además, esta arquitectura está muy bien soportada por Linux, lo que ha hecho que los procesadores ARM estén muy extendidos en aplicaciones empotradas. El diseño del ARM se ha convertido en uno de los microprocesadores más usados del mundo. Hoy en día, cerca del 75% de los procesadores de 32 bits poseen este chip en su núcleo.

PowerPC

La arquitectura PowerPC (PPC) nació de la colaboración entre IBM, Motorola y Apple. De las ideas de las tres firmas se creó la arquitectura POWER (Performance Optimization With Enhanced RISC). Esta arquitectura es conocida principalmente por su uso en los ordenadores Macintosh de Apple, pero se utiliza también en estaciones de trabajo y sistemas empotrados. Al igual que ARM, existe un gran soporte para esta arquitectura en Linux. Existen una gran variedad de CPUs PowerPC en las que Linux funciona correctamente.

Adicionalmente, hay un gran número de aplicaciones para procesadores x86 (arquitectura de los PCs de escritorio) disponibles para la arquitectura PPC. También hay soporte para Java y aplicaciones como OpenOffice han sido portadas a PowerPC. La comunidad Linux es muy activa en muchas áreas de desarrollo para esta arquitectura. La arquitectura PowerPC es utilizada en videoconsolas de última generación.

MIPS

La arquitectura MIPS (Microprocessor without Interlocked Pipeline Stages)

surgió obra de un proyecto de John Hennessey para la universidad de Stanford. Esta arquitectura es famosa por ser la base de las videoconsolas Nintendo 64 y Sony Playstation 1 y 2, aunque también se puede encontrar en muchos sistemas empotrados.

Al igual que ARM, la compañía que diseña las CPUs MIPS, MIPS Technologies Inc. vende sus diseños a terceros. Pero, a diferencia de ARM, existen varias implementaciones de sets de instrucciones que difieren según su versión. Compañías como IDT, Toshiba, Alchemy y LSI disponen de implementaciones de 32 bits de MIPS. Existen diseños MIPS de 64 bits que son propiedad de las compañías IDT, LSI, NEC, QED, SandCraft y Toshiba.

La primera versión de Linux en arquitecturas MIPS se realizó para dar soporte a estaciones de trabajo basadas en MIPS. Más tarde se comenzó a portar Linux a sistemas empotrados MIPS. Comparado con otras distribuciones como ARM o PowerPC, el uso de Linux en arquitecturas MIPS se encuentra muy limitado. Muy pocas de las grandes distribuciones de Linux han sido portadas a esta arquitectura.

SS.OO. en sistemas empotrados

Aunque actualmente existen varios sistemas operativos empotrados disponibles (Wind River's VxWorks, Microsoft Windows CE, QNX Neutrino, etc), Linux es claramente el líder en este tipo de sistemas.

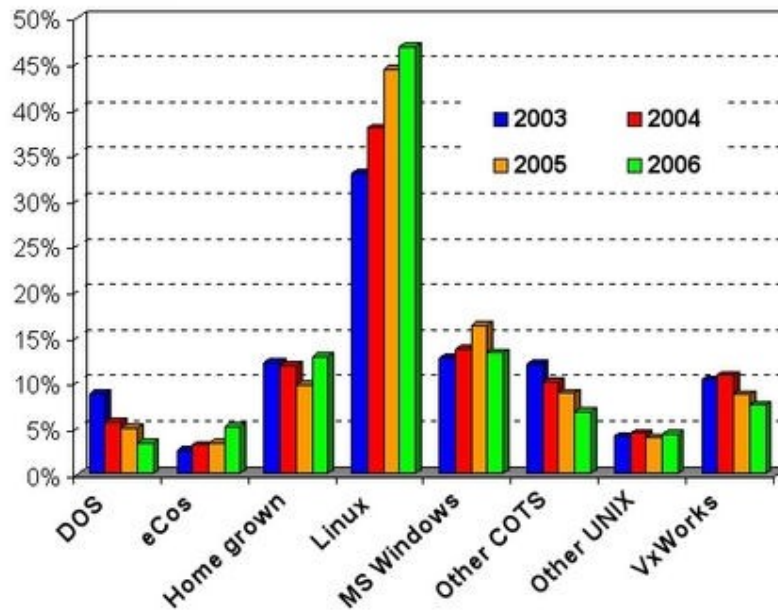


Figura 1: Cuota de mercado de los SS.OO. para sistemas empotrados

Microsoft Windows CE

Windows CE es una variante del sistema operativo Windows de Microsoft para sistemas empotrados. No se trata de un sistema Windows recortado en prestaciones sino más bien de un núcleo diferente al de las versiones de Windows para PCs. Está soportado en procesadores Intel x86 y compatibles, MIPS, ARM y SuperH.

Windows CE es un sistema operativo comercial optimizado para dispositivos con una capacidad de almacenamiento (el núcleo podría ejecutarse con algo menos de 1MB de memoria). A menudo los dispositivos donde se integra Windows CE no disponen de almacenamiento en disco y pueden constituir sistemas cerrados que no admiten ampliaciones. Windows CE puede ser considerado como un sistema operativo en tiempo real, con una latencia de interrupción determinista. Soporta 256 niveles de prioridad. La unidad fundamental de ejecución es el hilo. Esto ayuda a simplificar la interfaz y mejorar los tiempos de ejecución.

La primera versión incluía una interfaz gráfica similar a la de Windows y un número de aplicaciones populares de Microsoft, todas ellas reducidas en espacio y optimizadas en velocidad para ser ejecutadas en dispositivos tales como PDAs. Desde entonces Windows CE ha evolucionado hacia un modelo basado en componentes, orientado a aplicaciones empotradas y en tiempo real. En los últimos años ha sido portado a otras plataformas como Pocket PC, Smartphone e incluso consolas de

videojuegos.

A diferencia de otros sistemas operativos de Microsoft, parte de Windows CE se ofrece en forma de código fuente. Los primeros fragmentos de código fuente fueron ofrecidos a diversos fabricantes para que pudieran ajustarlo a su hardware. Más tarde productos como Platform Builder ofrecieron varios componentes en forma de código fuente. Los componentes que no dependen del entorno hardware y que constituyen el núcleo del sistema siguen distribuyéndose en forma de código binario únicamente.

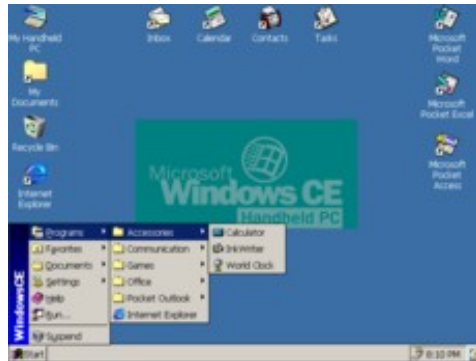


Figura 2: Apariencia del escritorio de Windows CE

Symbian

Symbian es un sistema operativo propietario diseñado para dispositivos móviles con librerías asociadas, interfaz de usuario e implementaciones de herramientas comunes producido por Symbian Ltd. Symbian es un descendiente del sistema EPOC de Psion y funciona exclusivamente sobre procesadores ARM.

Detrás de Symbian existe un grupo de importantes empresas del sector de las telecomunicaciones móviles como Nokia, Sony Ericsson, Panasonic, Siemens y Samsung.

Symbian está estructurado, como muchos otros sistemas operativos, de manera que admite multitarea “pre-emptiva”, multihilo y protección de memoria.

Symbian está dirigido principalmente a dispositivos de mano tales como teléfonos móviles, PDAs, etc. que tienen recursos limitados y deben permanecer funcionando durante largos periodos de tiempo (meses o incluso años). Una característica importante de Symbian es que la programación está orientada a eventos y la CPU se desactiva cuando las aplicaciones no están tratando directamente un evento. Esto se logra gracias a un lenguaje de programación denominado “active objects”. El uso

correcto de estas técnicas permite aumentar la duración de las baterías. Esta forma de programación favorece el uso de C++ aunque también pueden crearse programas para Symbian escritos en OPL, Python, Visual Basic, Perl, Java ME, etc.

VxWorks

VxWorks es un sistema operativo en tiempo real parecido a Unix creado y comercializado por Wind River Systems. Como la mayoría de los SS.OO. de tiempo real, VxWorks incluye un núcleo que soporta multitarea, planificación “pre-emptiva”, rápida respuesta a interrupciones, facilidades para sincronización y comunicación entre procesos, y un sistema de ficheros. Las versiones más recientes de VxWorks soportan ahora llamadas al sistema pSOS ya que Wind River posee ambos sistemas operativos en tiempo real.

Entre las características distintivas de VxWorks figura la eficiente compatibilidad con el estándar POSIX, el manejo de la memoria, las facilidades para sistemas multiprocesadores y la depuración simbólica y a nivel de código.

VxWorks utiliza herramientas de compilación cruzada, al igual que otros muchos sistemas operativos para sistemas empotrados. Dichas herramientas están preparadas para ser ejecutadas en diversos entornos y son capaces de generar código para diferentes arquitecturas empotradas así como para el propio host mediante VxSim.

Actualmente VxWorks ha sido portado a múltiples arquitecturas como, por ejemplo, Intel x86, MIPS, PowerPC, SuperH y ARM.

Distribuciones GNU/Linux

La popularización de los sistemas GNU/Linux para empotrados se ha visto favorecida por las ventajas inherentes al uso de software libre:

- Independencia del proveedor
- Bajo coste
- Licencia GPL
- Comunidad de código abierto

Debido a la gran heterogeneidad de los sistemas embebidos, no existe una versión estándar de Linux, sino que hay que hablar de muchas distribuciones de Linux que

cubren una o varias arquitecturas procesadoras. Todas ellas tienen en común los mismos componentes básicos, incluyendo el kernel de Linux, drivers, comandos, entornos de ventanas, utilidades básicas y librerías. Las diferencias entre las distribuciones se centran normalmente en cuál de los cientos de utilidades existentes se incluyen, en los módulos añadidos (tanto en código fuente como propietario), en las modificaciones del kernel y en la gestión de los procesos de instalación, configuración, mantenimiento y actualización.

Precisamente la necesidad de adaptación del kernel de Linux a las diferentes arquitecturas ha creado comunidades enteras de desarrolladores para cada una de ellas. En la siguiente tabla se muestran las comunidades existentes para el desarrollo de sistemas GNU/Linux sobre empotrados:

Arquitectura	Soporte en Internet
ARM	http://www.arm.linux.org.uk/developer
x86	http://www.kernel.org
PowerPC	http://penguinppc.org
MIPS	http://www.linux-mips.org
SuperH	http://linuxsh.sourceforge.net
M68K	http://www.linux-m68k.org

Tabla 1: Arquitecturas y comunidades de desarrolladores

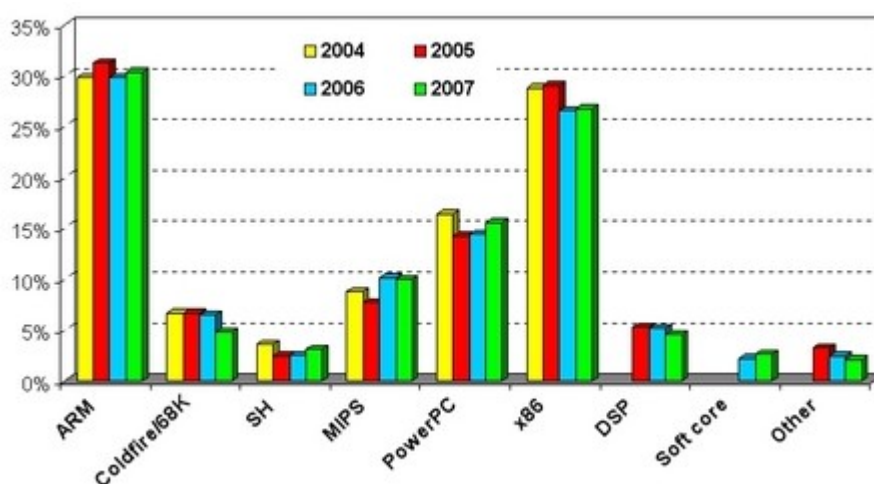


Figura 3: Arquitecturas empotradas soportadas en GNU/Linux

Las distribuciones de sistemas empotrados contienen, fundamentalmente, herramientas de desarrollo. Algunas de las distribuciones de empotrados más famosas

son FSM Labs, Linuxworks, Metrowerks, Montavista, Snapgear, SysGO, TimeSys.

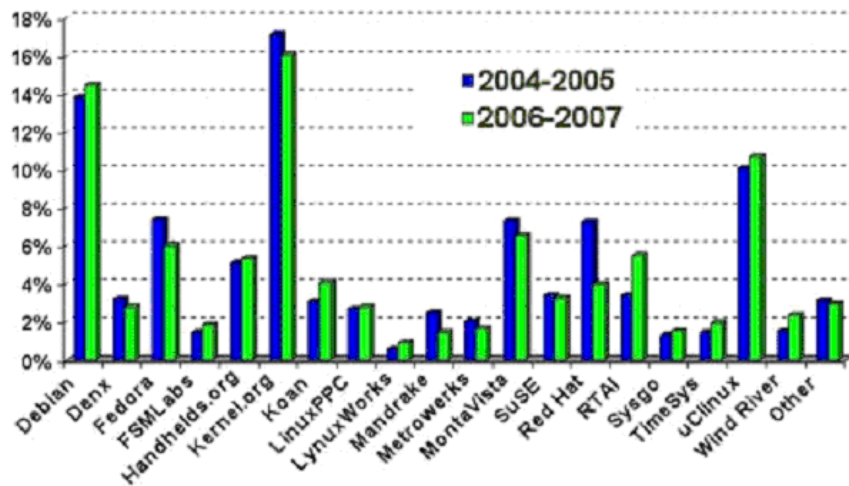


Figura 4: Distribuciones GNU/Linux en sistemas empotrados

En el año 2002, y al amparo del Quinto Programa Marco para la investigación y desarrollo tecnológico, la Comisión Europea financió un proyecto destinado a promover la adopción del Software libre en sistemas empotrados. El proyecto conocido como INES [9] (Industrial Embedded Systems) está formado por participantes de seis países diferentes. El objetivo de INES es el desarrollo y la popularización de aplicaciones comerciales para sistemas empotrados.

Aplicaciones de visión artificial

En la actualidad existe una amplia gama de productos de visión artificial orientados a aplicaciones de tráfico y dependiendo de la aplicación concreta a la que estén destinados tienen unas características u otras.

Los sistemas de visión artificial para tráfico se pueden clasificar atendiendo a dos criterios fundamentales: la funcionalidad y la arquitectura.

Clasificación por funcionalidad

Dentro de la clasificación por funcionalidad podemos distinguir dos grupos:

- Sistemas de detección y clasificación
- Sistemas de detección de infracciones (Enforcement)

Los sistemas de detección y clasificación utilizan un flujo de vídeo para obtener las variables de tráfico deseadas. Entre ellos podemos encontrar los siguientes tipos de equipos:

Sistemas de obtención de datos de tráfico

Los sistemas para la obtención de datos de tráfico se emplean generalmente en vías interurbanas. Los datos se obtienen por cada carril, siempre que hayan sido definidas correctamente las regiones de detección y tienen un carácter macroscópico. Entre las variables que pueden ser estudiadas están:

- Intensidad
- Velocidad
- Distancia entre vehículos
- Lonitud de los vehículos
- Nivel de servicio

Sistemas de detección de incidentes

Los sistemas para detección de incidentes generalmente se instalan en carretera o en túneles y sirven para detectar situaciones anómalas e informar de ello a un centro de control para que puedan tomarse las medidas oportunas y evitar accidentes. Entre las situaciones que pueden detectarse están las siguientes:

- Vehículo parado
- Conducción en sentido contrario
- Presencia de peatones
- Pérdida de la carga
- Fuego y humo
- Colas
- Descenso de velocidad

Sistemas urbanos

Los sistemas urbanos, como su propio nombre indica, se instalan en vías urbanas. Funcionan de forma parecida a los sistemas de detección de datos de tráfico, pero se centran más en la obtención de las siguientes variables:

- Presencia
- Contaje
- Longitud de colas

Los sistemas de detección de infracciones pueden procesar también flujos de vídeo, pero por la naturaleza de sus aplicaciones, es más habitual que procesen fotografías. Entre los sistemas de detección de infracciones encontramos los siguientes tipos:

Sistemas para detección de exceso de velocidad

Estos sistemas registran la velocidad de los vehículos y capturan imágenes de aquellos que excedan la velocidad permitida. Existen diversos métodos para detectar la velocidad de los vehículos:

- Radar
- Láser
- Detectores piezoeléctricos
- Espiras
- Por procesamiento del propio flujo de vídeo capturado

El resultado de la detección suele ser una fotografía del vehículo que ha cometido la infracción aunque en algunos casos estos equipos recurren a algoritmos de OCR (Optical Character Recognition) para obtener el número de la matrícula del vehículo y poder ofrecer el dato en formato electrónico para su tratamiento automatizado.

Sistemas para detección de paso en rojo

Estos sistemas capturan imágenes y/o fragmentos de vídeo de aquellos vehículos que rebasan semáforos en rojo. Suelen grabar breves secuencias de vídeo segundos antes y después de la infracción. Al igual que en el caso anterior, se pueden apoyar en algoritmos de OCR para el tratamiento automatizado de las infracciones.

Sistemas de control de acceso

Estos sistemas se utilizan para permitir o denegar el paso en función de la matrícula del vehículo que intente acceder. Algunas de las aplicaciones típicas de estos sistemas son:

- Parkings privados
- Acceso a carriles bus
- Peajes

En estos casos los sistemas sí recurren a algoritmos de OCR para contrastar datos con una base de datos de matrículas.

Clasificación por arquitectura

En cuanto a la arquitectura del sistema existen tres líneas principales

Basada en Rack

El procesamiento se centraliza en uno o varios racks donde se apilan varias

tarjetas y al que llegan las señales de vídeo (normalmente analógicas) procedentes de las cámaras instaladas. El hardware de las tarjetas es específico para el procesamiento de vídeo. Esta arquitectura tiene como ventaja la simplicidad de los equipos terminales (cámaras) pero plantea un problema a la hora de transmitir el vídeo analógico hasta el rack si la distancia a cubrir es alta.

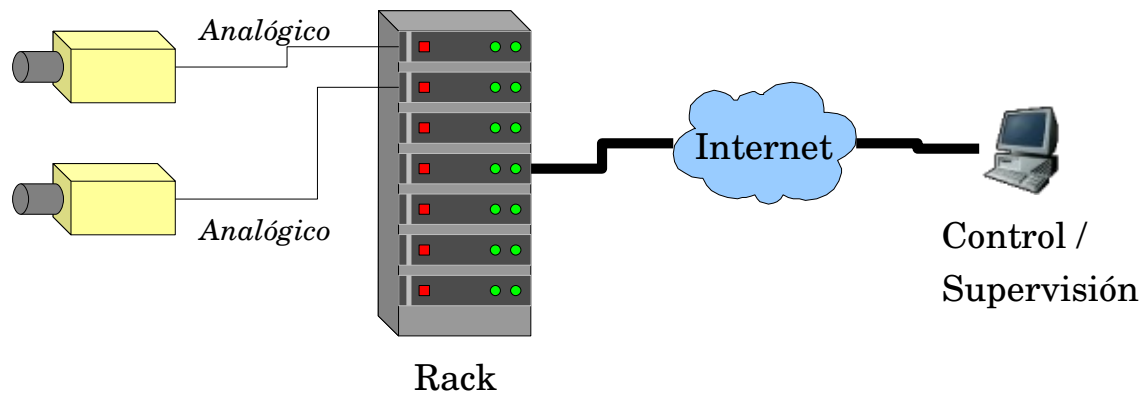


Figura 5: Arquitectura basada en rack

Basada en PC

Esta arquitectura también es centralizada, pero en lugar de usar un conjunto de tarjetas insertadas en un rack, el elemento central es un ordenador industrial. Las diferentes cámaras instaladas están dotadas de una “cierta inteligencia” y transmiten las imágenes y/o el flujo de vídeo en digital sobre IP. Estas imágenes y secuencias de vídeo se almacenan en el PC industrial para su procesamiento, por lo que es habitual encontrar servidores FTP en estos ordenadores y, por tanto, los elementos terminales deben tener la capacidad suficiente como para implementar un cliente FTP. Los elementos terminales se complican un poco pero a cambio se ganan las ventajas inherentes al uso de redes de área local:

- Infraestructura común para datos, vídeo e imágenes
- Las redes de transmisión de datos están mejor preparadas para transmitir la información a grandes distancias que las redes de cable coaxial para señales analógicas.
- Se pueden aprovechar infraestructuras existentes

Por otra parte, el uso de ordenadores industriales aporta las siguientes ventajas e inconvenientes:

- Abaratamiento de coste
- Mayor facilidad para el diseño de aplicaciones
- Menor especificidad del hardware, lo que resulta en un menor rendimiento.

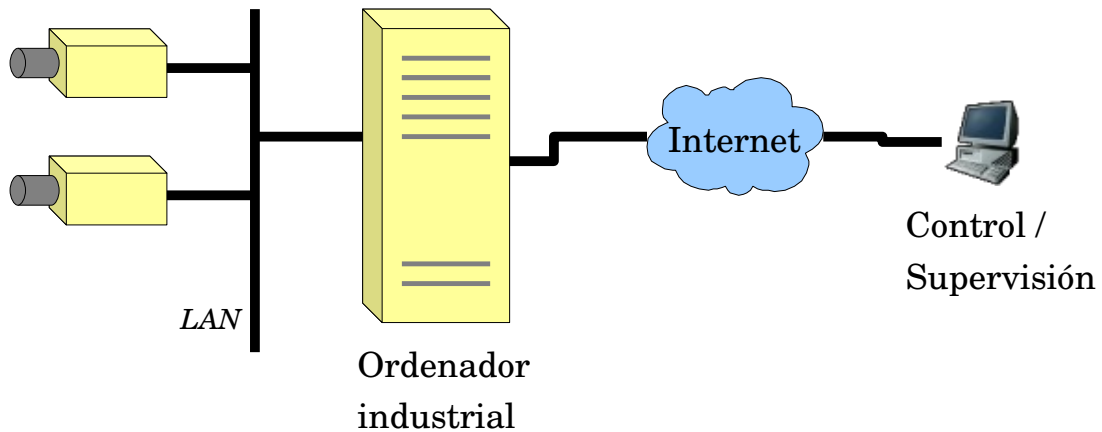


Figura 6: Arquitectura basada en PC

Arquitectura Compacta

En esta arquitectura la capacidad de procesamiento recae en cada uno de los elementos terminales, que se convierten así en “terminales inteligentes”. Estos terminales transmiten los resultados del procesamiento a través de una red IP, de modo que se puede reducir enormemente las necesidades de ancho de banda dependiendo de la aplicación. La arquitectura es claramente distribuida y permite implementar nuevos escenarios en los que los diferentes terminales pueden interactuar entre ellos, aumentando las posibilidades del sistema completo. El hardware de procesamiento, al igual que en la arquitectura basada en rack, es específico. Como contrapartida está el incremento del coste y la complejidad de los elementos terminales, pero hereda también las ventajas de usar redes de área local descritas anteriormente.

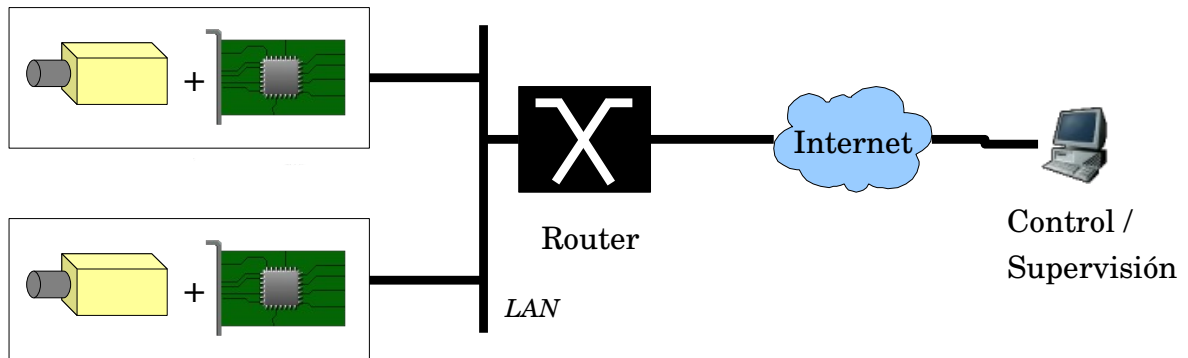


Figura 7: Arquitectura compacta

Clasificación por Funcionalidad		
Detección y Clasificación	Obtención de datos de tráfico	
	Detección de incidentes	
	Urbanos	
Detección de infracciones (Enforcement)	Exceso de velocidad	con OCR
		sin OCR
	Paso en rojo	con OCR
		sin OCR
	Control de acceso (con OCR)	

Tabla 2: Resumen de la clasificación por funcionalidad

Clasificación por Arquitectura
Basada en Rack
Basada en PC
Arquitectura compacta

Tabla 3: Resumen de la clasificación por arquitectura

Sensores de visión

Las tecnologías actuales empleadas para la captura de una imagen digital pasa por el uso de sensores CCD (Charge Coupled Device) [10] o sensores CMOS (Complementary Metal Oxide Semiconductor) [10]. Ambos sensores convierten la luz en carga eléctrica que es, a su vez, procesada para generar una señal de tensión analógica.

En un sensor CCD, la carga de cada píxel se transfiere a través de un reducido número de nodos de salida (habitualmente sólo uno) para convertirse en tensión y enviar la señal eléctrica fuera del integrado mediante un búfer. Los píxels se dedican sólo a la recogida de la luz y la uniformidad de la imagen obtenida es elevada. Este factor es importante para la obtención de imágenes de calidad.

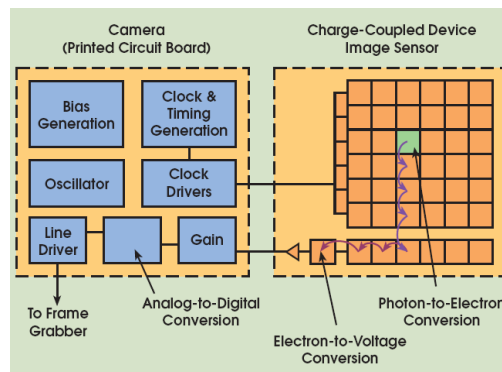


Figura 8: Sensor CCD y cámara

En un sensor CMOS cada píxel tiene su propio convertidor de carga eléctrica a tensión. El sensor también incluye amplificadores, corrección de ruido y circuitos digitalizadores de tal manera que la salida de un sensor CMOS es video digitalizado. Esta funcionalidad adicional aumenta la complejidad del diseño y reduce la cantidad de área de silicio dedicada a la recogida o captura de la luz. Como cada píxel realiza su propia conversión analógico-digital, la uniformidad de la imagen es inferior al caso anterior aunque como ventaja permite disminuir el número de dispositivos electrónicos externos necesarios para desarrollar una cámara básica.

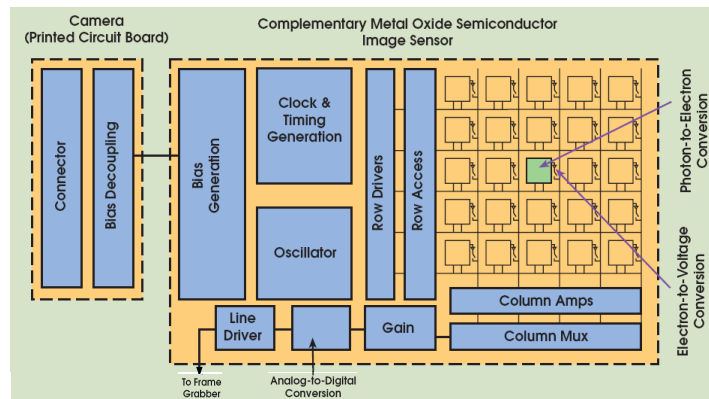


Figura 9: Sensor CMOS y cámara

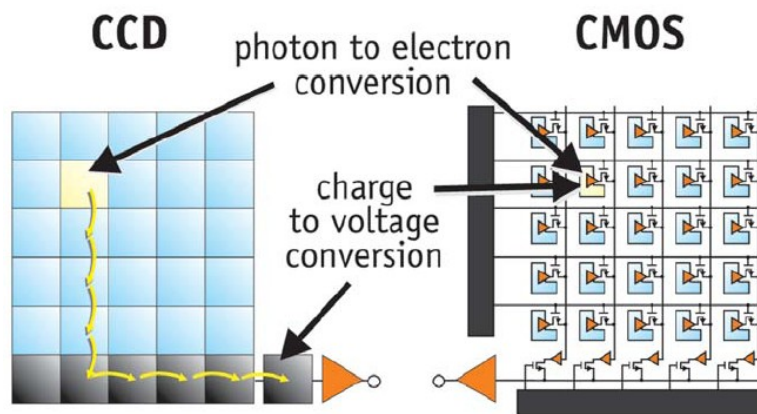


Figura 10: Conversión de la carga en tensión

Ambos dispositivos aparecen a finales de 1960 y comienzos de los 1970. Inicialmente los sensores CCD predominaron puesto que ofrecían mayor calidad de imagen con la tecnología disponible. Los sensores CMOS cobraron interés a partir de la década de los 1990, cuando se prevé el desarrollo de este tipo de sensores con menores consumos, mayores densidades de integración (camera-on-chip) y menores costes de fabricación que los sensores CCD.

Los sensores CCD se han venido empleando tradicionalmente en sistemas que requieran una elevada calidad de la imagen (estimada en ruido y eficiencia cuántica) y a costa del tamaño del dispositivo, mientras que los sensores CMOS se emplean en sistemas que requieren una elevada integración y un bajo tamaño y consumo, obtenidos a costa de peor calidad de imagen y mayor coste. En la actualidad, sin embargo, no está claro el rango de aplicación de cada tipo de sensor debido a que los diseñadores de sensores CMOS han elevado notablemente la calidad de la imagen obtenida (aproximándose a la ofrecida por los dispositivos CCD) mientras que los fabricantes de sensores CCD han disminuido los requerimientos en consumo de potencia y tamaño de los píxeles de los mismos. Como resultado, podemos encontrar

aplicaciones de bajo consumo que emplean sensores CCD y dispositivos con elevada calidad de imagen (cámaras fotográficas de altas prestaciones) que incorporan sensores CMOS.

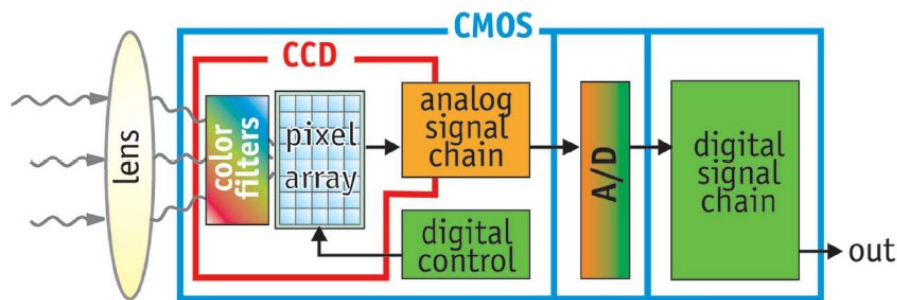


Figura 11: Integración en cámaras basadas en sensores CMOS y CCD

A nivel de chip, el coste es similar, a pesar de que los sensores CMOS prometían inicialmente mucho en este sentido. En principio las cámaras basadas en sensores CMOS requieren menos componentes aunque para conseguir equiparar, con los sensores CCD, la calidad en la imagen obtenida requieren habitualmente de otros chips adicionales que aumentan el coste y el consumo del sistema final. Por su parte, los sensores CCD son menos complejos y menos costosos de desarrollar que los CMOS. Así, los procesos de fabricación de los sensores CCD tienden a ser más maduros y optimizados y, en general, costaría menos desarrollar un sensor CCD que uno CMOS para una determinada aplicación de altas prestaciones.

Las ventajas en cuanto a coste e integración no se han conseguido mediante el empleo de sensores CMOS. Sin embargo, la velocidad de respuesta de este tipo de dispositivos es muy superior al de los sensores CCD debido al fácil paralelismo de las estructuras internas de salida de datos, lo que los hace interesante en aplicaciones industriales.

CCDs y CMOS continúan siendo complementarios. La elección de una u otra tecnología depende de la aplicación y del vendedor más que de la propia tecnología.

Formatos de vídeo

Formatos analógicos

VGA

Utiliza una línea para cada una de las componentes RGB de cada punto de la pantalla. Se utiliza para transmitir la señal a los monitores de los ordenadores. Debido a que las conexiones que utilizan esta interfaz no alcanzan grandes longitudes, se transmite en banda base directamente cada componente. Esto, junto con el hecho de que las componentes RGB son directamente interpretables por un tubo de rayos catódicos, hacen del VGA un formato con una excelente calidad. Dependiendo de la configuración del monitor se pueden obtener distintas resoluciones (640x480, 800x600, 1024x768) y distintas tasas de imágenes por segundo (suele estar en torno a las 70 imágenes por segundo). Se suele utilizar un conector sub-D de 15 pines, sobre los cuales se transmiten además las señales de sincronismo horizontal y vertical.



Figura 12: Conector VGA

Component Video

Utiliza un cable coaxial para cada una de las componentes YUV ó RGB. YUV es un espacio de color distinto al RGB formado por una señal de luminancia (Y) y dos de diferencia de color o crominancia (U y V). El formato YUV aparece para garantizar la compatibilidad con los televisores en blanco y negro, que solamente son capaces de interpretar la señal de luminancia. La conversión de RGB a YUV está descrita en la norma de la ITU-R BT.601 y el paso de un espacio a otro se reduce a aplicar una matriz de transformación o su inversa, si bien hay que tener en cuenta que el espacio de color YUV es más amplio y algunos colores YUV no son representables en el espacio RGB. Las señales Component Video tienen también una alta calidad. Todos los formatos analógicos basados en YUV (Component Video, S-Video y Vídeo Compuesto) admiten diferentes resoluciones y tiempos de sincronismo siendo los más habituales los correspondientes a los estándares PAL y NTSC.



Figura 13: Conectores Component Video

S-Video

Utiliza dos líneas para transmitir la señal YUV. Por una de ellas transmite la luminancia, mientras que por la otra transmite las dos señales de crominancia moduladas en fase y cuadratura de modo que sobre el mismo ancho de banda se transmiten las dos. Utiliza un conector especial de 4 pines de tipo Mini-DIN. Se utiliza fundamentalmente en los sistemas Hi8 y S-VHS. El hecho de combinar las señales de crominancia modulándolas en fase y cuadratura hace que el formato S-Video pierda calidad frente a Component Video. En cuanto a la resolución y tiempos de sincronismo, se aplican los mismos estándares que sobre Component Video.



Figura 14: Conector S-Video

Vídeo Compuesto (Composite Video)

Utiliza una sola línea coaxial para transmitir la señal YUV. Para formar la señal de vídeo compuesto primero se modulan en fase y cuadratura las crominancias tal y como ocurre en S-Vídeo, pero a diferencia de éste, esa señal de crominancia resultante se desplaza en frecuencia y se suma a la señal de luminancia. La crominancia se aloja en la parte alta del espectro de frecuencia de la señal para evitar interferencias sobre la señal de luminancia de modo que los televisores en blanco y negro puedan discriminarla fácilmente y no les afecte. Dada la naturaleza de estas señales, existen zonas del espectro donde la densidad espectral de potencia es muy baja, y las frecuencias correspondientes a estas zonas están relacionadas armónicamente, de modo que se aprovechan estos “huecos” de la señal de luminancia para colocar en ellos las partes de mayor densidad espectral de la señal de crominancia. A esto se le conoce como “imbricación de espectros”. Como resultado de este proceso, la señal de

crominancia sale más distorsionada y con un menor ancho de banda. El motivo para hacer esto se justifica en el hecho de que el ojo humano es más sensible a una variación de la luminosidad que a una variación de una tonalidad de color. El vídeo compuesto es ampliamente utilizado en los sistemas de televisión convencionales tanto para la radiodifusión como para la transmisión de señales de vídeo entre equipos. Se suelen utilizar jacks RCA (amarillo). Evidentemente, el video compuesto tiene menos calidad que el S-Vídeo. En cuanto a la resolución y tiempos de sincronismo, se aplican los mismos estándares que sobre Component Video.



Figura 15: Conector RCA de vídeo compuesto

Como ya se ha mencionado anteriormente, los formatos de vídeo analógico basados en YUV utilizan ciertos estándares para determinar su resolución y sincronismo. Estas señales de sincronismo van incluidas dentro de las propias señales de luminancia y crominancia y son claramente diferenciables de la propia señal que transporta la imagen. Esto se consigue utilizando valores de tensión notablemente inferiores al nivel del color negro durante los periodos denominados “blanking-time”. Entre una línea y otra siempre hay un periodo que no contiene información de vídeo denominado “blanking horizontal”, y entre una imagen y otra existe un periodo que tampoco contiene información de vídeo y que se denomina “blanking vertical”. Ambos periodos sirven para ajustar las señales de sincronismo horizontal y vertical de modo que el tubo de rayos catódicos pueda representar la imagen correctamente. El resto de la señal sí contiene información de vídeo útil y es denominada “active video”.

Así, el estándar que se ha impuesto en Europa se denomina PAL y se caracteriza por transmitir primero el conjunto de líneas horizontales pares y luego las impares (video entrelazado) de modo que muestran 50 semi-imágenes o campos por segundo, o lo que es lo mismo, 25 imágenes completas por segundo. El estándar define 625 líneas horizontales por cada imagen completa, aunque algunas de ellas no contienen información útil porque forman parte del periodo de blanking vertical. Al tratarse de señales analógicas, no se puede hablar de resolución en pixels para evaluar la calidad de una señal PAL frente a otros estándares. La calidad de una señal que utilice PAL va a depender fundamentalmente del formato de la señal (Composite Vídeo, S-Vídeo o Vídeo Compuesto), que son los que realmente marcan el ancho de banda y la distorsión de la señal. Existen chips que realizan la conversión de señales PAL a digital

obteniendo una resolución de 864x625 pixels totales de los cuales, la imagen real se reduce a 720x576 pixels. Se puede tomar esta referencia cuando se habla de sistemas PAL, pero se insiste en que este dato puede ser engañoso y que la calidad realmente depende del ancho de banda y distorsión de la señal analógica, puesto que si ésta ya viene en mal estado, de nada sirve que tener mucha resolución.

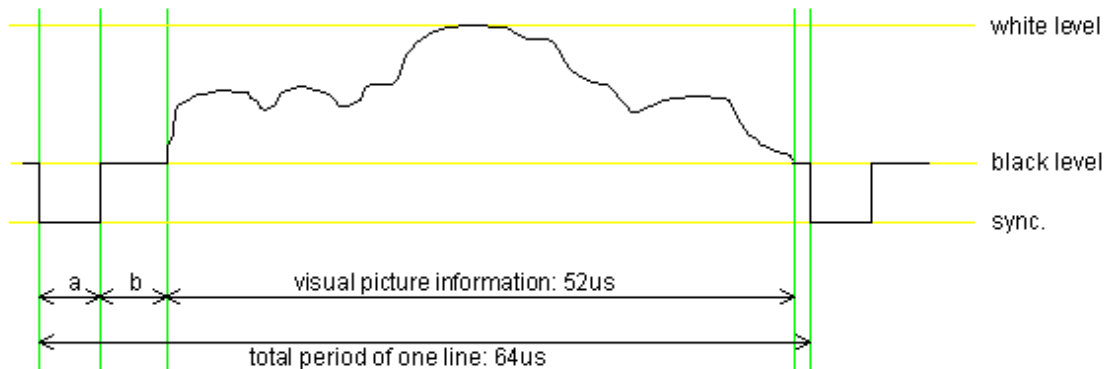


Figura 16: Aspecto de una señal de vídeo compuesto en el dominio del tiempo

El estándar utilizado en U.S.A. y Japón se denomina NTSC y, a diferencia de PAL, define 525 líneas y en vez de funcionar a 50 campos por segundo, lo hace a 60 campos por segundo.

Formatos digitales

Sin compresión

- YUV. Existen varias maneras de componer una señal YUV digital. La diferencia entre unas y otras está en la cantidad de muestras para cada una de las componentes. La relación entre la cantidad de muestras se denota mediante tres números. Así por ejemplo, en YUV 4:4:4 hay una muestra de cada componente (Y, U, V) por cada píxel. En YUV 4:2:2, hay una muestra Y por cada píxel, pero sin embargo sólo hay una muestra de U y de V para cada dos píxeles. En YUV 4:2:0 sólo aparece una de las componentes de crominancia por cada dos píxeles en cada línea (realmente se alterna YUV 4:2:0 en una línea con YUV 4:0:2 en la siguiente línea). La otra componente de crominancia que falta se obtiene de la siguiente línea. El formato YUV 4:2:0 es utilizado por los encoders / decoders MPEG. Como se puede ver, casi todas las alternativas submuestran la crominancia, apoyándose de nuevo en el hecho de que el ojo humano es menos sensible a la crominancia que a la luminancia.

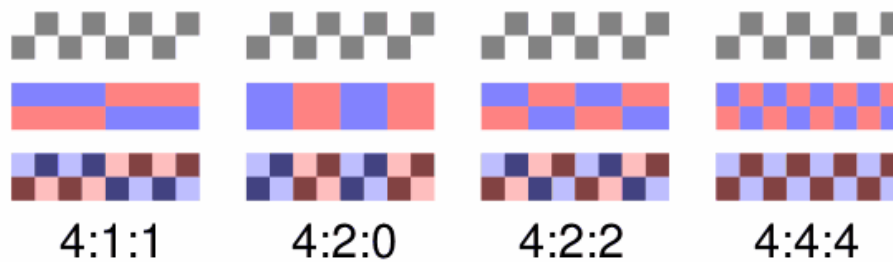


Figura 17: Diferentes tipos de submuestreo

- ITU-R BT.656 o CCIR 656. Esta norma regula la transmisión de vídeo digital sin comprimir. Utiliza el formato YUV 4:2:2 representando el valor de las componentes con una profundidad de color de 8 ó 10 bits. En el caso de sistemas de 625 líneas (asociados a PAL) la resolución es de 864x625 pixels totales de los cuales, la imagen real se reduce a 720x576 pixels. La forma habitual de transmisión es en paralelo, intercalando las muestras de la siguiente manera: Y0, U0, Y1, V0, Y2, U2, Y3, V2, etc. De esta manera, se consigue generar un flujo a 27 MHz (13,5 MPixel/s) para los sistemas con relación de aspecto 4:3. La información de sincronismo está incrustada dentro del propio flujo de bytes durante los periodos de blanking horizontal y vertical. Muchos dispositivos digitales generan y aceptan vídeo en este formato. Esta norma también afecta a los sistemas de 525 líneas (asociados a NTSC).

Con compresión

Evidentemente, el flujo de información del vídeo digital es bastante grande y resulta casi indispensable recurrir a la compresión cuando se necesita almacenar o transmitir por una red el vídeo digital. Para ello existe una serie de algoritmos, de los cuales los más importantes son los de compresión MPEG. MPEG se basa, a grandes rasgos, en describir los cambios respecto a una escena que sirve como base. Existen varias versiones como MPEG-1, MPEG-2, y MPEG-4 [11], siendo esta última ampliamente utilizada para transmitir vídeo por Internet debido a su eficiencia en la compresión. MPEG-4 permite obtener vídeo comprimido con una relación calidad-bitrate superior a la de sus antecesores. Existe una serie de parámetros que puede ayudar a estimar la calidad de un vídeo con compresión MPEG:

- fps (frames per second). Indica el número de imágenes por segundo. Si este parámetro tiene un valor bajo, se apreciarán saltos o discontinuidades entre las imágenes.
- Resolución en pixels. Indica el número de puntos por cada imagen. Existen varias resoluciones estándar, entre las que se pueden destacar las siguientes:

Resolución	Puntos
CIF	352x288
D1 (PAL)	720x576
VGA	640x480
QVGA	320x240
4CIF	704x576
QCIF	176x144

Tabla 4: Tamaños de imagen estandarizados

- **Bitrate.** Indica el número de bits por segundo que se genera tras la compresión. El bitrate puede ser constante o variable a lo largo del flujo de vídeo. A mayor bitrate, mayor será la calidad de la imagen aunque también será mayor el ancho de banda necesario para transmitirlo o el espacio de memoria para almacenarlo.

Formato	VCD	SVCD	DVD	DivX XviD WMV	MOV	ASF SMR nAVI	RM	DV
Resolución NTSC PAL	352x240 352x288	480x480 480x576	720x480 720x576*	640x480*	640x480*	320x240*	320x240*	720x480 720x576
Compresión de vídeo	MPEG1	MPEG2	MPEG2 MPEG1	MPEG4	Sorenson Cinepak MPEG4	MPEG4	RM	DV
Video bitrate kbit/sec	1150	1000 a 2500	3000 a 9000	300 a 1000	300 a 2000	100 a 500	100 a 500	25000
Compresión de audio	MP1	MP1	MP1 MP2 AC3 DTS PCM	MP3 WMA OGG AAC AC3	Sorenson Cinepak MP3	MP3 WMA	RM	DV
Audio bitrate kbit/sec	224	128 a 384	192 a 448	64 a 448	64 a 192	64 a 128	64 a 128	1000 a 1500
Tamaño/Minuto (MB/min)	10	1020	30 a 70	1 a 10	1 a 20	1 a 5	1 a 5	216
Min/74min CD	74min	35 a 60min	15 a 20min	60 a 180min	60 a 180min	120 a 300min	120 a 300min	3min
Horas/DVDR	N/A	N/A	2 a 4hrs (3 a 7hrs ^{***})	13 a 26hrs	13 a 26hrs	26 a 40hrs	26 a 40hrs	20min

Compatib. Reprod. DVD	Muy buena	Buena	Excelente	Poca	Ninguna	Ninguna	Ninguna	Ninguna
Carga de CPU	Baja	Alta	Muy alta	Muy alta	Alta	Baja	Baja	Alta
Calidad	Buena	Muy buena**	Excelente**	Muy buena**	Muy buena**	Decente**	Decente**	Excelente

* valor aproximado, puede ser mayor o menor.
** La calidad del video depende del bitrate y de la resolución de video. Un mayor bitrate generalmente significa una mayor calidad, pero también un mayor tamaño de archivo.
*** DVD con baja calidad de vídeo, similar a la calidad de VCD/SVCD.

Tabla 5: Formatos de vídeo comprimido más habituales

Formatos analógicos			
<i>Tipo de señal</i>		<i>Temporización / Tamaño</i>	<i>Espacio de color</i>
Component Video		VGA	RGB
		PAL	YUV
		NTSC	YUV
S-Video		PAL	YUV
		NTSC	YUV
Video Compuesto		PAL	YUV
		NTSC	YUV
Formatos digitales			
	<i>Espacio de color</i>	<i>Submuestreo</i>	<i>Transmisión</i>
Vídeo sin comprimir	RGB	888	raw
		565	raw
	YUV	4:4:4	raw
		4:2:2	BT.656
		4:2:0	raw
	<i>Compresión</i>		
Vídeo Comprimido		MPEG-1	
		MPEG-2	
		MPEG-4	

Tabla 6: Resumen de los formatos de vídeo

Transmisión de vídeo digital

Existen diversos métodos para la transmisión de vídeo digital en la actualidad. Algunos de estos sistemas, utilizados para la difusión de vídeo, no son realmente sistemas de transmisión en tiempo real, pues almacenan en buffers fragmentos del stream de vídeo que luego transmiten por protocolos como HTTP de la misma manera que se haría para transmitir un archivo. Nuestro interés, por el contrario, se centra más en los sistemas de transmisión en tiempo real por ser más adecuado para los sistemas de visión artificial aplicados al tráfico. Dentro de este grupo de sistemas nos centraremos en dos: los estándares para videoconferencia y el protocolo RTP y sus derivados.

Protocolos para videoconferencia

Actualmente el área relacionada con la videoconferencia está en plena transformación y estandarización. Diferentes esfuerzos y trabajos convergen hacia una compatibilidad de las soluciones, sin importar cuál es el tipo de red y su velocidad. La UIT (Unión Internacional de Telecomunicaciones) ha propuesto varios estándares. Desde su nacimiento, la videoconferencia ha sido definida por un estándar, el H.320. Sin embargo, en la actualidad han sido creados otros estándares para la transmisión de videoconferencia: H.321, H.322, H.323, H.324 y H.310.

- El estándar H.320 define una técnica para el transporte de videoconferencia sobre RDSI (Red Digital de Servicios Integrados) o ISDN, ofreciendo una calidad apropiada para comunicaciones de negocios. Sin embargo, los nuevos estándares implementan la transmisión de videoconferencia en diferentes niveles de calidad:
 - H.321 - Videoconferencia sobre ATM: Buena calidad para comunicaciones relacionadas con negocios.
 - H.322 - Videoconferencia sobre redes locales con calidad de servicio garantizada.
 - H.323 - Videoconferencia sobre IP/Ethernet (redes de calidad de servicio no garantizada).
 - H.324 - Videoconferencia sobre POTS (Plain Old Telephone Systems), que ofrece una baja calidad.
 - H.310 - Videoconferencia sobre ATM, utilizando MPEG-2: Ofrece la mayor calidad; es utilizada especialmente en aplicaciones médicas.

La transmisión de datos en videoconferencia se basa en los elementos siguientes:

Protocolo de Video.

Es un conjunto de códigos para asegurar la interoperabilidad entre equipos de videoconferencia. Permite la intercomunicación entre sistemas de videoconferencia de diferentes proveedores. Especifica los protocolos de vídeo, audio y de control para tener todos los servicios disponibles de una videoconferencia y está basado en la codificación MCT (Motion Compensation Transform). Hay tres clases de implementaciones de este estándar de datos multimedia definidos ITU (Unión Internacional de Telecomunicaciones):

Class 1: usa los requerimientos mínimos para ser compatible con H.320. La clase 1 soporta H.261 para compresión de vídeo, con resolución QCIF (Quarter Common Intermediate Format), 7.5 frames por segundo, codificación Motion compensation, y G.711 para protocolos de audio.

Class 2: usa los requerimientos mínimos para ser H.320 pero aparte provee de la implementación de una serie de funcionalidades añadidas. Esta clase provee todo lo visto en la clase 1, pero además soporta un grupo de las funcionalidades de la clase tres que describimos a continuación.

Class 3: soporta lo mínimo para ser un estándar H.320 pero además reúne todas las funcionalidades extras. Un sistema de clase tres tiene H.261 para compresión de vídeo, resolución CIF (or FCIF - Full Common Intermediate Format) ; 7.5, 10, 15, or 30 frames por segundo; pre- or post-procesado; Motion Compensation decodificador y codificador; y G.711, G.722, y G.728 como protocolos de audio.

Protocolos de audio

El Protocolo de audio G.711 usa el sonido de entre 48 y 64 Kbps; da una calidad de audio de teléfono y es el único protocolo de momento para que un sistema sea H.320 compatible. El protocolo de audio G.722 también utiliza audio entre 48 y 64 Kbps y provee una calidad estéreo, utilizada en sistemas de vídeo de clase 2. El Protocolo G.728 usa solamente 16Kbps, muy útil para videoconferencia con un ancho de banda por debajo de 256Kbps y deja mucho más ancho de banda para el vídeo.

CIF (Common Interface Format o Formato de Interfaz común)

CIF (Common Interface Format o Formato de Interfaz común –conocido también como FCIF - Full Common Interface Format) obtiene una resolución de 352x288.

QCIF (Quarter Common Interface) da un cuarto de la resolución completa CIF de 176x144. Esta resolución se suele ver bloqueada (pixelada) y suele tener numerosos patrones. QCIF es soportada por todos los sistemas de clase 1. CIF suele estar implementada en muchos sistemas de clase 2 y también es soportada por todos los sistemas de clase 3.

Frecuencia de Frames y Motion Compensation

Frecuencia de Frames (medida en frames por segundo o fps) tienen el mayor impacto sobre la “suavidad” del vídeo. Una alta frecuencia hace que el vídeo sea más fluido y con más saltos. Los sistemas de clase 1 deben soportar al menos 7.5 frames por segundo. Las frecuencias de vídeo del estándar H.320 son 7.5, 10, 15, y 30 fps. Por otra parte, el Motion Compensation reduce básicamente los saltos del vídeo que se producen codificando cada frame. Esto reduce el ancho de banda utilizado por algunos frames. Todos los sistemas H.320 tienen la posibilidad de utilizar el decodificador motion compensation. La clase 1 sólo soporta decodificación, los sistemas de clase 2 soportan alguna decodificación y los de clase 3 soportan mejoras en la decodificación de este sistema.

H.323 – Videoconferencia sobre redes TCP/IP

Hace poco tiempo se han concluido los trabajos relacionados con un nuevo estándar, el H.323. Este nuevo estándar fue diseñado para establecer videoconferencia sobre redes basadas en arquitecturas como Ethernet, Token Ring, FDDI, etc., utilizando los protocolos TCP/IP. H.323 no tiene las características que poseen los estándares H.320 y H.321, que fueron diseñados para aprovechar las ventajas de RDSI y ATM, para proporcionar una videoconferencia de alta calidad. El estándar H.323 es independiente del transporte, permitiendo la implementación de cualquier arquitectura de transporte, como por ejemplo ATM.

Los estándares para transmisión de videoconferencia sobre redes IP/Ethernet comienzan a ser una realidad. La diferencia básica con los anteriores es que esta videoconferencia, basada en este tipo de redes, no posee en su arquitectura una capa dedicada a la calidad del servicio, en la cual basar el transporte del vídeo. Como resultado de esta implementación se obtiene una videoconferencia con desfases entre

voz y audio y con baja calidad. Esta videoconferencia no puede ser considerada para aplicaciones de negocios serias.

El transporte de vídeo sobre redes Ethernet también tiene el desafortunado efecto de permitir la interacción entre el tráfico de datos y vídeo. Esto hace que el ancho de banda disponible para el tráfico de datos se vea disminuido por el tráfico de vídeo.

En este sentido, este tipo de videoconferencia podría utilizarse, por ejemplo, para establecer discusiones entre los individuos participantes en un proyecto; sin embargo, para establecer videoconferencia con alta calidad y con características multipunto es necesario utilizar ATM o RDSI.

Debido a la carencia de calidad de servicio en estas arquitecturas Ethernet, los diseñadores de los sistemas de transporte han propuesto un nuevo protocolo, RSVP. Resource ReSerVation Protocol (RSVP), actúa sobre la red para canalizar su comportamiento y hacerlo compatible con las necesidades del transporte en tiempo real.

RSVP se integra en una evolución hacia una nueva arquitectura, que pretende asegurar las comunicaciones multipunto en tiempo real conservando la filosofía del mejor esfuerzo (best effort) y la arquitectura IP. Esta evolución prevé los siguientes puntos:

- Establecer y mantener un camino único para un flujo de datos gracias a los protocolos de encaminamiento multipunto. Este mantenimiento del camino es indispensable para el funcionamiento de RSVP.
- Establecer un módulo de control que gestione los recursos de la red.
- Instaurar un sistema de ordenación de paquetes en la cola de espera para satisfacer la calidad de servicio solicitada.

En general, RSVP es un protocolo de control que permitirá obtener el nivel de calidad de servicio optimizado para un flujo de datos.

H.320 – Videoconferencia sobre RDSI

El estándar H.320, que define la implementación de videoconferencia sobre RDSI, ha estado vigente durante una década y hoy día es muy común implementarla sobre RDSI. Esto es debido a que RDSI permite la transmisión de videoconferencia en diversos niveles de calidad. RDSI es capaz de proveer una elevada calidad de transmisión de videoconferencia, principalmente por su carácter síncrono que permite

el transporte de vídeo con una baja tasa de retardo. Las características de transporte de RDSI permiten proveer a la videoconferencia de la sensibilidad que ésta demanda. Además, es capaz de implementarla en una gran variedad de velocidades de transmisión: desde 64 kbps hasta 2 Mbps. Hasta 128 kbps la videoconferencia es considerada de baja calidad, no siendo apropiada para aplicaciones de negocios. Sin embargo, a velocidades iguales o superiores a 384 kbps, RDSI provee una muy buena calidad de transmisión, ideal para aplicaciones de negocios.

La velocidad de transmisión de la videoconferencia está directamente relacionada con las aplicaciones que se le dan a esta:

- 64 kbps: Generalmente para aplicaciones recreacionales, donde la baja resolución y los desfases entre el audio y el vídeo son aceptables.
- 128 kbps: Utilizada en conferencias dentro de empresas y organizaciones (cortas distancias).
- 384 kbps: Calidad para aplicaciones de negocios. El audio y el vídeo están sincronizados y los movimientos son uniformes.
- 512 kbps: Alta calidad para aplicaciones de negocios. Alta resolución y movimientos muy uniformes; el desfase entre audio y vídeo es prácticamente indetectable.
- 768 kbps ó más: Excelente calidad de transmisión de videoconferencia. Ideal para aprendizaje a distancia, aplicaciones médicas, etc.

RDSI permite obtener una buena calidad en la transmisión de videoconferencia a velocidades iguales o superiores a 384 kbps. Sin embargo, es muy costoso y presenta ciertas complejidades. Por ejemplo, es necesario implementar tres interfaces de 128 kbps y llevarlas a cada uno de los dispositivos de videoconferencia. Estas líneas deben entonces conectarse formando un solo canal a través de un multiplexor (MUX).

ATM puede utilizarse para implementar un sistema puramente para propósitos de videoconferencia, tal como RDSI; con la ventaja de que esta implementación sobre ATM utiliza el cableado existente que está típicamente presente en las arquitecturas de redes actuales.

H.321 – Videoconferencia sobre ATM

Para implementar las características del estándar H.320 en cuanto a calidad de transmisión, con un costo y una complejidad menores, el estándar H.320 ha sido adaptado y ha surgido el estándar H.321. El estándar H.321 describe los métodos para

implementar videoconferencia sobre ATM con ventajas sobre el modelo RDSI, y es totalmente compatible con el estándar H.320.

El estándar H.321 basado en ATM implementa la videoconferencia en el mismo estilo que RDSI, con los mismos incrementos en velocidad de transmisión (128 kbps, 384 kbps, 768 kbps, etc.). La diferencia fundamental es que la videoconferencia sobre ATM es más fácil y más barata de implementar. ATM logra esto debido a aspectos como los siguientes:

- Las tarjetas ATM tienen un coste inferior.
- Se utiliza una pasarela RDSI-ATM como punto de acceso centralizado para la red WAN RDSI. Esta metodología permite el acceso fuera de la red y sirve también de centro de multiplexión sustituyendo los multiplexores para cada estación utilizados en la implementación RDSI. Esto proporciona un ahorro importante.
- Se utilizan switches ATM en lugar de RDSI, disminuyendo costos en la implementación.
- La topología ATM no necesita de múltiple cableado como ocurre con la implementación RDSI, que requiere de tres cables UTP individuales.

La implementación de ATM no sólo proporciona beneficios en cuanto a la disminución de costos para implementar la transmisión de videoconferencia, sino que provee las bases de una arquitectura de red que puede utilizarse para el transporte de voz y datos en adición a la videoconferencia. Esta capacidad está haciendo de ATM la elección tecnológica en un amplio espectro de aplicaciones.

H.324 – Videoconferencia sobre POTS

El estándar H.324 para transmisión de videoconferencia define una metodología para su transporte a través de la red telefónica ó lo que se conoce como POTS (Plain Old Telephone Systems). Específicamente el estándar H.324 describe terminales para comunicaciones multimedia trabajando a bajas velocidades, utilizando módems V.34. Estos terminales pueden transmitir voz, datos y vídeo en cualquier combinación en tiempo real.

El estándar H.324 está diseñado para optimizar la calidad de la transmisión de videoconferencia sobre los enlaces de baja velocidad asociados con los POTS, típicamente estas velocidades están en el rango de 28.8 kbps a 56 kbps. Estas bajas velocidades de transmisión sumadas a la naturaleza impredecible del medio de

transmisión, restringe este tipo de videoconferencia a unos pocos cuadros por segundo.

Sin embargo, se espera que el estándar H.324 tenga cierta aceptación entre el mercado de consumidores. Primero, porque este tipo de videoconferencia está orientada a aplicaciones recreacionales donde no se requiere de una elevada calidad y en segundo lugar debido a la facilidad de implementación donde sólo se requiere de un PC equipado con un módem y utilizar la red telefónica convencional (POTS).

H.310 – Videoconferencia sobre ATM – MPEG2

Mientras los estándares H.320 y H.321 pueden proporcionar una elevada calidad de videoconferencia, especialmente cuando se utilizan elevadas velocidades de transmisión (768 kbps ó mas), el estándar H.310 define una metodología para implementar videoconferencia basada en MPEG-2 (estándar del ISO) sobre ATM a velocidades que van entre 8 y 16 Mbps. La videoconferencia basada en el estándar H.310 provee una elevadísima calidad en la transmisión de audio y vídeo, estando este tipo de videoconferencia orientada a aplicaciones como la transmisión de procedimientos quirúrgicos en vivo, donde el grupo de médicos asesores están ubicados a grandes distancias. Estas elevadas velocidades de transmisión ofrecidas por este estándar permiten el establecimiento de una videoconferencia con elevada interactividad entre los participantes. Aplicaciones como el establecimiento de procesos educativos, donde existen expertos situados a distancia y donde el nivel de calidad de la videoconferencia debe ser máximo requieren del uso de este estándar.

RTP

RTP (Real-time Transport Protocol) define un formato de paquete estandarizado para la distribución de audio y vídeo por Internet. Fue desarrollado por el grupo de trabajo de transporte de audio y vídeo del IETF y fue publicado por primera vez en 1996 como RFC 1889, que sería reemplazado posteriormente en 2003 por la RFC 3550. RTP puede ser usado también en combinación con RSVP, mejorando el campo de las aplicaciones multimedia.

RTP no tiene un puerto TCP o UDP estandarizado por donde comunicarse. El único estandar que obedece es que las comunicaciones por UDP se hacen por un puerto par y el siguiente puerto impar más alto se usa para el protocolo de control de RTP (RTCP). Aunque no está estandarizado, se suele usar la pareja de puertos 16384-32767. RTP puede transportar cualquier dato con características de tiempo real, como audio y vídeo interactivo. El establecimiento de la comunicación así como su cierre

normalmente se lleva a cabo por el protocolo SIP. El hecho de que RTP use un rango dinámico de puertos dificulta su paso a través de los cortafuegos. Para solucionar este problema, a menudo se hace necesario utilizar un servidor STUN.

RTP fue diseñado originalmente como un protocolo multicast, pero desde entonces ha sido aplicado en muchas aplicaciones unicast. Se usa frecuentemente en sistemas de transmisión multimedia (en combinación con RTSP) así como en videoconferencia (en combinación con H.323 o SIP), constituyendo el fundamento técnico de la industria de voz sobre IP. RTP va acompañado por RTCP y está ubicado encima de UDP. Las aplicaciones que usan RTP son menos sensibles a la pérdida de paquetes pero son típicamente sensibles a los retrasos, por este motivo, trabajar sobre UDP es una opción más adecuada que TCP en estas aplicaciones.

De acuerdo con la norma RFC 1889, entre los servicios ofrecidos por RTP se incluyen:

- Identificación del tipo de contenido. Indica qué tipo de tráfico transporta.
- Número de secuencia. Permite establecer una numeración de las unidades de datos del protocolo.
- Marcas de tiempo. Permiten la sincronización y el cálculo del jitter.
- Monitorización de la distribución.

Los protocolos por sí mismos no proporcionan mecanismos que aseguren una entrega a tiempo. Tampoco ofrecen ninguna garantía de calidad del servicio (QoS). Este tipo de cosas deben ser resueltas por otros mecanismos. Debido a esto se pueden producir situaciones de entrega de paquetes fuera de contexto y el control de flujo y congestión no están soportados directamente. Sin embargo, los protocolos distribuyen los datos necesarios a la aplicación para asegurarse de que pueden reordenar los paquetes de manera correcta. Además, RTCP proporciona información acerca de la calidad de la recepción, que puede ser usada por la aplicación para hacer ajustes locales. Por ejemplo, si se está formando congestión, la aplicación podría decidir bajar el ancho de banda de transmisión.

RTP también fue publicado por la ITU-T como H.225.0, pero fue retirado más tarde una vez que el IETF tuvo una RFC estable publicada. Existe como un estándar de Internet (STD 64) definido en la RFC 3550 (que reemplaza la RFC 1889). La RFC 3551 (STD 65) define un perfil específico para conferencias de audio y vídeo con mínimo control. La RFC 3711 define SRTP (Secure Real-time Transport Protocol), que es realmente una extensión al perfil de RTP para conferencias de audio y vídeo, y

puede ser usado opcionalmente para proporcionar confidencialidad y autenticación de mensajes.

La posición de RTP en la torre de protocolos puede resultar algo extraña. Se decidió poner RTP en el espacio de usuario usando a su vez los servicios de UDP. El funcionamiento se describe a continuación:

La aplicación multimedia consiste en múltiples flujos de audio, vídeo, texto, etc. Estos flujos son introducidos en las funciones de librería de RTP, que está en el espacio de usuario, junto con la aplicación. La librería multiplexa los streams y los introduce en paquetes RTP, que a su vez son enviados a través de un socket. Al final de los sockets (en el núcleo del sistema operativo), los paquetes UDP son generados e introducidos a su vez en paquetes IP. Llegados a este punto y dependiendo de las capas física y de enlace estos datagramas IP se fragmentarán a su vez en unidades de datos del protocolo de enlace para ser transmitidos.

Como consecuencia de este diseño, pueden presentarse dudas a la hora de clasificar RTP dentro de la torre de protocolos. Como se ejecuta en el espacio de usuario y es enlazado a la aplicación, puede parecer un protocolo del nivel de aplicación. Por otra parte, es un protocolo genérico e independiente de la aplicación que sólo proporciona facilidades de transporte, por eso puede clasificarse dentro del nivel de transporte, y es ahí donde normalmente se clasifica a RTP.

A continuación se muestra la estructura de un paquete RTP:

Offset/Bits	0-1	2	3	4-7	8	9-15	16-31
0	Version	P	X	CC	M	PT	Nº Secuencia
32	Marca de tiempo						
64	Identificador SSRC						
96	Identificadores CSRC						
96 + CC x 32	Cabecera adicional. (Indica la longitud AHL)						
96 + CC x 32 + (X x (AHL + 16))	Datos						

Tabla 7: Estructura de un paquete RTP

Los bits “Version” indican la versión del protocolo. La versión actual es la 2. El bit P es utilizado para indicar si hay bytes extra de relleno al final del paquete RTP. El bit

X indica si las extensiones al protocolo están siendo usadas en el paquete. Los bits CC contienen el número de identificadores CSRC que siguen a la cabecera fija. El bit M es usado a nivel de aplicación y está definido por el perfil. Si está activado significa que los datos contenidos en el paquete tienen especial relevancia para la aplicación. El bit PT indica el formato de la carga de datos y determina su interpretación por parte de la aplicación . SSRC indica la fuente de sincronización.

El conjunto RTP + RTCP se usa habitualmente. RTP es usado para transmitir los datos (audio y vídeo) mientras que RTCP es usado para monitorizar los parámetros de calidad de servicio (QoS). La monitorización de la calidad de servicio es muy importante en aplicaciones modernas. En aplicaciones de gran escala como IPTV hay un retraso inaceptable entre informes de RTCP, lo que puede derivar en problemas de calidad de servicio.

Diseño del sistema

Especificaciones iniciales

Una vez estudiado el estado del arte de los sistemas de visión artificial aplicados al tráfico se procedió a elaborar una serie de especificaciones que debería cumplir el sistema a implementar. La lista de especificaciones incluye los siguientes requisitos de cara al diseño hardware:

- **Formato de tarjeta 3U.** Se decide escoger este formato estándar para dotar de una mayor facilidad de instalación al sistema.
- **Alimentación.** La alimentación debe ser a 5V DC.
- **Entradas/Salidas.** El sistema debe contar al menos con 8 entradas y 8 salidas de propósito general.
- **Conexión Ethernet.** Se hace especial hincapié en la necesidad de un puerto Ethernet para el control y operación del sistema.
- **Vídeo de entrada.** Los canales de vídeo de entrada deben aceptar tanto señales analógicas de vídeo compuesto como señales de vídeo digital sin comprimir del tipo BT.656 [12]. Dependiendo de la aplicación final pueden necesitarse un número u otro de entradas de vídeo y en determinados formatos (analógico y digital).
- **Vídeo de salida.** En determinadas aplicaciones puede ser necesaria una salida de vídeo analógico (vídeo compuesto) para transmitir imágenes con datos sobreimpresos.
- **Puertos serie.** Se requiere que el sistema tenga dos puertos serie y que puedan funcionar tanto en modo RS-232 como en modo RS-485.
- **Memorias.** El sistema debe incluir suficiente memoria RAM para poder ejecutar los algoritmos de visión artificial. El sistema también debe incluir memoria no volátil para almacenar las aplicaciones.
- **Soportes de almacenamiento masivo.** El sistema debe incorporar algún tipo de soporte de almacenamiento masivo, bien sea por USB, o bien por tarjeta de

memoria extraíble.

- **Compresión MPEG-4.** El sistema debe poder comprimir vídeo en formato MPEG-4.

Sistema desarrollado

En virtud de las especificaciones mencionadas anteriormente se desarrolló el prototipo de sistema de visión artificial EPI-ARM 1.0

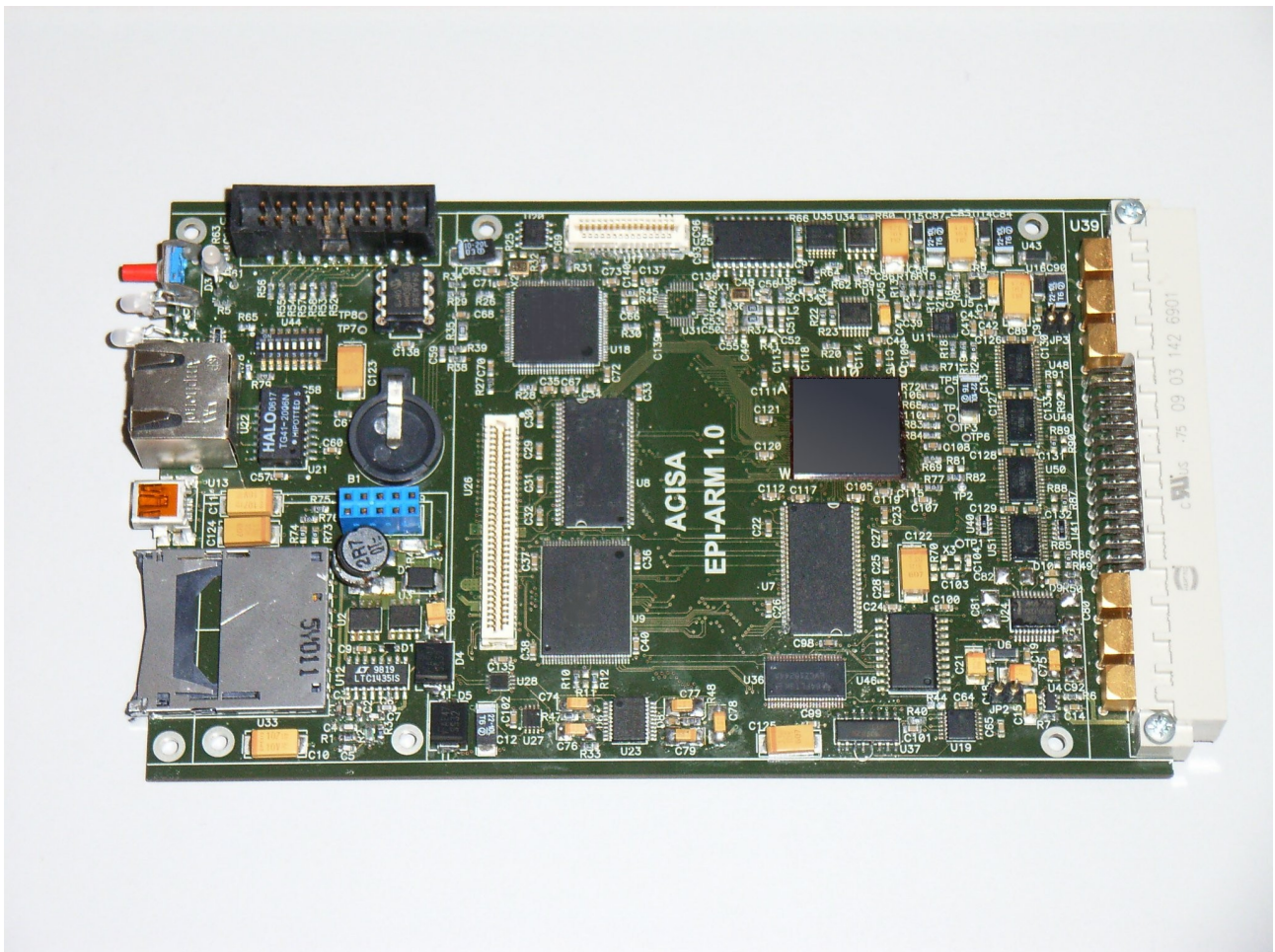


Figura 18: Prototipo EPI-ARM 1.0

El sistema se articula en torno a un núcleo ARM y su estructura de bloques responde a la figura 19:

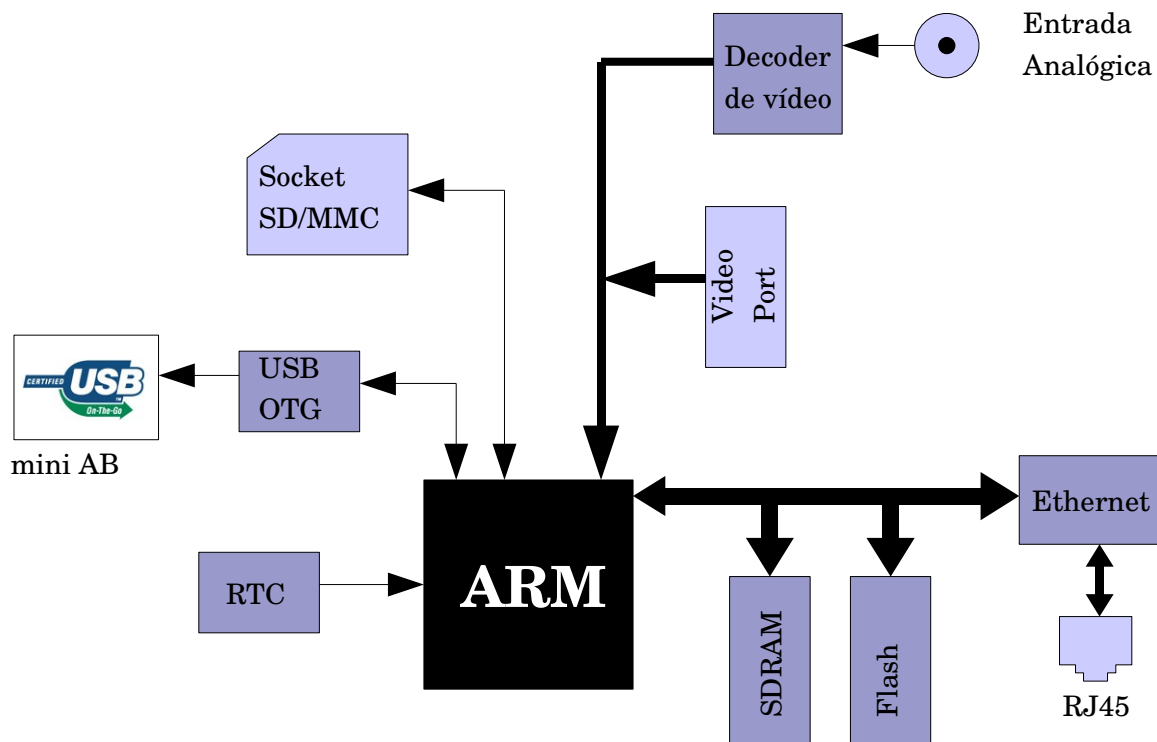


Figura 19: Diagrama de bloques del EPI-ARM 1.0

Tecnología empleada

La solución tecnológica empleada está basada en el uso de un procesador ARM, que se engloba dentro de los procesadores RISC. La elección de un procesador RISC como soporte hardware para las aplicaciones de visión artificial aplicadas al tráfico obedece a un conjunto de razones:

- Los procesadores RISC y, en concreto, el ARM permiten la instalación de sistemas operativos de amplia difusión. Esto ha permitido la utilización de GNU/Linux, lo que supone en la práctica que se le dota al sistema de importantes funcionalidades como por ejemplo la torre de protocolos TCP/IP, los módulos básicos de manejo de dispositivos USB, la gestión de la multitarea, etc. Además el hecho de tener un sistema operativo facilita el desarrollo y depuración de las aplicaciones de control.
 - Existe bastante soporte para este tipo de procesadores.
 - El procesador elegido integra, entre otros muchos periféricos de interés, un encoder MPEG-4 hardware, lo que evita tener que añadir chips específicos o tener que desarrollar o comprar librerías de compresión de vídeo. Además, el hecho de que el compresor sea un componente hardware ayuda a aliviar la carga

computacional sobre el procesador.

- El precio del procesador elegido es bastante competitivo.

Aunque el procesador ARM tiene suficiente capacidad para las primeras aplicaciones desarrolladas, la solución original incluye la posibilidad de combinar el procesador ARM con un DSP de alta gama. Esta arquitectura híbrida es muy adecuada en aplicaciones que requieren una gran capacidad de procesamiento o que tienen que procesar varios canales en paralelo. En cualquier caso, tanto los procesadores RISC como los DSPs se encuentran dentro del conjunto de soluciones tecnológicas con mayor flexibilidad. Si bien es cierto que otras soluciones como los ASSPs o los Media Processors pueden mostrarse más eficientes en algunos casos, en estas aplicaciones el requisito de la flexibilidad era mucho más importante. Hay que tener en cuenta que lo que se pretendía diseñar es una plataforma básica a partir de la cual irán surgiendo diferentes productos con finalidades distintas dentro del ámbito de las aplicaciones de tráfico, lo que justifica la elección de una tecnología flexible y fácilmente reprogramable.

Formato de la tarjeta

La elección del formato 3U (160 x 100mm) para la tarjeta no es casual. El hecho de estar estandarizadas permite que las tarjetas 3U puedan ser insertadas en racks con lo que se puede concentrar la instalación de varios equipos en uno o varios racks. Esta solución es especialmente adecuada para centros de control donde las señales de vídeo a procesar están accesibles a través de una matriz. Así mismo, el reducido tamaño de la tarjeta también permite su instalación compacta dentro de una carcasa convencional para cámaras. Como resultado, el producto final se puede adaptar a diferentes entornos sin tener que modificar el PCB.

Debido a la complejidad del diseño, la tarjeta está fabricada a 6 caras en clase 6 (anchura mínima de pistas: 120 micras) y hace uso de vías ciegas para permitir una mayor densidad de pistas. El encapsulado de los componentes utilizados (BGA y SMD, principalmente), también ha obligado a emplear técnicas modernas de soldadura como el reflujo por convección.

Alimentación

La alimentación de la tarjeta es a 5V DC tal y como figura en las especificaciones, y su consumo es inferior a 2,5W.

Memorias

Se han escogido chips de memoria SDRAM por ser el tipo de memoria que tiene menores tiempos de acceso de todos los tipos de memoria que el procesador podía soportar. Para el almacenamiento de los programas y el sistema operativo se ha escogido una memoria Flash de tipo NOR. En principio también se barajaron otras opciones como memorias Flash NAND o memorias DiskOnChip pero finalmente se prefirió utilizar memorias NOR por su mayor facilidad de integración y velocidad en los accesos en lectura. La siguiente tabla muestra una comparativa entre los distintos tipos de memoria no volátil estudiados:

	NOR	DiskOnChip	NAND
Fiabilidad	Alta	Alta	Baja
Coste	Alto	Medio	Bajo
Densidad de memoria	Baja	Alta	Alta
Rendimiento en escritura	Muy Bajo	Alto	Medio
Rendimiento en lectura	Muy Alto	Alto	Medio
Facilidad de integración	Muy Alta	Alta	Baja
Capacidad de arranque	Sí	Sí	En algunos casos
XIP	Sí	No	No

Tabla 8: Comparativa entre memorias flash

Conexión de red

Para dotar al sistema de conectividad Ethernet, se incluyó un controlador Ethernet de tipo 10/100BaseT, con su correspondiente conector RJ-45. Como ya se apuntaba en las especificaciones, la conectividad Ethernet es muy importante porque permite aumentar la funcionalidad del equipo y facilita la operación con el sistema durante la fase de desarrollo. Durante el funcionamiento del equipo la conexión Ethernet puede ser utilizada con dos objetivos principalmente:

Transmisión de la información generada. Mediante la conexión Ethernet se pueden enviar los datos recogidos a un centro de control. Estos datos pueden tener formatos muy diversos, desde datos numéricos hasta imágenes y streams de vídeo. En el caso de aplicaciones destinadas a recoger parámetros de tráfico los datos numéricos

y alfanuméricos pueden ser transmitidos a un centro de control teniendo información en tiempo real de los diferentes puntos monitorizados. Para el caso en que la información a transmitir sean imágenes o vídeo comprimido la conexión Ethernet también es de gran utilidad. Con ayuda de protocolos como RTP se puede transmitir vídeo generado en tiempo real que puede ser visualizado en tiempo real en un centro de control. El hecho de transmitir vídeo por Internet permite aprovechar infraestructuras existentes y, por consiguiente, disminuir costes frente a las instalaciones convencionales basadas en redes específicas para la transmisión de vídeo analógico. Además, la tecnología RTP permite concatenar flujos de vídeo y datos sobre la misma infraestructura.

Control de equipos. Mediante la conexión Ethernet se puede controlar el equipo de forma remota, de forma que si existe conexión a una red, se pueden evitar desplazamientos innecesarios de personal. En estos casos, el control de los equipos se puede realizar desde un centro de control. El sistema ha sido dotado de un servidor web y un servidor SSH para este tipo de operaciones. Usando un navegador convencional se puede acceder al equipo, consultar sus registros y configurar su funcionamiento. En el caso concreto de los primeros equipos EPI-ARM, destinados al contaje de vehículos, es posible definir las regiones de detección desde la web.

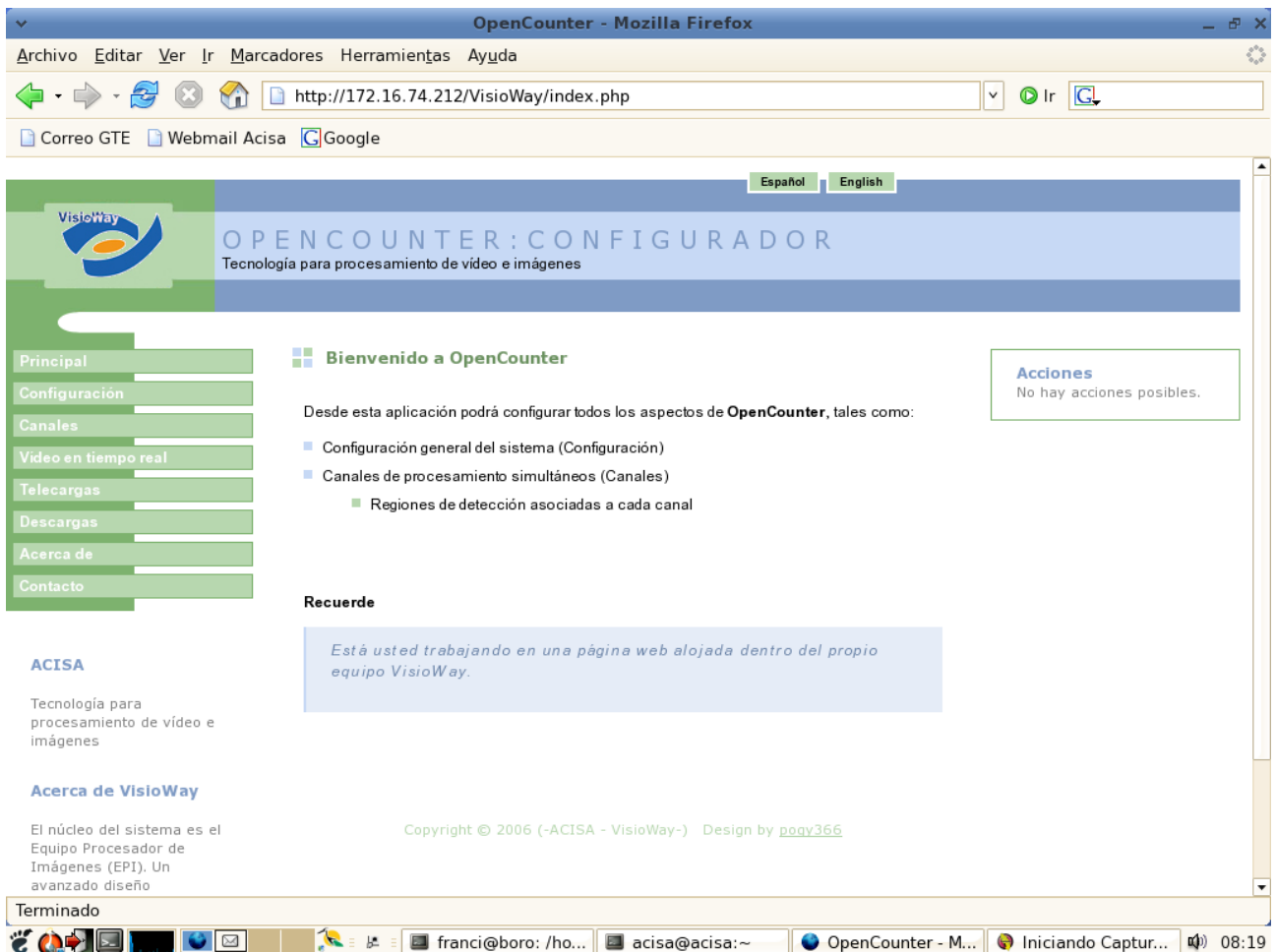
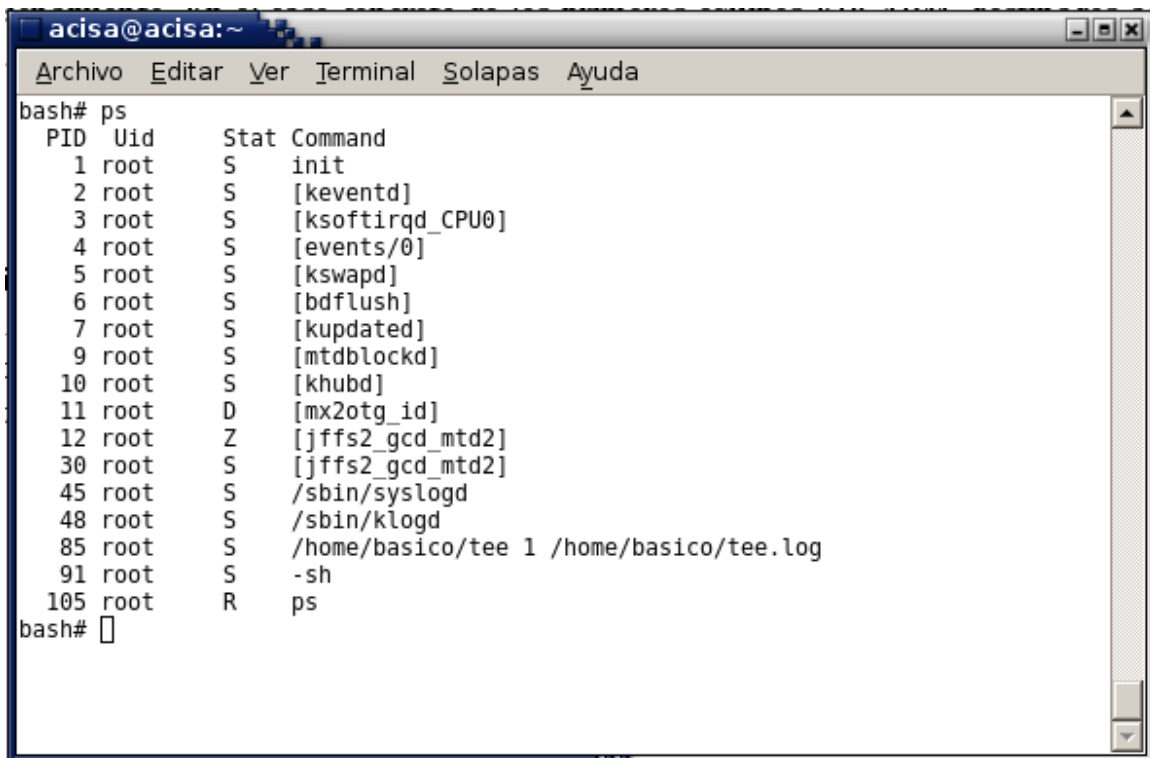


Figura 20: Acceso via web

El servidor SSH permite un acceso controlado a la consola del sistema pudiendo realizar operaciones de control a más bajo nivel. Esta conexión normalmente es utilizada para depuración o para la actualización de las versiones del software. Las sesiones están protegidas por contraseña y por el propio mecanismo de privilegios de Linux, lo que ayuda a evitar accesos no autorizados.



```
acisa@acisa:~
Archivo  Editar  Ver    Terminal  Solapas  Ayuda
bash# ps
  PID  Uid    Stat Command
   1  root    S     init
   2  root    S     [keventd]
   3  root    S     [ksoftirqd_CPU0]
   4  root    S     [events/0]
   5  root    S     [kswapd]
   6  root    S     [bdflush]
   7  root    S     [kupdated]
   9  root    S     [mtdblockd]
  10  root    S     [khubd]
  11  root    D     [mx2otg_id]
  12  root    Z     [jffs2_gcd_mtd2]
  30  root    S     [jffs2_gcd_mtd2]
  45  root    S     /sbin/syslogd
  48  root    S     /sbin/klogd
  85  root    S     /home/basico/tee 1 /home/basico/tee.log
  91  root    S     -sh
 105  root    R     ps
bash#
```

Figura 21: Acceso por SSH

Entradas / Salidas de vídeo

La tarjeta EPI-ARM está diseñada de forma que la entrada de vídeo puede provenir de una fuente analógica o digital. La tarjeta admite entradas analógicas con formato PAL en vídeo compuesto. En el caso de que la entrada sea digital, se admite, preferentemente señales con el formato expuesto en la norma ITU-R BT.656 [12], o en su defecto, señales digitales con interfaz paralela de 8 bits con sus correspondientes señales de reloj y sincronismos vertical y horizontal. El diseño de la etapa de entrada de vídeo se ha realizado permitir su adaptación a múltiples fuentes de vídeo y aumentar así la versatilidad del sistema, que puede ir equipado con una cámara propia o procesar imágenes provenientes de fuentes externas.

Existe también la posibilidad de expandir el número de entradas de vídeo mediante el uso de la tarjeta EPI-DSP, que, acoplada a la placa EPI-ARM, dota al sistema de una mayor capacidad y permite el procesamiento simultáneo de hasta cuatro señales de vídeo analógico.



Figura 22: Conectores coaxiales de vídeo en la tarjeta EPI-ARM



Figura 23: Tarjeta EPI-DSP

La salida de vídeo analógico sólo está disponible cuando el sistema funciona con la placa EPI-DSP acoplada. En este caso se puede obtener una salida de vídeo analógico (vídeo compuesto) en formato PAL o NTSC. Hay que resaltar que las imágenes de salida son tratadas por la tarjeta EPI-DSP, pudiendo ofrecer imágenes con sobreimpresión o incluso generar cualquier tipo de imagen.

Encoder MPEG-4

El sistema dispone de un compresor MPEG-4 Simple Visual Profile que puede ofrecer vídeo digital comprimido en tamaño CIF. El vídeo resultante de la operación del compresor puede ser almacenado o transmitido por Internet en tiempo real. Esto puede ser especialmente útil para aquellos casos en que se requiera monitorizar el tráfico de forma visual y el sistema instalado está lejos del centro de control. El almacenamiento de vídeo, por su parte puede ser interesante para recoger secuencias de algún incidente o infracción cometida. El hecho de escoger la compresión MPEG-4 se debe a su buena relación calidad/ancho de banda.

Dispositivos de almacenamiento masivo

Con vistas al almacenamiento de imágenes, fragmentos de vídeo y demás información de tráfico se ha dotado al sistema de interconexión a dispositivos extraíbles de almacenamiento masivo. Concretamente, el sistema dispone de un slot para tarjetas SD/MMC donde puede almacenar la información antes citada. Aunque en principio se barajaron también otros soportes de memoria como por ejemplo tarjetas

Compact Flash o incluso discos duros en miniatura finalmente se optó por usar tarjetas SD. Los motivos que llevaron a la elección de las tarjetas SD fueron:

- Amplia difusión en el mercado de las cámaras digitales para aficionados
- Menor coste que las tarjetas Compact Flash
- Menores dimensiones que las tarjetas Compact Flash
- No dispone de elementos mecánicos susceptibles de fallo como ocurre con los discos duros
- Mayor simplicidad de conexión que las tarjetas Compact Flash
- Aunque las tarjetas Compact Flash tienen mayor capacidad, las tarjetas Secure Digital cada vez avanzan más rápido y se prevé que pronto alcanzarán a las tarjetas Compact Flash en cuanto a capacidad.

De forma adicional se ha dotado al sistema de conexión por USB-OTG. OTG (On-The-Go) [13] es un suplemento al popular estándar USB que permite que el dispositivo pueda comportarse como maestro (host) o como esclavo (slave) según el tipo de dispositivo al que se conecte. El dispositivo externo se conecta al equipo a través de un conector USB MiniAB. Si se trata de un “host” (por ejemplo, un PC), el controlador USB del EPI-ARM se comportará como un esclavo, pero si se trata de un elemento esclavo (por ejemplo, un pen drive), el controlador se comportará como maestro. Esto permite utilizar la interfaz USB-OTG tanto para controlar el sistema desde un “host” como para manejar un dispositivo de memoria USB donde almacenar datos, imágenes, vídeos, etc.

Conectividad BlueTooth

El diseño de la tarjeta EPI-ARM prevé la interconexión de una placa BlueTooth específica a través de un canal serie. Esta placa adicional, fabricada por Acisa en colaboración con Cetecom se compone de un módulo BlueTooth controlable a través de un puerto serie con señales TTL. La tarjeta BlueTooth integra un conector coaxial para la antena, de forma que se le puede instalar una pequeña antena compacta como en el caso de la figura 24 o se le puede conectar un latiguillo de cable coaxial para poder situar la antena en otro lugar más favorable para la recepción de señales.

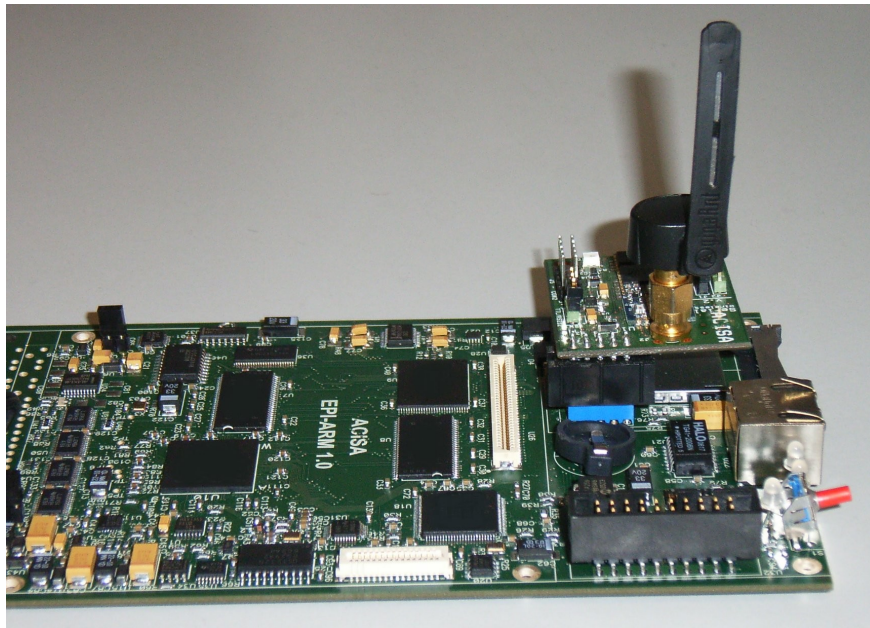


Figura 24: Detalle de la conexión entre la placa BlueTooth y la placa EPI-ARM

Entradas / Salidas digitales

La tarjeta EPI-ARM dispone de 28 líneas de E/S digitales de propósito general estructuradas en bancos de 8 señales, excepto el último que sólo posee 4. Las señales son configurables por hardware como entrada o salida por bloques enteros. Las señales están accesibles a través del conector de backpanel y funcionan como entradas/salidas TTL a 5V. Aunque las entradas/salidas de propósito general dan mucha versatilidad, en aplicaciones como el contaje de vehículos pueden ser utilizadas para generar pulsos cuando los vehículos pasan por las zonas de detección y emular, de este modo, a las clásicas espiras.

Puertos Serie

La tarjeta dispone de dos puertos serie asíncronos. El puerto serie 1 es utilizado para la consola principal del sistema. En este caso funciona como una interfaz RS-232 aunque prescinde de las señales de control de flujo. Mediante un cable especial puede conectarse a un PC desde el que se puede controlar la consola del sistema.

Ambos puertos serie pueden generar señales en formato RS-232 y RS-485, aunque el puerto serie 1 puede utilizar también niveles TTL para la comunicación con la placa BlueTooth.

Aunque en principio el puerto serie 2 no tiene asignada una función, se prevé que

sea usado para controlar la telemetría. Es habitual que los posicionadores y los mecanismos ópticos ajustables se controlen a través de un puerto serie RS-485. Lo más común es que se puedan controlar tres parámetros:

- Pan (Ángulo de giro en horizontal)
- Tilt (Inclinación)
- Zoom (Aumento)

Gracias al puerto serie 2 se pueden llevar a cabo las tareas de control de la telemetría pudiendo conseguir un sistema que varíe automáticamente la zona a enfocar dependiendo de las necesidades.

Instalación de GNU/Linux

En la actualidad existen multitud de sistemas empotrados que soportan el sistema operativo GNU/Linux. A diferencia del caso de los ordenadores personales (PC) las arquitecturas de los sistemas empotrados pueden variar mucho de un sistema a otro debido a su naturaleza mucho más específica que la de los PCs. Las diferencias no terminan en la arquitectura general del sistema, sino que también puede haber diferencias importantes dependiendo del microprocesador que utilicemos para desarrollar el sistema empotrado.

En vista de todas estas diferencias respecto a la arquitectura tradicional de los PCs, para la que surge la mayoría de los sistemas operativos, cabe preguntarse cómo se puede instalar un sistema operativo como Linux en un sistema empotrado específico. Evidentemente, la instalación en un sistema empotrado no es un proceso tan sencillo como puede ser instalar una distribución de Linux en nuestro ordenador. Afortunadamente hoy en día los fabricantes de los microprocesadores y otras empresas del sector desarrollan herramientas que facilitan el desarrollo de aplicaciones para Linux sobre sistemas empotrados.

En primer lugar se va a describir la secuencia de desarrollo e implantación que se suele aplicar a sistemas empotrados. La siguiente figura describe el proceso completo:

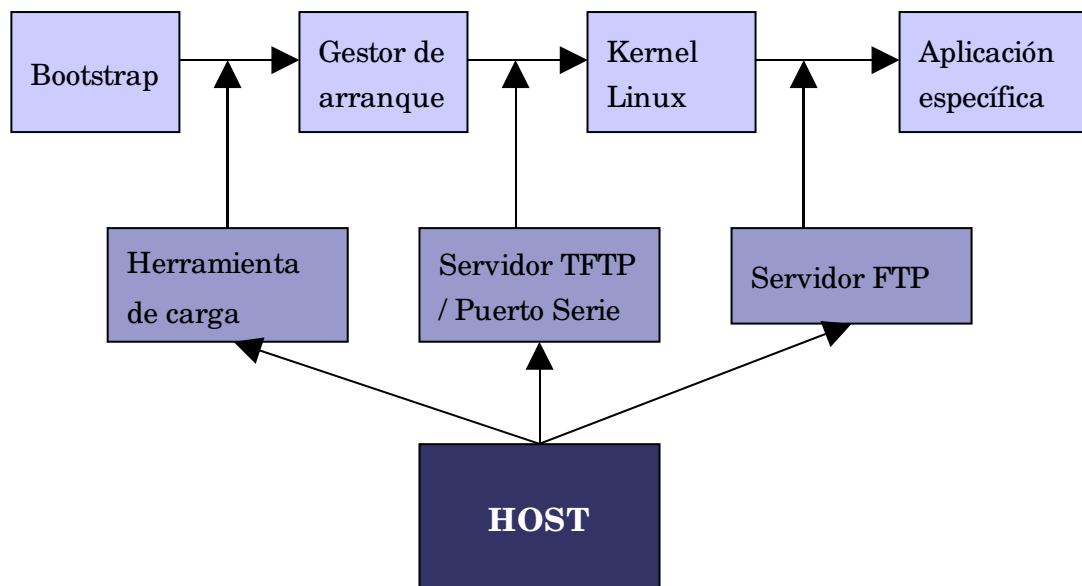


Figura 25: Secuencia de desarrollo e implantación

Debido a que un sistema empotrado normalmente no incorpora un teclado o una pantalla, tenemos que hacer uso de un PC (HOST) que nos permita visualizar el estado de los programas que se están ejecutando en el sistema y que nos permita interactuar con el mismo. De esta manera, el host está involucrado en todas las fases hasta la ejecución de la aplicación o aplicaciones específicas que dotan al sistema de la funcionalidad final para la que se diseñó.

Una vez que se ha realizado toda esta secuencia con éxito, el sistema empotrado estará preparado para funcionar de manera autónoma. La secuencia de programas ejecutados en el sistema varía ligeramente en esta situación puesto que al estar almacenados en una memoria flash o en disco, se puede empezar ejecutando directamente el gestor de arranque. Obviamente, ya no hará falta la intervención del host, tal y como se describe en la siguiente figura:

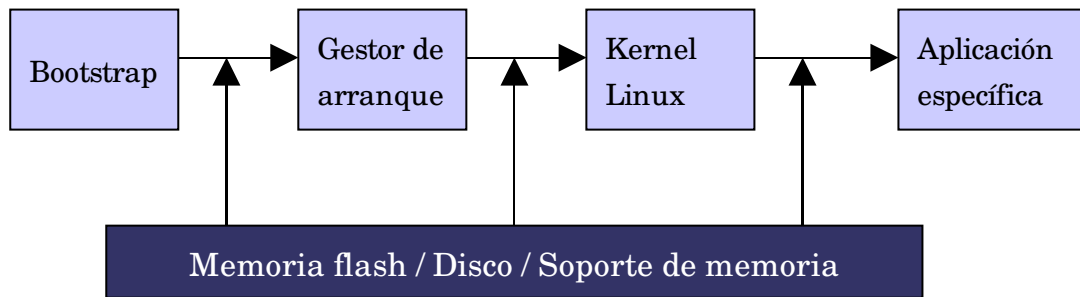


Figura 26: Funcionamiento autónomo

Fase 1: Arranque del microprocesador

En esta primera fase el sistema no tiene almacenado programa alguno en dispositivos de memoria no volátil (Flash, disco duro, etc.) por lo que no podemos dejar que arranque desde esos dispositivos. En su lugar utilizaremos el modo de arranque por bootstrap, mediante el cual el programa a ejecutar se le pasa a través de un canal externo que suele ser un puerto serie o un puerto USB. Se suelen usar estos dispositivos para arranque en modo bootstrap por la sencillez de uso de los mismos, lo que permite que el propio procesador pueda implementar el mecanismo de arranque bootstrap en forma de pequeño programa almacenado en una memoria ROM interna. La técnica del bootstrap está muy extendida en los sistemas empotrados y los propios fabricantes de microprocesadores facilitan el software que permite cargar programas desde un host puesto que estos programas son altamente dependientes del tipo de CPU que lleve nuestro sistema empotrado. Un ejemplo de programa de este tipo es HAB Tools.

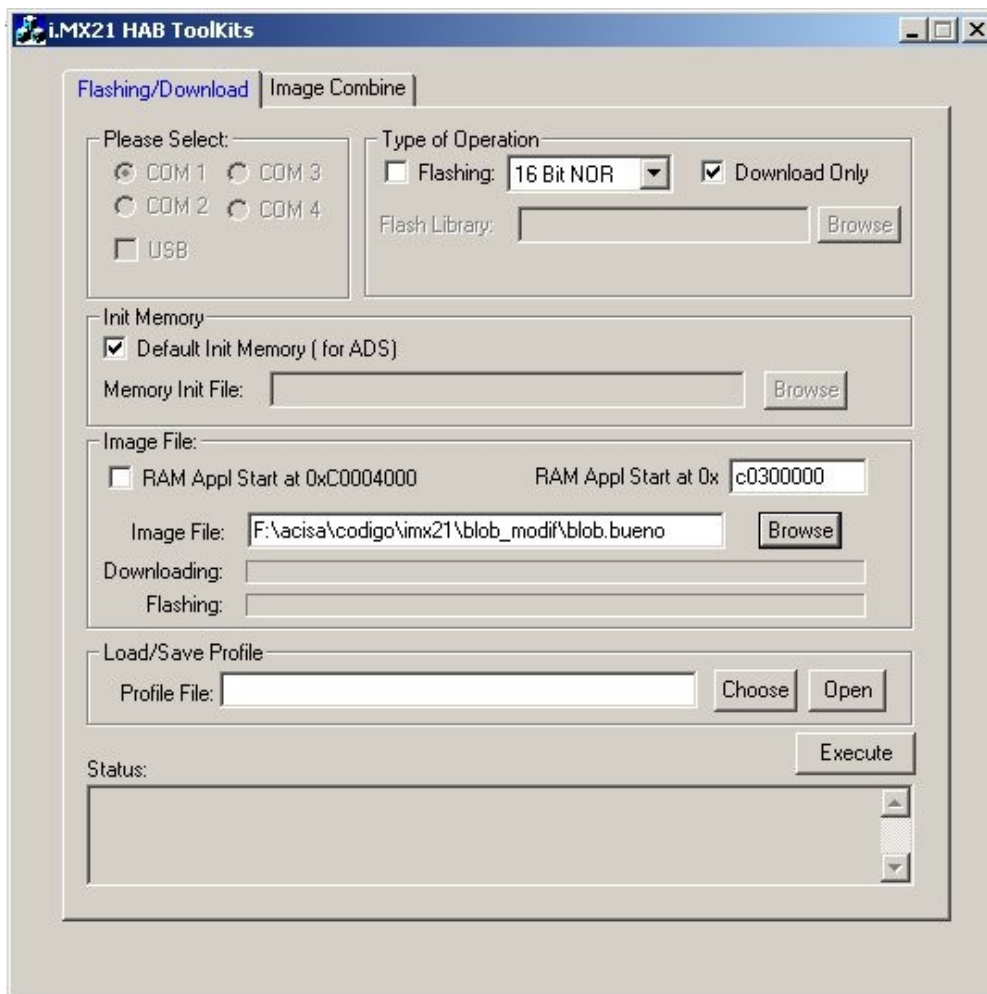


Figura 27: Interfaz de HAB Tools

Los programas cargados por bootstrap se almacenan en memoria RAM y una vez que termina la transferencia del mismo se le cede la ejecución. En este punto podemos cargar cualquier programa para su ejecución directa (sin sistema operativo). En el caso de querer instalar GNU/Linux, lo más normal es que el programa que se cargue por bootstrap sea el propio gestor de arranque, que nos ayudará a instalar el kernel y el sistema de ficheros.

Fase 2: El gestor de arranque

Un gestor de arranque es un programa que normalmente se ejecuta al inicio y cuya misión fundamental es preparar el sistema habilitando las funciones básicas para lanzar la ejecución de un sistema operativo. Además de esto, el gestor de arranque normalmente implementa una interfaz bastante simple que permite al usuario realizar algunas funciones de configuración y control. Existen varios gestores de arranque para Linux, como GRUB, LILO, BLOB, etc.

Para poder operar con el gestor de arranque es necesario disponer de una consola. En el caso de un PC el problema se resuelve mediante la pantalla y el teclado, pero en un sistema empotrado normalmente no disponemos de estos periféricos, por lo que hay que recurrir a otros métodos. Lo más habitual es que se derive la consola a un puerto serie. Este puerto serie se conecta a un PC (Host) sobre el que se ejecuta un programa que haga de terminal por puerto serie como Hyperterminal, minicom, gtkterm, etc. De esta manera disponemos de una pantalla y un teclado externos al sistema empotrado para poder comunicarnos con él.

Una vez establecida la comunicación con el gestor de arranque utilizaremos las funcionalidades que ofrece para configurar adecuadamente el sistema y proporcionar un entorno adecuado para lanzar la ejecución del kernel. El objetivo principal que se pretende conseguir es cargar el kernel y el sistema de ficheros en el dispositivo de memoria no volátil (memoria flash o disco), pero para ello es necesario definir previamente ciertos parámetros que nos permitirán efectuar la descarga. Es práctica habitual hacer este trasvase de datos a través de una interfaz de red, como por ejemplo Ethernet, usando para ello algún protocolo de transferencia de ficheros como TFTP, FTP ó SFTP. La descarga de datos se puede realizar también a través de una interfaz serie, pero resulta más lenta, por lo que es preferible hacerla a través de la red siempre que sea posible. Evidentemente, tenemos que fijar la información de red antes de poder transferir los datos, por lo que el gestor de arranque deberá proporcionar comandos para asignar la dirección IP del sistema empotrado, la dirección MAC, la dirección IP del Host, etc. Estos datos pueden ser cambiados posteriormente desde el propio gestor de arranque o desde Linux, por lo que no es estrictamente necesario que los datos consignados sean los definitivos para la tarjeta en cuestión, simplemente deberemos facilitar una información de red coherente para poder efectuar la transferencia.

Dependiendo de las necesidades del sistema transferiremos tantos bloques de datos como sean necesarios y se crearán las particiones necesarias. Normalmente sólo se necesitan cuatro particiones para poder arrancar el sistema con Linux:

Gestor de arranque
Parámetros de arranque
Kernel de Linux
Sistema de Ficheros

Tabla 9: Particiones básicas

En la posición más baja de la memoria se graba una imagen del propio gestor de arranque. La dirección de comienzo de esta partición debe coincidir con la dirección de comienzo de ejecución del microprocesador cuando se arranca en alguno de los modos de ejecución directa (sin bootstrap). De este modo, en posteriores arranques, el microprocesador ejecutará directamente el gestor de arranque tras un reset.

La segunda partición corresponde a los parámetros de arranque, donde el gestor de arranque almacena tanto parámetros relativos a su propio funcionamiento como parámetros de configuración que se le pasan al kernel en el momento de su lanzamiento.

En la tercera partición se almacenaría la imagen del kernel de Linux (normalmente comprimida debido a las restricciones de capacidad que suelen tener los sistemas empotrados).

Y en la cuarta partición se volcaría el sistema de ficheros, que contendría tanto los ficheros y programas necesarios para las aplicaciones como los ficheros y directorios que forman parte del sistema operativo completo (GNU/Linux).

En los cuatro casos el procedimiento a seguir sería el mismo:

1. Borrar el contenido de la partición (en el caso de memorias Flash)
2. Transferencia del contenido de la partición desde el host a la memoria RAM del sistema por FTP.
3. Grabación de los datos en el dispositivo físico de memoria no volátil.

Fase 3: El Kernel de Linux

Llegados a este punto el sistema está preparado para arrancar Linux. Si las particiones han sido grabadas correctamente y seleccionamos por hardware un arranque desde la memoria no volátil, el sistema ejecutará directamente el gestor de arranque y éste cederá el control al kernel de Linux siempre que el usuario no desee ejecutar algún comando del gestor.

El gestor de arranque cede la ejecución al kernel en el punto de entrada, que corresponde a un bloque de instrucciones que se limitarán a descomprimir el kernel (si la imagen estaba comprimida en forma de zImage ó bzImage). Estas instrucciones forman parte del código del kernel pero están escritas en ensamblador debido a que en este punto de la ejecución, el sistema operativo no está en marcha y toda la funcionalidad que éste ofrece no está disponible. Por regla general, la imagen del kernel se extrae de la memoria no volátil, normalmente más lenta, y la imagen descomprimida se almacena en memoria RAM para conseguir una ejecución más rápida. A pesar de este hecho, es posible ejecutar el kernel directamente desde el dispositivo de memoria no volátil. Concretamente, en el caso de las memorias flash, esta técnica se denomina XiP (eXecute in Place). De todas formas, salvo casos excepcionales, lo habitual es ejecutar desde memoria RAM.

Si la descompresión de la imagen del kernel se ha desarrollado con normalidad, el kernel tomará el control y empezará a inicializar servicios y a medida que lo va haciendo irá mostrando mensajes por la consola. Un detalle importante a tener en cuenta en relación a la consola es que normalmente Linux utiliza las señales de control de flujo en el caso de consolas por puerto serie. Si las líneas de control de flujo (RTS, CTS) no están conectadas al host es posible que la UART del sistema empotrado se bloquee interpretando que el host no está preparado para recibir datos. En este caso, aunque el kernel esté en condiciones de arrancar, no veremos nada en la consola y no podremos utilizar el sistema. En estos casos es conveniente efectuar las modificaciones oportunas sobre el código del driver de la UART, que se encuentra incluido en las fuentes del kernel, para ignorar las señales de control de flujo.

Cuando el kernel ha inicializado todos los servicios básicos se dispone a montar el sistema de ficheros principal en el directorio raíz. Dependiendo de la configuración del kernel y del tipo de sistema de ficheros que utilicemos el kernel utilizará un driver u otro para acceder a los ficheros. Es muy importante que el kernel pueda montar el sistema de ficheros, puesto que existen muchos ficheros que también forman parte del sistema operativo.

Una vez montado el sistema de ficheros el kernel continúa la inicialización hasta llegar al último paso: la ejecución de los scripts de inicialización. Estos scripts son editables por el programador y en ellos se pueden especificar todos los procesos que deben ejecutarse justo después de arrancar Linux. Dependiendo de la distribución utilizada, dichos scripts se organizarán de una manera u otra. El primero de los scripts en ejecutarse es “/etc/inittab” éste invoca a otros que a su vez pueden invocar a otros scripts. El script inittab tiene un formato especial y define los procesos que hay que ejecutar ante determinados eventos como por ejemplo al iniciar Linux. El resto de los scripts de inicialización suelen encontrarse en el directorio “/etc/rc.d” y tienen el formato habitual de los scripts de Linux. En el directorio “/etc/rc.d/init.d” se encuentran los scripts de inicialización y parada de los demonios y otros servicios. El último de los scripts en ejecutarse es “/etc/rc.d/rc.local” pero no se recomienda ejecutar en él demonios o programas que vayan a estar activos durante mucho tiempo.

Llegados a este punto, ya debe haber aparecido el prompt de Linux en la consola y a partir de este momento el sistema se puede utilizar de la misma manera que cualquier sistema Linux a través de la consola, teniendo en cuenta las evidentes limitaciones de los sistemas empotrados.

Sistemas de Ficheros

Existen muchos tipos de sistemas de ficheros soportados en Linux, la mayoría de ellos pensados para el almacenamiento de archivos en discos duros. Debido al tamaño, peso y otras características de los discos duros normalmente resulta inviable su utilización en sistemas empotrados. En estos sistemas se suele recurrir a la utilización de dispositivos de memoria no volátil de estado sólido. Concretamente, desde hace algunos años, lo que predominan son las memorias flash. Si bien estas memorias, a diferencia de los discos duros, no contienen elementos mecánicos y tienen unas dimensiones muy reducidas, su capacidad no es demasiado grande y, lo que es más grave, el número de escrituras está limitado. Por este último motivo se han desarrollado algunos sistemas de ficheros especialmente diseñados para aplicarse sobre memorias flash. Estos sistemas de ficheros aplican técnicas de detección de nivel de uso de los sectores de las memorias flash para conseguir aumentar la vida útil de las mismas. Además pueden tener mecanismos de aislamiento de sectores defectuosos. Un sistema de ficheros de este tipo es JFFS (Journalling Flash File System) en sus diferentes versiones.

Como ejemplo citaremos algunas características de JFFS2:

- Funciona sobre memorias de tipo NOR y de tipo NAND
- Permite el uso de hard links
- Puede utilizar algoritmos de compresión (zlib, rubin, rtime)
- Dispone de un mecanismo “recolector de basura” que evita el número de operaciones de E/S innecesarias
 - Basado en “inodes” y “dirent nodes”
 - El montaje de dispositivos JFFS2 puede ser algo lento si la capacidad es elevada.

Existen otras alternativas más simples para el sistema de ficheros como por ejemplo CRAMFS (Compressed ROM File System).

De CRAMFS se pueden destacar las siguientes características:

- Es muy simple y eficiente en lo referente al espacio utilizado, lo que lo hace adecuado para sistemas empotrados.
 - No permite la escritura.
 - Utiliza la compresión zlib por páginas, lo que permite un acceso aleatorio sin necesidad de descomprimir la imagen entera del sistema de ficheros.
 - El tamaño de los ficheros está limitado a 16MB
 - El tamaño total del sistema de ficheros puede llegar hasta 256MB

Existe otro método para montar el sistema de ficheros especialmente indicado para las etapas de desarrollo de un sistema empotrado. En situaciones en las que la memoria no volátil del sistema empotrado es insuficiente o existen problemas con los drivers para poder ser accedida puede plantearse el uso de un sistema de ficheros externo montado con NFS (Network File System). En este caso el sistema de ficheros real reside en un host externo y el sistema empotrado lo monta usando NFS haciendo uso de una interfaz de red. Este hecho aporta algunas ventajas:

- La limitación de capacidad viene impuesta por el tamaño del disco en el host, que normalmente es muy superior a la capacidad que pueda tener un sistema empotrado.
 - El acceso es en lectura y escritura sin restricciones.

Herramientas de desarrollo

Debido a las limitaciones de los sistemas empotrados resulta inviable el

desarrollo de aplicaciones dentro del propio sistema, como ocurre con los PCs. En su lugar el desarrollo de las aplicaciones tiene lugar fuera, en un host, mientras que las pruebas se hacen sobre el sistema real. Este hecho da lugar a la existencia de los denominados “compiladores cruzados” o “cross-compilers”, que son herramientas de compilación diseñadas para ser ejecutadas en un PC pero que generan ejecutables para la arquitectura que hayamos utilizado en el sistema empujado. Dependiendo del tipo de procesador (ARM, PowerPC, MIPS, etc.) tendremos que seleccionar un compilador u otro.

En el caso concreto de las herramientas de desarrollo para aplicaciones sobre Linux en sistemas empujados se suelen utilizar versiones modificadas de GCC, el conjunto de herramientas de compilación por antonomasia para Linux. Estas herramientas se manipulan de la misma forma que el GCC para las arquitecturas de PC pero además contienen opciones específicas para el tipo de procesador.

En el caso particular de los procesadores ARM las herramientas se denominan igual que en un gcc normal pero añadiendo el prefijo “arm-linux-“. Así, por ejemplo, el compilador de C se invoca con “arm-linux-gcc”, el de C++ como “arm-linux-g++”, etc. Los binarios generados con estos compiladores se transfieren al sistema empujado, donde pueden ser ejecutados.

También existen otras variantes del GCC que resuelven por software el tratamiento de flotantes en procesadores sin coprocesador matemático, como es el caso de algunos procesadores ARM. Estos compiladores (soft-float) resuelven las operaciones con flotantes de una manera mucho más eficiente que la que utiliza Linux por defecto en estos casos.

Aunque con las herramientas de compilación es suficiente para hacer desarrollos con sistemas empujados, existen empresas dedicadas a la creación de entornos de desarrollo para determinadas arquitecturas utilizadas en sistemas empujados. Estos entornos suelen ser de pago, a diferencia de las herramientas derivadas de GCC pero ofrecen un entorno mucho más amigable y facilitan la configuración, el desarrollo y la depuración de aplicaciones. Ejemplos de entornos de este tipo son Code Warrior y Target Wizard, ambos de Metrowerks.

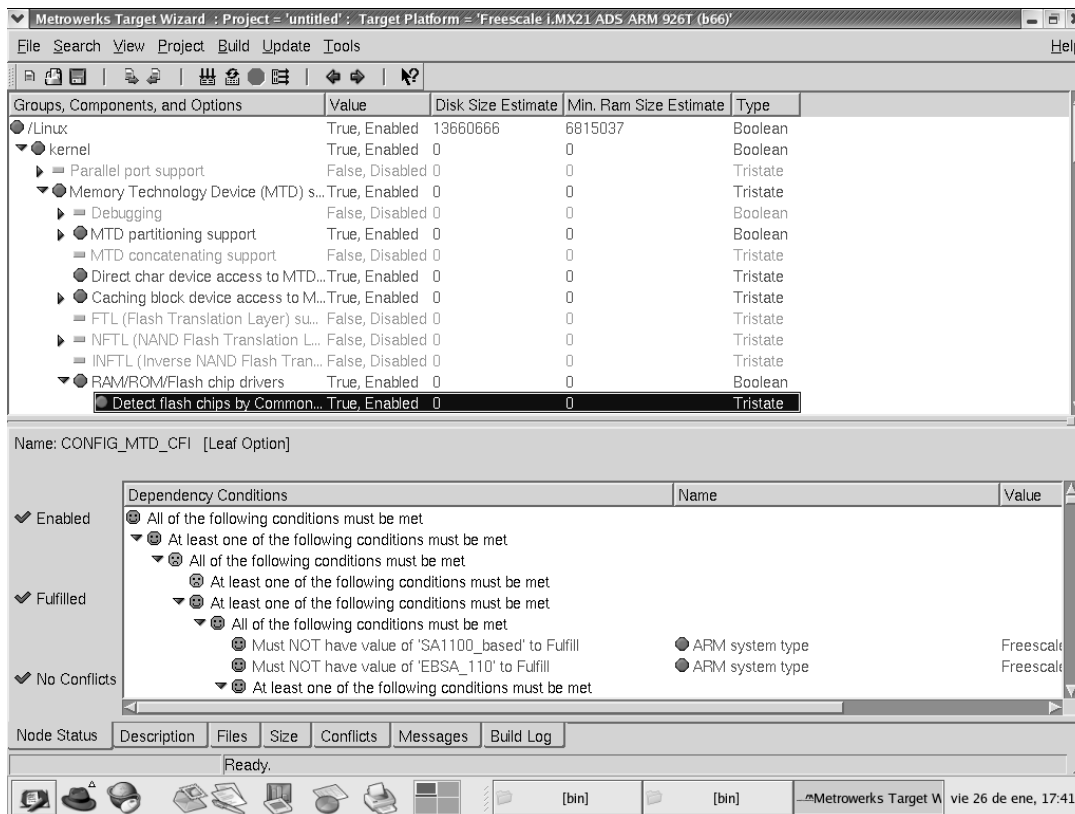


Figura 28: Vista de la aplicación Target Wizard

Estos entornos son muy útiles para configurar las opciones del kernel y los ficheros esenciales para el funcionamiento del sistema operativo. También nos permiten cargar paquetes con aplicaciones de utilidad.

Configurar correctamente el kernel es esencial para un buen funcionamiento, pero en sistemas empujados esto solo no es suficiente. Debido a la heterogeneidad de los mismos se hace necesario particularizar determinadas zonas de código del kernel de acuerdo con la arquitectura empleada, y con frecuencia hay que recurrir a drivers específicos o incluso construirlos [14]. Hacer cambios sobre el código fuente del kernel y sus drivers no es algo trivial y debe hacerse con mucho cuidado. Afortunadamente, algunos cambios son comunes dentro de una misma arquitectura y se describen en forma de parches (patches). El conjunto de parches se reúne en lo que se denomina una BSP (Board Support Package). Las BSPs se aplican al código fuente del kernel original (único para todas las arquitecturas) particularizándolo para la arquitectura concreta que se vaya a emplear.

La depuración de aplicaciones para sistemas empujados no es tan sencilla como en los sistemas Linux para PC. En sistemas empujados los posibles fallos pueden provenir de muchas partes del código, y suelen ser más difíciles de detectar. En

algunos casos podemos encontrar una versión de GDB para la arquitectura que estemos utilizando, pero esto sólo vale para depurar aplicaciones en el espacio de usuario. Para depurar código incrustado en las fuentes del kernel la cosa se complica. Existen otras técnicas como gdbserver o módulos “espía” que se insertan en el kernel y que permiten realizar la depuración enviando información a un host.

También existe la posibilidad de depurar con ayuda de un emulador. Los procesadores utilizados en sistemas empujados suelen incluir un puerto JTAG al que se puede conectar un emulador. La gran ventaja de estos dispositivos es que nos permiten observar la memoria y los registros internos del procesador para poder determinar dónde está el fallo. Los puertos JTAG también permiten controlar la ejecución, pudiendo detenerla, insertar breakpoints, etc. En determinadas circunstancias los emuladores pueden ser de gran ayuda para poner en marcha un sistema embebido con Linux.

Desarrollo de aplicaciones

Una vez instalado el sistema operativo sobre la tarjeta EPI-ARM y las herramientas de compilación apropiadas sobre el host se procedió al desarrollo de las aplicaciones que le confieren al sistema la funcionalidad prevista. El desarrollo comprende tres fases principales:

- Desarrollo y/o adaptación de drivers
- Desarrollo de los algoritmos de procesamiento
- Automatización y puesta a punto del sistema completo.

Drivers

Junto con la distribución utilizada ya existía una serie de drivers para determinados periféricos de uso común como por ejemplo el driver del controlador Ethernet o el driver para la tarjeta SD/MMC. Aunque muchos de estos drivers fueron aprovechables también es cierto que hubo que realizar modificaciones puntuales sobre algunos de ellos para adaptarlos a la arquitectura hardware concreta de la tarjeta EPI-ARM.

PRP

En el caso del módulo de preprocesamiento de la imagen hubo que llevar a cabo modificaciones importantes en el driver. El driver se distribuye dentro de un paquete que contiene los siguientes elementos:

- El driver del preprocesador como módulo insertable (emma_prp)
- Una librería de manejo del driver (prplib)
- Programas de ejemplo

El driver se presenta como un dispositivo de caracteres y se puede acceder a él a través de los medios convencionales (funciones open, ioctl, read, etc.), pero para facilitar su manejo y teniendo en cuenta la finalidad del driver se creó la capa de abstracción prplib, que fija de antemano algunos parámetros de configuración y proporciona una interfaz más sencilla. De esta manera, el diseñador de la aplicación sólo tiene que enlazar prplib con su aplicación y utilizarlo con la interfaz definida en prplib.h.

A continuación se detallan las principales características del driver:

- Formatos de entrada:
 - YUV 4:2:2
 - RGB 24 bits
- Formatos de salida para el canal 1:
 - YUV 4:2:2
 - RGB 24 bits
- Formatos de salida para el canal 2:
 - YUV 4:2:0
 - YUV 4:2:2
- Escalado desde 1:1 (100%) a 8:1 (12,5%).
- Conversión de color YUV a RGB y viceversa
- Detección de ausencia de señal de video
- Salida
 - Desentrelazada / No desentrelazada
 - Blanco y negro / color

Internamente, el driver maneja dos canales de vídeo que parten de la misma información de vídeo pero que pueden ser procesados de forma distinta. De esta manera, se puede tener, por ejemplo, una salida adaptada para ser procesada por un

algoritmo de OCR obteniendo una imagen en blanco y negro a tamaño completo y por otra parte tener la misma imagen pero en color y reducida a tamaño CIF para ser comprimida en formato MPEG-4.

Para cada canal, el driver dispone de un juego de dos buffers que se llenan de forma alterna de modo que cuando se le solicita al driver una imagen, éste devuelve la última imagen completa disponible, sin perjuicio de que en ese momento se esté recibiendo la imagen siguiente. Este mecanismo, unido a unas marcas de tiempo evita la pérdida de imágenes y ayuda a mantener la coherencia de la secuencia de vídeo. El empleo del doble buffer además evita esperas cuando se llama a la función de lectura, aunque siempre es posible sincronizarse con la llegada de una nueva imagen mediante algunas variantes de la función de lectura.

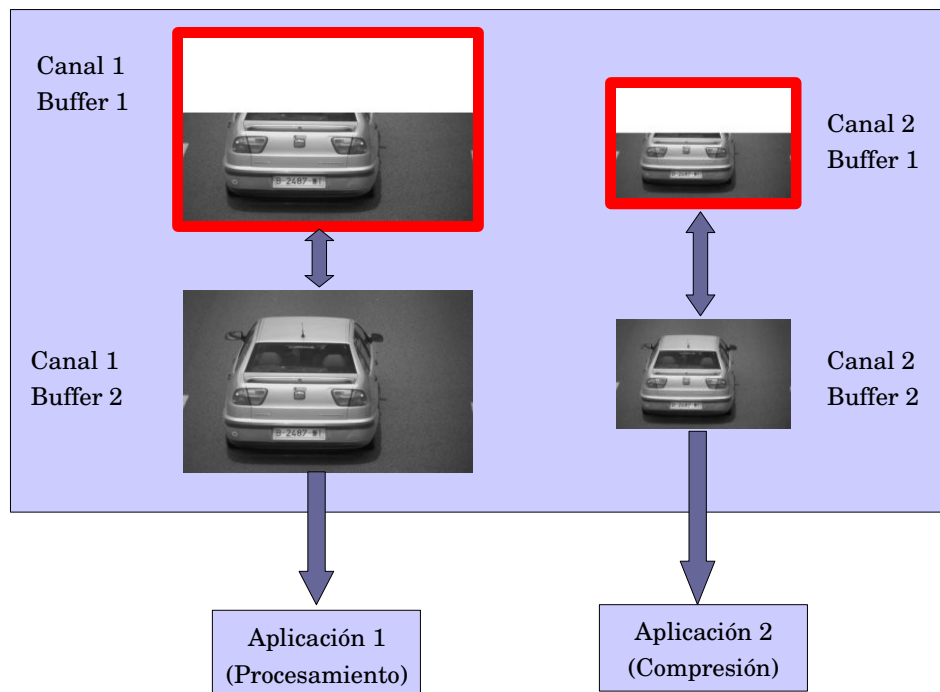


Figura 29: Mecanismo de doble buffer del driver del PRP

GPIO

El driver GPIO gestiona las entradas/salidas de propósito general del EPI-ARM. Al igual que el driver del PRP, éste también se distribuye con un paquete que incluye el módulo del driver, una librería de manejo (gpiolib) y programas de ejemplo. El driver también se representa como un dispositivo de caracteres.

Mediante una interfaz muy sencilla se pueden activar y desactivar las líneas de

salida y leer el valor presente en las líneas de entrada. Dado que la dirección de cada bloque de líneas GPIO viene fijada por la configuración hardware que se le haya dado a la placa EPI-ARM, se ha optado por hacer un driver genérico que pueda tratar las líneas GPIO como entradas o salidas y el control de la dirección se ha dejado en manos de la librería gpiolib, que debe ser específica para cada configuración hardware. De esta manera, se garantiza que la aplicación de usuario que utiliza el driver no realiza configuraciones incorrectas que pudieran dar lugar a cortocircuitos.

Aplicaciones

Una vez que se ha conseguido instalar el sistema operativo y que se han construido los drivers necesarios puede procederse al desarrollo de las aplicaciones. En este caso concreto se han desarrollado tres aplicaciones que, en conjunto, llevan a cabo el conteo de vehículos. La primera de ellas se dedica al procesamiento de las imágenes para obtener parámetros de tráfico tales como el número de vehículos, ocupancia, etc. La segunda aplicación se dedica a la compresión de video en formato MPEG-4 y a la transmisión del mismo por Internet. Una tercera aplicación gestiona el arranque de las dos aplicaciones anteriores y registra los eventos ocurridos en un logger.

La interacción entre las aplicaciones y los drivers se muestra en la siguiente figura:

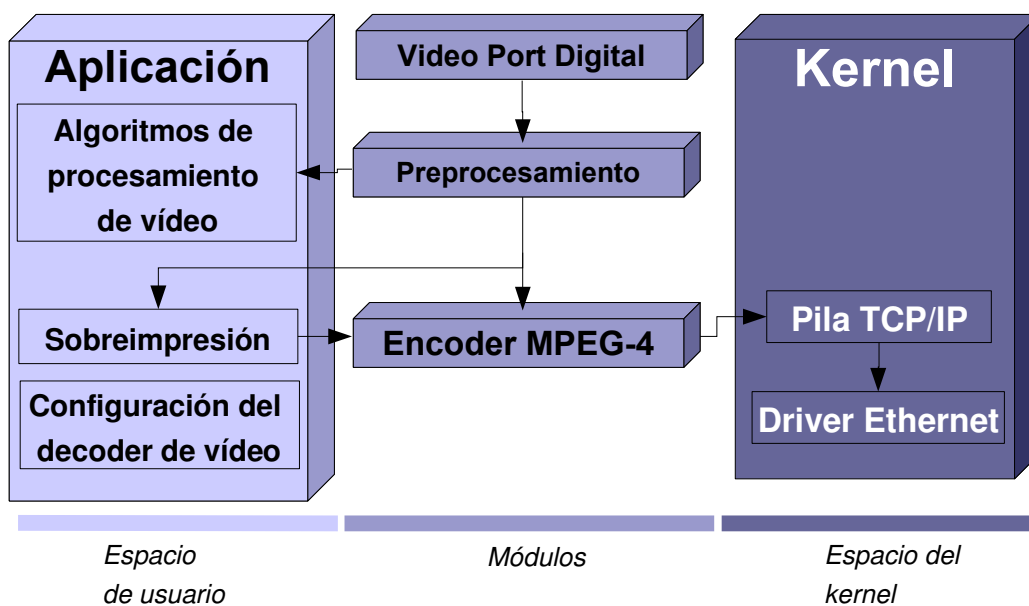


Figura 30: Interacción entre los drivers y las aplicaciones

El primero de los productos comerciales derivados de este proyecto está destinado al conteo de vehículo y obtención de datos del tráfico. Las aplicaciones desarrolladas le permiten establecer regiones de detección con las siguientes características

- Pueden ser de forma arbitraria, siempre que se trate de polígonos cerrados.
- Cada región puede realizar varias funciones simultáneamente.
- Se puede configurar un número arbitrario de regiones aunque hay restricciones en el número de regiones y en el área conjunta de las mismas.
- Función presencia:
 - No direccional.
 - Principalmente pensada para detección de incidentes o regiones que emulen espiras magnéticas.
- Función conteo.
 - Puede ser direccional
 - La diferencia frente al conteo por presencia, es que éste último trabaja por activaciones, sin individualizar cada uno de los vehículos que puedan aparecer simultáneamente en la región.
- Función cola.
 - Se ha ideado un algoritmo que denominamos "Detección de presencia detenida", para detectar cuándo aparecen vehículos detenidos dentro de la región.
 - Activación de salida digital asociada como alarma.
- Algoritmo de obtención del modelo de fondo.
- Detección y eliminación de sombras: Para reducir el efecto que pueden tener las sombras o los cambios de iluminación (autoiris, nublados, proyección de los faros de los vehículos sobre la calzada en condiciones nocturnas, etc) provocando numerosas falsas detecciones, se ha optado por localizar regiones que presenten cierta densidad de bordes de primer plano, intentando al mismo tiempo, filtrar aquellos bordes que pertenezcan al propio fondo.

El último paso en la cadena de desarrollo consiste en automatizar la ejecución de las aplicaciones. En este punto intervienen los scripts de inicialización, cuya misión es la de ejecutar automáticamente los procesos necesarios durante el arranque del sistema. En estos scripts se cargan de forma ordenada los módulos necesarios y a continuación todos los demonios y procesos involucrados.

Para la caracterización del comportamiento del sensor IP en una red se hace necesario el uso de diferentes herramientas que permitan monitorizar e intervenir el tráfico que circula por la red. Estas herramientas ayudan a simular cargas de tráfico, probar estrategias de transmisión del vídeo sobre IP y monitorizar la calidad del servicio ofrecido. Para la intervenir sobre el tráfico circulante se pueden utilizar las propias funciones de red que nos ofrece el S.O. (Linux en este caso) y para la monitorización se pueden emplear las librerías Libpcap.

Libpcap

Las librerías Libpcap[15] son unas librerías escritas en lenguaje C y publicadas bajo licencia open source que permiten programar un monitorizador de paquetes en redes locales de manera rápida y sencilla. Están escritas para soportar tanto sistemas operativos Windows como Linux.

Un monitorizador de red es un programa de escucha de una red y se encarga de recoger todos los paquetes que circulan por ésta. Para poder escuchar la información que circula por una red, se deben de reunir, como mínimo, dos requisitos. El primero es que la red sea de tipo broadcast. El segundo es que la tarjeta de red del usuario que quiera realizar la escucha se encuentre en modo monitor.

Las redes de tipo broadcast son aquellas en la que la información que envía un usuario a otro se expande por toda la red, llegando a todas las máquinas conectadas pero que solo la procesan los destinatarios. Cuando un usuario de estas redes envía una trama de información, todas las máquinas leen la dirección de destino de su cabecera. Si al comparar la dirección de destino con la suya ven que no son iguales, descartan la trama. Si ambas direcciones son iguales se procesará el paquete. El tipo de redes broadcast más conocido en la actualidad son las redes Ethernet.

Si se desea procesar una trama de información que no va dirigida a nosotros es necesario especificarle a nuestra tarjeta de red que lo haga. Para ello hay que activarle el flag de monitor. En otras palabras, una máquina con el flag de monitor activado procesa todas las tramas que recibe independientemente de que vayan dirigidas a ella

o no. Cuando una máquina tiene activado el flag de monitor se dice que trabaja en modo promiscuo.

Cuando una máquina recibe una trama Ethernet, ésta se procesa y se entrega a las capas superiores hasta llegar a la capa de aplicación. A partir de esta capa, la aplicación correspondiente procesa los datos recibidos. En el caso de un monitor de red, la información que éste procesa no sólo son los datos de aplicación, sino todos los datos recibidos incluyendo las cabeceras de las capas de Enlace, Red, Transporte y Aplicación. De esta manera se puede obtener información sobre quién envía datos en una red, de qué tipo son, a qué puertos van dirigidos, etc.

El funcionamiento básico de un monitorizador programado con las librerías libpcap se muestra en la siguiente figura:

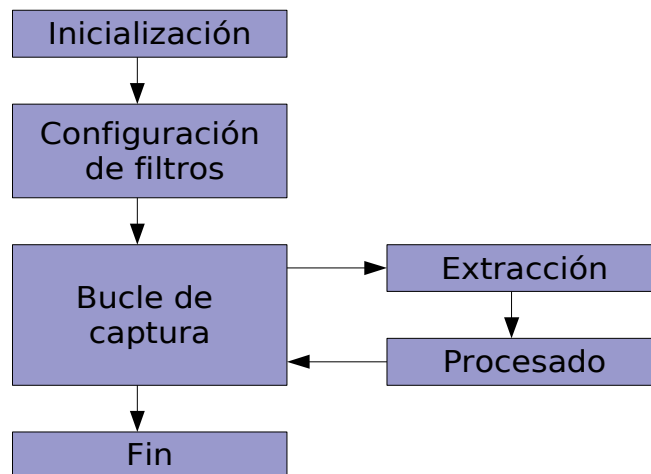


Figura 31: Esquema de uso de libpcap

Primero se inicializa el programa. En esta fase se asigna una interfaz (tarjeta) de red a monitorizar, y se ejecutan ciertas funciones necesarias para la correcta inicialización del programa. En la fase de activación de filtros, se activan los filtros de captura que permitirán decidir de qué máquinas o redes capturamos tráfico o qué tipo de tráfico deseamos capturar. La tercera fase es la fase de captura, donde se realiza el bucle que escucha la red hasta recibir algún paquete. Cuando se recibe información entramos en la fase de extracción de datos, en la cual se realiza el proceso de extraer la información de las cabeceras y del payload (datos útiles) del paquete. Una vez extraída, entramos en la fase de procesado de datos, donde podemos mostrarlos por pantalla o guardarlos en un fichero entre otras funciones. Tras procesar los datos volvemos al bucle del programa, el cual puede acabar si cumple cierta condición, como por ejemplo

que se hayan recibido un número de paquetes determinado o que haya pasado cierto tiempo. Una vez que se sale del bucle el programa entra en la fase de salida, en la cual finaliza el programa.

Inicialización del programa

La fase de inicialización del programa engloba a las funciones capaces de obtener información del sistema: interfaces de red instaladas, configuración de estas interfaces (Dirección IP, Máscara de Red), etc.

Las funciones más relevantes de esta fase son las siguientes:

*char *pcap_lookupdev(char *errbuff)*

Esta función busca el primer dispositivo de red válido para captura y devuelve un puntero a éste. En caso de error devuelve NULL y una descripción del error en la cadena *errbuff*.

*int pcap_lookupnet(char *device, bpf_u_int32 *netp, bpf_u_int32 *maskp, char *errbuf)*

Una vez obtenido el nombre de una interfaz válida podemos consultar su dirección de red (no su dirección IP) y su máscara de subred. *device* es un puntero a una cadena de caracteres que contiene el nombre de una interfaz de red válida. *Netp* y *maskp* son dos punteros a *bpf_u_int32* en los que la función almacenará la dirección de red y la máscara de red respectivamente. Si se produce un error la función devuelve -1 y una descripción del error en la variable *errbuff*.

*int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf)*

Esta función devuelve todas las interfaces de red que pueden ser abiertas para captura de datos. Puede darse el caso de que existan interfaces de red que no puedan ser abiertas (por ejemplo por no tener permisos). Con esta función sólo aparecen las interfaces que han sido abiertas. Para llamar a esta función es suficiente pasarle un puntero sin inicializar de tipo *pcap_if_t*. La función transforma ese puntero en una lista enlazada que contendrá cada una de las interfaces y sus datos asociados (dirección y máscara de red). En caso de error la función devuelve -1 y una descripción del error producido en *errbuff*.

*int pcap_datalink(pcap_t *p)*

Esta función devuelve el protocolo de enlace de datos asociado a una interfaz de red. Existen numerosos valores, cada uno asociado a un tipo de red. Los más habituales son:

DLT_EN10MB Ethernet (10Mbps, 100Mbps, 1000Mbps)

DLT_IEEE802_11 IEEE 802.11 wireless LAN

Filtros de captura

Como se ha comentado anteriormente, libpcap es una librería de funciones que ejecuta un monitor en espacio de usuario (user space). Sin embargo la captura de datos debe realizarse en la zona del Kernel (Kernel space). Esto significa que es necesario un mecanismo capaz de traspasar esta frontera y este mecanismo debe ser eficiente y seguro ya que cualquier fallo en las capas interiores del espacio del Kernel puede ocasionar una degradación en el rendimiento de sistema. En el caso de no usar un filtro de captura, todos los paquetes que fueran recibidos traspasarían la frontera del espacio del kernel para llegar al espacio de usuario. Esto es muy ineficiente si solo se desea capturar un cierto tipo de paquetes (por ejemplo, los que van dirigidos al puerto 21). La solución a esta ineficiencia es establecer un filtro en la zona Kernel que solo deje pasar los paquetes que nos interese capturar. Ésta es la principal labor de un Packet/Socket Filter.

Prácticamente cada sistema operativo posee un sistema de filtrado propio. Nosotros nos centraremos en el más extendido en Linux, el BPF (BSD Packet Filter). El funcionamiento de BPF se basa en dos componentes: el Network Tap y el Packet Filter. El Network Tap se encarga de recopilar los paquetes desde el driver del dispositivo de red y de entregárselos a las aplicaciones a las que estén destinados. El Packet Filter tiene por función decidir si el paquete debe ser aceptado o no. En caso afirmativo también se encarga de decidir qué cantidad de información del paquete le entrega a la aplicación. Esto se hace porque no tiene sentido que ésta reciba información de, por ejemplo, las cabeceras Ethernet.

En la Figura 32 puede apreciarse el esquema general del funcionamiento de BPF. Cuando la interfaz de red recibe un paquete, el driver de red comprueba que este paquete vaya dirigido a la máquina. En caso afirmativo lo entrega a la pila de protocolos (protocol stack), si no es así descartará el paquete. Si cuando se recibe un paquete el BPF se encuentra activo, dicho paquete será procesado por el filtro en vez de ser enviado a la pila de protocolos o descartado.

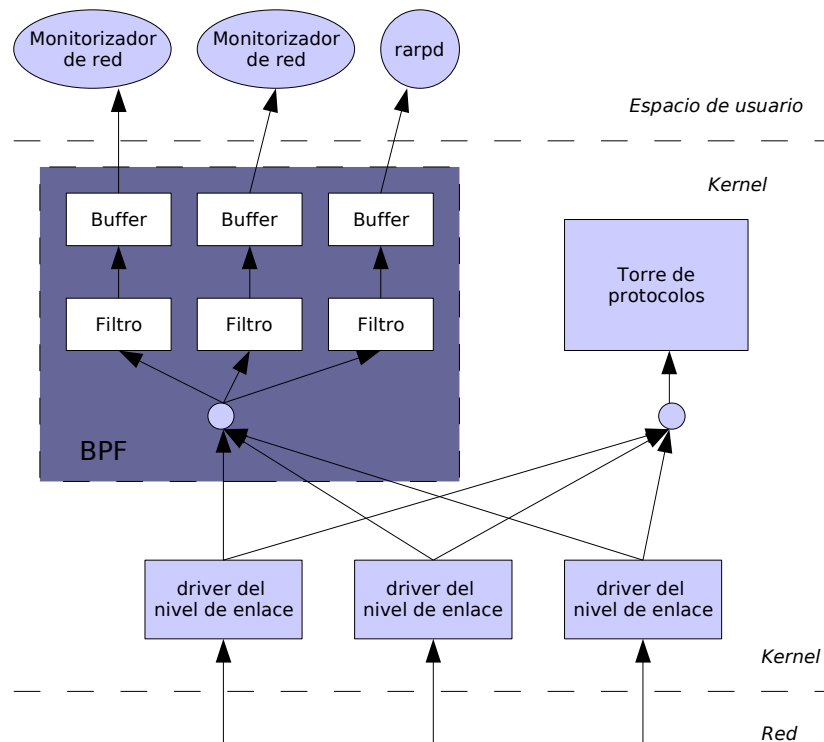


Figura 32: Esquema de filtrado BPF

El BPF se encarga de comparar el paquete con cada uno de los filtros establecidos, entregando una copia de dicho paquete a los buffers de las aplicaciones cuyo filtro se ajuste al contenido del paquete. Si no existe ninguna coincidencia filtro-paquete, se devuelve el paquete al driver, el cual actuará como si el BPF no estuviera activo. Todo este proceso sucede sin que el paquete salga de la zona Kernel. Existen procesos que pueden estar interesados en consultar cada uno de los paquetes que circulan por la red, lo que echa por tierra todos los intentos de maximización del rendimiento introducidos por el BPF. Una manera de aumentar la eficiencia es el uso de buffers, lo que permite entregar varios paquetes de una sola vez, evitando así cambios continuos de zona. Para mantener la secuencialidad en que se reciben los paquetes, BPF utiliza una marca de tiempo, tamaño y offset a cada uno de los paquetes del grupo. BPF utiliza su propio lenguaje para la programación de sus filtros.

Las librerías Libpcap implementan una interfaz que permite utilizar los filtros BPF mediante un lenguaje más amigable al usuario. Una vez escrito el filtro en el lenguaje Libpcap, estas librerías se encargan de compilar el filtro a lenguaje BPF para que pueda ser aplicado. El lenguaje de alto nivel desarrollado por Libpcap ha sido popularizado por TCPdump gracias al cual se ha convertido en el estándar para definir filtros de captura.

La expresión utilizada para definir un filtro contiene una serie de primitivas y tres posibles modificadores a las mismas. La expresión puede ser verdadera o falsa. En caso de ser verdadera el paquete se entrega a la zona de usuario, mientras que si es falsa se devuelve al controlador de red. Los tres modificadores posibles son:

tipo: Puede ser `host`, `net` o `port`, indicando respectivamente a una máquina, una red completa o un puerto concreto. Por defecto se asume el tipo `host`.

dir: Especifica el sentido del tráfico que se desea capturar. Las dos opciones son `src` y `dst`. Por defecto se captura el tráfico que circula en ambos sentidos (`src or dst`).

proto: En este caso se especifica el protocolo que se desea capturar. Las opciones son `tcp`, `udp`, `ip`, `ether` (en este caso se capturan tramas a nivel de enlace), `arp` o `rarp` (peticiones `reverse-arp`).

Todas las expresiones anteriores pueden ser combinadas con la ayuda de operadores lógicos: negación (`!` `not`), concatenación (`&&` `and`) o adición (`||` `or`).

Algunos ejemplos de estas expresiones serían:

- `192.168.1.1 and proto tcp and (dst port 21 or dst port 22)`
- `(dst net 192.168 or dst host 127.0.0.1) and proto udp and port 55500`

Si se desea más información sobre el lenguaje de filtros de `libpcap` puede consultarse el manual de `tcpdump` usando la orden “`man tcpdump`”.

Las funciones específicas que `libpcap` proporciona para la fase de filtrado son las siguientes:

*int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask)*

Esta función se encarga de compilar un programa de filtrado en formato `tcpdump` (`char* str`) en su BPF equivalente (`bpf_u_int32 netmask`). `netmask` es la máscara de red local que puede obtenerse mediante la función `pcap_lookupnet()`. Si se produce un error la función retorna `-1`. Para una descripción más detallada de lo sucedido se puede emplear la función `pcap_geterr()`.

*int pcap_setfilter(pcap_t *p, struct bpf_program *fp)*

Esta función se utiliza para aplicar el filtro ya compilado. Para ello hay

que pasarle como parámetro a esta función el resultado de compilar el filtro con `pcap_compile()`. En caso de error la función devuelve -1 y puede obtenerse una descripción más detallada con `pcap_geterr()`.

Bucle de captura

Una vez que se ha compilado y aplicado el filtro de captura el programa entra en su siguiente fase, el bucle de captura. En esta fase se espera la llegada de algún paquete. Cuando éste llega se activa una función que permite capturarlo. Durante la fase del bucle es posible utilizar varias funciones que permiten distintos modos de funcionamiento del programa durante esta fase. Las principales diferencias entre estas funciones son el número de paquetes que se quieren capturar, el modo de captura (modo promiscuo o normal) y la manera en que se definen las funciones de llamada o Callbacks (la función invocada cada vez que se captura un paquete). A continuación se describen estas funciones con más detalle.

*pcap_t *pcap_open_live (char *device, int snaplen, int promisc, int to_ms, char *errbuf)*

Para que el programa pueda entrar en el bucle de captura hay que obtener un descriptor de tipo `pcap_t`, por lo cual se utiliza esta función. El parámetro `char *device` es el nombre del dispositivo de red en el que se realizará la captura. Si a este parámetro se le asignan los valores ANY o NULL se fuerza la captura en todos los dispositivos disponibles. El argumento `int snaplen` indica el número máximo de bytes que serán capturados. El siguiente parámetro, `int promisc`, indica el modo de captura. Si su valor es 0 se realizará una captura en modo normal, cualquier otro valor indica que la captura se realizará en modo promiscuo. El parámetro `int to_ms` especifica cuantos milisegundos se quiere que el Kernel agrupe paquetes antes de entregárselos a la zona de usuario. Esto se hace por razones de rendimiento, ya que como se ha explicado antes, realizar esta operación para cada paquete sería muy costoso. La función devuelve NULL en caso de producirse un error, para el cual puede encontrarse una descripción en el parámetro `errbuf`.

*int pcap_dispatch (pcap_t *p, int cnt, pcap_handler callback, u_char *user)*

Con esta función se capturan y procesan los paquetes. El parámetro `cnt` especifica el número máximo de paquetes a procesar antes de salir del bucle. Si `cnt` vale -1 se capturará indefinidamente. El argumento `callback` es un puntero a la función que se invocará para procesar el

paquete. El puntero es de tipo `pcap_handler()`, al cual se le pasan los dos siguientes parámetros:

Puntero `u_char`: Aquí es donde se encuentra el paquete.

Estructura `pcap_pkthdr`

La función devuelve el número de paquetes capturados o -1 en caso de producirse un error. Para mostrar un mensaje más detallado del error pueden utilizarse las funciones `pcap_perrorr()` y `pcap_geterr()`.

*int pcap_loop (pcap_t *p, int cnt, pcap_handler callback, u_char *user)*

Esta función es muy parecida a `pcap_dispatch()`. La principal diferencia reside en que `pcap_loop()` no finaliza cuando se produce un error por timeout. En caso de error la función devuelve un número negativo y 0 si el número de paquetes indicados en `cnt` se ha completado con éxito.

*u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)*

Esta función lee un único paquete y devuelve un puntero a `u_char` con su contenido. Esta función es una alternativa a las anteriores. Con esta función no existe la necesidad de declarar ninguna función de callback.

Extracción de datos

Una vez que se captura un paquete se entra en la fase de extracción de datos. En esta fase se tienen que separar datos útiles de cabeceras para que luego puedan ser procesados. Hasta este momento, el paquete que se ha entregado es un conjunto de bytes en bruto llamados RAW bytes. Para poder obtener información inteligible es necesario realizar la función que la pila de protocolos hubiera hecho si el paquete no hubiera sido capturado por el filtro. Esto significa que es necesario interpretar las cabeceras que tiene el paquete, lo cual no puede realizarse si no se conocen como están estructuradas las cabeceras. En esta sección se explicará de manera genérica algunos de los campos de éstas, pero si se quiere realizar una lectura más profunda de éstas se pueden consultar los RFC y estándares que las definen. Los más importantes son:

RFC 768 (UDP)

RFC 791 (IP)

RFC 792 (ICMPv4)

RFC 793 (TCPv4)

RFC 826 (ARP)

Cuando se recibe un paquete la primera cabecera que se extrae es la de la capa más baja para luego ir subiendo capa a capa. En este caso la primera cabecera a extraer es la de Ethernet. El primer dato que hay que conocer de la cabecera de una trama Ethernet es su tamaño: 14 bytes. Los campos más importantes de esta cabecera son la dirección ethernet origen (`ether_shost`), la dirección ethernet destino (`ether_dhost`) y el tipo de paquete que porta. Los tres tipos de paquetes más típicos son:

<code>ETHERTYPE_IP</code>	(paquete IP)
<code>ETHERTYPE_ARP</code>	(paquete tipo ARP)
<code>ETHERTYPE_RARP</code>	(paquete de tipo RARP)

En cuanto a las direcciones, Linux proporciona una serie de funciones que permiten transformar direcciones Ethernet de 48 bits a texto legible. Si se desea más información sobre estas funciones puede consultarse el fichero `/usr/include/netinet/ether.h`.

Una vez conocida la estructura de la cabecera Ethernet es posible extraerla del paquete `u_char* packet`. Para ello solo hay que desplazarse por el `u_char* packet` 14 bytes. El resto de información del paquete dependerá del valor que tenga el campo `ether_type`, por lo que es necesario consultarlo.

En Linux, la cabecera IP se encuentra definida en `/usr/include/netinet/ip.h`. Si se consulta este archivo puede observarse que la cabecera IP puede variar de tamaño. La longitud de la cabecera IP se encuentra en el campo `ip_hl`. Otros campos importantes de la cabecera IP son la direcciones de origen y destino (`saddr` y `daddr` respectivamente) y el protocolo que se encuentra dentro del payload de IP (`protocol`). Los protocolos más importantes indicados por este campo son:

ICMP	(protocol = 1)
TCP	(protocol = 6)
UDP	(protocol = 17)

Si suponemos que el paquete recibido contenía un datagrama UDP, el valor del campo `protocol` será 17. Para extraer el datagrama tendremos que recorrer `ip_hl` bytes del paquete `u_char* packet`. En este caso obtendremos el datagrama UDP. La cabecera UDP tiene un tamaño fijo de 8 bytes, por lo que si se quiere extraer el payload que contiene un datagrama UDP solo hay que avanzar 8 bytes en el paquete `u_char*`

packet. Para el caso de otros protocolos indicados por el campo protocol de la cabecera IP el proceso de extracción sería similar. Los pasos a seguir serían los siguientes:

1. Extracción de la cabecera Ethernet (14 primeros bytes del u_char* packet).
2. Consulta del campo ether_type de la cabecera Ethernet para saber si el paquete es IP.
3. Si el paquete es IP, consulta del campo protocol de la cabecera IP.
4. Consulta del archivo indicado por protocol (alojado en /usr/include/netinet/)
5. Consulta del tamaño de la cabecera del protocolo. Si existe payload, éste viene a continuación de la cabecera del protocolo.

Procesado de datos

En la fase de procesado de datos ya se dispone, tanto de las cabeceras del paquete recibido como del paquete en sí. Con estos datos se pueden realizar las funciones que se deseen, desde realizar estadísticas del tráfico que fluye por la red hasta espiar la información de los usuarios de ésta. Al ser tan extensas las posibles aplicaciones existentes, éstas no se tratarán en este apartado. Sin embargo sí que se explicarán las funciones que ofrece Libpcap para el almacenamiento de estos datos en ficheros y el análisis de éstos.

*pcap_dumper_t *pcap_dump_open(pcap_t *p, char *fname)*

Esta función se utiliza para abrir un fichero en el que se guardarán los datos que sean capturados. Si no hay problemas la función abrirá el fichero char* fname en modo escritura y devolverá un puntero a un descriptor de tipo pcap_dumper_t. En este descriptor es donde se podrán volcar los datos de la captura. Si se produce un error la función retornará NULL. Mediante la función pcap_geterr() se puede ver una descripción de dicho error. Las siguientes funciones se ejecutan una vez se ha realizado la captura.

pcap_t pcap_open_offline(char fname, char *errbuf)*

Esta función permite abrir un fichero en modo lectura cuyo contenido es una captura de paquetes en formato tcpdump. fname indica la ruta del fichero. La función devuelve NULL en caso de error. Para consultar éste se usa errbuf.

*void pcap_dump_close(pcap_dumper_t *p)*

Si se ha terminado con el fichero de salida, éste puede cerrarse con la

función `pcap_dump p_close`.

Transmisión de vídeo y redes

Existen diferentes soluciones para la transmisión de vídeo a través de redes IP, de las cuales estudiaremos tres, describiendo las ventajas e inconvenientes de cada una de ellas:

Unicast

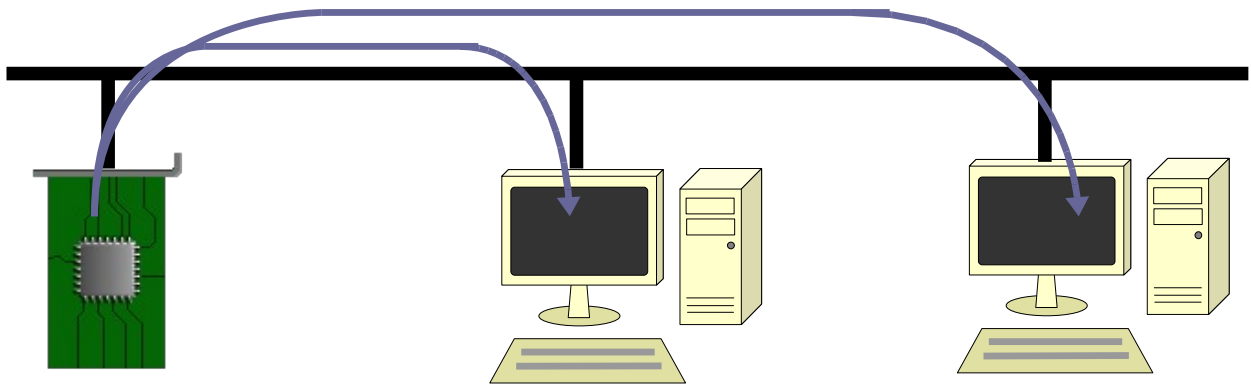


Figura 33: Esquema de transmisión Unicast

Esta solución es la más sencilla de todas aunque es la menos eficiente cuando hay más de un cliente que quiere visualizar un determinado stream de vídeo. De acuerdo con la figura, el emisor envía un paquete por cada receptor conteniendo una copia del stream. Los clientes tienen que hacer una petición al emisor para que les envíe el stream de vídeo y deberían indicarle también en qué momento desean dejar de recibir el stream.

Ventajas:

- El esquema es simple tanto para el emisor como para los receptores.

Desventajas:

- Si N receptores piden visualizar el stream, el emisor tiene que enviar N copias de cada paquete, con la consiguiente sobrecarga para el sistema emisor.
- No optimiza el ancho de banda.

- Obliga a implementar mecanismos de solicitud de conexión/desconexión al stream para poder servir el vídeo y evitar situaciones en las que el emisor siga enviando paquetes inútilmente a un equipo que ha dejado de escuchar el stream o que simplemente ha caído.

Multicast

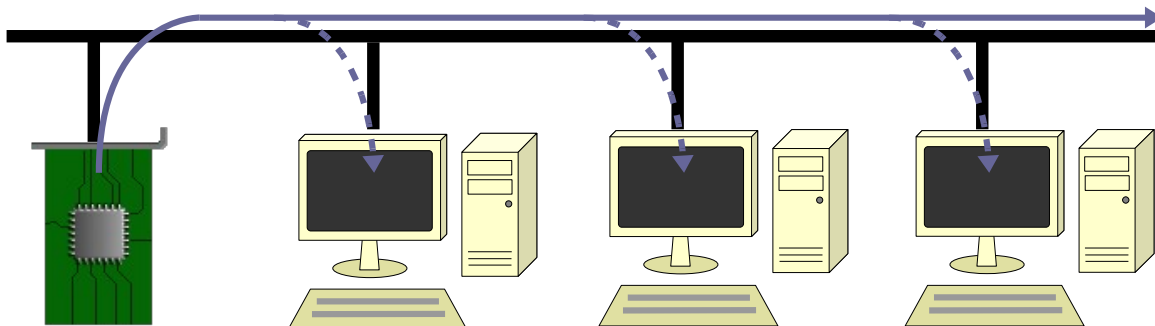


Figura 34: Esquema de transmisión Multicast

Esta solución es una de las más eficientes para la transmisión de vídeo a diferentes equipos. La transmisión multicast permite que el emisor envíe una única copia del stream de vídeo para todos los receptores. Los receptores se suscriben a la dirección IP multicast hacia la que el emisor está dirigiendo el stream de vídeo de modo que sólo aquellos equipos interesados recibirán el stream de vídeo, evitando sobrecargas al sistema EPI-ARM. Para más información acerca de multicast consultar el Anexo A.

Ventajas:

- Alta eficiencia para la distribución de datos a múltiples receptores.
- Siempre supone para el sistema emisor la misma carga computacional, independientemente del número de receptores suscritos.
- Es sencillo de implementar en el lado del emisor.
- No es necesario que el emisor lleve un control del número de receptores que están escuchando.

Desventajas:

- Mayor complejidad de implementación en el lado del receptor.
- Es necesario configurar los routers de la red adecuadamente dependiendo de las subredes que el tráfico multicast tenga que atravesar hasta sus destinos.

Unicast + Servidor

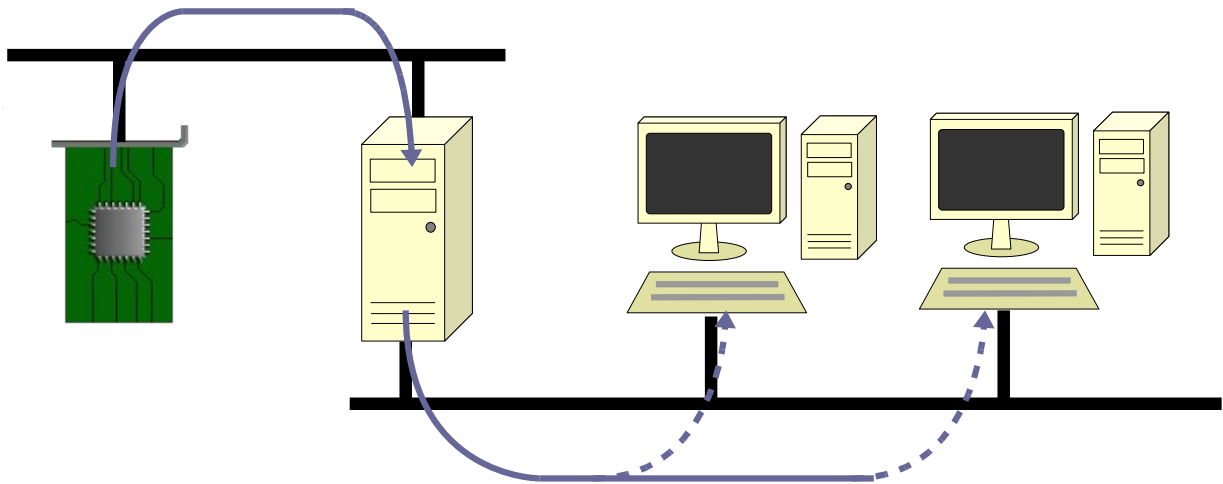


Figura 35: Esquema de transmisión Unicast + Servidor

Esta solución, aunque algo más compleja que la anterior, también permite simplificar las cosas desde el punto de vista del emisor. En este caso concreto, el emisor transmite en unicast pero siempre hacia un único destino, el servidor. Todos los equipos EPI-ARM de la red transmitirían a dicho servidor desde el que se redistribuirían los flujos de vídeo hacia los destinatarios. Dentro de la subred de los receptores es posible transmitir el vídeo mediante unicast o multicast, según convenga.

Ventajas:

- Alta eficiencia para la distribución de datos a múltiples receptores.
- Siempre supone para el sistema emisor la misma carga computacional, independientemente del número de receptores suscritos.
- Es sencillo de implementar en el lado del emisor.
- Se puede realizar un control más exhaustivo del acceso a los flujos de vídeo mediante mecanismos de usuarios y permisos implementados en el servidor.
- El tráfico, al ser unicast, es mucho más fácil de dirigir a través de los routers desde los emisores hasta el servidor.

Desventajas:

- Aparece un nuevo equipo que encarece el sistema, el servidor de vídeo.
- La difusión de varios flujos de vídeo se centraliza en un equipo, y aunque se

pueden incluir servidores redundantes, esta medida encarecería aún más el coste del sistema.

- Dependiendo del método de transmisión en la subred de los receptores la complejidad en el lado del receptor puede ser similar al caso de transmisión en multicast.

Desarrollo de las pruebas

En el siguiente apartado se describen algunas de las pruebas que se han efectuado sobre el sistema EPI-ARM para observar su comportamiento en red.

Motivación de las pruebas

Son dos los motivos principales que han dado lugar al desarrollo de las pruebas. Por un lado, y más importante, aparece la preocupación por el comportamiento del sistema en “redes sucias” y por otro lado, se quiere comprobar que el flujo de vídeo generado se puede reproducir en directo sin problemas.

La experiencia ha demostrado que algunos equipos utilizados en las redes de tráfico no se comportan bien cuando son conectados a las denominadas “redes sucias”, es decir, redes en las que hay una cierta cantidad de tráfico que no va dirigido al equipo pero que es recogido por éste afectando a su rendimiento. Concretamente, se ha observado que determinados conversores Ethernet/Serie se colapsan cuando se insertan en redes con tráfico multicast, que evidentemente, no va dirigido a ellos.

La segunda de las pruebas va encaminada a comprobar que el flujo de vídeo transmitido se distribuye en los paquetes de forma adecuada y que alcanza el receptor a tiempo. Dada la naturaleza de los datos, los flujos de vídeo suelen transmitirse usando protocolos del nivel de transporte no orientados a conexión, máxime cuando existen múltiples receptores. En las transmisiones de vídeo en tiempo real, como es el caso, la información debe llegar a tiempo, y no tiene sentido representar las imágenes que lleguen fuera de tiempo, por lo que no suelen ser necesarias las retransmisiones. A pesar de que el hecho de que lleguen todas las tramas no es un parámetro crítico, evidentemente, no es conveniente que se pierdan tramas a menudo, ya que la calidad de la transmisión se verá resentida [16].

Prueba 1: Robustez del sistema en redes sucias

Para el desarrollo de esta prueba se ha conectado una tarjeta EPI-ARM con un PC usando un cable de red cruzado para emular una red de área local. Se han simulado las condiciones de una red sucia utilizando el PC y observando el rendimiento de la tarjeta sobre la propia consola del EPI-ARM. Para evaluar el rendimiento se ha monitorizado el número de frames por segundo (fps) que registra la aplicación de procesamiento de imágenes. Desde el PC se ha ejecutado una aplicación de transmisión de datos denominada “morralla” para emular la red sucia. Esta aplicación dirige un flujo de datos arbitrarios hacia un destino configurable con un ancho de banda también ajustable.

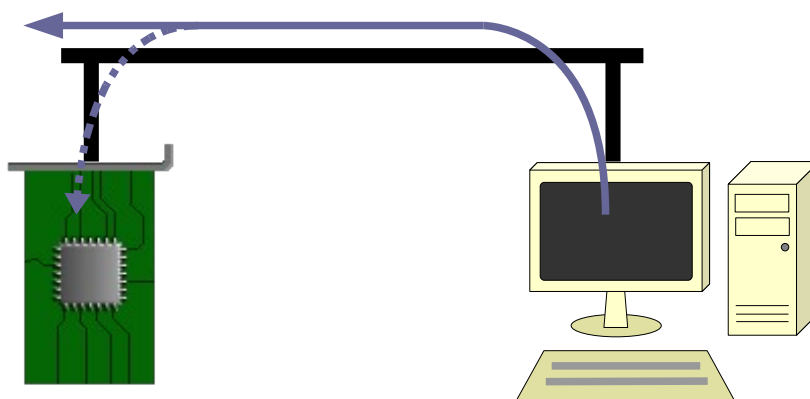


Figura 36: Montaje realizado para la prueba 1

Durante la prueba se generó tráfico multicast desde el PC dirigido al puerto 456 a distintas intensidades. Aunque evidentemente en el EPI-ARM ninguna aplicación espera recibir esos datos, lo que se pretende determinar es la carga computacional que pudiera derivarse del simple rechazo de los paquetes y cómo esto afecta al rendimiento del sistema.

Los resultados de las pruebas fueron los siguientes:

Situación	Sin tráfico	Tráfico Multicast	Tráfico Multicast
Tráfico efectivo (bps)	0	960000	8000000
fps procesados	6,88	6,97	6,98

Tabla 10: Resultados de la prueba 1

Como se observa, los resultados son muy parecidos, lo que indica que el tráfico multicast no afecta. A pesar de que se detectan pequeñas diferencias en el número de fps procesados entre las diferentes situaciones, hay que aclarar que la velocidad de

procesamiento del algoritmo depende de las imágenes que se estén capturando. Este hecho explica esa pequeña diferencia.

De los resultados también se desprende que el rechazo del tráfico multicast se hace en el nivel de enlace, por lo que no resultan afectadas las capas superiores de la torre de protocolos.

Para contrastar los resultados con el efecto que podría haber tenido en la situación más desfavorable se hizo una última prueba en la que sustituyó el tráfico multicast por tráfico unicast, que tampoco iba a ser recibido por ninguna aplicación pero que alcanzaría el nivel de red antes de ser rechazado. En esta situación, con un tráfico efectivo de 8000000 bps dirigidos al EPI-ARM el rendimiento bajó hasta los 5,57 frames por segundo, es decir, unos 1,37 fps menos que en el resto de los casos.

Prueba 2: Medida del tiempo entre paquetes

Para medir el tiempo de paquetes se utilizó un montaje similar al de la prueba 1, pero en este caso se utiliza el EPI-ARM para transmitir vídeo y el PC tanto para recibirlo y visualizarlo como para analizar el tiempo de llegada de los paquetes con libpcap.

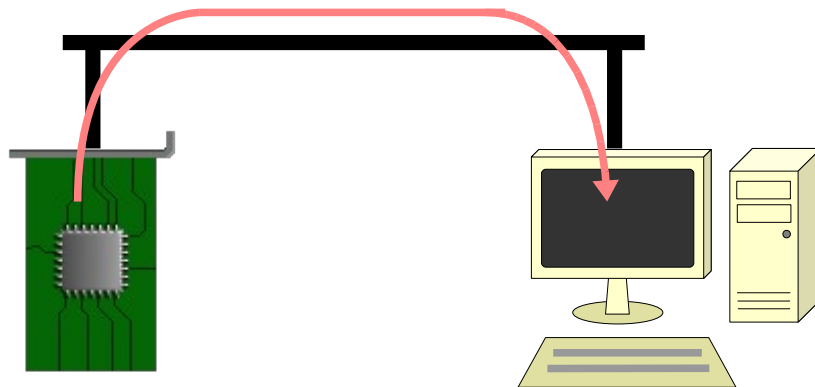


Figura 37: Montaje realizado para la prueba 1

El EPI-ARM envía un paquete UDP por cada fragmento del stream de vídeo que resulta de la compresión de una trama. El vídeo transmitido tiene las siguientes características:

- Tipo: MPEG-4 Elementary Video
- Bitrate: 384Kbps

- Tamaño: CIF (352 x 288)
- Tasa de frames: 25 fps

Para la monitorización se ha utilizado un programa basado en libpcap configurado para filtrar el tráfico UDP procedente del EPI-ARM. Los resultados obtenidos son los siguientes:

Variables	Resultados
Nº de paquetes	53
Media	0,030153 s
Varianza	0,001907
Desviación típica	0,043664
Máximo	0,0260425 s
Mínimo	0,000080 s

Tabla 11: Resultados de la prueba 2

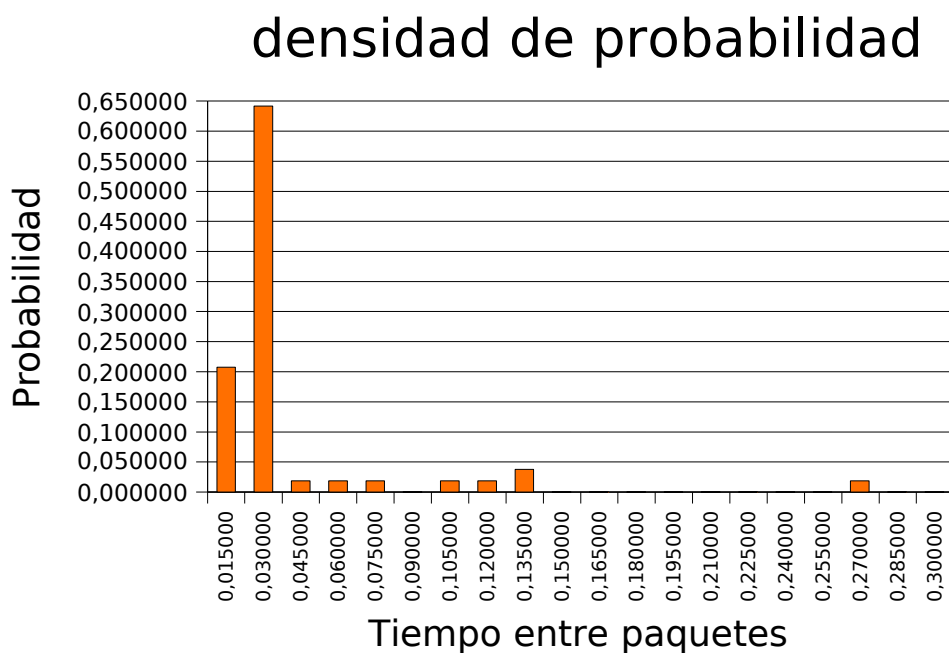


Figura 38: Densidad de probabilidad para el tiempo entre paquetes

De los resultados anteriores se desprende que el tiempo entre paquetes tiene valores aceptables en el 85% de los casos, lo cual es suficiente para la mayoría de los reproductores. En cualquier caso, muchos de los reproductores de vídeo disponen de un buffer de recepción con el que se suaviza el efecto de la aleatoriedad del tiempo entre paquetes.

Aportaciones del proyecto

La participación en este proyecto ha sido prolongada y activa, y ha comprendido prácticamente todas las fases del mismo, desde la selección de componentes y elecciones tecnológicas previas hasta la industrialización del producto pasando por el desarrollo de esquemáticos y PCBs o el desarrollo de drivers. La diversidad de elementos tecnológicos utilizados ha reportado bastantes beneficios en forma de conocimientos y experiencia, fundamentalmente en el ámbito del desarrollo de sistemas electrónicos digitales y en el ámbito de los sistemas operativos para sistemas empujados, en particular, de GNU/Linux. El hecho de haber participado en varias fases a lo largo del proyecto confiere una visión global del mismo que ha ayudado a una mejor coordinación e integración entre las partes que componen el proyecto.

Resultado del proyecto

ACISA ostenta la propiedad industrial de todos los productos derivados de este proyecto y el primer resultado ha sido el producto comercial VisioWay OpenCounter®, destinado al contaje de vehículos y la obtención de variables de tráfico de interés.

Gracias al lanzamiento de este producto, el Ayuntamiento de Sevilla ha adjudicado a ACISA el contrato de “Nuevas Instalaciones para el Control del Flujo del Tráfico” en el que se apuesta por este tipo de tecnología en más de 100 puntos de control.

La empresa pretende ejecutar el contrato con la tecnología desarrollada en el proyecto por lo que ya han sido instaladas cuatro unidades de prueba en el cruce de Avda. Bueno Monreal con la Avda. de La Palmera. Los primeros resultados han sido satisfactorios, lo que ha propiciado que se prevea la instalación de más equipos VisioWay OpenCounter en el resto de los puntos de control indicados en el contrato.



Figura 39: Imágenes de la primera instalación en Sevilla

Ampliaciones y futuros trabajos

Este proyecto supone la apuesta de la empresa por un nuevo tipo de tecnología para desarrollar sus productos. El sistema realizado tiene carácter de plataforma sobre la que se pueden implementar aplicaciones muy diversas, lo que permite su

aprovechamiento en otros proyectos y productos.

La empresa está desarrollando en la actualidad otros productos de visión artificial aplicada al tráfico, concretamente está trabajando en la lectura automática de las placas de matrícula (ALPR). Este tipo de sistemas pueden ser usados en las ciudades para controlar el uso inadecuado de los carriles bus, el acceso a zonas restringidas, detección de infracciones, etc.

Multicast es un método de envío de datagramas IP a un grupo de receptores interesados.

Usos de Multicast IP

Multicast IP ha tenido cierto éxito en la distribución unidireccional de streams como vídeo de alta calidad a un gran número de receptores. De hecho, muchos operadores de TV por cable en USA y algunas instituciones educativas han utilizado multicast IP precisamente para ello. De forma adicional, también ha sido utilizado para aplicaciones de videoconferencia.

Otro de los usos que ha tenido multicast dentro de las redes comerciales y de campus es la distribución de ficheros, particularmente para distribuir imágenes de sistemas operativos y actualizaciones a equipos remotos.

Aunque multicast IP ha tenido cierto éxito en cada una de estas áreas, no ha sido implantado masivamente y generalmente no está disponible como servicio para el usuario medio. Hay al menos dos factores fundamentales para la falta de implantación de multicast, y ambos están de alguna manera relacionados entre sí. Por un lado, enrutar tráfico multicast, particularmente para una comunicación bidireccional, requiere una gran complejidad en el manejo de protocolos. Por otra parte, hay un número de problemas adicionales de operatividad para gestionar una red multicast, entre ellos los problemas asociados a ataques de denegación de servicio (DoS), ya que los sistemas multicast son más vulnerables a estos ataques.

Direccionamiento

Tipos de direcciones

Existen cuatro formas de direccionamiento IP, cada una con sus propiedades particulares:

Unicast

El concepto más común de dirección IP es una dirección unicast. Normalmente se refiere a un único emisor o receptor, y puede ser usada tanto para enviar como para recibir. Generalmente, una dirección unicast está asociada con un único dispositivo o host, pero puede no ser una correspondencia directa. Algunos PCs, por ejemplo, pueden usar varias direcciones unicast a la vez, cada una para un propósito distinto. Enviar los mismos datos a múltiples direcciones unicast requiere que el emisor envíe todos los datos por cada uno de los receptores.

Difusión (Broadcast)

Enviar datos a todos los posibles destinatarios permite al emisor enviar los datos una sola vez, y todos los receptores pueden tener una copia de los mismos. En el protocolo IP, 255.255.255.255 representa una difusión limitada a la red local. Adicionalmente, una difusión limitada puede realizarse combinando el prefijo de la red con un sufijo compuesto en su totalidad por '1' binarios. Por ejemplo, para enviar a todas las direcciones dentro de una red con el prefijo 192.0.2, la dirección de la difusión limitada será 192.10.2.255.

Multicast

Una dirección multicast está asociada a un grupo de receptores interesados. De acuerdo con el RFC 3171, el rango de direcciones que va de la 224.0.0.0 a la 239.255.255.255 está reservado para direcciones multicast. Este rango era conocido anteriormente como "Clase D". El emisor envía un único datagrama (desde la dirección unicast del emisor) con destino a la dirección multicast, y los routers se encargan de hacer copias y enviarlos a todos los receptores que se hayan suscrito a esa dirección multicast.

Anycast

Al igual que broadcast y multicast, anycast es una topología de rutado punto a multipunto. Sin embargo, el flujo de datos no se transmite a todos los receptores, sólo al único que esté "más cercano" en la red. Anycast se usa para balancear cargas de datos. Es usado en DNS y UDP.

Asignación de direcciones multicast

El rango de direcciones de clase D, que está todavía asociado con grupos de direcciones multicast, no está distribuido como las direcciones unicast tradicionales. De hecho, la asignación y distribución de las direcciones multicast ha sido un problema que puede resultar en múltiples soluciones insatisfactorias.

El bloque 224.0.0.0/24 está dedicado a multicast sobre redes locales únicamente. Los datagramas dirigidos a estas direcciones nunca deben ser reenviados a otra red por un router.

Gran parte del espacio de direcciones restantes dentro de 224.0.0.0/8 ha sido asignado a determinadas aplicaciones o simplemente han sido reservadas por el IANA.

El bloque 232.0.0.0/8 está reservado para su uso por parte de SSM (Source-specific multicast).

El bloque 233.0.0.0/8 está reservado para direcciones GLOP.

El bloque 239.0.0.0/8 es un espacio de direcciones reservado para tareas administrativas. Algunos operadores han tratado este bloque entero de acuerdo con el RFC 1918. Una lectura cuidadosa del RFC 2365 muestra que sólo un subconjunto de este espacio de direcciones debería ser tratado de esta manera. Hay porciones de este espacio relativas a la región de asignación, que son muy similares al espacio de direcciones unicast de una red privada.

Protocolos y aplicaciones

Dado que multicast usa un modo de transmisión diferente de unicast, sólo los protocolos diseñados para multicast deben ser usados para este propósito.

La mayoría de los protocolos de aplicación existentes que usan multicast están encima de UDP. En muchas aplicaciones, RTP es usado para la transmisión de contenidos multimedia sobre multicast; El protocolo RSVP puede ser usado en estos casos para la reserva de ancho de banda en una distribución multicast.

En una red local, la distribución del tráfico multicast está controlada por IGMP. Dentro de un dominio de rutado se usan PIM o MOSPF. Entre dominios de rutado se

usan protocolos tales como MBGP.

Pueden ocurrir errores si algún paquete que iba dirigido a una dirección unicast accidentalmente se envía a una dirección multicast. En particular, el envío de paquetes ICMP a una dirección multicast se ha usado en el contexto de ataques DoS como una manera de lograr una amplificación de paquetes.

Rutado

Cada host (y de hecho cada aplicación dentro del host) que quiere ser un miembro receptor de un grupo multicast debe usar el protocolo IGMP (Internet Group Management Protocol) para unirse al grupo. Los routers adyacentes también usan este protocolo para comunicarse.

En el rutado unicast, cada router examina la dirección de destino de un paquete entrante y la comprueba con su tabla de enrutamiento para determinar qué interfaz usar para que el paquete llegue lo más cerca de su destino. La dirección de origen es irrelevante para el router.

Sin embargo, en el rutado multicast, la dirección del emisor es usada para determinar la dirección del flujo de datos. El tráfico hacia el emisor del flujo de datos multicast se considera como tráfico ascendente. El router determina a través de qué interfaces se canalizará el flujo descendente (downstream) para un grupo multicast, y envía el paquete por las interfaces apropiadas. El término “reverse path forwarding” es usado para describir este concepto de rutado de paquetes lejos de la fuente.

Distribución a nivel de enlace

Los paquetes unicast son distribuidos hacia un determinado receptor dentro de una subred Ethernet indicando una dirección MAC correspondiente al equipo destinatario. Los paquetes multicast son distribuidos usando una dirección MAC dentro del rango 01:00:5E:00:00:00 – 01:00:5E:7F:FF:FF. El primer octeto (01) incluye el bit broadcast/multicast. Los 23 bits menos significativos de la dirección IP multicast son mapeados en los 23 bits menos significativos de la dirección MAC. Esto significa que pueden existir ambigüedades a la hora de distribuir los paquetes. Si dos equipos de la misma subred se suscriben cada uno a un grupo multicast diferente cuyas direcciones IP difieran sólo en los bits 27 a 23, los paquetes Ethernet para ambos

grupos multicast serán distribuidos a ambos equipos, requiriendo la actuación del nivel de red para descartar los paquetes no solicitados.

Multicast, por su naturaleza, no es un mecanismo orientado a conexión, por eso los protocolos como TCP, que permiten la retransmisión de paquetes perdidos, no son apropiados. Para aplicaciones como la difusión de audio y vídeo, la pérdida ocasional de un paquete no es un problema. Pero para la distribución de datos críticos, es necesario un mecanismo de petición de retransmisión.

En estos casos, es interesante utilizar PGM (Pragmatic General Multicast), documentado en la RFC 3208. En este escenario, los paquetes multicast tienen números de secuencia y cuando un paquete se pierde, un receptor puede pedir la retransmisión usando una conexión unicast convencional.

Protocolos usados en multicast IP

A continuación se citan algunos de los protocolos usados en multicast IP:

- IGMP (Internet Group Management Protocol)
- PIM (Protocol Independent Multicast)
- DVMRP (Distance Vector Multicast Routing Protocol)
- MBGP (Multicast BGP)
- MSDP (Multicast Source Discovery Protocol)
- MLD (Multicast Listener Discovery)
- GMRP (GARP Multicast Registration Protocol)
- mDNS (Multicast DNS)

- [1] Peden M. et al, The World report on traffic injury prevention, ISBN: 92-415-9131-5, 2004
- [2] C. Setchell, E. L. Dagless, Vision-based road traffic monitoring sensor, IEEE Proc. - Vis. Image Signal Process, 2001
- [3] G. Li, J. Zhang, H. Lin, D. Tu, M. Zhang, A moving object detection approach using integrated background template for smart video sensor, Instrumentation and measurement technology conference, 2004
- [4] X. Y. Wang, K. W. Zhang, X. Y. Yang, Research of the road traffic incident characteristics, 4th International conference on machine learning and cybernetics, 2001
- [5] Artemis Strategic Research Agenda, 2006
- [6] Agenda Estratégica Prometeo, 2006
- [7] B. Hori, J. Eyre, Processors for video, 2005, http://www.dspdesignline.com/howto/benchmark_product_sel/187002905
- [8] A. Dasu, S. Panchanathan, A Survey of Media Processing Approaches, IEEE Transactions on circuits and systems for video technology, Vol. 12, Nº 8, Aug 2002
- [9] A. Romeo, J. García, La Pastilla Roja, ISBN: 84-609-0213-7, 2003
- [10] M. G. Kang, Selected papers on CCD and CMOS imagers, ISBN: 0819451142, 2003
- [11] T. Ebrahimi, F. Pereira, The MPEG-4 Book, ISBN: 0-130-61621-4, 2002
- [12] ITU, Recomendación ITU-R BT.656, Unión Internacional de Telecomunicaciones,
- [13] USB-IF, On-The-Go Supplement to the USB 2.0 Specification Revision 1.3, USB Implementers Forum, 2006
- [14] A. Rubini, J. Corbet, Linux Device Drivers, ISBN: 0-596-00008-1, 2001
- [15] A. López, Aprendiendo a programar con Libpcap, 2005, <http://www.e-ghost.deusto.es/docs/2005/conferencias/pcap.pdf>
- [16] M. Paredes, M. Fleury, M. Ghanbari, Accurate packet-by-packet measurement and analysis of video streams across an Internet tight link, Signal Processing: Image Communication, 2005

Agradecimientos

A Sergio Toral Marín y Federico Barrero García por darme la oportunidad de participar en este proyecto.