ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA

PLATAFORMAS SYSTEMS-ON-CHIP OMAP PARA EL DESRROLLO RÁPIDO DE SISTEMAS EMBEBIDOS

SANDRA ELIZABETH ARGÜELLO JÁCOME

SANGOLQUÍ – ECUADOR

2011

CERTIFICACIÓN

Certificamos que el presente proyecto de grado titulado: "PLATAFORMAS SYSTEMS-ON-CHIP OMAP PARA EL DESRROLLO RÁPIDO DE SISTEMAS EMBEBIDOS", ha sido desarrollado en su totalidad por la señorita SANDRA ELIZABETH ARGÜELLO JÁCOME con CC: 1714559802, bajo nuestra dirección.

Atentamente

_____

Ing. Byron Roberto Navas Viera

DIRECTOR

_____

Ing. Julio Cesar Larco Bravo

CODIRECTOR

RESUMEN

El presente proyecto tiene como objetivo el diseño y desarrollo de un Sistema de Referencia empleando una Plataforma SoC (Systems-On-Chip) TI OMAP(Open Multimedia Application Platform) para el desarrollo rápido y robusto de aplicaciones multimedia el cual sirva como modelo para futuras aplicaciones.

El Sistema de Referencia incluyó las metodologías para el desarrollo rápido y económico de sistemas embebidos multimedia de alto desempeño orientados a diversas aplicaciones, empleando Plataformas OMAP, interfaces de última generación y código abierto Android.

Durante el proceso fueron analizadas la arquitectura y jerarquía de hardware y software de las plataformas TI OMAP3530. Se permitió evaluar las ventajas de esta propuesta frente a otras soluciones de Sistemas Embebidos, a través de diferentes pruebas de desempeño utilizando dos plataformas diferentes con similares características. Además fue posible comparar los resultados a los obtenidos con dos configuraciones distintas de manejo de señales digitales dentro del SoC OMAP3530.

La información que acompañe a este proyecto de grado, contendrá información específica sobre la plaforma adquirida, y puede ser utilizada como modelo para futuras aplicaciones sobre la plataforma, las cuales den soporte a distintos proyectos en cooperación con otros departamentos del DEEE o líneas de investigación de la ESPE.

DEDICATORIA

A mi padre, quien es mi inspiración para ser una mejor persona, a quien entre juegos despertó mi gusto por la electrónica y a quien siempre recuerdo con el más cálido sentimiento…

A mi familia, en especial a mis sobrinos quienes alegran todos los días de mi vida. A mi madre por su fortaleza, y porque con su ejemplo me inspira a seguir adelante siempre. A mis hermanos y cuñados por su apoyo incondicional y por sus buenos consejos.

A Dios, por sus bendiciones, pues sin su guía no seríamos capaces de seguir adelante y gracias a él es que he logrado conquistar cada etapa de mi vida.

Sandra Elizabeth Argüello Jácome

AGRADECIMIENTO

Mi más sincero agradecimiento hacia mis profesores Ing. Byron Navas, Ing. Julio Larco e Ing. Gonzalo Olmedo, PH.D., ya que sin su apoyo y dedicación este proyecto no hubiera podido salir adelante. Pese a las complicaciones y a la distancia, me supieron apoyar y colaborar de la mejor manera, siempre buscando ser entes facilitadores y enriquecido mis conocimientos día a día.

De manera especial, quisiera agradecer a mi hermano Ricardo Argüello, por su guía, por su apoyo incondicional y por siempre buscar un tiempo para ayudarme en lo que necesite. Su colaboración fue muy importante en el desarrollo de este proyecto.

Sandra Elizabeth Argüello Jácome

PROLOGO

La industria de los semiconductores se encuentra en constante cambio, busca la creación de dispositivos que cumplan con la tendencia actual en cuanto a tamaño, prestaciones, ahorro de energía y funcionalidad. Esta ha conducido a un diseño electrónico con tecnología que requiere disminución en tamaño mientras presenta mejoras en cuanto a tiempo de lanzamiento al mercado, eficacia, y diseño con menores fallas. Es así como surgieron los System-on-Chip, para llenar dichos requerimientos. Los System-on-Chip basan su diseño en la reutilización de bloques de Propiedad Intelectual y plataformas. Esta va con la tendencia en la industria, la cual busca usar una plataforma común que cubra un gran conjunto de funcionalidades.

Tradicionalmente en un país en desarrollo como el Ecuador y en el Departamento de Eléctrica y Electrónica DEEE, básicamente se compra tecnología ya desarrollada y se la conFigura para aplicaciones particulares. Diseñar todo un sistema embebido con microcontroladores, DSP (Digital Signal Processors), memorias, etc; ensamblarlos en una placa de circuito impreso; y luego diseñar la estructura de software, requiere tiempo. A pesar de que el prototipo final es funcional; la calidad, confiabilidad, y costo impide su comercialización. Se puede comprender muy bien los conceptos que están detrás de tecnologías avanzadas pero el no poder aplicar efectivamente estos conocimientos para diseñar productos propios es una limitación. La globalización y crisis mundial obligan a ser desarrolladores de tecnología electrónica y cambiar los modelos de negocios para poder sobrevivir.

Las estimaciones sobre Android para uso en smartphones han roto records, las ventajas del uso de plataformas SoC y sistemas abiertos como Linux en dispositivos embebidos indican que este es el camino a seguir en el futuro. Existen muchos teléfonos desarrollados sobre la plataforma OMAP y se conoce también, que existe la posibilidad de obtener tarjetas de desarrollo que contemplan mayores capacidades de desarrollo de aplicaciones multimedia a precios accesibles en el mercado. Por lo tanto, desarrollar dicha tecnología presenta mayores ventajas, tanto en la parte académica como en la investigación aplicada y científica de la ESPE, liderando el cambio en el país.

Cabe recalcar que no existen trabajos en el empleo de Plataformas SoC en el país y probablemente en varios países de la región, por esta razón se pretende que mediante este proyecto se presente una metodología de diseño para el desarrollo rápido de sistemas embebidos, realizando análisis y experimentos sobre una plataforma SoC, a fin de que se genere un sistema de referencia, el cual consiste en documentación, código y tutoriales para dar soporte a futuros trabajos que permitan desarrollar tecnología y no solo consumirla.

La documentación que acompaña a este proyecto incluye información acerca del Estado del Arte de las plataformas TI OMAP, es decir su evolución, historia, análisis de su arquitectura, jerarquía de hardware y software; además se presenta un estudio comparativo entre las diferentes tarjetas de plataforma OMAP existentes en el mercado, con lo cual fue posible adquirir la plataforma más conveniente y con mejores características. Se especificaron las características técnicas de la plataforma escogida y posteriormente fue adquirida por el DEEE. Se analizó su funcionamiento, arquitectura y sistemas de desarrollo.

Por otra parte, se incluye un Sistema de Referencia, el cual será la base para futuras aplicaciones sobre la plataforma adquirida. Se ha planteado un procedimiento para su configuración, su correcta utilización, instalación de herramientas para su uso, y se incluye también un tutorial de desarrollo de aplicaciones software basadas en el Sistema Operativo Android. En conclusión se consiguió describir una metodología de diseño para el desarrollo de aplicaciones multimedia en la plataforma. Toda esta experiencia permitirá evaluar las ventajas de esta propuesta frente a otras soluciones para el desarrollo rápido y confiable de Sistemas Embebidos y servirá como modelo para otras aplicaciones que den soporte a proyectos en cooperación con otros departamentos o líneas de investigación.

Finalmente se prosiguió a evaluar las ventajas de esta experiencia frente a otras soluciones para el desarrollo de sistemas embebidos y Sistem-on-Chip. Permitiendo un análisis cualitativo, comparativo y de desempeño sobre las aplicaciones multimedia realizadas, enfocandoce principalmente en el desempeño multimedia y procesamiento gráfico.

El desarrollo del proyecto se encuentra organizado por capítulos, de la siguiente manera. En el Primer Capítulo se recopilará la información necesaria para desarrollar el tema de tesis. Primero se realizará una pequeña investigación sobre los precedentes del tema, estos son la evolución del chip, la ley de Moore, y la Crisis de Productividad. Se investigará acerca de SoC, profundizando conceptos, estudiando su arquitectura y metodologías de diseño. Finalmente se describirá el concepto de plataforma y su arquitectura. En el Segundo Capítulo se presenta un estudio de la plataforma Omap TI, se analizarán sus características más sobresalientes, de esta manera se seleccionará cual es la familia más adecuada. A continuación se analizará las tarjetas de desarrollo que contengan la plataforma escogida, más representativas del mercado, y se procederá a la selección de la tarjeta que será adquirida para el desarrollo del presente proyecto. El Tercer Capítulo contempla una descripción detallada de la arquitectura de Hardware y Software de la Plataforma de procesamiento SoC TI OMAP 3530 y Plataforma de Aplicaciones EVM OMAP35x. En el Cuarto Capítulo se estudiarán los Entornos de Desarrollo Integrado (IDE) y System Development Kit (SDK), soportados para la tarjeta de desarrollo EVM OMAP3530. Además se analizarán las metodologías para la implementación del Sistema Operativo Android sobre dicha tarjeta. Finalmente se desarrollarán los tutoriales de los procedimientos realizados. En el Quinto Capítulo detalla las especificaciones necesarias para el diseño y desarrollo de aplicaciones Android sobre la tarjeta EVM OMAP35x, a través de un sistema de referencia, mediante metodologías de desarrollo a través de implementaciones básicas sobre el sistema operativo implantado, Android. El Sexto Capítulo contempla un análisis sobre las aplicaciones realizadas, enfocadolo principalmente al desempeño multimedia y procesamiento gráfico. El desempeño de la plataforma EVM OMAP3530 será comparado entre dos dispositivos de similares características. La información recolectada servirá de guía para futuras aplicaciones. Se concluye con el desarrollo de las respectivas Conclusiones y Recomendaciones del proyecto total, en el SéptimoCapítulo.

INDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

GDSII (Graphic Data System): es un formato de fichero de datos, que desde hace años ha sido en la industria de los circuitos integrados el estándar de facto para el intercambio de datos entre herramientas de diseño físico de chips.

Lenguaje de descripción de hardware (HDL, *Hardware Description Language*): permite documentar las interconexiones y el comportamiento de un circuito electrónico, sin utilizar diagramas esquemáticos.

Latencia: Es el tiempo o lapso necesario para que un paquete de información se transfiera de un lugar a otro.

H264 o MPEG-4 parte 10c: capaz de proporcionar una buena calidad de imagen con tasas binarias notablemente inferiores a los estándares previos (MPEG-2, H.263 o MPEG-4 parte 2), además de no incrementar la complejidad de su diseño.

H.263: es un estándar de la Unión Internacional de Telecomunicaciones (ITU) para la codificación de vídeos con compresión que data de 1995/1996. H.263 describe un Codec, que en primera línea se concibió para videoconferencias. Está optimizado para una tasa de bits baja (bajo 64 kbit/s, es decir, velocidad ISDN) y un movimiento relativo reducido. Si bien, el propio estándar no define una tasa de bits concreta.

AAC (*Advanced Audio Coding*): es un formato informático de señal digital audio basado en el algoritmo de compresión con pérdida, un proceso por el que se eliminan algunos de los datos de audio para poder obtener el mayor grado de compresión posible, resultando en un archivo de salida que suena lo más parecido posible al original.

3GP: es un formato contenedor usado por teléfonos móviles para almacenar información de medios múltiples (audio y video). Este formato de archivo, creado por 3GPP (3rd Generation Partnership Project), es una versión simplificada del "ISO 14496-1

Media Format", que es similar al formato de Quicktime. 3GP guarda video como MPEG-4 o H.263. El audio es almacenado en los formatos AMR-NB o AAC-LC.

MPEG-4: es una serie de códecs y estándares internacionales de vídeo, audio y datos creado especialmente para la web. Está formado por una serie algoritmos de compresión que codifica datos, audio, y vídeo optimizando su calidad de almacenamiento, codificación y distribución en redes. Con las cámaras de hoy, se integra, captura y codifica en una sola acción, lo que optimiza la potencialidad del usuario para emitir.

APIs (Application Programmng Interfaces): es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Framework: es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Data Stream: Una secuencia de señales codificadas digitalmente para representar la información en la transmisión.

Renderizado: es un término utilizado para referirse al proceso de generar una imagen desde un modelo. Este término técnico es utilizado por los animadores o productores audiovisuales y en programas de diseño en 3D.

Widget: En informática, un widget es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de widgets o Widget Engine. Entre sus objetivos están dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

CAPÍTULO 1

ESTADO DEL ARTE

Resumen: En el siguiente capítulo se recopilará la información necesaria para desarrollar el tema de tesis. Primero se realizará una pequeña investigación sobre los precedentes del tema, estos son la evolución del chip, la ley de Moore, y la Crisis de Productividad. Se investigará acerca de SoC, profundizando conceptos, estudiando su arquitectura y metodologías de diseño. Finalmente se describirá el concepto de plataforma y su arquitectura.

1.1. INTRODUCCIÓN

1.1.1. Evolución de los Circuitos Integrados

El circuito integrado (CI), también llamado chip es uno de los inventos más importantes en la evolución de la tecnología ya que son utilizados en casi todos los equipos electrónicos existentes en la actualidad. Se lo puede definir como un "Elemento compacto fabricado en una pequeñísima placa de silicio y que cumple las funciones de un circuito electrónico completo, más o menos complejo, en el cual intervienen componentes activos y pasivos"[1]. La idea de los circuitos integrados es hacer que todos los elementos electrónicos como capacitores, resistencias, transistores, diodos y todas sus conexiones se encuentren dentro de un material semiconductor.

El primer circuito integrado, ver Figura. 1.1[2], fue desarrollado en 1958 por Jack Killby[2] quien trabajaba para la Texas Instruments, descubrió que los elementos electrónicos podían ser constituidos del mismo material y que se los podía incluir en un solo circuito, desarrollando así un oscilador, el cual constaba de seis transistores integrados en una sola base semiconductora. El diseño de Killby tuvo algunos inconvenientes ya que

los elementos se interconectaban mediante cables de oro y esto dificultaba tener una mayor complejidad de diseño. A finales de 1958 fue desarrollado el transistor planar, por Jean Hoerni[2], de Fairchild. Para su fabricación se utilizaron técnicas de fotograbado y el material era sometido a determinados gases. Las ventajas que surgían de estos transistores eran su fiabilidad y sus reducidos costos de fabricación.

Figura. 1.1. Primer Circuito Integrado

El transistor planar fue desarrollado casi al mismo tiempo en que surgió el circuito integrado, su invención permitió el desarrollo del primer circuito integrado planar. El circuito planar fue creado en 1961 por Robert Noyce, quien también pertenecía a Fairchild[3]. Noyce adapto la técnica de fabricación en transistores planares a la producción de circuitos integrados superando los inconvenientes del circuito integrado original y llevándolos a su comercialización. En 1963, Fairchild fabricó un dispositivo llamado 907[4], el primer circuito integrado producido en masa, contenía dos compuertas lógicas, las que integraban cuatro transistores y cuatro resistores.

Este fue el comienzo de la microelectrónica, el siguiente paso en la evolución de los circuitos integrados fue su miniaturización. A partir de la invención del circuito electrónico el número de componentes que podían ser incorporados en un mismo chip incrementaba, pero su tamaño disminuía, de acuerdo al número de elementos activos al circuito se fijaron diferentes niveles de integración

SSI (Small Scale Integration) pequeña escala de integración, de 10 a 100 transistores, estos fueron los primeros circuitos integrados. MSI (Medium Scale Integration) media escala de integración, de 101 a 1.000 transistores, a los mediados de los sesenta. LSI (Large Scale Integration) gran escala de integración, de 1.001 a 10.000 transistores, en la

época de los setentas[2]. Para principios de los ochentas y hasta la actualidad se considera la tecnología VLSI (Very Large Scale Integration) integración en una escala muy grande.

En 1969 fue desarrollado, por Intel, el primer microprocesador llamado 4004, su lanzamiento fue en 1971[5]. El 4004 estaba compuesto de 2.300 transistores, con tecnología de 10 micrones (la tecnología suele expresarse en términos de dimensión crítica, que se refiere al espaciamiento entre cada elemento electrónico o línea de conexión del cual está conformado el chip), una transferencia de 4 bits a la vez, una memoria máxima de 640 bytes y un reloj de 108KHz de velocidad[6].

En un principio el microprocesador fue diseñado para utilizarlo en calculadoras (4004), pero gracias a su facilidad de programación se la empleó en diferentes aplicaciones, desde semáforos hasta analizadores de sangre, e incluso fue utilizado por la NASA. Se puede decir que este fue el punto de partida para difundir la tecnología y generalizar los sistemas digitales en sistemas complejos basados en el microprocesador.

1.1.2. Ley de Moore

La Ley de Moore plantea que "El número de componentes que pueden ser colocados en un circuito integrado se duplican aproximadamente cada dos años"[7], dicha afirmación se ha convertido en principio guía para la industria de los semiconductores proponiendo una tendencia tecnológica que rige hasta la actualidad.

Esta ley experimental fue descrita por Gordon Moore, Director de Investigación y Desarrollo de la compañía Fairdchild Semiconductor, el 19 de abril de 1965, en un artículo de la revista Electronics Magazine titulado *"Cramming more components onto integrated circuits*"[8]. La idea del documento se baso principalmente en asegurar que la electrónica tenía futuro, en concreto se habló de cómo evolucionarían los circuitos integrados apoyándose en su comportamiento y evolución a la fecha.

Para escribir el artículo, Moore simplemente observo la tendencia de integración de los chips para ese entonces y predijo empíricamente que la complejidad de los componentes de un circuito integrado se incrementa a un ritmo de aproximadamente un factor de dos por año[9]. En el año de 1975, en una reunión de la IEEE (Institue of

Electrical and Electronic Engineering), Moore actualizó su teoría original, afirmando que la capacidad de integración se duplicaría aproximadamente cada dos años. Con el tiempo la ley fue modificada, asegurando que se duplicaría cada 18 meses, esta afirmación fue planteada en los años 80 por Intel, teniendo en cuenta que además del número de transistores, existía un incremento en la frecuencia del reloj[10].

La ley de Moore ha trascendido en el tiempo. Cuando fue formulada, Moore predijo que esta aproximación podría ser constante por al menos 10 años, pero ha regido el diseño integrado hasta la actualidad. Es muy importante apreciar el alcance que ha tenido la Ley de Moore hacia diferentes aspectos tecnológicos, como es el caso de la capacidad de las memorias o ancho de banda, ya que se han desarrollado a la par que los microprocesadores. La evolución de la Ley de Moore se la puede apreciar en el siguiente gráfico (Figura. 1.2[11]).

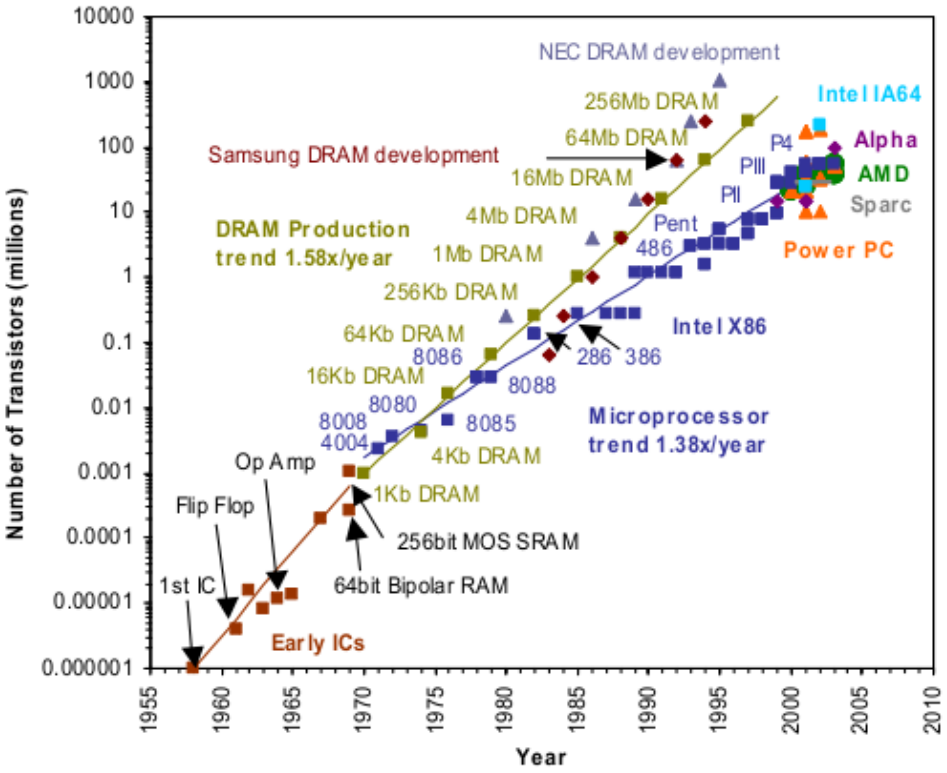


Figura. 1.2. Evolución de la Ley de Moore

El alcance de la ley fue incluso hacia la economía ya que la consecuencia directa de la Ley de Moore se ve reflejada en la disminución del costo de fabricación de circuitos integrados, al mismo tiempo que sus prestaciones suben. Mientras la tecnología basada en

silicio aumenta en rendimiento, se vuelve exponencialmente más barato fabricarla [10]. Podemos ver que la tecnología de hoy es obsoleta el día de mañana, como es el caso de los computadores, que una vez que son adquiridos, devalúan su precio a la mitad en un año y medio, al cabo de dos años se encuentran depreciados casi en su totalidad.

En momento en que Moore predijo la tendencia, trabajaba en los laboratorios de Fairchild, pero 1969, junto con Robet Noyce, fundó la compañía Intel, la cual es fabricante mundial de microprocesadores y es una de las empresas tecnológicas de investigación, desarrollo y fabricación con más trascendencia[8]. Intel y otras compañías importantes como AMD, IBM, XILINX, ALTERA, ARM, buscan estar en el mercado y competir con lo mejor en tecnología, lo que significa evolucionar a la par con la Ley de Moore, y esto no es fácil. Si un fabricante no evoluciona al ritmo de esta tendencia, simplemente sus productos no podrán ser vendidos.

El mercado semiconductor tiene un crecimiento acelerado, sin embargo la Ley de Moore puede expirar. Las afirmaciones del fin de esta teoría han existido por muchos años pero esto no ha sucedido todavía. Se debe considerar que la Ley de Moore no es una ley natural, es una ley formulada a consecuencia de la observación que se ha ido cumpliendo a través del tiempo y dejará de cumplirse en el momento en que sea imposible seguir reduciendo el tamaño de los transistores o cuando las inversiones en el equipamiento que sea necesario para su construcción sea tan alto que no justifiquen su producción. El primer microprocesador tenía 2300 transistores y ahora, se tienen procesadores con 2.3 billones de transistores[12], así que parece ser que la Ley de Moore se cumplirá por largo tiempo, gracias a nuevas tecnologías y avances tecnológicos.

1.1.3. Crisis de Productividad

El diseño de circuitos integrados complejos se ha visto sujeto a la Ley de Moore por casi cincuenta años, impulsando un increíble progreso tecnológico y económico. Como resultado, los semiconductores reducen su tamaño, mientras brindan mayor capacidad y mejor rendimiento, a un menor costo.

Figura. 1.3. Evolución de la Tecnología

Desafíos Tecnológicos para el diseño de Circuitos Integrados

Estar a la par con la tendencia, representa significativos desafíos para los diseñadores, ya que la tecnología de integración ha avanzado (Ver. Figura. 1.3[13, 14]) hasta llegar a dimensiones que son inferiores a 0.25 micrones, lo que es conocido como tecnología DSM (Deep Sub-micron). En la actualidad, las industrias ya dominan tecnologias de 32 nanometros. Estos desafíos pueden ser agrupados en tres categorías[14]: Timing Closure, Capacidad y Propiedades Físicas, los cuales van a ser descritos a continuación.

- **Timinig Closure**

Los flujos de diseño tradicionales han utilizado modelos estadísticos para estimar la longitudes de canal de los circuitos integrados, prediciendo las interconexiones del metal para un análisis de tiempos de interconexión dentro del circuito previos al diseño (pre – layout)[14]. A nivel de la tecnología DSM, los retrasos de interconexión son muy significativos y deben ser determinados correctamente para lograr el Timing Closure. Los modelos estadísticos para estimar la carga de alambre son inexactos porque ellos representan un valor estadístico basado en el tamaño de un bloque y el número de compuertas, como consecuencia se obtienen valores inexactos[14].

- **Capacidad**

Con la tecnología DSM, es posible integrar 10M compuertas en un solo circuito integrado utilizando una tecnología de 0.15 um y menor, esto significa un gran desafío en capacidad para muchas de las herramientas en el flujo de diseño[14].

- **Propiedades Físicas**

La evolución de la complejidad de los componentes y los altos niveles de integración tiene efectos físicos negativos, como consecuencia se tienen dispositivos geométricamente más finos, interconexiones que utilizan más capas metálicas, reducción de la tensión de alimentación, elevado aumento de componentes en los circuitos integrados, umbrales menores en los dispositivos y mayores frecuencias de reloj. Todos estos factores causan problemas con la integridad de la señal (SI, Signal Integrity) y con la integridad del diseño (DI, Design Integrity). Algunos problemas con SI incluyen crosstalk, IR drop y ground bounce. Respecto a los problemas DI, estos incluyen migración de electrones, aumento de temperatura de electrones y cables[14].

Los desafíos se agravan mientras la tecnología disminuye de tamaño y deben ser detectados antes de ser lanzados al mercado. Si algún problema es detectado, se deben ejercer procesos de corrección[14]. Los problemas deben ser corregidos dentro del flujo de diseño para buscar y corregir los problemas a tiempo, ya que, de ser detectados al final del proceso, puede ser que se encuentren millones de problemas y sea imposible su reparación.

Estos desafíos descritos junto con otros factores han contribuido al aparecimiento de la "Brecha de Diseño".

Brecha de Diseño

La Brecha de Diseño es la diferencia que se genera entre lo que la industria de la electrónica es capaz de construir, y lo que los equipos de diseño son capaces de diseñar en un periodo de tiempo razonable[15]. Actualmente existe tecnología que permite la fabricación de sistemas complejos, pero esto implica un costo muy elevado, en relación a la productividad de los diseñadores, para su desarrollo.

En la Figura. 1.4[16], se muestra la brecha de diseño, en ella se observa la diferencia entre la complejidad del diseño y la productividad de los diseñadores, se indica la complejidad en función de transistores incluidos en un chip y la productividad en función al número de diseñadores participes del diseño. Se debe prestar atención a que la densidad máxima de integración tiene un crecimiento de casi un 60 % por año, pero la productividad de diseño solo crece en un 21% por año, esto implica que existe una creciente brecha de productividad, la cual empeora debido a un time-to-market que disminuye.

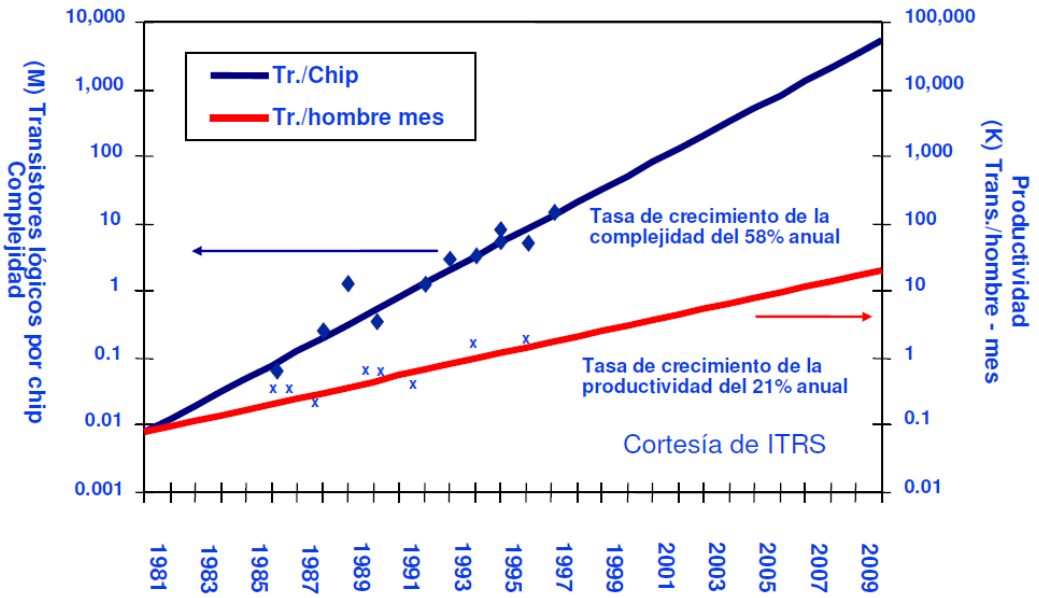


Figura. 1.4. Crisis de Productividad

El incremento en la complejidad de los circuitos integrados presenta grandes desafíos a los grupos de diseñadores encargados en la verificación y diseño de sistemas. El problema de la Brecha de Diseño no puede ser solucionado simplemente disponiendo de más ingenieros, ya que no existen muchos ingenieros calificados para solventar tales problemas, e incluso si existieran, no es conveniente trabajar con un grupo grande de diseñadores[14]. Si los grupos de diseño crecen, el nivel de coordinación requerido para que todos se encuentren en sincronía, también debería crecer.

Tiempo de Puesta en el Mercado (TTM, Time-to-Market)

El mercado demanda complejidad en los diseños electrónicos, así como también una reducción en el tiempo de puesta en el mercado de un producto. El tiempo de puesta en el mercado se relaciona a los ciclos de diseño en circuitos integrados y la tendencia indica que el tiempo se reduce[15]. En la actualidad, los productos electrónicos tienen tiempos de vida muy cortos, si existe un retraso en el TTM puede significar la pérdida de millones de dólares[15]. Esta tendencia es mostrada en la siguiente Figura. 1.5 [17].

Se observa también que las aplicaciones tradicionales como las militares e industriales son desplazadas por las de los consumidores, que requieren ciclos de diseño más cortos. Los desafíos de tecnología y mercado tienen un gran impacto en las

metodologías de verificación. Se estima que aproximadamente entre el 40 al 70 por ciento de los esfuerzos en el flujo de diseño es consumido en la verificación de tareas[14].

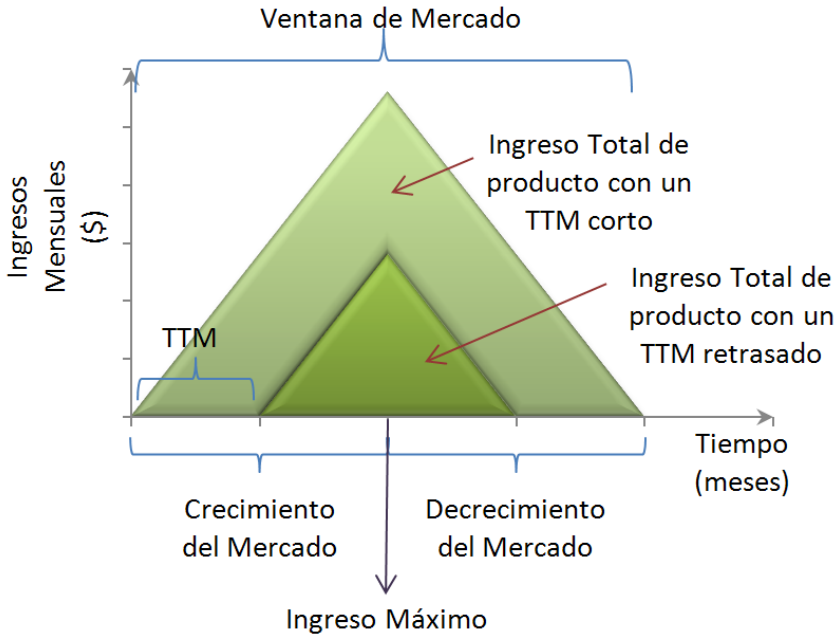


Figura. 1.5. Tendencia del Time-to-Market

1.2. SYSTEM-ON-CHIP (SOC)

La Brecha de diseño y el Time-to-Market, conllevó a una crisis en el diseño de circuitos integrados a mediados de los noventas[15], cuando la tecnología alcanzaba el punto de DSM y se podían colocar la mayoría de los elementos de procesamiento de un sistema completo, en un solo circuito integrado. Dichos sistemas se encontraban integrados en una placa de cableado impreso con múltiples encapsulados (System-on-Board, SoB)[18]. Se puede decir que estos fueron los inicios de lo que para muchos era una revolución de diseño de circuitos integrados, una revolución System-on-Chip [15], la cual fue posible gracias al avance de la tecnología permitiendo la integración de la mayoría de componentes y subsistemas de un producto electrónico completo sobre un solo circuito integrado y no solo eso, sino que además abarca muchas más ventajas que incluyen mejor rendimiento, menor consumo de energía, menor costo, tamaño, TTM, tiempo de desarrollo y verificación. La tendencia de diseño System-on-Chip, ver Figura. 1.6 [17], nace en medio de una crisis y junto con la aplicación de estrategias de reutilización de componentes que persigue una solución para los problemas en el desarrollo de CI.

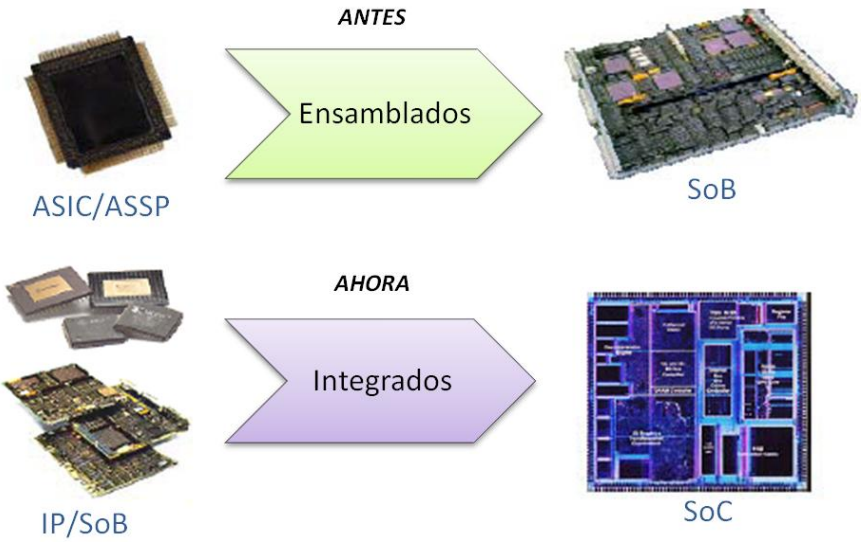


Figura. 1.6. Evolución hacia el diseño SoC

1.2.1. Evolución de SoC

Para contrarrestar los desafíos de diseño de CI, ya sea por su complejidad, brecha de diseño y tiempos de mercado, se iniciaron estrategias de como el Diseno y Reutilizacion de Bloques-de-Propiedad-Intelectual (IP-Core Design & Reuse). La estrategia con IP-Cores se basa en la reutilización de bloques preexistentes, de esta manera se reduce la cantidad de trabajo requerido en un nuevo diseño[15]. La estrategia de reutilización trae grandes ventajas simplificando la cantidad de trabajo en el diseño y además reduciendo los esfuerzos en la verificación del diseño final.

Otra herramienta es el diseño basado en plataformas, mediante el cual se puede ir más allá que la reutilización de bloques. El diseño basado en plataformas incorpora un conjunto de elementos básicos comunes para una familia de productos los cuales son identificados, integrados y verificados como una sola entidad[19]. Luego se crean nuevos elementos diseñados individualmente que se los agregan a la entidad. Los nuevos elementos pueden ser bloques diseñados desde cero o IP-Cores.

A continuación se presentará un poco de la historia de la revolución de los System-on-Chip, se dará un enfoque a lo que fueron los primeros pasos, éxitos y fracasos de la utilización de propiedad intelectual, el camino no fue del todo ideal. Su evolución puede separarse en cuatro fases.

La primera fase [15] (1995-1999) fue llamada la fase de esperanza, debido a las promesas que se tenía con la reusabilidad. Se caracteriza por ser el inicio de la tendencia y la base de lo que hoy es el diseño de SoC, se deben destacar los acontecimientos clave que trazaron el camino[15]:

- Establecimiento de la Alianza VSIA (Virtual Socket Interface Alliance, VSIA) en 1996, como una organización industrial, la cual se enfocaba en la definición de IP para el diseño de SoC. Fue formada por diferentes compañías de semiconductores, de sistemas y la industria de EDA (Electronic Design Automation). Se encargó de diferenciar las diferentes clases de IP, separándolas en Soft, Firm y Hard IP, sus resultados fueron descritos en un documento llamado "Reuse Architechture Document", de las cuales se hablará más adelante. Mediante VSIA diferentes grupos económicos combinaron sus destrezas y conocimientos para acelerar la revolución de SoC y en el camino fueron los más recompensados, pues definieron las bases de un mejor diseño de CI.
- La creación de metodologías de reuso "Reuse Metodology Manual (RMM)", en colaboración de Mentor Graphics y Synopsys en 1997. en el cual se desarrollaba y describía en detalle una serie de consejeros y reglas para la creación y reuso efectivos de bloques IP tipo Soft.
- La creación del Proyecto Alba en 1997-1999 en Escocia, el cual fue creado a por el gobierno para el desarrollo económico creando tecnología, esto se logró mediante la movilización de recursos del gobierno, universidad e industria enfocando sus esfuerzos en una acción colectiva de cambio a través del diseño de SoC basado en IP.

La una segunda fase [15] (1999-2001), fue la llamada fase de realidad y decepción, en una época donde la economía decaía, debido a la burbuja del punto com, el colapso del sector de las telecomunicaciones y especulaciones del Wall Street y de algunas empresas. Se produjo desempleo y debilitación de gran parte de la industria, el diseño de circuitos integrados no fue la excepción.

El área del diseño SoC, reutilización basada en IP y el diseño electrónico en general pasó por una etapa de recesión, la cual afecto al diseño de circuitos integrados, así como

también a la manufacturación, ventas y comercialización, incluso produjo pérdidas de trabajo. Desde mediados de los noventa, varias estadísticas se produjeron alrededor del inicio y culminación del diseño de ASIC (Application-Specific Integrated Circuit) y ASSP (Application Specific Standard Product), las cuales mostraron un menor descenso, seguido por caída precipitada en el periodo de 2000-2002, pero al año 2003-2007 se ve una disminución más suave a un valor constante[15].

La industria había pasado por una profunda transformación en los últimos años, y se debieron a una combinación de tres factores: un colapso económico, la transformación de los estilos de diseño, y a un incremento elevado en el Non-Recurring Engineering (NRE) (costo de investigación, desarrollo, diseño y pruebas de un nuevo producto[20]) en diseños ASIC y ASSP. Al momento, la electrónica posibilita la posibilidad de incluir múltiples ASIC´s combinados en un mismo SoC, o en un chipset altamente integrado o System-in-Package (SiP). Podemos decir que parte de la recesión fue debido a la transición hacia diseños SoC altamente integrados, los cuales eran demandados por la economía y rendimientos requeridos a una tecnología de 180nm y 130 nm.

La migración hacia el diseño SoC tuvo sus desatinos, empezando por comentarios que surgieron alrededor del diseño SoC y de la reutilización IP, haciéndolas pasar como conceptos que fracasaban. Se decía que si bien en el pasado se hicieron varias promesas en cuanto al concepto que representaban, la utilización apresurada de IP sin planificación cuidadosa en infraestructura, metodologías, herramientas y flujos, no la hacían muy efectiva.

Como resultado, muchas personas se desilusionaban de IP, experimentando por primera vez con los problemas de hacer la reutilización una realidad. Las estimaciones acerca de la importancia de la reutilización IP y las oportunidades de negocio relacionadas a este decayeron, los ingresos con IP no fueron tan altos como lo esperado, un gran número de compañías nuevas tuvieron que cerrar[21], algunas personas incluso declararon que el negocio de IP no iba a existir por más tiempo.

Sin embargo, al mismo tiempo el negocio de IP-Cores, se convirtió en una realidad para algunas compañías, quienes tenían los conceptos apropiados y realizaron el trabajo pesado, lo cual es indispensable para comenzar como un vendedor IP. Algunos ejemplos de

esto son ARM y SICAN[21], ambos son desarrolladores de productos IP exitosos y además prestan servicio de diseño.

La tercera fase [15] (2000-2002), muestra un cambio de actitud y de enfoque respecto al diseño basado en IP en un gran número de empresas. Ese cambio incluye una evolución hacia Diseño basado en Plataformas para SoC, este concepto ha evolucionado desde sus inicios para ser la base del diseño de Circuitos integrados en la actualidad. Se basa en un método por el cual se puede reducir el tiempo y riegos involucrados en el diseño y verificación de un SoC complejo, mediante la combinación de la reutilización IP tanto de Hardware (HW) como de Software (SW), integra grupo de componentes IP en una plataforma reutilizable[15].

Cuarta fase, en la actualidad se utiliza el diseño de SoC como una herramienta importante para el desarrollo de circuitos integrados, se encuentra basando en dos conceptos que hacen posible su éxito, estos son: la reusabilidad, y diseño basado en plataformas. El desarrollo de SoC ha sido muy bien acogido por los diseñadores de CI, pues con él se consigue un mejor rendimiento, permite gran capacidad de integración, con menor consumo de energía y tamaño, a un mejor costo.

1.2.2. Estructura SoC

Los SoC son sistemas embebidos donde todos sus componentes se encuentran integrados de dentro de un CI complejo. Los SoC se encuentran integrados por módulos pre diseñados de funciones complejas de IP que ofrecen gran variedad de aplicaciones.

El diseño y uso de SoC implica estructuración de hardware así como de software. Generalmente se encuentran integrados por[15]:

- Al menos un procesador programable, a menudo se utiliza una combinación de al menos un procesador de control RISC (Reduced Instruction Set Computer) y un Procesador Digital de Señales DSP (Digital Signal Processor)
- Estructuras de comunicaciones, son buses de comunicación pueden ser buses del sistema, de periféricos o buses de alta velocidad.

- Memoria on-chip, se refiere a cualquier tipo de memoria que exista dentro del sistema, esta puede ser tipo RAM (Randon Access Memory), ROM (Read Only Memory), etc.
- Aceleradores Hardware, utilizados para aplicaciones con procesamiento de señales, ofrecen un mejor desempeño y menor consumo de energía.
- Periféricos, se utilizan para la interacción con el exterior y pueden ser de tipo análogo o digital.
- Componentes adicionales, podrían incluirse sistemas microelectromecánicos (Microelecrtomechanical-Systems, MEMS), opto/microelectrónicos (Opto/microelecrto-mechanical-Systems, O/MEMS).
- En cuanto al software se generalmente se consideran drivers, middleware, RTOS, códigos de aplicaciones, etc.

La Figura. 1.7[22] muestra la arquitectura típica de un SoC para aplicaciones, la estructura de un dispositivo SoC puede variar dependiendo del diseño y del fabricante.

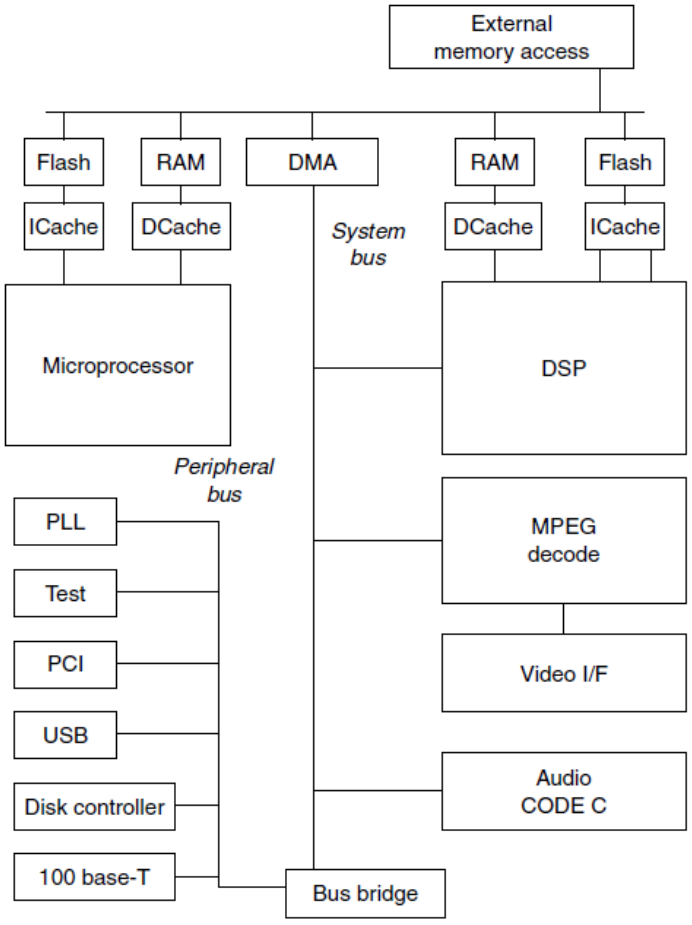


Figura. 1.7. Dispositivo System-on-Chip

1.2.3. Propiedad Intelectual

Como anteriormente se dijo, los IP-Cores nacieron de la necesidad de encontrar una metodología la cual permitiese gestionar sistemas complejos debido a la alta integración, produciendo la reducción de la complejidad y un menor tiempo de verificación. Debido a que, al ser anteriormente creados no es necesario verificarlos, lo cual permite reducir los ciclos de diseño de un sistema muy complejo.

El diseño de bloques IP requieren generalmente un mayor esfuerzo y costo. Sin embargo, debido a esta arquitectura reutilizable, una vez que el bloque IP se encuentra diseñado y verificado, su reuso en diseños futuros ahorra tiempo y esfuerzos significativos en ciclo de diseño[23]. Las IPs suelen estar respaldadas por un proveedor de tecnología, llamados Third-party o proveedores terceros, o pueden ser generados por los mismos diseñadores.

Componente Virtual

VSIA, que a lo largo de la historia ha influenciado el diseño de IPs, fue el encargado de describir todas las interfaces de diseño que un IP-Core debía satisfacer, además definió el diseño de los modelos y como debía efectuarse la integración de información, que debía ser provista junto con el IP entregado, a esto se lo conoció como *Socket Virtual[22]*. Un Socket Virtual, que más tarde fue conocido como *Componente Virtual* (Virtual Component, VC), fue concebido para que un IP-Core finalizado sea fácilmente utilizado e integrado sobre un diseño SoC.

Específicamente, en el contexto de VSIA, un Core IP representa el diseño de un bloque el cual *podría* ser reutilizable, pero un VC representa el diseño de un bloque el cual se encuentra *destinado* para ser reutilizado[22]. Entonces para un VC cumpla con un diseño calificado como altamente reutilizable, depende de un desarrollo y verificación bien establecidos que cumplan con estándares aceptados. En ese caso se observa que el diseño de VC debe tomar más tiempo, pero a la larga trae consigo mejor beneficios para con el desarrollo de SoCs.

Categorías de IP-Cores

Existen categorías IP-Cores (Figura. 1.8[23]), las cuales proveen diferentes niveles de flexibilidad para reutilización y optimización, se clasifican en tres tipos de categorías[23]:

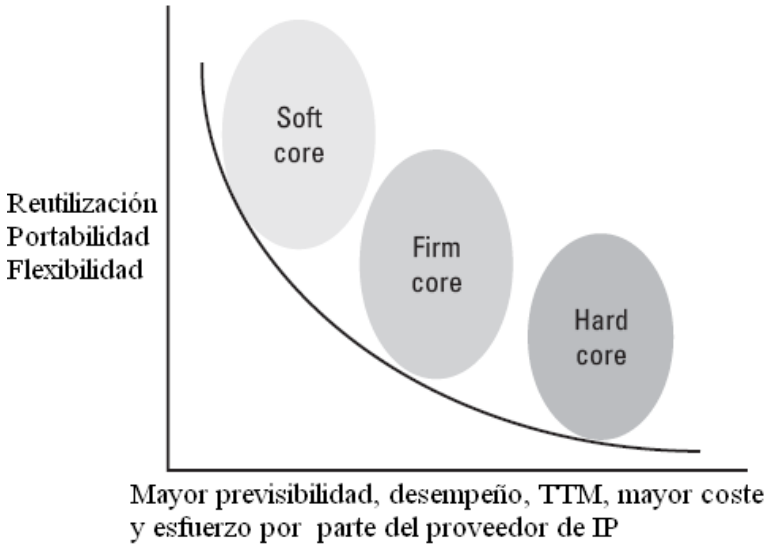


Figura. 1.8. Categorías de IP-Cores

- **Soft Core**

Los IPs son descritos en HDL (Hardware Description Language), para su descripción funcional. Este tipo de cores entregan la máxima flexibilidad y reconFiguración, ya que pueden ser personalizados para cada aplicación específica. Así del mismo modo deben ser sintetizados, optimizados y verificados por el usuario antes de integrarlo al diseño SoC completo.

- **Hard Core**

Son entregados en bloques enmascarados, formato GDSII (Graphic Data System), que incluyen layout y caracterización temporal por lo que no pueden ser modificados por el usuario. Ofrecen una implementación más optimizada de los tres cores, ya que su integración es simple, solamente se la debe incluir físicamente en el diseño SoC. Sin embargo, se caracterizan por ser dependientes de la tecnología y poseer mínima flexibilidad y portabilidad en lo que respecta a reconFiguración e integración con las diferentes tecnologías y diseños.

- **Firm Core**

Son entregados de manera de netlists sintetizadas las cuales pueden ser simuladas y cambiadas en caso de ser necesario, sin que esto signifique la realización de un nuevo layout. Provee una mezcla de las funcionalidades de soft y hard cores, por un lado trae el alto rendimiento y optimización de propiedades del Hard Core y por otro incluye la flexibilidad dada por el Soft Core.

La Tabla. 1.1[23] expuesta a continuación muestra la comparación de las características más importantes de los diferentes formatos de IP:

Tabla. 1.1. Comparación de formatos de IP-Cores

Formato IP	Representación	Optimización	Tecnología	Reusabilidad
Soft Core	RTL	Baja	Independiente de la tecnología	Muy Alta
Firm Core	Netlist	Alta	Tecnología Genérica	Alta
Hard Core	GDSII	Muy Alta	Dependiente de la tecnología	Baja

1.2.4. Metodologías de Diseño de CI

Cada avance de tecnología es siempre seguido por el desarrollo de una nueva tecnología de diseño, la cual lidera la próxima generación de metodologías de diseños. La metodología de diseño responde con una adaptación al nuevo proceso de diseño lo que resulta en un incremento gradual de productividad[19]. De esta forma se describen metodologías de diseño de manera serial, debido a que para la transición de alguna de estas primero se debe cruzar por las del medio, como se muestra en la Figura. 1.9[19].

Actualmente son utilizados tres tipos de métodos de diseño[19] Timing-Driven Desing (TDD), Block-Based Design (BBD) y Platform-Based Design (PBD), como muestra la Figura 1.9 Cada tipo puede ser seleccionado dependiendo de la tecnología a utilizar, la capacidad de diseño y el nivel de reusabilidad IP que sea invertida en la realización del diseño.

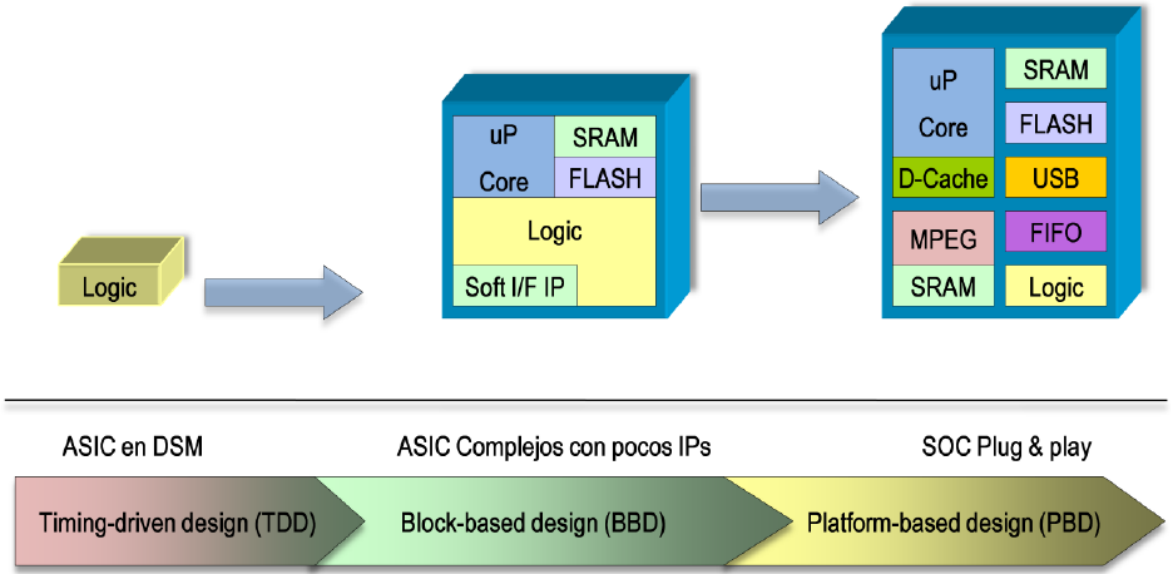


Figura. 1.9. Metodologías de Diseño de CI

Timing-Driven Desing (TDD) [19]

Se basa en la idea de considerar los aspectos de temporización en todos los niveles del proceso de diseño. TDD es utilizada para conseguir un diseño ASIC de tamaño moderado y complejo, para nuevos diseños de procesamiento DSM, y no posee una utilización significativa de diseño jerárquico. Los grupos de diseño designados para esta metodología tienen a ser pequeños y homogéneos.

TDD limita iteraciones debidas a las discrepancias de temporización, de manera que se pueda comprender fácilmente la transferencia del netlist. Pero aun así, los productos DSM de alta densidad y rendimiento requieren bucles iterativos.

El cálculo de retardos y análisis de tiempos son bastante precisos, y son muy bien soportados por las herramientas para planificación del diseño (*floor-planing tools*) y para análisis de tiempos (*Static-timing tools*) existentes. Este método crea soluciones efectivas, pero toma mucho tiempo en diseñar lo cual no era la mejor solución para sistemas más grandes.

Block-Based Design (BBD) [19]

Esta metodología es utilizada para sistemas con mayor complejidad, para diseño de ASICs de tamaño medio con complejidad. En comparación con TDD, BBD implementa reutilización oportunista, pero no genera componentes reutilizables. Se introduce el concepto de creación de un sistema mediante la integración de bloques de funciones de sistema pre-diseñados, a otras más complejas. Se caracteriza por soportar muchos grupos de trabajo complejos e incorporar diseño jerárquico. Las funciones de reutilización oportunista son pobremente caracterizadas, sujetas a modificaciones y requieren pre-verificación. BDD se encuentra modelado al nivel de sistema donde simuladores y emuladores son utilizados para la co-verificación de hardware y software.

Platform-Based Design (PBD) [19]

Es el siguiente paso de la evolución tecnológica, incorpora las capacidades acumuladas de las metodologías de diseño TDD y BBD, además integra la reutilización y el diseño jerárquico al máximo. Una de sus principales ventajas es que mediante esta, se puede disminuir el TTM del primer producto y de esta forma ampliar las oportunidades y la velocidad de entrega de productos derivados. Esta metodología de diseño planificado que reduce los esfuerzos de diseño y verificación, evitando riesgos posibles para SoCs complejos. Utiliza por metodología de diseño, la jerarquía la cual empieza al nivel de sistema, al igual que BBD, pero incorpora un diseño mediante reutilización extenso y planificado. PBD separa el diseño en dos partes: *Block Autoring* y *System Chip Integration*.

- **Block Autoring**

Generación de bloques para que el modo de comunicación entre los diseños de destino sea muy cómoda. Manejan dos conceptos de diseño: Estandarización de interfaz, los bloques creados deben ser distribuidos entre los equipos de diseño, bajo las mismas especificaciones de interfaces y directrices de metodología de diseño; y Diseño de Sistema Virtual, para establecer las limitaciones del sistema necesarios para el diseño de bloques (fuente, opciones de prueba, Hard, Firm o Soft)

- **System-On-Chip Integration**

Se enfoca en el diseño y la verificación de la arquitectura del sistema y las interfaces entre los bloques. Inicia con la partición del sistema en torno a las funciones ya existentes a nivel de bloque y la identificación de las nuevas funciones o diferenciando las funciones requeridas. La partición se realiza a nivel del sistema junto con un análisis de rendimiento, diseño de trade-offs para hardware y software y verificación funcional.

Las características relevantes de las diferentes metodologías de diseño, se encuentran resumidas en la siguiente Tabla 1.2. [19].

Tabla. 1.2. Características de diseño

Design characteristics	*Timing-Driven Design (TDD)*	*Block-Based Design (BBD)*	*Platform-Based Design (PBD)*
Design complexity	5k-250k gates	150k – 1.5M gates	300k and greater
Design level	Register-transfer level (RTL)I	Behavioral - RTL	Architectural
Design team	Small, focused	Multidisciplinary	Multigroup
Primary design	Custom logic	Blocks in context	Interfacing in system
Design reuse	None	Opportunistic, soft, firm, hard	Planned firm, hard
Primary optimization focus	Synthesis, gate-level design	Floorplanning, block arch	Silicon-compatible system
Primary design granularity	Gates and memory	Functional cluster, cores	Virtual components (VCs)
Bus architecture	None/ custom	Custom	Standardized/multiple/specific
Test architecture	None/scan	Scan/JTAG/BIST/custom	Hierarchical, parallel, BBD
Mixed-signal	None	A/D, D/A. PLL	Functions, interfaces
Constraints/goals	Logic constraints	Block-budgeted constraints	Interface constraints
Verification level	RTL/gate	Bus functional/cycle accurate	Mixed – ISS to RTL
Hardware/Software Co-Verif	None	Hardware/Software functional	Hardware/Software interface
Partitioning focus	Synthesis limitations	Functions	Function/communication
Placement	Flat	Hierarchical	Hierarchical
Routing	Flat	Flat	Hierarchical
Timing analysis	Flat	Flat/limited hierarchical	Hierarchical
Delay calculation	Flat	Flat	Hierarchical
Physical verification	Flat	Flat/limited hierarchy	Hierarchical

1.3. PLATAFORMAS SOC

Durante los últimos años el diseño de circuitos integrados se ha concentrado en el desarrollo de SoC complejos y en la reutilización de Hard and Soft IP-Cores, mediante el diseño basado en Plataformas. De acuerdo a VISIA, plataforma puede ser definido como “un conjunto de características comunes integradas y gestionadas, en el que se puede construir un conjunto de productos o una familia de productos”[15]. Se puede decir que

plataforma SoC es una familia coordinada de estructuras de hardware y software, las cuales satisfacen las constantes que permitan la reutilización de componentes de hardware y software.

Las Plataformas se componen de arquitecturas comunes como procesadores, memorias periféricos, estructuras de buses y tecnologías de soporte como librerías IP y sus herramientas las cuales controlan las arquitecturas comunes, y en base a estas se pueden obtener nuevos sistemas.

1.3.1. Beneficios de Plataformas SoC

Como se vio anteriormente el diseño basado en plataformas no es el único método utilizado para el diseño de SoCs. La elección de diseño y métodos de integración se basan en los requerimientos de capacidades tales como, costo, rendimiento, consumo de energía, riesgos de diseño y TTM. El diseño de plataformas es un método razonable, el cual ha ganado popularidad a través de los años gracias a sus beneficios, entre los que se pueden incluir:

- El aumento de la productividad de diseño, la reducción del riesgo, la facilidad de pre-integrar Componentes Virtuales de otros dominios de diseño y la facilidad de reutilizar arquitecturas SoC creadas por otros expertos[22].
- Hacer posible el diseño rápido derivado, utilizando modificaciones de hardware y software, reduce el tiempo de diseño e incrementa los resultados. El diseño rápido derivado, permite la optimización y adaptación de productos a necesidades de aplicaciones muy específicas y a mercados muy específicos[15].
- Gracias a las plataformas se puede capturar y reutilizar las mejores arquitecturas y métodos de diseño encontradas para particulares productos y mercados[15].

1.3.2. Clases de Plataformas

Existen tres tipos de plataformas utilizadas en el mercado, plataformas para aplicaciones, plataformas reconfigurables y plataformas centradas en el procesador.

Plataformas para Aplicaciones

Permiten que se puedan crear aplicaciones sobre su Arquitectura de Hardware y Software, se pueden utilizar las funciones que se incluyen con la plataforma para crear implementaciones propias. Entregan al diseñador grupos de librerías de componentes de hardware y software, junto con algunos ejemplos que son diseños de referencia. Sin embargo este tipo de plataformas requieren gran esfuerzo de modelado que significa un riesgo de inversión. Algunas familias de este tipo de plataforma son Philips, Nexperia y TI OMAP[15]. El desarrollo sobre plataformas para aplicaciones significa realizar el software que correrá sobre la plataforma utilizando el hardware entregado por el fabricante, es decir, los periféricos incluidos y en algunos casos incluir hardware no existente en la plataforma original. Pero también significa la utilización del software que puede ser firmware o drivers incluidos con la plataforma, para desarrollar la aplicación.

Plataformas Reconfigurables

Son entregados como procesadores tipo Hard Core con lógica reconfigurable, que permite a los usuarios personalizar la plataforma en cuanto a su hardware y software. Ejemplos de este tipo de plataformas son Xilixs Platform FPGA y SOPC de Altera[15].

Plataformas Centradas en el Procesador

Este tipo de plataformas concentran su diseño en procesadores específicos, requiere una arquitectura de buses y un set básico de periféricos para su funcionamiento. Además del procesador se incluye el software específico para su control como un kernel, o un sistema operativo en tiempo real (RTOS Real Time OS). Es de tipo más genérico, lo cual significa menor esfuerzo para su desarrollo y significa un menor riesgo de inversión, pero requiere mayor trabajo para convertirlo en un producto. Ejemplos de este pueden son las familias de productos como ARM, ST Mocroelectrónics[15].

CAPÍTULO 2

PLATAFORMA TI OMAP

Resumen: En el presente capítulo se presenta un estudio de la plataforma Omap TI, se analizarán sus características más sobresalientes, de esta manera se seleccionará cual es la más adecuada. A continuación se analizará las tarjetas de desarrollo que contengan la plataforma escogida, más representativas del mercado, y se procederá a la selección de la tarjeta que será adquirida para el desarrollo del presente proyecto.

2.1. INTRODUCCIÓN

La industria de los semiconductores se encuentra en constante evolución, desarrollando nuevos métodos para el diseño de circuitos integrados, los cuales han solventado los problemas de costo y tiempo de desarrollo. System-on-Chip representa la mayor evolución en el diseño de circuitos integrados, y fue posible gracias al avance de la tecnología, que permitió la integración de la mayoría de componentes y subsistemas electrónicos en un solo CI, pero más importante que esto, es entender como su diseño se encuentra integrado como un solo sistema. SoC implementa la reutilización de bloques de propiedad intelectual, además basa su diseño en plataformas, las cuales son muy utilizadas en la actualidad pues permiten reunir un conjunto de funcionalidades en arquitecturas comunes, las cuales pueden soportar una variedad de aplicaciones.

El desarrollo de esta tesis tiene como objetivo el estudio, evaluación, y desarrollo de aplicaciones sobre la plataforma SoC desarrollada por TI (Texas Instruments), llamada OMAP (Open Multimedia Application Platform), mediante la cual es posible desarrollar aplicaciones multimedia para dispositivos de aplicación de bajo costo en el Ecuador. OMAP es una Plataforma de Aplicaciones, la cual permite que funciones multimedia sean

incluidas en Smartphones 2.5G - 3.5G y PDAs (Personal Digital Assistants). Estas funciones incluyen video, mensajería, navegación web, video conferencia, juegos, comercio móvil y otras tareas computacionalmente exigentes. Esta plataforma es sumamente importante en la actualidad ya que muchos teléfonos móviles como Nokia, Sony Ericsson, Motorola, Samsung, LG, entre otros, la utilizan. Soportan diversos Sistemas Operativos como Symbian, Android, Windows CE, Linux, además se encuentran relacionados con desarrolladores 3rd Party OMAP como ActionEngine, Bitflash, Certicom, Comverse, Hi Corporation, Ideaworks 3D, Microsoft Windows Media, Packet Video, RealNetworks, SafeNet, SpeechWorks[24].

Las plataformas OMAP integran procesadores RISC (Reduced Instruction Set Computer) de propósito general, aceleradores de hardware, memorias, periféricos, e interfaces[24], algunas familias OMAP integran una arquitectura de doble núcleo que generalmente consiste de un procesador de propósito general ARM y uno DSP. Desde su lanzamiento, la plataforma OMAP fue diseñada para optimizar el desempeño, consumo de energía y el time-to-market de los clientes desarrolladores. En él se pueden diferenciar importantes características de diseño SoC basado en plataformas, como lo es el diseño jerárquico, considerando a la arquitectura software, tan crítica como la de hardware, además implementan la reutilización IP como parte de su infraestructura.

2.2. PLATAFORMA TI OMAP

TI OMAP identifica a una plataforma como una familia de arquitecturas que utiliza un conjunto de características comunes, aplicadas a distintos productos [25]. Una familia de arquitecturas se refiere a una plataforma que sirve como base para desarrollar nuevas plataformas, variando las funcionalidades que esta pueda poseer, y las características comunes se refieren a una instancia de una plataforma, es decir una arquitectura particular conformada por algunas de las capacidades que poseen como tal las familias de plataformas. Las funcionalidades que definen a una plataforma pueden ser empleadas en diferentes etapas de desarrollo de la plataforma, y mediante esto se puede reducir sus costos de desarrollo, su valor es medido en cuanto a su noción de reusabilidad y como se podrá apreciar a continuación el diseño de una plataforma servirá para la implementación de muchas más con diferentes variaciones, lo cual es muy beneficioso para el desarrollo rápido de sistemas embebidos.

2.2.1. Características

El rango de productos OMAP consiste en varias familias de dispositivos dirigidos a diferentes mercados, la fortaleza de este concepto beneficia a los fabricantes de dispositivos móviles, desarrolladores de software y grupos de desarrollo internos de TI, quienes pueden ampliar la utilización de su previo desarrollo, a través de sus diferentes productos que pueden ser aplicados a varios niveles, entre sus principales características se pueden enumerar las siguientes:

- Los dispositivos pueden utilizar periféricos e interfaces de memoria comunes, lo que implica un soporte común para su interconexión con otros componentes de un teléfono móvil o PDA.
- Beneficia a la creación de aplicaciones que pueden ser compartidas entre dispositivos, las cuales pueden ser creadas bajo ambientes de desarrollo comunes. Pueden utilizar un solo software framework de bajo nivel, mediante este, se garantiza que los desarrolladores puedan escribir código que será utilizado en diversos dispositivos.
- Con una sola plataforma SoC se puede diseñar y producir rápidamente un nuevo producto[24].
- Se encuentran definidos por estándares a través del MIPI (Mobile Industry Processor Interface)[26]. MIPI es una alianza de varias compañías que buscan definir estándares de interfaces comunes para dispositivos móviles, de esta manera se adaptan interfaces comunes para distintos dispositivos.

2.2.2. ARM (Advanced RISC Machines)

Es necesario definir que el tipo de procesador utilizado por OMAP ya que tiene la particularidad que es utilizado por todas las familias OMAP. Las plataformas OMAP, utilizan para su diseño, procesadores de Propósito General tipo ARM, los cuales se caracterizan por tener una arquitectura computacional tipo RISC. ARM es una compañía proveedora de Propiedad Intelectual (IP), provee diseño y licenciamiento IP de tecnología, en lugar de fabricarla y distribuirla. De esta forma soportan una gran cantidad soluciones de procesamiento utilizadas por una gran cantidad de fabricantes de tecnología. Sus soluciones van desde handsets móviles, set top boxes, sistemas de frenado de automóviles,

incluso routers, actualmente se utiliza en casi el 95% de los teléfonos móviles del mundo y más de una cuarta parte en dispositivos electrónicos[27].

Los procesadores ARM son utilizados para microcontroladores, DSP y aplicaciones Java, basan su diseño en procesadores RISC, son utilizados para aplicaciones que requieren la combinación de DSP y el rendimiento del microcontrolador, utilizan tecnología Jazelle [27], que permite la ejecución directa de bytecodes de Java. Los procesadores ARM se caracterizan por un bajo consumo de potencia, lo cual los hace ideales para utilizarlos en dispositivos como los antes mencionados. Son soportados por varios sistemas Operativos de Tiempo Real (RTOS) embebidos. Dentro de las familias OMAP se utilizan procesadores ARM9E, ARM11 y ARM Cortex-A. Cabe recalcar que los procesadores ARM9 fueron los pioneros, estos fueron utilizados como procesadores para funcionamiento clásico, su mejora ARM Cortex-A8, se diferencia de este, porque se encuentra dirigido específicamente hacia plataformas para aplicaciones orientadas a video.

2.2.3. Familia OMAP

TI OMAP se encuentra distribuida hacia el mercado de los celulares y smartphones, el cual es su principal canal de distribución, pero también cuenta con versiones de catálogo para aplicaciones, que además cuentan con modelos de soporte. La familia TI OMAP (tabla 2.1) se encuentra dividida en cuatro grupos[28] de productos clasificados de acuerdo a su desempeño y uso previsto.

Tabla 2.1. Familia TI OMAP

Grupos	Descripción	Familias
Alto Desempeño	Procesadores de aplicación en smartphones Corren sistemas operativos como Linux, Symbiam, Windows CE y Amdroid, Soportan conectividad Aplicaciones como audio y video	OMAP 1 OMAP 2 OMAP 3 OMAP 4
Multimedia Básica	Comercializados a fabricantes de teléfonos móviles Altamente integrados y de bajo coste Productos de consumo.	OMAP 331 OMAP 310 OMAP DM xx
Aplicaciones y Modems	Comercializados a fabricantes de teléfonos móviles Altamente integrados Utilizados en teléfonos móviles muy baratos.	OMAP Vxx OMAP 850 OMAP 7xx
OMAP-L1xx	Comercializados por catálogo, Aplicaciones orientadas a video	OMAP-L137 OMAP-L138

	Tecnológicamente diferentes a otros OMAP	

Procesadores de Aplicaciones Multimedia de Alto Desempeño

Familia de OMAP de procesadores de aplicaciones de alto desempeño, son las más utilizadas actualmente, en su mayoría se encuentran dirigidos al mercado de smartphones, pero además incluyen mercado de distribución por catálogo para aplicaciones específicas, a esta pertenece la familia OMAP35xx, junto con las cuales se distribuyen además, tarjetas de desarrollo que incluyen periféricos utilizados en la elaboración de aplicaciones. Son capaces de correr sistemas operativos como Linux, Symbiam, Windows CE y Android, soportan conectividad a computadores personales y varias aplicaciones como audio y video, incluidas en esta categoría se encuentran cuatro familias de plataformas:

- **OMAP1**

La familia OMAP1 se caracteriza por utilizar procesadores ARM 925EJ y 926EJ, sus versiones (ver tabla 2.2 [29]) se diferencian de acuerdo a su tecnología empleada o set de periféricos, algunos de sus modelos se encuentran fuera del mercado, pero otros como el OMAP 1710, se encuentran disponibles para mercado de celulares.

Tabla 2.2. Familia OMAP1

Modelo	Vel. Reloj	Procesadores	Tecnología	Observaciones
OMAP17110	220 MHz	ARM926EJ-S + C55x DSP	90 nm	Low-voltage

- **OMAP2**

La familia OMAP2 es utilizada por proveedores de celulares debido a su rendimiento en multimedia para handsets 3G y 2G y wireless PDAs (ver tabla 2.3 Tabla 2.3[29]), por lo general incluyen aceleradores de hardware y mejoras de procesamiento de video, además incluyen algoritmos de alta compresión de datos, permitiendo redes de datos que soportan más información, posibilita la implementación de servicios como intercambio de juegos multimedia a través de móviles, reproductor de música y videoconferencia.

Tabla 2.3. Familia OMAP2

Modelo	Vel. Reloj	Procesadores
OMAP2431	330 MHz - 220 MHz	ARM1136 + C64x DSP
OMAP2430	330 MHz - 220 MHz	ARM1136 + C64x DSP
OMAP2420	330 MHz - 220 MHz	ARM1136 + C55x DSP

- **OMAP3**

La familia OMAP3 proporciona productividad de alto desempeño y capacidades avanzadas en entretenimiento para dispositivos móviles de próxima generación, integrando por procesadores de aplicaciones multimedia de alto nivel, con desempeño como el de una portátil. Incluyen aceleradores de Hardware IVA (Image Video Audio), Procesadores de Señales de Imágenes (ISP Integrated Image Signal Processor), utiliza tecnologías para reducción de consumo energético, soporta Sistemas Operativos de Alto nivel[29].

OMAP3 se encuentra dividido en tres familias (ver tabla 2.4 [29, 30]) OMAP34xx, OMAP35xx y OMAP36xx. OMAP34xx y OMAP36xx se encuentran enfocadas al mercado de smartphones y actualmente existen algunos dispositivos que los utilizan. En cuanto a la familia OMAP35xx, solamente se encuentra solo mediante catálogo y se lo utiliza como procesador de aplicaciones, incluye un procesador Cortex-A8 y un procesador propio de TI C64x+DSP, entre sus mejoras se incluye que entrega una interface de usuario avanzada, gráficos de alta calidad, video y conectividad para aplicaciones, se distribuye junto con una tarjeta de aplicaciones.

Tabla 2.4. Familia OMAP3

Modelo	Vel. Reloj MHz	Procesadores	Tecnología	Dispositivos que lo utilizan
OMAP34x	600 800	ARM Cortex-A8 + IVA 2 + ISP+ aceleradores de Harware y de gráficos 2D/3D	65 nm	Motorola Droid/Milestone, Palm Pre, Samsung i8910, Nokia N900, Milestone XT720
OMAP35x	600 720	ARM Cortex-A8 + C64x + DSP y aceleradores de video	65 nm	Gumstix Overo Earth, OSWALD, Beagle Board, IGEPv2, Touch Book, Gumstix Overo Water
OMAP36x	800	ARM Cortex-A8 + IVA 2 + ISP+ aceleradores de	45 nm	Motorola Defy, Motorola Milestone 2, Pre 2, Motorola Droid

		Harware y de gráficos 2D/3D		2 Global

- **OMAP4**

La plataforma OMAP4xx se presenta como una mejora de la familia OMAP, utiliza Dual-Core mediante su procesador Cortex-A9, el cual emplea el más alto desempeño y menor consumo de energía, además posee mejoras en sus aceleradores de Hardware, empleando Full High Definition y codificador/codificador multiestándar de video. Es empleado en teléfonos móviles de última generación como se muestra en la tabla 2.5 [29].

Tabla 2.5. Familia OMAP4

Modelo	Vel. Reloj MHz	Procesadores	Tecnología	Dispositivos que lo utilizan
OMAP4430 **OMAP4440**	1 GHz 1.5 GHz	Dual-Core ARM Cortex-A9	45 nm	PandaBoard, RIM BlackBerry Playbook

Procesador de Aplicaciones Multimedia Básico

Son comercializados solamente a fabricantes de teléfonos móviles, son altamente integrados y de bajo costo. Utilizados como productos de consumo, enfocando su desempeño a mayor rendimiento multimedia de dispositivos móviles, incluyendo capacidades de alta calidad de captura de imágenes, para este propósito utilizan un ISP. La serie OMAP-DM (ver tabla 2.6[29]) se encuentra orientada hacia dispositivos móviles que incluyen cámaras digitales de alto desempeño.

Tabla 2.6. OMAP Multimedia Básico

Modelo	Procesadores
OMAP331 **OMAP310**	ARM9
OMAP-DM270	ARM7 + C54x DSP
OMAP-DM299	ARM7 + ISP
OMAP-DM500	ARM7 + ISP
OMAP-DM5xx	ARM926 + ISP

Procesador de Aplicaciones y Modem Integrado

Comercializados a fabricantes de teléfonos móviles, son altamente integrados y utilizados para teléfonos móviles muy baratos. Incluyen multimedia de alto rendimiento, multimedia básica y procesador de aplicaciones para handsets y PDAs de 2.5G y 3G, utilizan procesadores ARM926EJ y su solución integra funcionalidades de modem y de aplicación en la misma arquitectura (tabla 2.7[29]).

Tabla 2.7. OMAP Aplicaciones y Modem

Modelo	Características
OMAPV1035	ARM926EJ + GSM/GPRS/EDGE digital baseband, RF, y procesador de aplicaciones
OMAPV1030	ARM926TEJ + GSM/GPRS/EDGE digital baseband y procesador de aplicaciones
OMAP850	ARM926TEJ + EDGE digital baseband y procesador de aplicaciones

Procesador OMAP-L1xx

Comercializados solo por catálogo, utilizados para aplicaciones orientadas a video, se caracteriza por ser tecnológicamente diferentes a otros OMAP ya que estos, a diferencia de la Familia OMAP utiliza coma flotante (ver tabla 2.8 [29]).

Tabla 2.8. OMAP L1xx

Modelo	Vel. Reloj MHz	Procesadores	Dispositivos que lo utilizan
OMAP-L137	300	ARM926EJ-S + C674x floating point DSP	Hawkboard
OMAP-L138	300	ARM926EJ-S + C674x floating point DSP	

2.2.2. OMAP35xx

El estudio de plataformas SoC se enfocará en el análisis de la familia TI OMAP35x, ya que es la plataforma seleccionada para el estudio y desarrollo de aplicaciones en esta tesis. TI OMAP35x es la primera en el mercado con núcleo ARM-Cortex A8 y produce mejoras en su desempeño cuatro veces superiores al anteriormente utilizado ARM 9, su

principal diferencia entre las otras familias y la razón por la que se escogió para su estudio es que son plataformas dirigidas al desarrollo de aplicaciones y su rango de aplicación no se limita solamente a teléfonos celulares, sino que ofrecen prestaciones variadas que pueden ser utilizadas en diferentes mercados.

OMAP35xx ofrece cuatro diferentes combinaciones de prestaciones tecnológicas entre su procesador ARM, periféricos multimedia, gráficos OpenGL ES 2.0 (Open Graphics Library for Embedded Systems) [31], las cuales diferencian a los cuatro dispositivos de la familia OMAP35xx (OMAP3530, OMAP3525, OMAP3515, OMAP3503). Además puede correr sistemas operativos como Windows CE, Linux y Android, permite navegación web y compatibilidad con Java. Posee tecnología TI SmartReflex, que reduce el consumo de energía y optimiza el desempeño mediante técnicas adaptativas de hardware y software que controlan dinámicamente el voltaje y la frecuencia[31]. Su arquitectura se encuentra diseñada para proporcionar la mejor calidad de video, imagen y procesamiento de gráficos, puede soportar transmisión de video, imágenes 2D/3D, videoconferencia, alta resolución de imagen fija, soporta una amplia gama de aplicaciones entre las cuales se puede citar: Terminales de datos portables, Navegación, consolas de juego, interface humana, control industrial, ensayos y mediciones, computadores de placa[31].

Diagrama de Bloque TI OMAP35x

La arquitectura de los dispositivos OMAP35x, como muestra la Figura 2.1[31], se halla conformada por bloques, cada uno se encuentra integrado por módulos afines al bloque. La arquitectura presentada incluye todos los bloques que definen la familia, pero pueden ser distribuidos de forma diferente en sus categorías, es decir, que los bloques mostrados, pueden o no incluirse en el dispositivo final. Es importante analizar de una forma breve cada uno de los bloques que conforman la familia TI OMAP35x, pues las características que aporten, ayudarán a definir la plataforma sobre la cual se centra el desarrollo del proyecto.

- **Microprocesador ARM ® Cortex ™-A8 [32]**

Es un procesador de la familia de procesadores RISC ARM de alto desempeño, que hoy en día es utilizado en gran cantidad de dispositivos electrónicos, entre los cuales se

encuentran teléfonos móviles, netbooks, DTVs, impresoras, dispositivos de información y entretenimiento automovilístico y otras aplicaciones de alto desempeño. Trabaja en un rango de frecuencias desde los 600 MHZ a 1GHZ. Texas Instruments implementa estos procesadores en su familia OMAP35x, a los rangos de frecuencias de 600 MHz y 700 MHz.

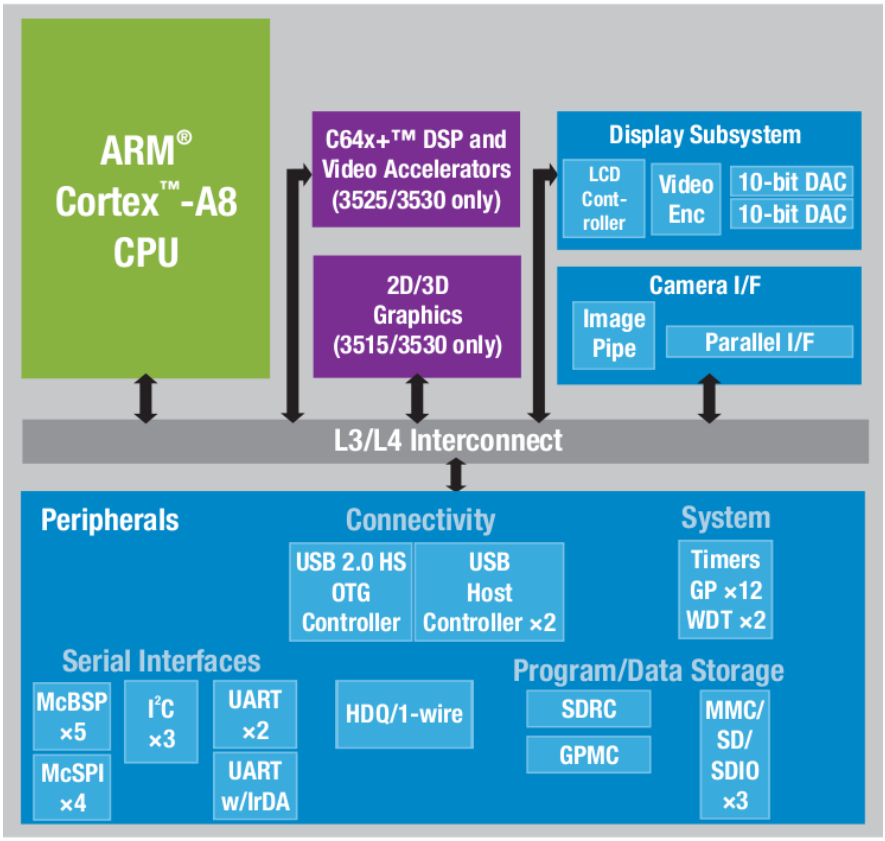


Figura. 2.1. Diagrama de Bloques OMAP35x

Basa su diseño en la tecnología NEON™[33]. La tecnología NEON puede acelerar multimedia y algoritmos de procesamiento de señales, mediante su arquitectura *SIMD* (Single Instruction, Multiple Data) de 128 bits, diseñada para proveer aceleración de aplicaciones multimedia.

En OMAP35x, el procesador ARM es adecuado para códigos de control y esta es su función principal, en él se corre el sistema operativo y aplicaciones de usuario, la tecnología ARM puede manejar periféricos, pero no es su tarea fundamental. Dentro del sistema OMAP, la arquitectura OMAP35xx, maneja el procesamiento de señales por separado, utilizando otro procesador el cual será analizado más adelante.

- **C64x+ digital signal processor (DSP) Core [34]**

TMS320DMC64X+ VLIW digital signal processor (DSP) core, define el subsistema IVA (Image Video and Audio Accelerator), el cual es propio de Texas Instruments y se encuentra implementado en OMAP35xx. Se encarga de manejar el procesamiento de señales, como video, procesamiento de voz y audio. El procesamiento de señales realizado por el procesador DSP consume mucha menos energía. VLIW (Very Long Instruction Word), es una arquitectura de CPU que implementa una forma de paralelismo a nivel de instrucción. Su objetivo es simplificar el diseño del hardware al dejar todo el trabajo de planificar el código en manos del compilador, e implica una menor potencia y consumo.

- **POWERVR SGX™ Graphics Accelerator subsystem (GPU)[35]**

POWERVR, es una división de la compañía Imagination Technologies, que se encarga de desarrollar hardware y software para interpretaciones 2D y 3D, para codificación y decodificación de video y procesamiento de imágenes.

SGX se refiere a la serie 5 de la familia POWER VR. POWERVR™ SGX GPU (Graphics Processing Unit), es un core IP que funciona como un subsistema para la aceleración de gráficos 2D y 3D, que soportan OpenGL 2.0 y DirectX 10.1 Shader Model 4.1. Se encuentran incluidos en SoCs de sistemas portables, soportan display y efectos de video para videojuegos. Texas Instruments utiliza estas prestaciones en la serie de familias OMAP 3 y 4, las cuales son utilizadas por un gran número de fabricantes de teléfonos móviles en el mercado. Utiliza una serie de características para generación de gráficas que adicionan efectos de sobras, profundidad y geometría que son esenciales en la creación de ambientes de gráficas 3D.

Una de las características que aporta POWERVR SGX es que es compatible con gráficas OpenGL ES 2.0 (Open Graphics Library for Embedded Systems)[36]. OpenGL ES 2.0 es definida por Khonos, un grupo de empresas dedicadas a hardware y software gráfico en APIs gráficas y multimedia. Definen una API (Application Programming Interface) multilenguaje y multiplataforma, para gráficos, mediante la cual se pueden escribir aplicaciones generadoras de gráficas 2D y 3D. Consiste en una librería de funciones que pueden ser utilizadas para dibujar escenarios 3D a partir de figuras geométricas primitivas.

- **Camera image signal processor (ISP) [34]**

El procesador de señales de imágenes para la cámara soporta múltiples formatos y opciones de interfaces para una amplia variedad de sensores de imagen.

- **Subsistema de Pantalla[34]**

El subsistema de Pantalla posee una gran cantidad de funciones para la manipulación simultánea de múltiples imágenes y una interfaz programable, la cual soporta una variedad de pantallas. Además soporta salida de video NTSC (resolución de 525 líneas) y PAL (resolución de 625 líneas). El sistema sintetiza tareas del procesador ARM, permitiendo el desarrollo de de aplicaciones gráficas, ya que suprime la necesidad de instalar tarjetas de video extra.

- **Interconexiones Nivel 3 (L3) and Nivel 4 (L4) [34]**

Es un subsistema de interconexiones que proporcionan transferencias de alto ancho de banda para indicadores múltiples y controladores de memoria externa y periféricos on-chip.

- **Periféricos[34]**

La familia OMAP35xx incluye numerosos interfaces para conectar periféricos o dispositivos externos, como se pudo observar en la Figura 2.1. Los interfaces integrados simplifican el diseño y reducen costos totales. Se tienen periféricos de interfaces seriales, de conectividad, de sistema, de almacenamiento de datos, para el subsistema de display, cámara, los cuales serán detallados más adelante

SoCs de la Familia TI OMAP35x

La familia TI OMAP35x se encuentra conformada por cuatro dispositivos SoC los cuales ofrecen diferentes prestaciones, en cuanto a procesador ARM Cortex-A8, set de periféricos multimedia, compatibilidad de gráficos 2D y 3D con OpenGL ES 2.0, aceleradores de Video y procesador C64x+ DSP.

La Figura 2.2[37], muestra la distribución de los diferentes atributos, los cuales minimizan el desarrollo de aplicaciones, aprovechando reutilización de hardware y software, la generación OMAP35xx incluye[31]:

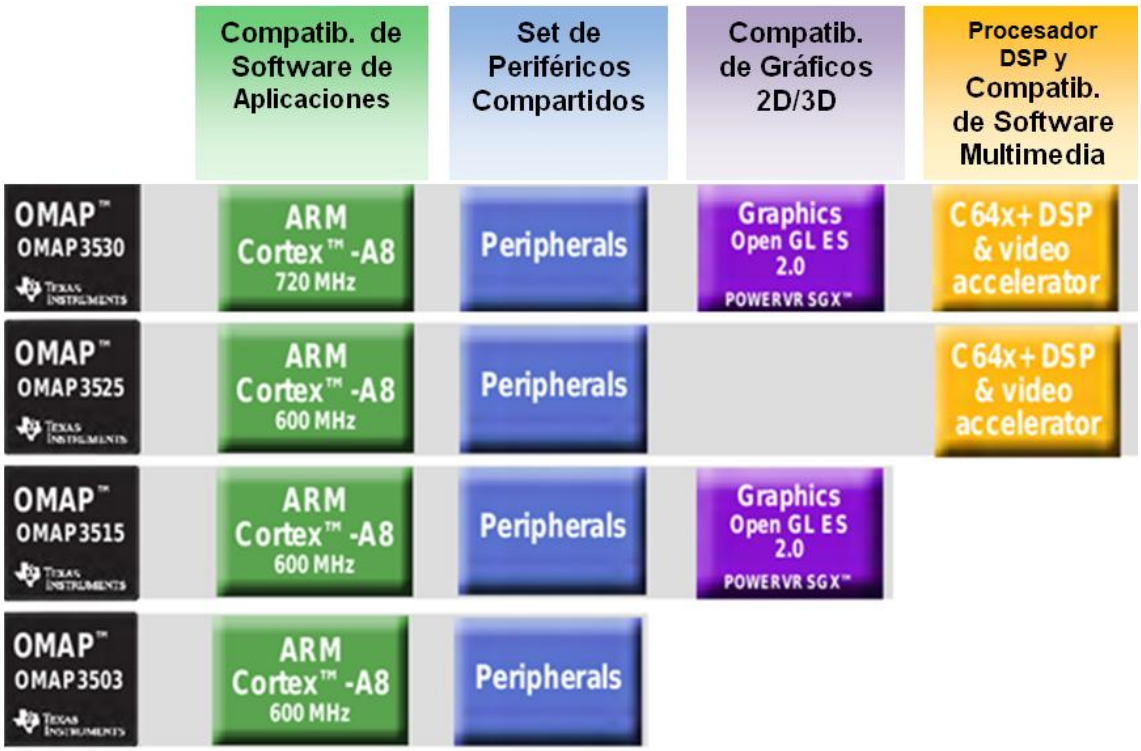


Figura. 2.2. SoCs OMAP 35xx

- **OMAP3503**

Cuenta con un procesador de 600-MHz ARM Cortex-A8, un conjunto de periféricos y memoria. Con 1200 Dhrystone MIPS de rendimiento, puede ejecutar sistemas operativos, como Windows CE y Linux y permite un arranque más rápido, navegación web y tiene compatibilidad con aplicaciones Java.

- **OMAP3515**

Consiste en el mismo set de periféricos y procesador ARM de OMAP3503, con la adición de OpenGL ES 2.0 como motor gráfico. El procesador OMAP3515 puede lograr una gran calidad gráfica en juegos, por lo que es de elección para los desarrolladores de consolas de o para desarrolladores de sistemas portátiles de navegación.

- **OMAP3525**

Tiene las mismas características que el OMAP3503, con la adición de servicios integrados para audio, vídeo, imágenes y aceleración multimedia combinado con hardware de vídeo habilitado

- **OMAP3525**

Trae todas las características de OMAP3503, 3515 y 3525. Con un procesador ARM integrado, DSP, motor gráfico, tecnología DaVinciTM, y un rico conjunto de periféricos. TI OMAP3530 tiene un procesador de alto rendimiento y ahorro de energía, para aplicaciones de entretenimiento.

2.3. TARJETAS CON PLATAFORMAS TI OMAP

El desarrollo actual de SoC, se ha convertido en un desafío de diseño cada vez mayor. La complejidad de las aplicaciones ha llegado a ser en un factor determinante en el ciclo de diseño de los sistemas y ha interferido en los tiempos de salida al mercado (TTM). Nació la necesidad de contar con un conjunto de herramientas de desarrollo que den un entorno adecuado para el diseño, lo simplifiquen, y que además permitan aprovechar al máximo las prestaciones de los sistemas. Para este propósito se han creado *Tarjetas de Desarrollo* en diferentes niveles, de acuerdo a las prestaciones que ofrecen frente al desarrollo de aplicaciones.

Las Plataformas SoC, que se encuentran integradas a una Tarjeta de Desarrollo, son consideradas en el mercado como el microprocesador de la tarjeta, ya que es su unidad central de procesamiento, pero en realidad son un System-on-Chip y en este caso pertenecen a la familia TI OMAP35x. La idea de una Tarjeta de Desarrollo, es dar una base estable para desarrollar aplicaciones, aprovechando las prestaciones del SoC utilizado, mediante el uso de periféricos incluidos en las plataformas. Es la manera para poder empezar el desarrollo de aplicaciones de software, antes de que la plataforma final, este completa y sea lanzada al mercado, lo que es una gran ventaja para un mejor TTM.

Generalmente una Plataforma SoC puede ser evaluada sobre un Tarjeta de Desarrollo llamada Modulo SoM (System-on-Module)[38], esta tarjeta de desarrollo cuenta con un microprocesador (SoC OMAP35x), memorias y entradas/salidas, es decir tiene las

prestaciones básicas, pero dispensa de todos los periféricos necesarios para el desarrollo de aplicaciones; estas y otras prestaciones pueden ser agregadas, y así se forman diferentes entornos de desarrollo Hardware. Las diferencias entre los entornos de desarrollo, se ven reflejados en su precio, el cual va aumentando en cuando aumentan las prestaciones que ofrecen, las mismas que se miden en cantidad de periféricos, medios de comunicación, soporte que brindan las casas de diseño e implementos adicionales para desarrollo (como pantallas Touch-Screem).

A continuación se analizarán las diferentes Tarjetas de Desarrollo: System-on-Module, Kit de Desarrollo y Módulo de Evaluación. En la Figura 2.3[39], se muestran los diferentes de entornos de desarrollo, empezando por SOM. Se analizarán las tarjetas OMAP existentes en el mercado, se tomaron en cuenta, costos referenciales indicados por los fabricantes, los cuales pueden diferir al momento de adquirirlos.

Figura. 2.3. Ambientes de Desarrollo Hardware

2.3.1. System-on-Module (SOM)

Características

- Desarrollo de prototipos
- Las más baratas del mercado, sus costos van desde los $55 a los $400
- Utilizadas como herramientas de desarrollo puramente de software

- Generalmente incluyen un SoC, memoria RAM y controladores de I/O
- No cuenta con periféricos de entrada y salida
- El cableado de estos periféricos es encaminado a través de buses hacia conectores sobre la placa.
- Para su funcionamiento, el módulo necesita ser montado sobre una plataforma de soporte o placa base
- Los buses deben ser conectados a la placa base para finalizar su recorrido sobre , conectores estándar de periféricos.
- Tienen soporte de Texas Instruments y de los fabricantes del módulo

Tarjetas TI OMAP

Tabla 2.9. System-on-Module TI OMAP35x

Tarjeta	Texas Instruments OMAP35x SOM-LV	Texas Instruments OMAP35x Torpedo SOM	CM-T3530 Computer-On-Module	OMAP3530 0.40mm Processor Module
Fabricante	*LOGIC PD*	*LOGIC PD*	*CompuLab*	*MISTRAL*
				
Dimensiones	31 x 76.2 x 7.4 mm	15 x 27 x 3.8 mm	66 x 44 x 7 mm	52 x 53.3 mm
Código	SOMOMAP3530-11-1672IFCR	SOMOMAP3530-21-1670AGCR	CM-T3530-D64-C600-N128	OMAP3530 0.40mm
Componentes	OMAP3530 processor, 600 MHz, 128 MB DDR, 256 MB NAND flash, 8 MB NOR flash, LAN Ethernet, audio, touch, Bluetooth, 802.11	OMAP3530 processor, 600 MHz, 128 MB DDR, 256 MB NAND flash, audio	Cortex-A8 OMAP3503 or OMAP3530 CPU, up to 720 MHz 256 Mbyte mobile DDR 512 Mbyte Flash Disk, incluye protección del filesystem	OMAP3530 Processor – 0.40mm, Memoria Micron: 128 MB Mobile DDR SDRAM & 128 MB NAND Flash
Referencia	http://www.logicpd.com	http://www.logicpd.com	http://www.compulab.co.il	http://www.mistralsolutions.com

2.3.2. Kits de Desarrollo

Características

- Desarrollo de aplicaciones parcial

- Diseño dedicado al desarrollo de hardware y software
- Kits de desarrollo de bajo costo, para evaluación de aplicaciones
- Su arquitectura de hardware generalmente comprende dos módulos, uno de procesamiento y otro de periféricos básicos.
- Incluyen un módulo SOM, una placa base que contiene periféricos básicos, herramientas de desarrollo y soporte.
- Desarrollo de aplicaciones limitado, en función a número de periféricos.
- Subutilizan el SoC, ya que no pueden aprovechar todo su potencial.
- Se los puede encontrar en dos tipos: **Community Board y eXperiments Kit**

Tatjetas TI OMAP

- **eXperimenter Kit**

Tienen soporte de Texas Instruments y de los fabricantes del módulo. Sus costos varían desde los $150 a los $600 dólares

- **Community Board**

No tienen soporte de Texas Instruments, pero la existe una comunidad de apoyo en la web. Sus costos varían desde los $125 a los $220 dólares

Tabla 2.10. Kits de Desarrollo TI OMAP35x

Kits de Desarrollo	Community Board	Community Board	eXperimenter Kit
Nombre	Overo	BeagleBoard	Tsunamipack
Fabricante	Gumstix	BeagleBoard	TechNexion
			
Dimensiones	17mm x 58mm x 4.2mm	76.2 x 76.2 mm	120 x 50 mm
Código	GUM3503W/A/E/ F/FE	BEAGLE	Tsunamipack

Componentes	OMAP3530 Application Processor Conectividad wireless 802.11b/g y Bluetooth, microSD card slot, TPS65950 Power Management	OMAP3530 Application Processor, periféricos: DVI-D, USB OTG, MMC+/SD/SDIO, salida S-Video, I/O de audio Stereo, Energía vía cargadores USB	Procesador OMAP3530W, LCD de 4.3", periféricos: Wi-Fi 802.11b/g, MMC/SD, USB, LAN, audio, entrada y salida de S-video y TTL, software, adaptor de 60W 12VDC
Referencia	http://www.gumstix.net	http://beagleboard.org/	http://www.technexion.com

2.3.3. Módulos de Evaluación

Características

- Tarjetas de desarrollo de aplicaciones completas.
- Son orientados al desarrollo de aplicaciones multimedia de alto desempeño
- Sus costos varían desde los $995 a los $1500 dólares
- Adicionalmente de lo que incluyen los Kit de Desarrollo, los Módulos de Evaluación incluyen: en cuanto a Hardware pantalla LCD, Teclado, módulos Wireless, y en cuanto a Software incluyen Sistemas Operativos (Linux, WinCE, Android), herramientas de desarrollo Software Development Kit (SDK).
- Poseen una ventaja frente a las anteriores herramientas, ya que no subutilizan el potencial del SoC, pues poseen todos los periféricos posibles.
- Tienen soporte de Texas Instruments y de los fabricantes del módulo

Tarjetas TI OMAP

Tabla 2.11. Módulos de Evaluación TI OMAP35x

Nombre	**Zoom™ OMAP35x Development Kit**	**Zoom™ OMAP35x Torpedo Development Kit**	**OMAP35x EVM**
Fabricante	*LOGIC PD*	*LOGIC PD*	*TI&MISTRAL*
			
Dimensiones	146.1 x 158.8 x 17.1 mm	134.0 x 94.5 x 18.9 mm	114.3 x 177.8 mm

Código	TMDSMEVM3530-L	SDK-OMAP3530-20-128256R	TMDSEVM3530
Componentes	OMAP35x SOM-LV module, LCD WQVGA Touch Screen, Plataformas de Expansion, periféricos: USB, MMC/SD, CompactFlash, Conectividad Wireless LAN y Bluetooth, Cables y Software	OMAP35x Torpedo SOM module, periféricos: Ethernet, USB, UART, MMC/SD 3 Boards, Battery Pack, Cables, Power Supply, LCD WQVGA Touch Screen Cables y Software	OMAP35x EVM Processor Module, Módulo WL1271 Daughter Card con Wireless LAN and Bluetooth, soporte de USB host, entradas de video, TPS65950 Power Management, memoria de 256 MB LPDDR/256 MB NAND Flash memory y OMAP35x Linux Digital Video Software Development Kit (DVSDK).
Referencia	http://www.logicpd.com	http://www.logicpd.com	http://www.mistralsolutions.com

2.4. SELECCIÓN DE TARJETA CON PLATAFORMA TI OMAP

Las plataformas TI OMAP se encuentran disponibles en el mercado a través de diversas tarjetas de desarrollo, mediante las cuales se pueden evaluar sus capacidades. Las tarjetas de desarrollo son soportadas por varios fabricantes, los cuales proporcionan una base de hardware, software y soporte para empezar a desarrollar tecnología. Su rango de precios varía entre cientos y pocos miles de dólares. Como uno de los objetivos planteados del proyecto, se encuentra la adquisición de una plataforma TI OMAP para el DEEE. Para esto fue necesario tomar en cuenta algunos criterios técnicos y no, los cuales se describirán a continuación.

2.4.1. Enfoque de Selección

Si bien se tienen algunas posibilidades en cuanto a plataformas y tarjetas de desarrollo, su selección debe cumplir con los objetivos planteados y debe respetar el presupuesto asignado, a la vez que se adecue a un dispositivo con las mayores y mejores prestaciones. Es así que, a continuación se realizará el análisis adecuado para su adquisición.

Situación del Proyecto

La presente tesis es parte de un Proyecto de Iniciación Científica, el cual costa de dos partes, se pretende el desarrollo de un sistema de referencia que utilice las capacidades de

una plataforma Texas Instruments de la Familia OMAP, su propósito, el desarrollo tecnología en el país y más concretamente en el DEEE.

Su primera fase, *Multimedia*, será desarrollada en la presente tesis, la segunda fase del proyecto, llamada *Wireless*, será desarrollada posteriormente. Ya que la selección y adquisición se realizo en la primera parte del proyecto, se debe tomar en cuenta que la tarjeta seleccionada cuente de alguna forma de comunicación inalámbrica, que será útil en la segunda parte del proyecto.

La comunicación inalámbrica puede estar presente de varias maneras. La tarjeta de desarrollo puede incluir un módulo Wireless, la comunicación puede estar incluida dentro de la placa o puede ser adquirido como una plataforma separada y adaptarla a la plataforma adquirida. La mejor opción sería que sea obtenida como módulo de la tarjeta, de esta forma podrá ser adaptada cuando se desarrolle la segunda parte del proyecto.

Situación del Ecuador

Si bien el presente proyecto pretende estudiar, analizar y emplear tecnología de última generación para transferir conocimiento y desarrollar aplicaciones en el pais, las plataformas necesarias para este cometido son desarrolladas en el extranjero, para su adquisición lo ideal sería contar con un proveedor directo, pero eso no es posible y aquello repercute directamente en el precio final de adquisición. Para el caso del Ecuador, se tienen dos consideraciones importantes, que no existe un mercado para plataformas de desarrollo SoC de ningún tipo y los aranceles que deben ser cancelados para adquirir equipos electrónicos en el país es muy alto.

La adquisición de plataformas SoCs, es un mercado totalmente nuevo para el país e incluso para Latinoamérica, debido a que no somos considerados como países desarrolladores de tecnología, sino más bien, consumidores de la misma. Además, para este tipo de tecnología muchas veces es necesario contar con un permiso especial y declaración de que no será usado para aplicaciones consideradas peligrosas o en contra de la seguridad. Los proveedores de dispositivos electrónicos locales deben considerar la importación de pocos dispositivos, debido a su poca demanda, además deben costear los

altos aranceles obligatorios. Esto genera un incremento en el precio de la plataforma, en relación al indicado por los fabricantes en sus páginas web, y debe ser considerado.

Consideraciones Técnicas

El punto más importante a tomar en cuenta es la parte técnica. Si el presupuesto lo permite, la tarjeta adquirida debe tener las mayores prestaciones posibles, solo así se asegurará que pueda ser aprovechada al máximo. Se debe considerar una plataforma y tarjeta que permitan utilizar todas las características que nos brindan las plataformas TI OMAP35x. Las características de las plataformas OMAP35x son conocidas, así que se debe buscar la plataforma que mejor se sujeta a las mismas, y que soporte entre otras cosas lo siguiente:

- Periféricos: se debe considerar que existan una gran cantidad de periféricos estandarizados en la industria.
- Conectividad: se debe asegurar que se cuenten con los periféricos básicos para la comunicación.
- Wireless: es importante tener en cuenta la segunda parte del proyecto, por lo que se deben incluir un módulo de comunicaciones inalámbricas que como mínimo soporte Bluetooth y Wi-Fi
- Pantalla: uno de los requerimientos que deben considerarse es que se incluya una pantalla Touch-Screen, que es lo que se utiliza en los dispositivos actuales.
- Soporte: Es esencial contar con el soporte oficial necesario para el desarrollo, al ser oficial, quiere decir que Texas Instruments soporta al dispositivo y se puede acceder a colaboración directa, la cual es contestada inmediatamente, además de contar con soporte de los fabricantes de la plataforma.
- Herramientas de Desarrollo de Software: se precisan de herramientas de desarrollo, las cuales deben ser incluidas con el dispositivo y se permita acceder a actualizaciones.
- Android: es importante que cuente con soporte para la instalación del sistema operativo Android y su SDK, ya que al ser un SO relativamente nuevo, no existen muchos periféricos que lo soporten oficialmente.

2.4.2. Selección

Selección Plataforma SoC TI OMAP35x

Texas Instruments, con su familia OMAP, presenta una gama de posibilidades para el desarrollo de dispositivos de aplicaciones como PDA, tablets, media players portables, etc y sus productos se encuentran ya en el mercado dentro de dispositivos que son muy utilizados actualmente. La facilidad que aportan las plataformas System-on-Chip, al permitir crear sistemas portables entre plataformas de desarrollo, hace que consideremos la posibilidad de utilizar una plataforma que no se enfoque solamente en dispositivos telefónicos sino que además brinde la oportunidad de realizar aplicaciones varias, las cuales puedan ser luego trasladadas a un dispositivo similar. Es por esta razón que se decidió escoger la familia de plataformas TI OMAP35x, la cual se enfoca al desarrollo de aplicaciones.

Conforme a lo indicado en la sección 2.2.2., la familia TI OMAP35x se encuentra conformada por cuatro dispositivos SoC los cuales ofrecen diferentes prestaciones, en cuanto a procesamiento que incluyen núcleos ARM, DSP e ISP, set de periféricos multimedia estándar, compatibilidad de gráficos 2D y 3D, aceleradores de video, imagen y audio, y un sistema de interconexión de alto rendimiento. A continuación se muestra una tabla 2.12[34] de lo más representativo de estas familias.

Tabla 2.12. Soporte de Subsistemas/Co-Procesador/Periféricos OMAP35x

Subsistemas/Co-Procesador/Periféricos	OMAP3530	OMAP3525	OMAP3515	OMAP3503
Subsistema IVA2.2	✓	✓		
Subsistema Acelerador Gráfico POWERVR SGX™	✓		✓	
Co-Procesador Cortex-A8 Neon	✓	✓	✓	✓
Controlador SDRAM	✓	✓	✓	✓
Controlador de Memoria de Propósito General	✓	✓	✓	✓
Package-on-Package	✓	✓	✓	✓
Procesador de Cámara ISP	✓	✓	✓	✓
Interfaces				
Ancho de entrada @ Max Clk	12 bit input @ 75MHz 9 bit input @ 130MHz	12 bit input @ 75MHz 9 bit input @ 130MHz	12 bit input @ 75MHz 9 bit input @ 130MHz	12 bit input @ 75MHz 9 bit input @ 130MHz
Modo de entrada en Paralelo	✓	✓	✓	✓
Modo de entrada de Sync	✓	✓	✓	✓
Modo de entrada ITU	✓	✓	✓	✓

Múltiples Cámaras	Soporta 2 a la vez	Soporta 2 a la vez	Soporta 2 a la vez	Soporta 2 a la vez
Subsistema Display	✓	✓	✓	✓
Interface LCD and salida de TV	✓	✓	✓	✓
Display Serial Interface (DSI)	✓	✓	✓	✓
McBSP1/2/3/4/5	✓	✓	✓	✓
McSPI1/2/3/4	✓	✓	✓	✓
Controlador High-Speed USB OTG	✓	✓	✓	✓
Controlador High-Speed USB Host	✓	✓	✓	✓
HDQ/1-Wire	✓	✓	✓	✓
UART1/2	✓	✓	✓	✓
UART3/IrDA/CIR	✓	✓	✓	✓
I2C1/2/3	✓	✓	✓	✓
MMC/SD/SDIO1/2/3	✓	✓	✓	✓
GP Timer (x12)	✓	✓	✓	✓
Watchdog Timer (x2)	✓	✓	✓	✓
32-kHz Sync Timer	✓	✓	✓	✓
GPIO	✓	✓	✓	✓
General-purpose Device	✓	✓	✓	✓

De acuerdo a toda la información presentada y el correspondiente análisis, se concluye que la plataforma SoC TI OMAP3530, ofrece las mayores prestaciones de la familia OMAP35x y debe ser adquirido para su análisis. Es un procesador de aplicaciones de alto rendimiento de la familia OMAP3, diseñado para ofrecer el mejor vídeo en su clase, imagen y de procesamiento de gráficos. Incluye un procesador súper escalar ARM Cortex-A8, un subsistema IVA2.2., basado en el procesador TMS32DMC64x + WLIW DSP, el subsistema POWERVR SGX^{TM} de aceleradores gráficos 2D/3D con OpenGL ES 2.0, aceleradores IVA^{TM} 2 (imagen, video, audio), un procesado de cámara ISP (Image Signal Processor), un sistema de interconexión avanzado y compatibilidad con periféricos estándares en la industria como se muestra en la tabla.

Selección Tarjeta de Desarrollo

Las tarjetas de desarrollo son entornos de evaluación hardware que se ajustan a las necesidades de los diseñadores, para la evaluación de aplicaciones generales y específicas. Fueron analizadas tres tipos de tarjetas existentes en el mercado, en la tabla 2.13 se muestran las capacidades de cada una en cuanto a factores importantes como cantidad de periféricos, dispositivos incluidos (display), conectividad, conectividad wireless, costo, soporte y herramientas de desarrollo. Estas características nos darán la pauta para

seleccionar el mejor tipo de tarjeta, con mejores prestaciones, para posteriormente analizar marcas y equipos específicos.

Tabla 2.13. Plataformas de Evaluación

Plataforma / Características	System-on-Module	Kits de Desarrollo	Módulos de Evaluación
Periféricos		**	***
Conectividad	*	**	***
Wireless		*	***
Display		**	***
Aplicaciones	*	**	***
Costo	***	**	*
Soporte	**	**	***
Herramientas de Desarrollo	*	**	***

Queda claro de los Módulos SOM no satisfarían a las necesidades del Departamento, por falta de periféricos e incluso, porque su mayoría no se encuentra a la venta ya que son incluidos como módulo de una placa base. Podría ser una ventaja en cuanto a precio, pero el presupuesto alcanza para escoger cualquiera de ellas.

Los Kits de Desarrollo, pueden integrar algunos periféricos, pero no todos los que soportan las plataformas OMAP3530. Son una solución económica de evaluación, y se los puede orientar a soluciones específicas de diseño. No todos soportan comunicación inalámbrica y eso debe ser tomado en cuenta al momento de adquirir una de estas plataformas, ya que la inclusión de este módulo puede aumentar el valor del equipo. El caso de la plataforma BeagleBoard es especial, esta se encuentra orientada a la comunidad, y a pesar de no contar con soporte oficial por parte de Texas Instruments, cuenta con una gran cantidad de foros que la soportan y tienen gran acogida por desarrolladores. Su problema radica en que no cuenta con todos los periféricos soportados por OMAP, a pesar de que se le pueden añadir módulos para obtener más funcionalidades, esta no sería la solución más adecuada.

La solución más efectiva se centra en tener una plataforma que cuente con todos los periféricos soportados, posea soporte, incluya herramientas de desarrollo, sea orientado a

muchas aplicaciones y sea asequible al presupuesto. Los Kits de desarrollo que se encuentran en el mercado tienen una gran ventaja por su precio, pero el presupuesto favorece a que se los Módulos de Evaluación puedan ser tomados en cuenta, incluyendo la adquisición de comunicación inalámbrica. En este caso analizaremos dos plataformas: *Zoom™ OMAP35x Development Kit* de LOGIC PD y *OMAP35x Evaluation Module (EVM)* de Mistral.

- **Contenido de Tarjetas**

OMAP35x Evaluation Module (EVM)

Figura. 2.4. OMAP35x Evaluation Module (EVM)

Fabricante

MISTRAL
... Partners in Real Time

Mistral Solutions - http://www.mistralsolutions.com/

Precio

Mistral
$1499
http://www.mistralsolutions.com/products/omap_3evm.php

TI eStore
$1495
http://focus.ti.com/docs/toolsw/folders/print/tmdsevm3530.html

Código

TMDSEVM3530

Contenido de OMAP35x EVM

Módulos

- OMAP EVM Main Board
- OMAP35x Processor Module
- TPS65950 Power Module
- WL1271 Daughter Card.

Software y manuales

- CD/DVD incluyendo herramientas de desarrollo WLANSDK and DVSDK
- WinCE 6.0 R3 BSP en una Tarjeta SD
- CD with Code Sourcery Toolchain
- CDs with Microsoft Visual Studio 2005 con Platform builder para WindowsCE6.0 R3
- Set-up Guide
- Hardware Users Guide, Getting Start Guide

Cables y accesorios

- Serial Cable
- USB Cable
- Ethernet Cable
- Display LCD con Touch Screen de 3.7” VGA/QVGA
- Stylus
- Cable y adaptador de energía.

Zoom™ OMAP35x Development Kit

Figura. 2.5. Zoom™ OMAP35x Development Kit

Fabricante

LOGIC

Logic PD - www.logicpd.com/ti

Precio

$995
http://www.logicpd.com/products/development-kits/texas-instruments-zoom%E2%84%A2-omap35x-development-kit#tabs-som-1

Código

TMDSMEVM3530-L

Contenido del kit de Desarrollo Zoom OMAP35x:

Módulos

- OMAP35x SOM-LV System on Module (SOM) SOMOMAP3530-10-1672IFCR SOM-LV
- LV form factor application baseboard

Módulos de expansión

- High-precision analog (HPA) adapter board (permite expansión agregando kits de desarrollo TI diferentes)

Software

- LogicLoader (bootloader/monitor)
- Windows Embedded CE 6.0 BSP
- Open source Linux BSPs
- Imagen pre instalada de SO en una Tarjeta SD

Cables y accesorios

- Cable Serial Null-modem
- Cable de Ethernet cruzado
- Cable USB A to mini-B cable
- Cable RF Coaxial
- Display LCD de 4.3" WQVGA con Touch Screen
- Antena Omnidireccional
- Cable y adaptador de energía.

- **Análisis Comparativo**

Como se pudo observar, a simple vista, estos dispositivos ofrecen cualidades similares. Ahora se compararán las funcionalidades específicas que soporta cada una, es importante analizar el tipo de soporte que ofrecen, la cantidad y calidad de periféricos, las herramientas de desarrollo incluidas, tipo de conectividad que soportan, etc.

Diferencias Importantes

Soporte

La primera diferencia que se pudo encontrar radica en el origen de las plataformas, *OMAP35x EVM*, es una plataforma producida y soportada por Texas Instruments en cooperación con Mistral Solutions, y además es utilizada para evaluar otros procesadores de Texas Instruments. Esta característica le otorga soporte total por parte de TI y es una gran ventaja sobre *Zoom OMAP35x.*

Módulos

Las plataformas se encuentran conformadas por módulos, los cuales se interconectan para formar la tarjeta de evaluación. La Placa Base, es sobre la cual se montan los demás módulos, entre los cuales se incluye el más importante, el de procesamiento, donde se encuentra la plataforma OMAP3530.

Zoom™ OMAP35x incluye sobre su módulo de procesamiento (OMAP35x SOM-LV), la conectividad Wireless. *OMAP EVM* por su parte incluye, un módulo especial (WL1271 Daughter Card), el cual es añadido a su placa base para este cometido. *OMAP EVM*, además adiciona el módulo TPS65950 Power Module, el cual se encarga de la generación y distribución de varios requerimientos de alimentación para la tarjeta.

Periféricos

Las plataformas incluyen varios periféricos que hacen posible la evaluación de codecs en el procesador. Todos se encuentran adheridos a una placa base de pruebas. Además de periféricos, la tarjeta contiene conectores que adaptan los demás módulos, conectores para añadir periféricos no incluidos con el paquete y pines de prueba. La tabla 2.14[40-42] presentada a continuación, indica los diferentes periféricos, pertenecientes a cada dispositivo.

Tabla 2.14. Tabla Comparativa: Periféricos

	INTERFACES	**OMAP35x EVM**	**OMAP35x Development Kit**
Display	LCD and Touch Screen	1	1
Audio	Line In	1	1

	Line Out	1	1
Video Inputs	Component	1	
	CVBS	1	
	S-Video	1	
Video outputs	CVBS	1	
	S-Video	1	
	Micron Camera Module Connector	1	
	DVI-D	1	
	LCD con Touch Screen	1	1
USB	High Speed USB OTG 2.0 mini-AB	1	1
	High speed USB Host interface	1	1
	USB B		1
Ethernet	RJ45 Ethernet	1	1
Serial	UART	2	1
Wireless	Ethernet 802.11b/g /n	1	
	Ethernet 802.11b/g		1
	Bluetooth 2.0 + EDR	1	1
	Antena	1	1
Tarjetas de Memoria	MMC/SD/SDIO	1	
	MMC/SD		1
	Slot de tarjeta CompactFlash Tipo I		1
Debug	JTAG	1	1
	ETM	1	1
	High-density breakout board		1
Fuente de Poder	DC Jack	1	1
	Entrada de Batería	1	
Key Pad	Teclado 3x5	1	
	Botón System Reset	1	1
	Botón Warm Reset	1	
	Botón System power on/off	1	1
	Botón System wake-up		1
Conectores de Expansión	De proposito general	1	
	High-precision analog		1

Software

Las plataformas incluyen software para desarrollo, se observa que ambos incluyen binarios de Linux y WinCE. En el caso de *Zoom™ OMAP35x*, se incluye el programa LogicLoader, el cual es capaz de bajar Sistemas Operativos y Aplicaciones a la plataforma. *OMAP EVM* incluye DVSDK para el desarrollo de aplicaciones Linux, para el desarrollo de aplicaciones Windows incluye el IDE Microsoft Visual Studio.

Ventajas y Desventajas

Se presentaron las características sobresalientes de cada una de las plataformas, es momento de analizar cuáles de estas diferencias son beneficiosas o no y como influencian en el desarrollo de aplicaciones, se pueden observar sus ventajas y desventajas en la tabla 2.15, presentada a continuación.

Tabla 2.15. Ventajas y Desventajas

	OMAP35x EVM	**OMAP35x Development Kit**
Características Principales		
Fabricante	***TI&MISTRAL***	***LOGIC PD***
Precio	$1499 - $1495	$ 995
Soporte	**TI y Mistral** Ventaja Plataforma producida y soportada por Texas Instruments en cooperación con Mistral Solutions	**Logic PD** Desvenaja Soportada por Logic PD, no existe soporte directo de Texas Instruments
Periféricos/Conectores/Otros		
Comunicación Serial	**COM** Desventaja Laptops no incluyen Puerto serial, es necesario adquirir un cable USB-COM para adaptación	**USB B** Ventaja Soporta comunicación serial a una PC a través de su periférico USB B
Video	**Soporta entradas y salidas de video** Ventaja Se pueden realizar toda clase de aplicaciones dirigidas a video.	**No soporta periféricos de video** Desventaja Solamente incluye el periférico de pantalla LCD
Wireless	**A través de módulo WL1271** Ventaja Se puede acceder y conFigurar de forma separada	**Incluido en SOM-LV** Desventaja No existe modularidad
	Soporta estándar Ethernet 802.11b/g/n Ventaja Soporta estándar 802.11n posee máxima velocidad con el mejor rango de señal, es más resistente a interferencias de fuentes externas.	**Soporta estándar Ethernet 802.11b/g** Desventaja No soporta estándar 802.11n
Antena	**Soporta antena On-board chip y antena externa.** **Incluye On-board chip antenna** Ventaja Soporta tipos de antena que son más utilizados en el mercado Desventaja	**Soporta e incluye antena omnidireccional.** Desventaja Soporta un tipo de antena.

	No incluye antenas externas	
Tarjetas de Memoria	**MMC/SD/SDIO** Ventaja SDIO permite la funcionalidad de extender periféricos a través del slot SD.	**MMC/SD** Desventaja No soporta SDIO
	No incluye Slot de tarjeta CompactFlash Tipo I Desventaja No soporta otro dispositivo de almacenamiento directo, aunque este puede ser realizado en la misma tarjeta SD que contenga 3 particiones.	**Incluye Slot de tarjeta CompactFlash Tipo I** Ventaja Soporta otro tipo de dispositivo de almacenamiento
Pantalla	**3.7" VGA/QVGA LCD Display con Touch Screen** Ventaja Si la aplicación va a dirigida a smartphones, la visualización sería más acertada.	**4.3" WQVGA LCD Display con touch screen** Ventaja Comodidad en la visualización.
Key Pad	**Incluye Teclado** Ventaja Ofrece más opciones para aplicaciones que utilicen comando por botones.	**No incluye teclado** Desventaja Limita a las aplicaciones
Prueba	**Incluye interfaces JTAG y ETM (Embedded Trace Macrocell Interface)** Ventaja Se puede realiza funciones de depuración y seguimiento para procesadores ARM Desventaja No incluye módulo especializado para acceso a señales del procesador	**Incluye interfaces JTAG, ETM y Módulo High-density breakout board** Ventaja Se puede realiza funciones de depuración y seguimiento para procesadores ARM Ventaja Mediante el Módulo High-density breakout board se permite el acceso a las señales del SOM-LV
Alimentación	**Incluye TPS65950 Power Module** Ventaja Se encarga de la generación y distribución de varios requerimientos de alimentación para la tarjeta.	**No incluye manejo de energía** Desventaja No existe un eficaz manejo de Energía
	Soporta fuente DC de 5V y Bateria de 4.2V **Incluye Adaptador 5VDC y entrada de batería** Ventaja Ofrece dos fuentes de energía Desventaja No incluye batería	**Soporta e incluye Fuente de poder de 5VDC** Desventaja No ofrece fuente opcional

Conectores de Expansión	**Incluye conectores de expansión dual de propósito general** Ventaja Pueden soportar lo siguiente: - Canales de entrada ADC - Interface paralela de cámara de 12 bit - Interfaces UART(3), McBSP(2), SPI(2), SD/MMC(2), I2C(2) - Buses DSS(24 bit), GPMC(16-bit) - Interface de depuración ETM	**Incluye módulo High-precision analog (HPA) adapter board** Ventaja Permite expansión agregando kits de desarrollo TI diferentes
Software Incluido		
Software Incluido	- Linux Software Package (LSP) from TI - Linux™ Digital Video Software Development Kit DVSDK package - WinCE 6.0 R3 BSP en tarjeta SD - Code Sourcery Toolchain - Microsoft Visual Studio 2005 with Platform Builder for WindowsCE6.0 R3 Ventaja Incluye binarios y herramientas de desarrollo para aplicaciones	- LogicLoader (bootloader/monitor) - Windows Embedded CE 6.0 BSP - Open source Linux BSPs - Imagen pre instalada de SO en una Tarjeta SD Ventaja Incluye programa que facilita la instalación de programas y sistemas operativos. Desventaja Incluye binarios pero no incluye herramientas de desarrollo.

2.4.2. Tarjeta Seleccionada para el Proyecto OMAP

Las características más representativas entre las tarjetas presentadas, tienen relación al soporte y periféricos. Se puede observar que *OMAP EVM*, cumple con la mayoría de requisitos antes mencionados. Además, al ser desarrollada por Texas Instruments y Mistral Solutions, presenta un mejor soporte; posee mayor cantidad de periféricos, con lo cual se aprovechan las prestaciones que posee OMAP3530 como por ejemplo audio y video que serán utilizadas en esta tesis para la primera parte del proyecto OMAP

Ambas incluyen conectividad Wireless, que es importante para seguir con el proyecto, a futuro. Pero es importante resaltar, que *OMAP35x EVM* proporciona una segunda fuente de transmisión como ventaja sobre *Zoom™ OMAP35x*. Las dos poseen soporte de Software, pero se observa que *OMAP35x EVM* incluye herramientas de

desarrollo. Cabe mencionar, que el Demo del Sistema Operativo Android se encuentra disponible para las dos versiones de plataformas.

Por las razones presentadas se decidió seleccionar, para el DEEE, la plataforma SoC OMAP3530, la cual se encuentra incluida sobre la tarjeta de desarrollo *OMAP35x EVM* de Mistral Solutions. Adquiridas por un proveedor en Ecuador a un valor superior en un 53% al indicado en este proyecto (referencias de las páginas web oficiales de las empresas desarrolladoras).

CAPÍTULO 3

ANÁLISIS DE LA PLATAFORMA Y TARJETA SELECCIONADA

Resumen: En el siguiente capítulo se realizará una descripción detallada de la arquitectura de Hardware y Software de la Plataforma de procesamiento SoC TI OMAP 3530 y Plataforma de Aplicaciones EVM OMAP35x.

3.1. JERARQUÍA DE PLATAFORMAS

La plataforma OMAP constituye en sí una Jerarquía de Plataformas, las cuales pueden ser utilizadas en diferentes etapas de desarrollo. Es importante notar, que entre un elemento de silicón y un sistema final desarrollado, existen muchas plataformas que pudieron ser creadas y utilizadas en varias etapas del proceso. Mediante la reutilización de plataformas, se reducen los costos de desarrollo, además estas pueden ser empleadas en otros sistemas e incluir hardware, software y herramientas apropiadas para su utilización. Por lo tanto, las plataformas son apreciadas en relación a su noción de reusabilidad tanto en Hardware y como en Software.

3.1.1. Plataforma Hardware (HW) y Plataforma Software (SW)

Las plataformas permiten reusabilidad. La reusabilidad se lleva a cabo modificando aplicaciones pero manteniendo su arquitectura básica, esto implica que sus principales componentes deben mantenerse fijos y dentro de cierto grado de parametrización.

Una plataforma se encuentra formada por la combinación de hardware y software. La arquitectura básica hardware consiste generalmente de un procesador programable, un subsistema de I/O y memoria, este conjunto de elementos se lo conoce como *Plataforma*

Hardware (HW). El concepto de plataforma hardware por sí solo no es suficiente para alcanzar el nivel de reusabilidad necesaria para aplicaciones [25].

La plataforma HW debe ser abstraída a un nivel, donde el software de aplicación vea una interface de alto nivel con el hardware, la cual es llamada API (Application Program Interface). Existe una capa de software que es utilizada para realizar esta abstracción, llamada *Plataforma Software* (SW). Esta capa envuelve las diferentes partes de la plataforma HW: los procesadores programables y el subsistema de memoria, a través de un Sistema Operativo en Tiempo Real (RTOS), el subsistema de I/O a través de los Controladores de Dispositivos o Drivers, y la conexión de red a través del Subsistema de Comunicación de Red. El API en la única representación abstracta de la plataforma HW [25]. Con la capa de abstracción definida las aplicaciones de software pueden ser reutilizadas en diferentes instancias de plataformas.

3.1.2. Jerarquía OMAP

La Jerarquía de Plataformas OMAP (ver Figura 3.1) [24], desde abajo hacia arriba, consta de las siguientes plataformas: Tecnología de Silicón, Librería ASIC y herramientas, Plataforma SoC, Plataforma de Aplicaciones, y Diseño de Referencia. El análisis será mayormente enfocado a los niveles superiores, ya que es frecuente que los niveles más bajos que se acercan a la implementación física, cambien debido a los avances en la tecnología, mientras que los niveles más altos tienden a ser estables a través de las versiones del producto[43].

Tecnología de Silicón

La Tecnología de Silicón es la capa de más baja en la jerarquía de plataformas. Esta contiene la base hardware. Comprende los dispositivos electrónicos semiconductores que componen a los circuitos integrados, los transistores. Estos integrados pueden formar circuitos reutilizables.

Librería ASIC y Herramientas

El diseño de un ASIC se basa en componentes predefinidos llamados células, los cuales son distribuidos como IPs y deben ser definidos junto a librerías que contienen diferentes aspectos del mismo, como comportamiento en simulación, parámetros de temporización, datos físicos para el diseño, parámetros de ruido para chequeo de integridad de la señales, etc [43].Junto con la Tecnología de Silicón, representan los niveles más bajos de plataforma HW.

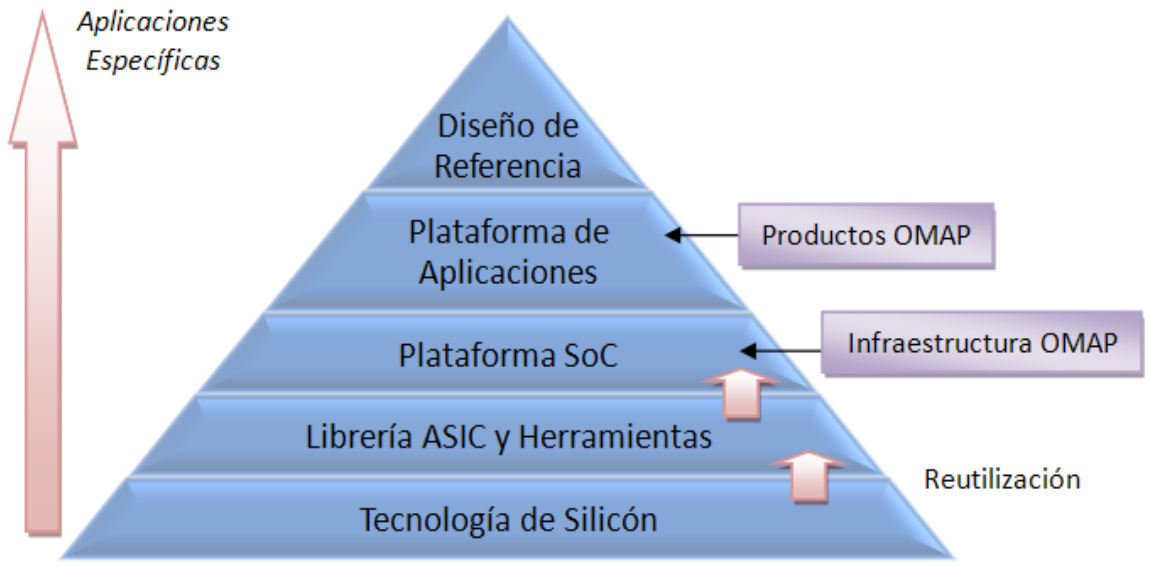


Figura. 3.1. Jerarquía de Plataformas en los Procesadores OMAP

En cuanto a Plataforma SW, se encuentra descrita por la arquitectura de set de instrucciones y lenguaje de ensamblador[24], que son las herramientas software que abstraen el uso de HW al lenguaje de máquina, y son el más bajo nivel de plataforma software seguido por los lenguajes de alto nivel.

Plataforma SoC

El hardware y software de OMAP se encuentran desarrollados sobre una infraestructura llamada Plataforma SoC. Esta infraestructura es la plataforma HW, y cuenta con al menos una librería de Componentes de módulos hardware reutilizables y una arquitectura para su interconexión, es decir una serie de reglas que determinan cuales componentes pueden agruparse entre sí en un producto, y definen su interconexión [24].

Las plataformas SoC OMAP se encuentran definidas como una familia de procesadores, los cuales son desarrollados por diferentes equipos de desarrollo, desde fabricantes de dispositivos móviles, desarrolladores de software y equipos internos de TI, y juntando esfuerzos crean una reutilización extensiva que es portable a través de sus diferentes familias de plataformas SoC. La plataforma *OMAP3530* seleccionada para el proyecto es una plataforma SoC, pero las aplicaciones que se realicen sobre esta pueden ser portadas a cualquiera de su familia.

Plataforma de Aplicaciones

La Plataforma de Aplicaciones es un producto OMAP, el cual es la base para la mayoría de Sistemas Embebidos, que generalmente utilizan una o más de estas plataformas. La Plataforma HW se encuentra compuesta generalmente por una pieza estándar de Hardware, típicamente incluyen un procesador y periféricos. Además incluyen una Plataforma SW compuesta por software de bajo nivel y entorno de desarrollo.

Las Plataformas de Aplicaciones son importantes ya que ayudan a amortizar los costos del desarrollo circuitos integrados para aplicaciones específicas a niveles de DSM y por lo tanto son la mejor opción para el desarrollo de aplicaciones que requieran un óptimo manejo de energía, rendimiento y superficie.

La tarjeta de desarrollo *OMAP35x EVM*, es una plataforma OMAP que se encuentra a nivel de jerarquía de Plataforma de Aplicaciones, ya que proporciona una placa base con procesador, periféricos y manejo de energía, y permite el desarrollo de implementaciones que pueden ser efectuadas en dispositivos finales.

Diseño de Referencia

La plataforma superior, que es el Diseño de Referencia, se la conoce también, como plataforma de Sistema, y sus usuarios (desarrolladores) pueden aplicarla rápidamente con mínimo esfuerzo para construir un producto completo.

Las plataformas HW incluyen múltiples circuitos integrados heterogéneos. Pueden desarrollarse muchas aplicaciones como procesadores de banda base, procesadores de aplicaciones, Circuitos para gestión de energía y RF. Incluyen código que controla todos los aspectos del sistema, desde drivers del dispositivo, hasta la interface de usuario. [24]

Se puede observar que al nivel más alto de jerarquía las plataformas se orientan a aplicaciones específicas. Texas Instruments enfoca estas plataformas a clientes que necesiten el rápido desarrollo de productos y una de sus mayores ventajas es que brinda una gran base de conocimientos tanto para sus clientes como para su equipo de trabajo TI.

3.2. PLATAFORMA HARDWARE (OMAP HW)

La plataforma SoC OMAP3530 es un procesador de aplicaciones de alto desempeño basado en la arquitectura de la familia OMAP™ 3 de Texas Instruments. Se encuentra diseñado para proveer procesamiento de alto desempeño en video, imagen, gráficos y comunicaciones, soportando Streaming de video, videojuegos para móviles 2D y 3D, Videoconferencia y alta resolución de imagen fija [44]. Su arquitectura se encuentra conformada por un procesador de propósito general GPP (General Purpose Processor): ARM Cortex™-A8, un procesador DSP (Digital Signal Processor): TMS320DMC64X+ VLIWX, interfaces de memorias y periféricos para aplicaciones. A continuación se muestra una tabla de resumen (Tabla 3.1[44]) de las características más representativas de OMAP3530:

Tabla 3.1. SoC OMAP3530 Periféricos

Característica	DSP	GPP
CPU	TMS320C64x+™	ARM Cortex-A8
Frecuencia	520 MHz	720 MHz
On-Chip L1/SRAM	112 KB	32 KB
On-Chip L2/SRAM	96 KB	256 KB
RAM(KB)	64 KB	
ROM	16 KB	32 KB
EMIF	1 32-Bit SDRC,1 16-Bit GPMC	

Tipos Memoria Externa	Soporta LPDDR, NORFlash, NANDflash, OneNAND, Asynch SRAM
DMA(Ch)	64-Ch EDMA,32-Bit Channel SDMA
Video Port (Configurable)	1 Salida Dedicada,1 Entrada Dedicada
Acelerador Gráfico	1
Pin/Package	423FCBGA, 515POP-FCBGA
POP Interface	Yes (CBB)
Core Supply (Volts)	0.8 V to 1.35 V
IO Supply(V)	1.8 V,3.0 V (MMC1 Only)
Rango de Temperatura de Operación(°C)	0 to 90,-40 to 105

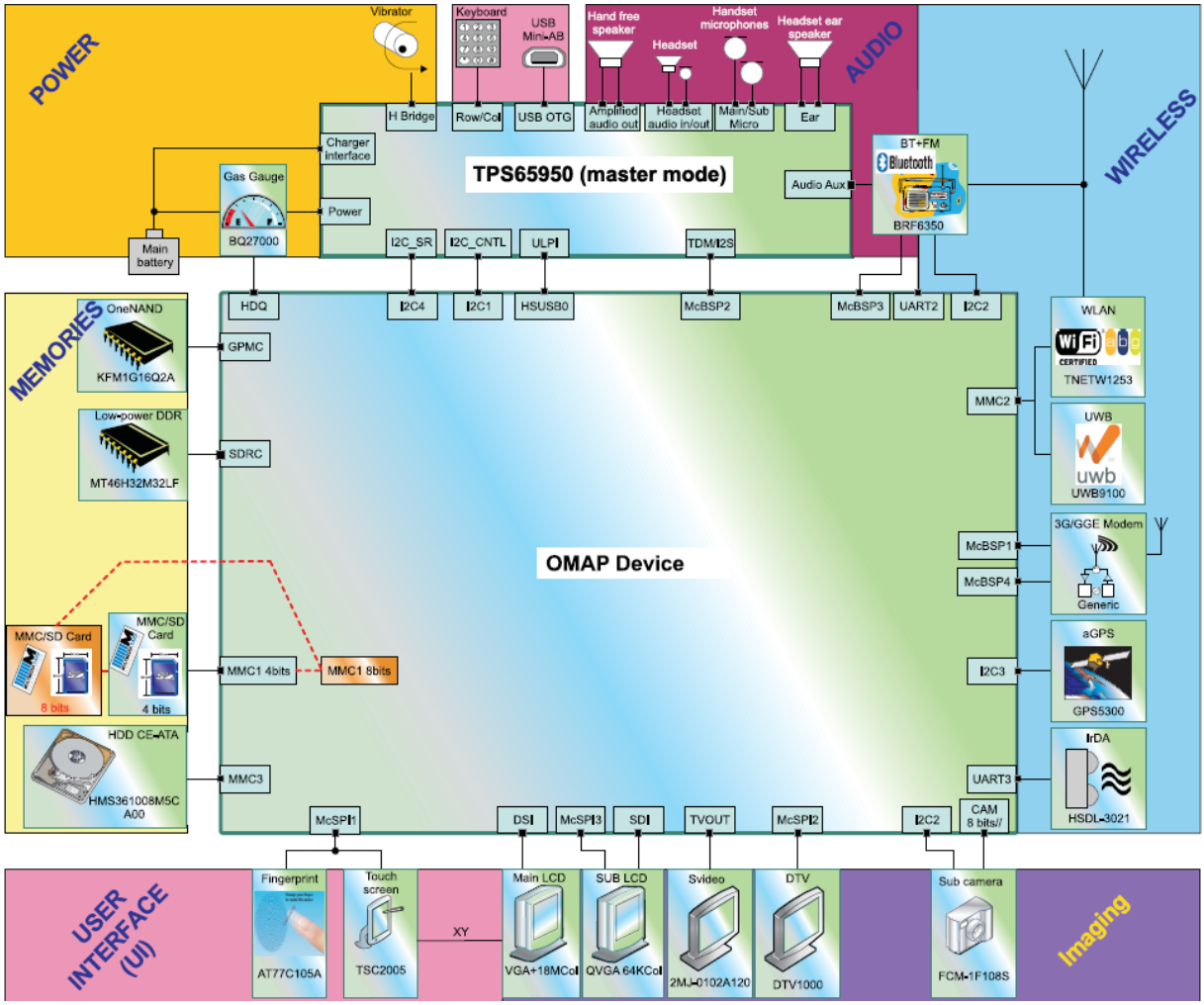


Figura. 3.2. Esquema Funcional SoC OMAP3530

En la Figura 3.2[44], se muestra el esquema funcional del SoC OMAP3530, que es una visión general del entorno OMAP3530. Se puede apreciar que se encuentra asociado a al dispositivo TPS65950, el cual es un circuito integrado de potencia que soporta la alimentación de otros dispositivos adicionales. Texas Instruments ofrece una solución global para dispositivos de aplicaciones, utilizando OMAP3530 junto con TS65950.

La plataforma OMAP35x viene en tres diferentes tipos de encapsulado. El procesador OMAP3530 adquirido para el proyecto tiene encapsulamiento CBB[trm] (12 x 12 mm), que posee una interface de tipo POP (Package-on-Package), el cual define conexiones BGA (Ball Grid Array) (conexiones en forma de soldaduras como bolitas de estaño que sirven para fijar dispositivos pequeños a placas base[45]). En este caso OMAP 3530, posee 515 bolitas de estáño, dispuestas en su cara posterior (calibre 0.4mm) para conexión a placa base y en su cara superior permite apilar dispositivos de memoria, los cuales se conectarán directamente con las interfaces de memoria externa propias de OMAP3530. Esta distribución puede ser observada a continuación (Figura 3.3[44]):

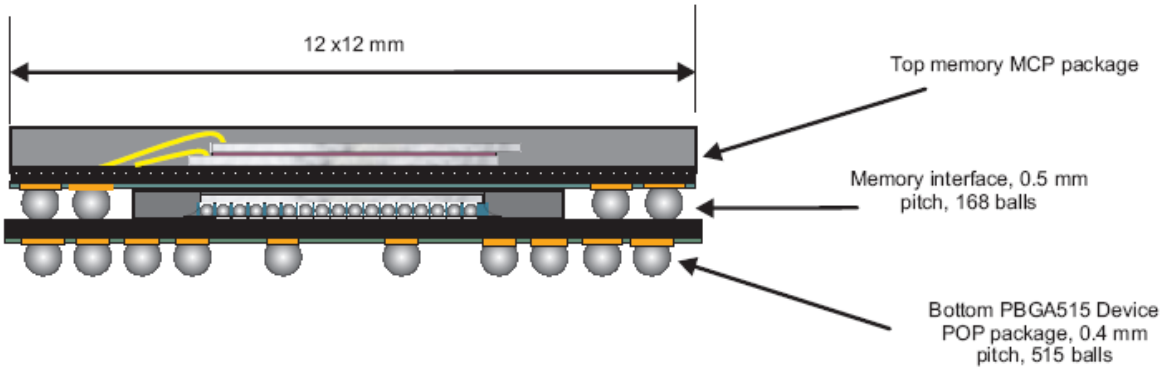


Figura. 3.3. Encapsulado SoC OMAP3530

La plataforma Hardware OMAP3530, se encuentra conformada por varios subsistemas, para su estudio han sido clasificados y agrupados de la siguiente manera:

- Subsistema de Procesamiento
    - Subsistema MPU (Microprocessor Unit), el cual se basa en el microprocesador ARM® Cortex™-A8.
    - Subsistema IVA2.2 (Image Video and Audio Accelerator) con el procesador digital de señales C64x+.

- Subsistema de Memoria
    - Subsistema Memoria On-Chip
    - Controlador GPMC (General-Purpose Memory Controller)
    - Subsistema SDRAM Controller (SDRC)
    - Módulo SDMA (System Direct Memory Access)
- Subsistema de Interconexiones
    - Interconexiones de chip Nivel 3 (L3) y Nivel 4 (L4) para transferencias de datos de alto ancho de banda entre controladores de memoria interna y externa, y periféricos on-chip.
- Subsistema de Procesamiento Gráfico
    - Subsistema Acelerador Gráfico POWERVR SGX
- Subsistema Multimedia
    - Procesador ISP (Image Signal Processor) para la cámara y el Subsistema de Display.
- Gestión de Energía Integral
    - Control de Encendido, Reset y Reloj.
    - SmartReflex™ technology
- Periféricos
    - Soporte de Periféricos

La Arquitectura de Hardware del procesador OMAP3530, puede ser apreciada en la Figura 3.4 [44]. A continuación se realizará una breve descripción de los subsistemas que componen a la plataforma, para mayor información de la arquitectura del sistema, por favor referirse al documento "OMAP35x Applications Processor Technical Reference Manual"[44], propio de Texas Instruments.

3.2.1. Subsistema de Procesamiento

Subsistema de Procesamiento de Propósito General GPP

Unidad de Microprocesador MPU [44] es el subsistema macro que maneja al subchip ARM, propio de la familia de procesadores diseñados por la empresa Acorn Cmputers y desarrollados por la compañía Advanced RISC Machines Ltd. ARM es utilizado como Procesador de Propósito General dentro de la arquitectura OMAP. En el módulo MPU

incluye lógica adicional para comunicación con su protocolo, emulación, manejo de interrupciones y mejoras en depuración. MPU se encarga de realizar las transacciones entre el procesador ARM, la interconexión L3 y el Controlador de Interrupciones.

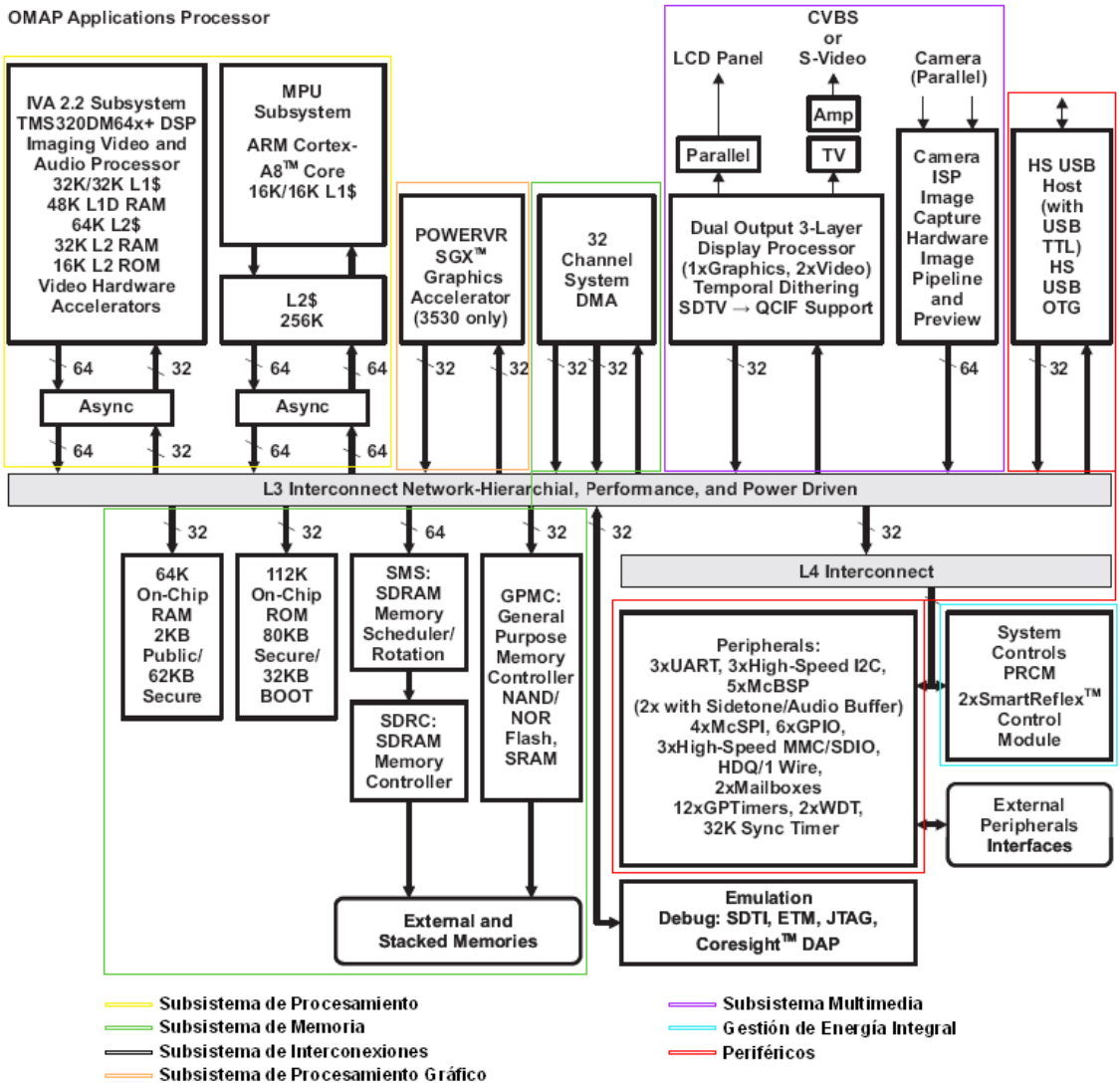


Figura. 3.4. Esquema de Bloques SoC OMAP3530

El Subchip ARM comprende el Microprocesador ARM ® Cortex TM-A8. El conjunto de instrucciones ISA (Instruction Set Architecture) soportadas son el set de instrucciones estándar de ARM, Thumb-2™, Jazelle® RCT Java accelerator. El ISA ARM incluye tecnología existente en la arquitectura ARMv7, tecnología Thumb-2™ para densidad de código y optimización en desempeño, además incorpora Jazelle® RCT para aceleración de compiladores utilizando una maquina virtual Java [32].

ARM provee alta capacidad de procesamiento e incluye tecnología NEON para aceleración multimedia móvil mediante su arquitectura SIMD (Single Instruction, Multiple Data). Además incluye memorias cache a dos niveles, de Nivel 1: cache de instrucciones de 16-KB y de datos de 16-KB, y Cache Nivel 2 de 256-KB [44].

El sub chip ARM Cortex TM-A8 incluye soporte para emulación, seguimiento y depuración de señales a través de los módulos In-Circuit Emulation Crusher (ICECrusher™), Embedded Trace Macrocell (ETM). ICECrusher™ funciona reemplazando la tarjeta del procesador con un dispositivo diferente que actúa o emula el dispositivo original, pero que contiene pines adicionales que hacen posible que estructuras internas dentro del dispositivo puedan ser visibles [44] (por ejemplo estructuras de buses internos). ETM proporciona un alto ancho de banda en el seguimiento de instrucciones para depuración de secuencias a través de sus pines de acceso [46]. Además, TI implementa JTAG (Joint Test Action Group), utilizado para emulación basado en escaneo, se comunica directamente con el procesador evitando un dispositivo de emulación, se puede mover datos dentro o fuera del dispositivo sin interrumpir su funcionamiento[47].

Subsistema de Procesamiento Digital de Señales DSP

El subsistema IVA2.2 [44] (Image Video and Audio Accelerator) se basa en un Megamódulo DSP, que es un Procesador Digital de Señales TMS320DMC64X+ VLIW DSP core, propio de TI. El cual incorpora aceleración de video mediante hardware dedicado a estimaciones de movimiento (Improved motion estimation (iME)) para codificación y decodificación de video. Integra un módulo hardware dedicado (Improved loop filtering (iLF)) para efectuar filtrado de video para codificación y decodificación. Y además incluye Secuenciador de video que se encarga de tareas de mantenimiento para los aceleradores de videos implementados en IVA2.2. Los codecs que son soportados por IVA2.2 son H264, H263-MPEG4, WMV9, REAL8 y 9[44].

Incluye memoria cache local, la cual se encuentra distribuida jerárquicamente en dos niveles L1 y L2. La memoria L1 es de tipo RAM y es utilizada de dos formas, como memoria directa dedicada a cache, manejada por el controlador de memoria de programa y como memoria dedicada a instrucciones de programa, utilizada por el controlador de memoria de datos para leer y escribir datos para/hacia memorias locales[44]. La memoria

L2 puede ser RAM/ROM y es un espacio de memoria/cache de 32KB y 16KB respectivamente, destinados a programas o datos[44].

Comunicación Interprocesador (IPC) [44]

El Módulo de Comunicación de Interprocesador IPC (Interprocessor Communicaction) es el que permite comunicación entre los procesadores on-chip del dispositivo. La comunicación se la realiza a través de un mecanismo de interrupciones de cola buzón (queued mailbox-interrupt). Este mecanismo permite que el software establezca un canal de comunicación entre los dos procesadores a través de registros y señales de interrupción asociadas (MAIL_U0_MPU_IRQ para MPU, MAIL_U1_MPU_IRQ para IVA2.2), mediante el envío y recepción de mensajes (mailboxes).

3.2.2. Subsistema de Memoria

Subsistema de Memoria On-Chip [44]

La configuración de memoria on-Chip OCM (On-Chip Memory) ofrece recursos para programación y almacenamiento de datos: 112 32 KB ROM, 64KB RAM. Cuenta con dos controladores de memoria OCM_RAM y OCM_ROM conectados al RAM Y ROM on-chip respectivamente, los cuales tienen su interface dedicada hacia la interconexión L3.

Muchos periféricos pueden acceder a la RAM a través de del Procesador Gráfico, Subsistema MPU, DMA, subsistema de cámara y display, IVA2 y USB (Universal Serial Bus). La memoria ROM es utilizada como área donde se guarda el programa de arranque o buteo, puede ser para código de buteo directo o para buteo desde una flash NAND externa.

Controladores de Memoria Externa

- **Controlador de Memoria de Propósito General GPMC [44]**

El GPMC (General-Purpose Memory Controller) es un controlador de memoria unificado UMC (Unified Memory Controller) propio de la familia OMAP 2 el cual se encarga de la interconexión de dispositivos de memoria externos como: Memorias SRAM asincrónicas, dispositivos ASIC, flash NOR, Flash NAND o dispositivos pseudo SRAM.

- **Subsistema SDRAM Controller (SDRC) [44]**

El subsistema SDRC provee conectividad de alto rendimiento entre el dispositivo OMAP y memorias externas SDRAM, que pueden ser memorias de tipo Mobile Single Data Rate SDRAM (M-SDR) y Low-Power Double Data Rate SDRAM (LPDDR). Además trabaja junto con el módulo Planificador de Memoria SDRAM SMS (SDRAM Memory Scheduler) el cual implementa calidad de servicio, optimiza la latencia y ancho de banda utilizado e incluye el módulo VRFB (Virtual Rotated Frame-Buffer) para facilitar rotaciones de gráficos 2D.

- **Módulo de Acceso Directo al Sistema de Memoria (SDMA) [44]**

El Módulo SDMA (System Direct Memory Access) ejecuta la transferencia de datos de alto rendimiento entre memorias y periféricos, sin la necesidad del apoyo de la Unidad de Microprocesador MPU, durante dicha transferencia. Una transferencia DMA es programada a través de un canal lógico DMA, que permite una transferencia óptima adaptada a los requisitos de la solicitud. Un ejemplo de su uso se puede dar cuando se necesita copiar datos desde un dispositivo externo (micrófono) a una memoria DRAM externa.

3.2.3. Subsistema de Interconexiones

Son llamadas Sistemas de Interconexión Chip-Level, y consisten en una Interconexión L3 y cuatro Interconexiones L4, las cuales se encargan de manejar las comunicaciones entre módulos y subsistemas del dispositivo. Las interconexiones L3 y L4 describen espacios de memoria de acceso a registros del módulo y registros de configuración.

Las interconexiones L3 y L4 son sistemas de redes de comunicación on-Chip NoC (Network-on-Chip) para SoC avanzados, los cuales son creadas por la compañía Sonics, Inc[48]. Texas Instruments implementa IPs de Sonics con su tecnología NoC, para proporcionar comunicaciones inteligentes dentro de su estructura.

Interconexión L3 [44]

Es una instancia de SonicsMX, la cual es marca registrada de Sonics, Inc. Maneja transferencia de datos de muchos tipos. Se encarga especialmente de intercambios entre el system-on-chip y memorias externas. Su bus de datos tiene un ancho de 64 bits del origen a su destino. Posee un espacio de memoria 16MB para registros de control que permiten la configuración de los parámetros de Subsistema de Interconexión L3.

L3 se encarga de interconectar IP Cores en una topología flexible a bajo consumo y alto rendimiento. Innovadoras estructuras físicas y protocolos avanzados garantizan ancho de banda y latencia personalizadas para cada IP Core, asegurando conexiones dedicadas entre IP Cores y conexiones lógicas sobre interconexiones compartidas. Además posee características de seguridad, iniciador (módulo que inicia pedidos de lectura y escritura al chip interconectado) autorizado, firewalls basados en regiones distribuidas para compartir recursos de sistema y gestionar la protección.

Interconexión L4 [44]

Es una instancia de Sonics3220, la cual es marca registrada de Sonics, Inc. La L4 del sistema maneja la transferencia de datos a los periféricos, soportando un bus de datos de ancho 32-bits de transferencias de datos. Se halla optimizada para soportar la interconexión de varios periféricos, los cuales se encuentran en diferentes *Dominios de Energía*.

Una manera efectiva para minimizar el consumo de energía del dispositivo, es colocar los módulos que lo conforman en distintos Dominios de Energía. Un dominio de energía es una sección del dispositivo el cual se encuentra alimentado de forma independiente y dedicada, puede ser encendido o apagado sin afectar otras partes del dispositivo.

L4, se encuentra formada por 4 interconexiones: L4-Core, L4-Per, L4-Wakeup y L4-Emu, cada una se encuentra en los Dominios de Poder CORE (mayoría de los periféricos), WKUP (Wakeup), PER (periféricos que no pertenecen a Dominio Core) y EMU (emulación) respectivamente. Los cuales poseen registros de memoria para configuración y direccionamiento.

3.2.4. Subsistema de Procesamiento Gráfico

El subsistema de procesamiento gráfico GPU (Graphics Processing Unit) se basa en el Acelerador Gráfico SGX[44], el cual es una instancia de POWERVR^{TM} SGX530, propio de la compañía Imagination Technologies, implementado en OMAP3530 por Texas Instruments. El subsistema se encarga de la aceleración de aplicaciones gráficas bidimensionales (2D) y tridimensionales (3D). La arquitectura POWERVR^{TM} SGX530 v1.2.1 es escalable y puede ser aplicada a varios segmentos de mercado desde dispositivos móviles a High-end desktop graphics. Entre sus aplicaciones tenemos, PDAs, celulares, consolas de video juego portátiles, etc[44].

SGX utiliza dos tipos de interconexiones hacia L3 una maestra y una esclava. Proporciona soporte API estándar en la industria para Direct3D Mobile (API que provee soporte en aplicaciones de gráficas 3D en plataformas Windows CE [49]), OpenGL ES 1.1 y 2.2, y Open VG v1.0.1(API que provee aceleración de Hardware de bajo nivel para librerías de gráficos vectoriales [50]).

Se caracteriza por realizar el procesamiento de varios tipos de datos para procesamiento de imágenes como: pixel y vertex, mediante los cuales se construye una imagen. La información de una imagen es manejada por *Data Masters*. Vertex indica la representación de una imagen a través de puntos vectoriales en el espacio y pixel es la representación de una imagen mediante un conjunto de cuadros de colores[44].

La información de una imagen procesada en forma vectorial y luego se inicia el proceso de rasterización el cual hace que una imagen descrita en formato vectorial se convierta en un conjunto de pixeles que pueden ser representados en una pantalla. Incluye una unidad de procesamiento (Universal Scalable Shader Engine (USSE)), la cual se encarga de realizar procesamiento de vértices y pixeles mediante *Shading* que agrega efectos y características a la imagen. Además se encarga del procesamiento de imágenes y video[44].

3.2.5. Subsistema Multimedia

Procesador de Imágenes para Cámara [44]

El procesador de señales de imágenes ISP (Image Signal Processor) es empleado exclusivamente para el manejo de cámara, soporta múltiples formatos y opciones de interfaces para una amplia variedad de sensores de imagen. El procesador ISP es un componente esencial para aplicaciones que incluyan imágenes y video. Puede ser utilizado para vista previa de video, grabación de video, captura de imagen inmóvil con o sin zoom. Proporciona la interface del sistema y la capacidad de procesamiento para conectar módulos sensores digitales de imágenes al dispositivo. Los formatos de imágenes soportadas pueden ser RGB, RAW, YUV y JPEG

Subsistema de Pantalla[44]

El Subsistema de Pantalla maneja dos funciones principales, soporta LCD y Pantalla de TV. Provee la lógica para presentar un frame de video desde el buffer de memoria (SDRAM o SRAM) a un panel LCD (Liquid-Crystal Display) o u set de Televisión.

Para LCD, soporta dos tipos de tecnologías display de matrices activas (pantallas de alta resolución) y display de matrices pasivas (pantallas monocromo). Posee un Controlador de Pantalla que provee distintos modos de pantalla (XGA(1024×768), WXGA(1440×900), SXGA(1280×1024), HD 720p. Además incluye tecnología TI FlatLink™3G para pantallas de teléfono de alta resolución. Para TV, soporta codificación de video que convierte señales de video RGB a los estándares de video análogo NTSC (National Television System Committee) y PAL (Phase Alternating Line). Ofrece codificación de video tipo Luminancia (componente de video: luz) y Crominancia (componente de video: color) para proveer formatos de video tipo CVBS (Composite Video Baseband Signal) y S-video (Separate Video).

Soporta el protocolo RFB (remote framebuffer) para acceso remoto a interfaces de usuario gráficas, a través de su interface RFB (RFBI). Además soporta funciones de video como solapamiento de imágenes (Picture-in-Picture), conversión espacio-color, rotación, rotación de fase de color y redimensionamiento.

3.2.6. Gestión de Energía Integral

Subsistema Administración de Energía, Reset y Reloj

El subsistema PRCM (Power, Reset and Clock Management) es el módulo central para la administración de energía del dispositivo, además es el módulo central de control para el reloj, reset y señales de administración de energía en el dispositivo. Asegura una reducción del consumo de energía dinámica y fugas de corriente estática para extender la vida de la batería en el producto final, garantizando un funcionamiento óptimo del dispositivo con consumo energético significativamente reducido [44].

Implementa Tecnología SmartReflex™ que es una gama de hardware inteligente y adaptativo, ademá implementa técnicas software para control dinámico de voltaje, frecuencia, potencia, modos de operación y temperatura del dispositivo [51].

3.2.7. Periféricos

El dispositivo es compatible con un amplio conjunto de periféricos para proporcionar interfaces flexibles y de alta velocidad, y recursos de programación on-chip. A continuación se proporciona una lista y descripción de los periféricos disponibles en dispositivos OMAP3530 (Tabla 3.2) [44].

Tabla 3.2. SoC OMAP3530 Periféricos

Tipo	Nombre	Cant	Descripción
Comunicación Serial	Multi-channel Buffered Serial Ports (McBSPs)	5	McBSPs proporciona una interfaz serial directa full-duplex entre el dispositivo y otros dispositivos en el sistema como códecs de audio y voz y otros chips de aplicación. McBSP1, McBSP2 y McBSP3 sirven como puertos seriales de propósito general, McBSP2 y McBSP3 incluyen capacidad adicional de bucle audio invertido.
	Multi-channel Serial Port Interface (McSPI)	4	McSPIs proporciona una interfaz maestro/esclavo para dispositivos SPI (Serial Peripheral Interface).

	High-speed Multi-port USB Host Controller	1	Controlador host USB2.0 de alta velocidad con tres puertos host, cada uno ofrece transacciones de datos en high-speed (hasta 480 Mbps), full-speed (12 Mbps) y low-speed (1,5 Mbps) respectivamente. En modo high-speed el controlador de host USB utliza una interfaz USB PHY (capa física) externa ULPI (UTMI (USB 2.0. Transceiver Macrocell Interface) low-pin Interface) de 12 pines u 8 pines. En los modos full-speed y low-speed utiliza una interfaz USB PHY externa serial de 6-/4-/3-pines. Además, cada puerto puede ser conectado directamente a un dispositivo USB (sin la necesidad de USB PHY) mediante un adaptador TLL (transceiver-less link logic) on-chip.
	High-speed USB OTG controller	1	Controlador OTG USB 2.0 de alta velocidad que ofrece transacciones de alta velocidad de datos (hasta 480 Mbps) en un puerto USB con controlador DMA embebido. El controlador OTG USB 2.0 de alta velocidad utiliza una interfaz USB PHY ULPI, como su interfaz externa.
	HDQ/1-Wire	1	El interface HDQ/1-Wire es compatible con el protocolo Benchmark HDQ y el protocolo Dallas Semiconductor 1-Wire. Se encarga de manejar la comunicación serial hacia dispositivos externos a traves de un solo cable de comunicación y tierra. Una aplicación típica del módulo HDQ/1-Wire es la comunicación con el circuito integrado de monitor de batería (indicador de gasolina).
	Universal Asynchronous Receiver/Transmitter (UART)	3	Interfaces de comunicación serial compatibles con el elemento de comunicaciones asíncronas TL16C550 estándar de la industria. UART1 y UART2 son interfaces de comunicación serial generales. UART3 proporciona apoyo adicional para el estándar de comunicación de datos por infrarrojo IrDA (Infrared Data Association) y CIR (Consumer Infrared).
	High-speed (HS) Inter-integrated Circuit (I^2C) Controllers	3	Interface I^2C Estándar Maestro/Esclavo de alta velocidad con soporte para modo Standar (hasta 100K bits/s), modo Fast (hasta 400Kbits s), y modo High-speed (hasta 3,4 M bits/s). Cada una de ellas proporciona una interfaz entre un host local, como el Subsistema MPU, y cualquier dispositivo I^2C-bus-compatible que se conecte a través del bus serie I^2C.
Medios extraíbles	Multimedia Card/ Secure Digital I/O (MMC/SDIO) Card Interface	1	Soporte de tarjeta de memoria MMC, tarjeta de memoria SD, o interfaz SDIO.
Misceláneos	Timers GP	12	Doce Timers de propósito general
	Watchdog timers (WDTs)	2	Dos watchdog timers
	32-kHz synchronization Timer	1	Timer de reloj 32-kHz
	General-purpose input/output (GPIO)		Pins de entrada/salida de propósito general manejados por seis controladores GPIO

3.3. PLATAFORMA SOFTWARE (OMAP SW)

La arquitectura OMAP incluye una infraestructura de software abierta que soporta el desarrollo de aplicaciones sobre una base heterogénea multiprocesador. Las tecnologías de software utilizadas por OMAP permiten a los desarrolladores de software obtener ventajas sobre su arquitectura, sin la necesidad de un conocimiento ni manejo previo de la arquitectura hardware subyacente mediante abstracción, y además brinda la posibilidad de portar dichas aplicaciones sobre diferentes plataformas OMAP.

OMAP3530 utiliza una arquitectura basada en doble procesador, uno de propósito general GPP (General Purpose Processor): ARM Cortex™-A8 y un procesador de señales digitales DSP: TMS320DMC64X+ VLIWX. Aplicaciones como full-motion video, videoconferencias, reconocimiento de voz, consolas de videojuegos, procesamiento de voz, aplicaciones multimedia, entre otros son cada vez más utilizadas en dispositivos portátiles y son posibles mediante la utilización de procesadores DSP. Además estas pueden ser utilizadas sobre un Sistema Operativo (OS, Operating System) de alto desempeño, sobre el procesador GPP.

El procesador DSP es eficiente para trabajar con código orientado a manejo datos y tareas relacionadas con el procesamiento señales digitales en tiempo real, mientras que el procesador ARM es mejor en códigos de control, desarrollo de software e integración. La configuración de doble procesador permite un menor consumo de potencia, permite además separar la interface de usuario, del procesamiento de señales [24]. Esto lo hace eficiente en el manejo de operaciones en tiempo real, por ejemplo: se puede realizar una videoconferencia, mientras se corre un programa sobre el Sistema Operativo implementado, puede ser el caso de utilizar un Programa Office, propio del Sistema Operativo Windows CE.

Una arquitectura abierta es importante ya que permite un campo de programación libre utilizado por desarrolladores de aplicaciones independientes, fortaleciendo la creación de una gran comunidad de desarrolladores para cada sistema operativo de alto nivel y fortaleciendo a compañías de software Third-party [24].

3.3.1. Arquitectura de Software OMAP 3530

En OMAP3530, el desarrollo de aplicaciones se realiza mediante la abstracción de la arquitectura de software DSP desde el entorno GPP, esto permite que el procesador ARM Cortex™-A8 utilice funciones de TMS320DMC64X+ VLIWX remotamente, y para el desarrollador la percepción de este enlace es transparente, es decir que GPP es visto como el maestro del sistema o como el procesador "host".

Texas Instruments utiliza este arquitectura basándose en el trabajo realizado por la compañía Spectron, ahora TI de Santa Barbara, para arquitecturas multiprocesador, este consiste de tres elementos [24]:

- Un conjunto de *APIs* (Application Programming Interfaces) muy bien definidas sobre un Sistema Operativo de alto nivel corriendo sobre el GPP. Alto nivel se refiere a la capacidad para el manejo de espacios de memoria virtualizada y la aplicación de una interface de usuario sofisticada.
- Un sistema de software que *vincule* aplicaciones del GPP a componentes del DSP como codecs de video y audio.
- Un *Estándar* para componentes DSP bien definido, que permita que sean fácilmente encapsulados en el framework OMAP.

Los *APIs* son utilizados para el procesamiento de video, imagen, voz y señales de audio, a través de algoritmos estándares para cada componente. El *estándar* de componentes se refiere a los estándares eXpressDSP™ propios de TI, los cuales pretenden que los componentes DSP sean llamados desde un programa C arbitrario, y que se comporten correctamente en términos de uso de la memoria y acceso a los recursos de hardware. Esto permite que los componentes se *vinculen,* a través de nodos, al sistema que hace posible la comunicación multiprocesador conocido como DSP/BIOS Bridge [24].

DSP/BIOS Bridge

DSP/BIOS Bridge es el sistema que "permite multiprocesamiento asimétrico en plataformas que contienen un procesador GPP y uno o más procesadores DSP adjuntos"[52]. DSP/BIOS Bridge es una combinación de software, para ambos procesadores que vincula ambos sistemas operativos. Este enlace de vinculación permite a

las aplicaciones de los procesadores GPP y el DSP una forma eficaz de comunicación de mensajes y datos en un dispositivo independiente. Entre sus características más representativas tenemos [53]:

- DSP/BIOS provee un *perfil estándar* de comunicación multiprocesador.
- Permite que procesadores maestros *controlen la ejecución* de procesadores esclavos. (ARM puede controlar buteo, ejecución o restablecimiento del DSP).
- Proporciona *APIs genéricos* para aplicaciones (para OS independientes y plataformas independientes).
- Oculta al sistema operativo del GPP detalles específicos de aplicaciones, necesarias para establecer comunicación con el hardware.
- *Ofrece portabilidad de aplicaciones*, las aplicaciones escritas para una plataforma específica puede trabajar directamente en otras plataformas de combinaciones de OS o plataforma, sin requerir o con menores cambios en códigos de aplicación.
- Provee *protocolos peer-to-peer* para comunicación entre múltiples procesadores en el sistema.
- Administra una arquitectura *escalable*, que permite que el sistema integrador utilice la configuración óptica que desee para footprint y desempeño.

- **Arquitectura DSP/BIOS Bridge**

En la arquitectura DSP/BRIDGE el procesador GPP es considerado el procesador host y el procesador DSP adjunto es considerado como un recurso de procesamiento que puede ser utilizado por aplicaciones y drivers corriendo en el GPP.

La abstracción que proporciona el DSP/BIOS BRIDGE brinda un enlace directo entre un programa GPP y una tarea DSP (ver Figura 3.5 [52]). El enlace de comunicación está dividido en dos sub-enlaces: Enlace Messaging (paquetes pequeños de longitud fija) y una Enlace Data Stream (múltiples, buffers grandes). Cada sub-enlace funciona de manera independiente, y cuenta con entrega de datos ordenada. Mediante este enlace GPP puede especificar que entradas y salidas son utilizadas por las tareas DSP. Generalmente las tareas DSP utilizan objetos mensajes para pasar información sobre estados de control y estatus, y objetos stream para transmisión eficiente de flujo de datos en tiempo real [52].

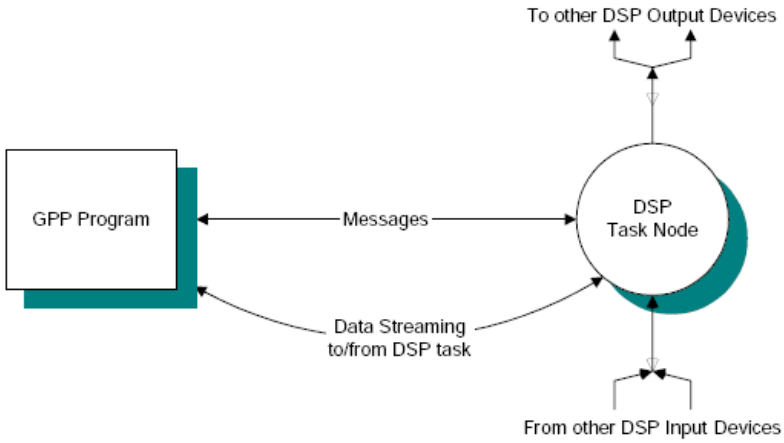


Figura. 3.5. Comunicación GPP con Nodo de tareas DSP

Los componentes de la arquitectura de software para OMAP3530 Figura 3.6, se encuentran divididos por: la parte izquierda, la configuración GPP que en este caso será el lado del procesador ARM Cortex™-A8, por la parte derecha se observa la configuración DSP que en este caso será utilizada por el procesador DSP TMS320DMC64X+ VLIWx+.

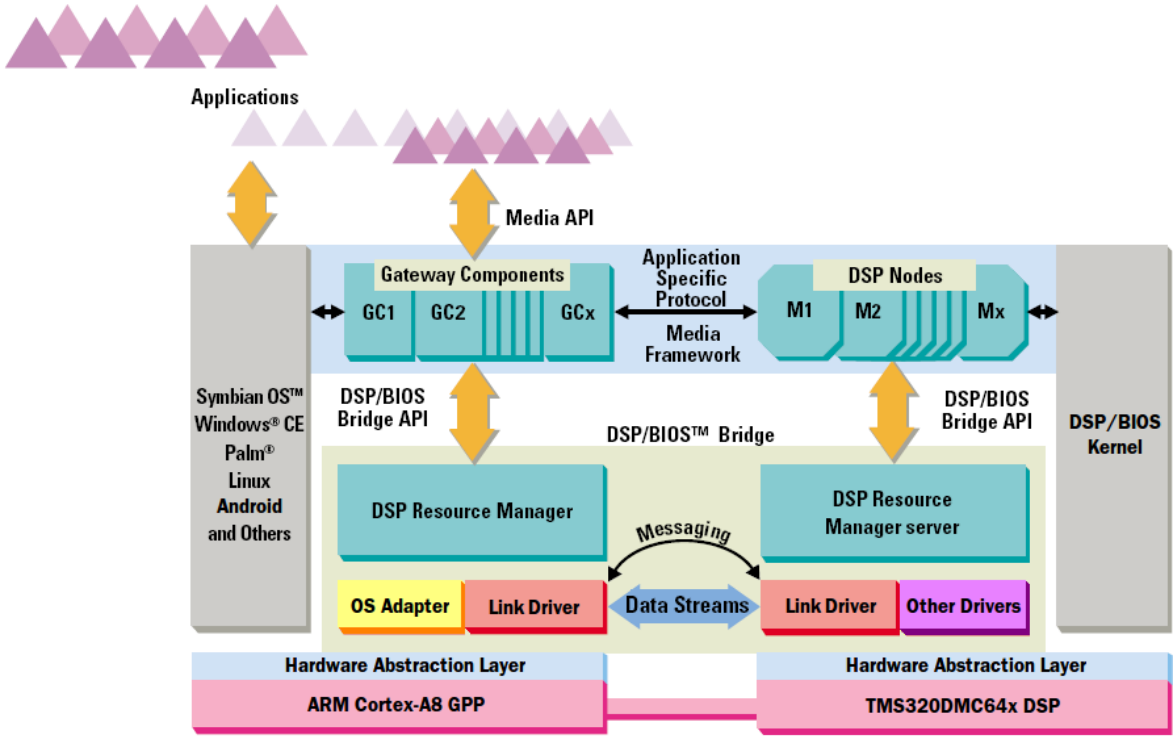


Figura. 3.6. Configuración DSP/BIOS Bridge

A continuación se analizará la distribución de dichos componentes [54]. En el lado de GPP se tiene:

- Hardware Abstraction Layer (HAL): es una capa de abstracción del hardware en el procesador ARM, funciona como una interfaz en el software y el hardware del sistema GPP.
- OS ARM: es el sistema operativo de alto nivel que corra sobre ARM, puede ser el caso de Symbian, Windows CE, Palm, Linux, Nucleus, Android, etc.
- OS Adaptor: se encarga de encapsular los servicios genéricos del SO que son requeridos por otros componentes del DSP/BIOS Bridge. Este componente exporta una API genérica que aísla a los otros componentes de los detalles de un sistema operativo específico. Todos los demás componentes utilizar esta API en lugar de dirigir las solicitudes directamente al OS. Esto hace DSP/LINK Bridge portable a través de diferentes sistemas operativos.
- Link Driver: es utilizado para interacción interprocesador de bajo nivel. Encapsula las operaciones de control de bajo nivel en el enlace físico entre GPP y DSP. Este módulo es el responsable de controlar la ejecución de DSP y transferencia de datos utilizando un protocolo definido.
- Resource Manager: es el administrador de recursos, responsable de crear dinámicamente instancias de recursos DSP para atender las solicitudes de asignación, monitoreo de recursos DSP, para carga dinámica de código DSP según sea necesario, y aplicar políticas para la gestión de los recursos DSP cuando hay conflicto entre las peticiones. Se encarga de crear instancias a bajo nivel de Nodos DSP.
- DSP/BIOS Bridge API: identifica a un conjunto de APIs exportados que son utilizados por los componentes Gateway para interactuar con el DSP/BIOS Bridge. Incluyen APIs para administración de nodos DSP, así como APIs para transferencia de datos.
- Gateway Components: se encarga de abstraer las funcionalidades proporcionadas por los nodos DSP a un simple conjunto de APIs disponibles para el desarrollo de aplicaciones. Dichos APIs pueden ser implementados a través de una serie de llamadas al DSP/BIOS Bridge API.
- Aplicaciones: son las aplicaciones finales que se encuentran disponibles para el usuario final, tales como el correo electrónico, reproductor multimedia, etc.

En el lado DSP:

- Hardware Abstraction Layer (HAL): es una capa de abstracción del hardware en el procesador TMS320, funciona como una interfaz en el software y el hardware del sistema DSP.
- DSP/BIOS Kernel: es el sistema operativo en tiempo real utilizado por el procesador DSP. Está diseñado para aplicaciones que requieren programación en tiempo real y sincronización, comunicación Host (PC) – Target (plataforma), o instrumentación en tiempo real.
- Link Driver: es el driver de comunicación DSP-GPP de bajo nivel.
- DSP Resource Manager Server: es el servidor del Gestor de Recursos que se encarga de crear, ejecutar y destruir nodos de procesamiento DSP dinámicamente. Además se encarga del enrutamiento de mensajes entre el GPP y los nodos individuales, de la alteración de prioridades en las tareas, y responde a la configuración de comandos y consultas de estado del Gestor de Recursos.
- Nodos DSP: son subprocesos de ejecución separada, que se ejecutan en DSP e implementan control o algoritmos de procesamiento de señales. Los nodos de tareas DSP, se comunican entre sí a través de mensajes cortos de longitud fija o a través de flujos de datos. Los algoritmos xDAIS son los encargados de realizar el procesamiento de señales y controlan la operación de los nodos.

Estándares eXpressDSP™ [55]

Los estándares eXpressDSP™ fueron desarrollados por Texas Instruments con el fin de aprovechar al máximo las capacidades del DSP, a través de una amplia gama de funciones multimedia sobre un dispositivo. eXpressDSP™ lleva a cabo sus funciones DSP, mediante estándares como eXpressDSP™ Algorithm Interoperability Standard (xDAIS) y eXpressDSP Digital Media (xDM). Dichos estándares entregan un conjunto de convenciones de código y APIs que hacen posible que algoritmos sean integrados rápidamente en la ejecución de aplicaciones DSP multifunción.

- **XDAIS (eXpress DSP Algorithm Interoperability Standard)**

El estándar XDAIS, también llamado TMS320 DSP Algorithm Standard, se encarga de proveer reglas y guías necesarias para habilitar la integración y ejecución de algoritmos.

Proveen cabeceras C/C++ para interfaces definidas que los algoritmos pueden implementar y las aplicaciones pueden invocar.

XDAIS hace posible que varios algoritmos coexistan en un sistema y compartan recursos, mediante la prevención de la utilización sobrecargada de recursos del sistema críticos, como son la memoria DMA y otros aceleradores. Esto es logrado por medio del manejo de interfaces. Las interfaces posibilitan que los algoritmos compartan recursos de forma segura y son definidos por el estándar que el algoritmo requiere implementar

- **xDM (eXpressDSP Digital Media)**

La norma XDM define un conjunto uniforme de APIs a través de diversos códecs multimedia para facilitar la integración y garantizar la interoperabilidad. XDM está construido sobre la especificación del algoritmo xDAIS. XDM hace posible que el framework de una aplicación soporte múltiples codecs del mismo tipo sin muchas modificaciones. Las clases de codecs soportados pueden ser codificadores y decodificadores de video, audio, voz e imagen.

Application Programming Interfaces (APIs)

En el SO GPP, DSP/BIOS Bridge agrega una API, que permite múltiples clientes, de esta forma se puede acceder a los recursos DSP. La aplicación del procesador GPP se comunica con una tarea DSP que corre sobre el procesador DSP por medio del DSP/BIOS Bridge. Desde la perspectiva del OS GPP, el DSP es tratado como un periférico más [52], por ejemplo en el OS Linux, DSP se encuentra en el directorio /dev junto a los demás periféricos.

En GPP, las aplicaciones o drivers de dispositivo pueden utilizar la API con los siguientes propósitos [52]:

- Inicializar tareas de procesamiento digital de señales en el DSP.
- Intercambio de mensajes con tareas DSP.
- Envía de buffers de flujo de datos para/desde tareas DSP.
- Tareas DSP pause resume y delete
- Realizar consultas de estado de los recursos.

Desde el punto de vista de DSP, DSP/BIOS Bridge agrega una API DSP, que permite intercambio de mensajes y bloques de datos entre el GPP y DSP. Además define una abstracción Nodo, que agrupa bloques relacionados (código y datos) y los junta dentro de un solo bloque funcional llamado *Nodo*. Los clientes GPP pueden realizar solicitudes a través de su API y en DSP se crearán nodos apropiados a las solicitudes y cuando ya no sean necesarios serán eliminados. Los nodos pueden ser utilizados con fines de procesamiento de señales así como el procesamiento de control general [52].

- **Codec Engine (CE) [47]**

Es un conjunto de APIs utilizados para instanciar y correr algoritmos xDAIS y codecs. Se puede decir que CE es un framework que hace posible que aplicaciones utilicen algoritmos y codecs de una manera rápida y efectiva. Puede manejar algoritmos tanto locales, en el mismo procesador que la aplicación (GPP), como remotos, en un procesador diferente a la aplicación (DSP).

Utiliza una interface VISA (Video, Image, Speech, and Audio) para realizar interacción entre instancias de codecs multimedia. Para que un algoritmo sea ejecutado por el CE, necesita implementar las interfaces definidas por el estándar XDM.

- **Multimedia Application Interface (DMAI) [47]**

DMAI es una interface que se encuentra como una capa delgada en la parte superior del Sistema Operativo y Codec Engine (Figura 3.7) que es utilizada para el rápido desarrollo de aplicaciones. Es una abstracción del buffer que lleva datos actuales (video, voz, audio, etc.) y además datos que describen el tipo de buffer que es utilizado por Codec Engine y controladores para realizar operaciones en los datos.

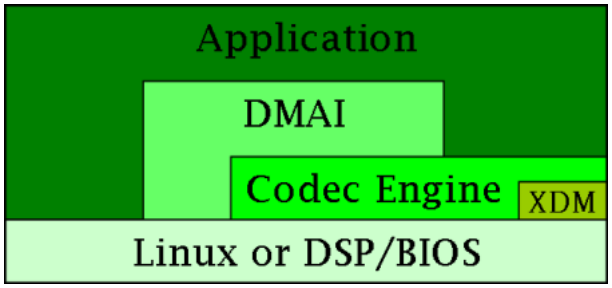


Figura. 3.7. Diagrama de Bloques DMAI

DMAI es un diseño funcional, cuando los módulos suelen describir una determinada operación (por ejemplo copiar frame usando "framecopy", el cambio de tamaño con "resize", etc), pero la implementación en el módulo puede cambiar entre los dispositivos y sistemas operativos en función de que controladores de dispositivos periféricos estén disponibles. En otras palabras DMAI no realiza abstracción de periféricos sino que abstrae las operaciones (por ejemplo "framecopy", "resize"), luego las implementa sobre los periféricos a su disposición. Esta es la clave para que las aplicaciones puedan ser portadas hacia distintos dispositivos sin ningún ajuste.

3.3.2. Sistemas Operativos Soportados

La plataforma de procesamiento TI OMAP3530 se basa en una arquitectura heterogénea multiprocesador, que consta de un procesador de propósito general y un procesador digital de señales. Sobre el porcesador TMS320DMC64X+ VLIW corre DSP/BIOS Kernel, propio de TI y sobre ARM Cortex™-A8 existen una variedad de Sistemas Operativos soportados de alto nivel y de tiempo real (RTOS), como muestra la tabla 3.3 [56]. Cada sistema operativo incluye herramientas de desarrollo SDK (Software Development Kit) que incluyen software especializado en desarrollo de aplicaciones multimedia, binaries, drivers, software para actualizaciones, además incluyen documentación y ejemplos.

Tabla 3.3. Sistemas Operativos soportados por ARM

Tipo OS	Proveedor	Producto	Observaciones
Android	TI (Community)	N/A	Gratis
Android	Mentor Graphics	Android OMAP3	$
Linux	TI (Community)	N/A	Gratis. Utiliza CodeSourcery
Linux	MontaVista	MontaVista Linux	$
Linux	RidgeRun	RidgeRun SDK	$
Linux	TimeSys	LinuxLink	$
Linux	WindRiver	WindRiver Linux	$
Windows Embedded CE 6.0	TI		Gratis
Windows Embedded CE 6.0	Adeneo	N/A	$
Windows Embedded CE 6.0	Logic PD	N/A	$ BSP; WinCE OS requires royalties.
RTOS	Green Hills	Integrity	$

RTOS	Mentor Graphics	Nucleus+®	$
RTOS	QNX	Neutrino®	$
RTOS	WindRiver	VxWorks®	$

Android [56]

Android se encuentra disponible para OMAP a través de proyectos como Rowboat que brindan soporte en dispositivos TI. En comparación con otros OS que requirieren múltiples pasos para funcionamiento, Android proporciona un entorno Redy-made que simplifica el desarrollo de aplicaciones multimedia con gráficos sofisticados o requisitos de interfaz de usuario. TI ha incluido a sus socios MentorGraphics aportando su experiencia en programación Android sobre OMAP35x.

Linux [56]

TI y sus socios ofrecen servicios y software Linux especializados en dispositivos TI, existen distribuciones libres y pagadas, que incluyen herramientas de desarrollo para aplicaciones multimedia como DVSDK (Digital Video SDK). Posee además una amplia red de apoyo en línea. Linux es el OS más utilizado en OMAP35x, actualmente existe en la red mucha información, soporte y referencias de ejemplos para el mismo.

Windows Embedded CE [56]

Es un OS optimizado para dispositivos embebidos que requieren mínimo de almacenamiento. Permite a los desarrolladores utilizar herramientas como Visual Studio para desarrollo multimedia en la plataforma incluyendo todas sus capacidades. TI ha hecho posible que WinCE porte si DVSDK para posibilitar el desarrollo de aplicaciones multimedia utilizando DSP.

RTOS [56]

Para aplicaciones que requieren operación en tiempo real, como en el sector industrial, aeroespacial, médico, automoción, OMAP35x soporta RTOS. Los socios RTOS proporcionan herramientas de desarrollo sofisticadas que permiten ejecución rápida de programas certificados conteniendo todas las capacidades de OMAP3530.

3.3.3. Proceso de Inicialización

El proceso de inicialización describe la secuencia de arranque OMAP, que es el proceso que inicia el sistema operativo que se ejecutará sobre la plataforma estudiada. Describe principalmente de 3 pasos que son: Bootloader, Kernel y Filesystem. Finalmente, luego de esta secuencia, la plataforma se encontrará cargada por el sistema operativo escogido funcionando y listo para realizar aplicaciones sobre la misma. El proceso se encuentra descrito en el siguiente gráfico:

Figura. 3.8. Secuencia de Arranque

Bootloader

Esta etapa es necesaria para inicializar el sistema y prepararlo para cargar el Kernel. El proceso inicia al ubicar el periférico el cual va a cargar el sistema, luego se obtiene el software de buteo y se lo carga. A continuación se describirá el proceso.

- **Hardware ROM Bootsystem**

Esta fase es necesaria para inicializar el hardware que cargará el bootoader contenido en un periférico. La ROM carga solamente los drivers necesarios para buteo de periféricos: Serial(UART3), SD Card(MMC1 and MMC2), Nand y USB.

El periférico a butear será seleccionado de acuerdo a la configuración del SW1 de la plataforma, el cual se encuentra en la parte superior. Las diferentes combinaciones de buteo pueden ser encontradas en [42].

- **X-Loader**

Es un software de bajo nivel que es leído desde el periférico de buteo, conFigura relojes y memorias, y prepara el sistema para cargar el bootloader U-Boot.

- **U-Boot**

Es el gestor de aranque U-Boot (Universal Bootloader) que se encarga de cargar el Kernel. Se trata de un software que puede ser manipulado a través de comandos para establecer configuraciones del dispositivo y buteo del kernel.

Kernel

Es un sistema operativo utilizado en la familia Linux, el cual se encarga de inicializar el resto de hardware y drivers. Es el encargado de gestionar recursos de hardware que serán utilizados por aplicaciones.

Filesystem

Es un sistema de archivos que estructuran la información guardada en el sistema de almacenamiento seleccionado. Se encarga de almacenar un conjunto archivos y aplicaciones de manera ramificada siguiendo un orden jerárquico.

3.4. TARJETA TI OMAP

EVM OMAP35x de Mistral es una plataforma aplicación de bajo costo y de fácil utilización. Basa su funcionamiento en el procesador TI OMAP3530. Está diseñado con una arquitectura modular y extensible. La OMAP3 EVM ayuda a los desarrolladores a aprovechar las ventajas de un procesador OMAP3 de gran alcance.

EVM incluye un conjunto de dispositivos on-board que se adaptan a una amplia variedad de entornos para aplicaciones wireless multimedia. En la Figura 3.9 [42], se puede observar el diagrama de bloques funcionales de la plataforma, que incluye módulos de procesamiento, interfaces de audio, video, memoria, de comunicación, módulos multimedia y botones.

El Módulo de Evaluación permite desarrollar aplicaciones de baja potencia, tales como: reproductores multimedia portátiles, dispositivos de navegación, consolas portátiles, ordenadores single board, terminales de datos, punto de venta/servicio, radio definido por

software, aplicaciones médicas, controladores de medios de comunicación, y muchos otros productos que requieren de baja potencia y alto rendimiento[42].

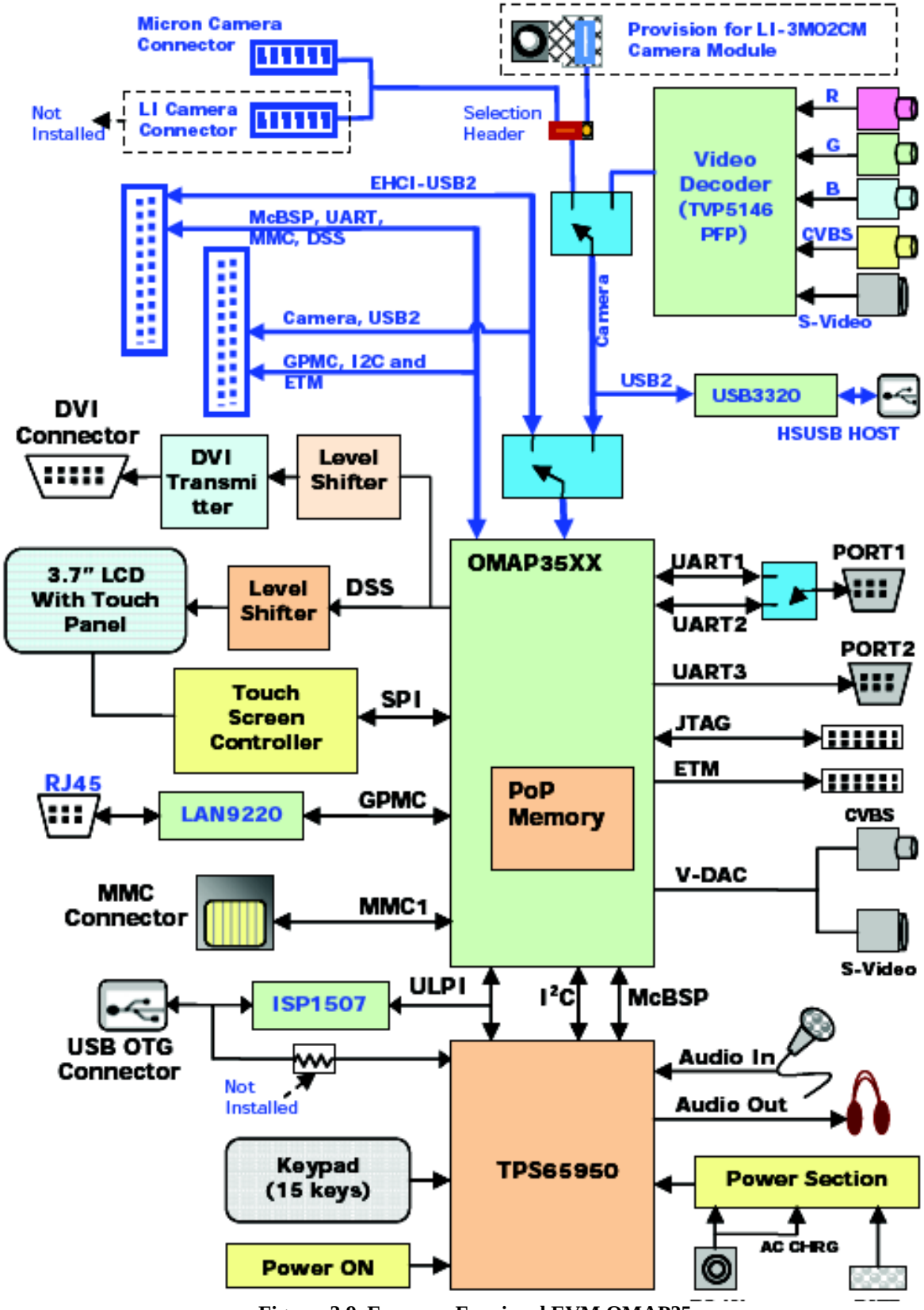


Figura. 3.9. Esquema Funcional EVM OMAP35x

EVM se encuentra diseñado con una arquitectura modular y extensible, compuesta por cuatro módulos interconectados:

- OMAP35x EVM Main Board
    - Interfaces de periféricos y conectores.
- OMAP Processor Module
    - Procesador de Aplicaciones OMPA3530 de alto rendimiento
    - Memoria PoP Micron
- TPS65950 Power Module
    - TPS65950 Integrated Power Management
- WL1271-based module
    - Ethernet 802.11b/g/n y Bluetooth 2.0 + EDR.

La Arquitectura de Hardware del la plataforma de evaluación EVM, puede ser apreciada en la Figura 3.10. A continuación se realizará una breve descripción de los subsistemas que componen a la plataforma, para mayor información de la arquitectura del sistema, por favor referirse al documento "OMAP35x Evaluation Module Hardware User Guide" [42].

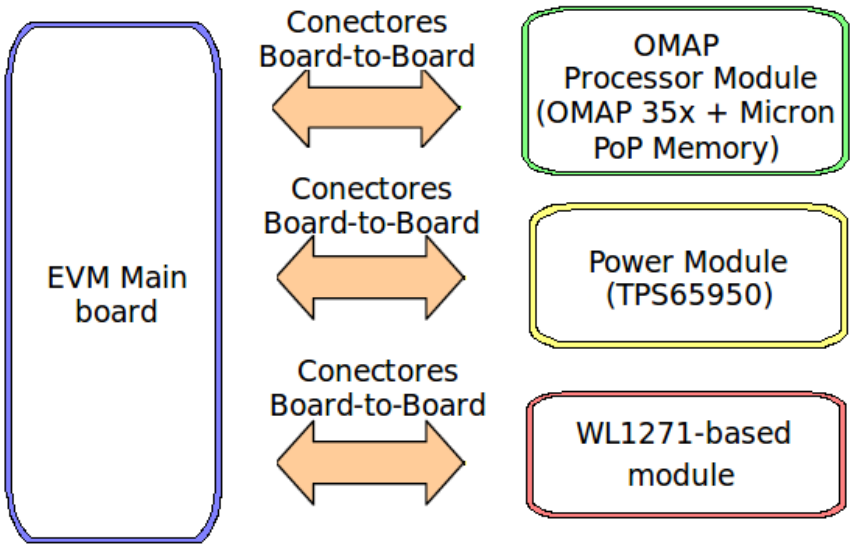


Figura. 3.10. Esquema de Bloques EVM OMAP35x

3.4.1. OMAP35x EVM Main Board

La plataforma Main Board provee todas las interfaces de periféricos y conectores necesarios para el desarrollo de aplicaciones multimedia, incluye además una pantalla LCD. Sus dimensiones son: 203.2 × 143.51 mm, circuito impreso de PC de 8 capas [42]. En la Figura 3.11 y Figura 3.12 [42], se presenta a la plataforma de vista frontal y posterior respectivamente, aquí se pueden observar los diferentes periféricos que ofrece la plataforma para el desarrollo de aplicaciones, los módulos restantes vienen conectados a través de conectores de expansión board-to-boad como puede observarse en la Figura 3.8 de la vista posterior. Por lo tanto, Main Board ofrece una base fija sobre la cual se puedan realizar un sin número de aplicaciones multimedia o wireless y cada de las interfaces serán analizadas a continuación.

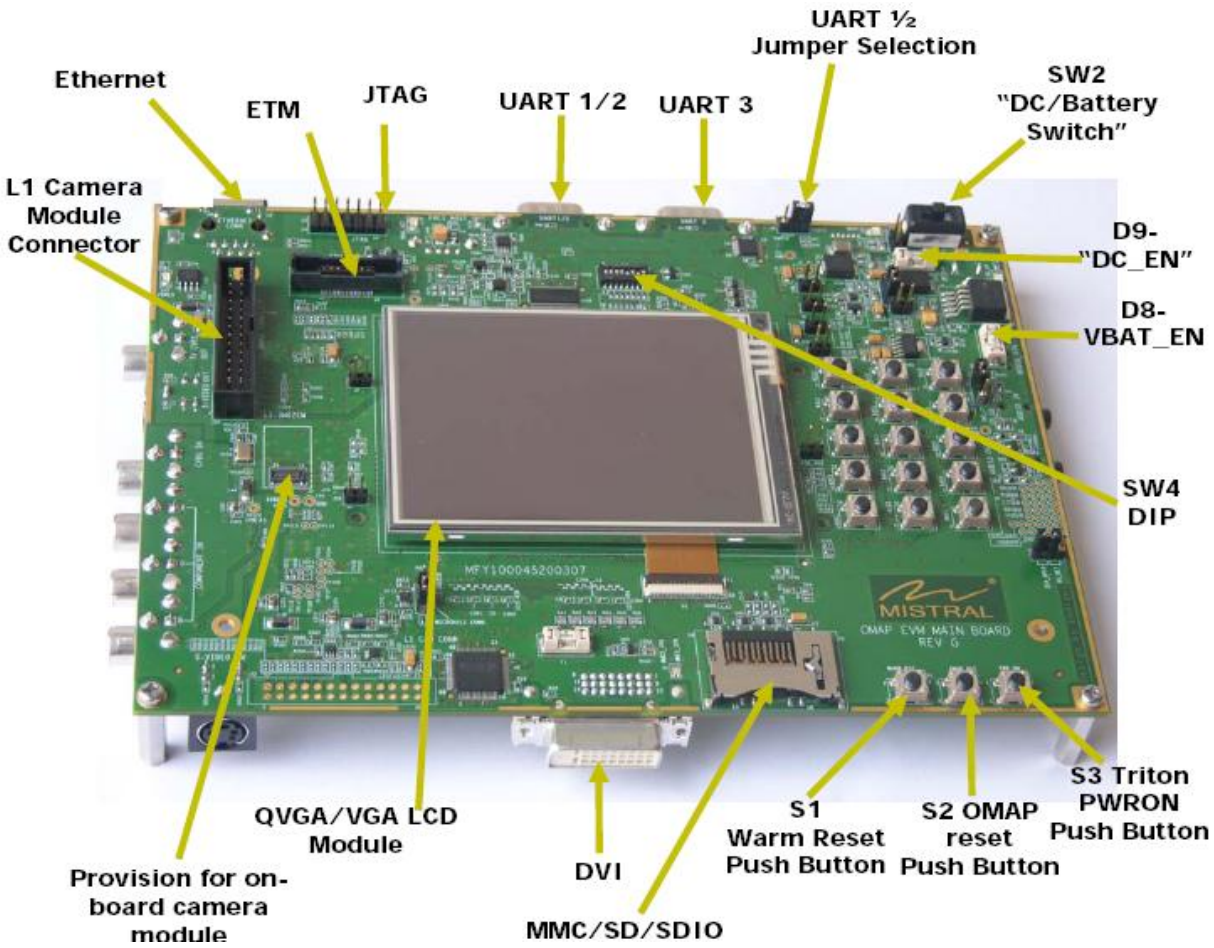


Figura. 3.11. EVM OMAP35x Main Board (Vista Superior)

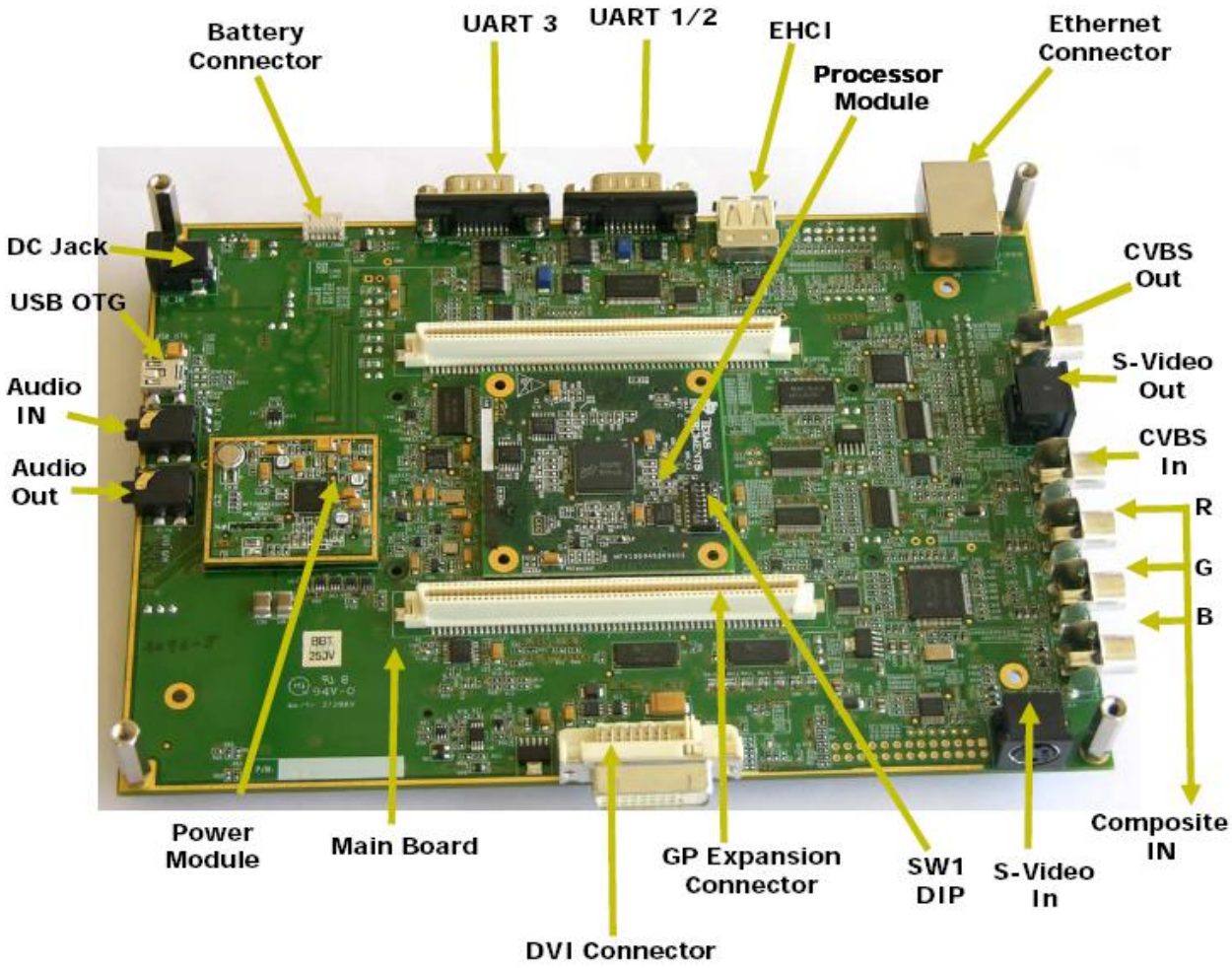


Figura. 3.12. EVM OMAP35x Main Board (Vista Posterior)

Interfaz de Video [42]

- **Salidas de Video**

EVM incluye un módulo de pantalla LCD LS037V7DW01, propio de la compañía Sharp Electronics Corporation. El módulo LCD es de 3.7", soporta formato RGB de 18-bits, utiliza dos tipos de resolución VGA (640×480) y QVGA (320×240), soporta Touch Sreem a través del controlador TSC2046. Su comunicación hacia el procesador OMAP3530, hacia el subsistema de Display es realizada utilizando la interfaz McSPI1.

Contiene además una interface de video DVI, el cual puede ser utilizado para conectar pantallas LCD utilizando cables HDMI a DV o para TV con interface HDMI.

EVM incluye dos salidas de Video para TV tipo *CVBS* y *S-Video*. Solamente uno de los dos se encontrará disponible al mismo tiempo, ya que sus salidas son multiplexadas de

una misma interface, se encuentran conectadas al Conversor Análogo Digital ADC (Analog-to-Digital Converter) del SoC OMAP3530.

- **Entradas de Video**

EVM posee varias entradas de video disponibles en la plataforma Main Boad y a través de conectores de expansión, los cuales pueden ser utilizados a cualquier momento. EVM soporta tres interfaces de cámara a través de conectores de expansión, además soporta entradas de video *CVBS*, *S-Video* y *Composite* sobre la plataforma, que se comunican con el procesador a través de interfaces GPIO.

Las opciones de video existentes son: Decodificador de Video TVP5146PFP (conversión componente RGB, Composite y S-Video a YCbCr), Sensor de cámara, ver Figura 3.13, (conectores de expansión *Micron Camera*, *LI-3M02CM*, y *LI/Image* (no incluye conector)). Los datos adquiridos son enviados al procesador ISP del SoC OMAP3530 y son mostrados en el LCD.

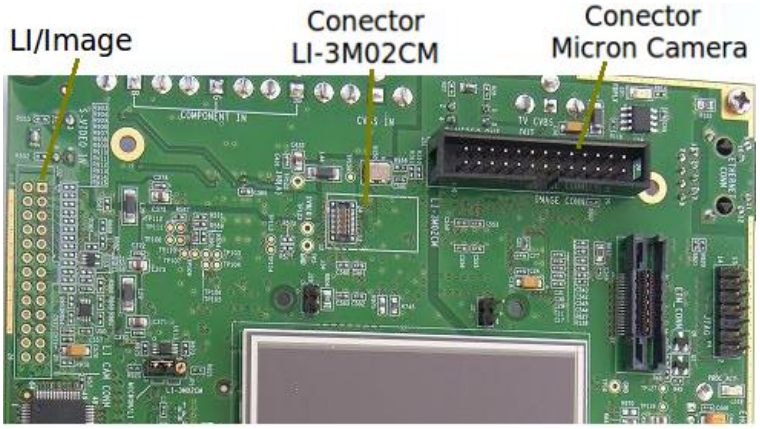


Figura. 3.13. EVM OMAP35x Main Board Conectores Cámara

Interface Audio [42]

Consiste en una entrada y una salida de audio estéreo, se comunica a través del códec de audio en el módulo TPS65950 Power Module. Para el envío y recepción de audio, sus periféricos se encuentran interconectadas las interfaces McBSP2 en el SoC OMAP350 Y TDM en TPS65950.

Interface Ethernet [42]

EVM incluye una interface de Ethernet a 10/100 Mbps, utiliza un controlador PHY y MAC (Media Access Control) externo LAN9220, propio de la compañía SMSC (Smart Mixed-Signal Connectivity) que soporta. El controlador LAN9220 se interconecta con el procesador OMAP a través del controlador GPMC.

Interface Tarjeta de Memoria SD/MMC/SDIO [42]

EVM soporta dos interfaces SD/MMC para tarjetas de memoria MMC/SD/SDIO: interface SD/MMC1 (soporte 1.8/3.0 V), e interface SD/MMC2 (1.8 V). El interface SD/MMC1 corresponde al slot SD/MMC sobre la Main Board, mientras que SD/MMC2 puede ser ruteada a través de un conector de expansión. El módulo TPS65950 es el encargado de proveer energía a la tarjeta. Las señales son transmitidas hacia el SoC OMAP3530 a través de la interface GPIO.

USB [42]

- **High Speed USB OTG Interface**

La interface High Speed USB OTG se encuentra soportada en EVM a través del transceptor USB ISP 1507 propio de la compañía NXP Semiconductors. El procesador es interconectado hacia el transceptor a través de su interface ULPI. Es utilizado para la implementación de un Host Hi-Speed USB host, periféricos o dispositivos OTG.

- **High Speed USB Host Interface**

EVM soporta la interface High Speed USB HOST, a través del transceptor HSUSB USB3320 propio de SMSC. El procesador se interconecta al tanseptor a través del puerto USB2 (EHCI). Es utilizado para la implementación de un Host Hi-Speed USB host.

Interface Serial [42]

- **UART Interface**

EVM soporta tres interfaces UART por medio de 2 puertos RS232. Las interfaces UART1 y UART2 son multiplexadas en un solo puerto (selección a través de jumpers).

UART1 y UART2 proveen comunicación serial y UART3 es utilizada para buteo por periférico en el procesador OMAP.

- **I2C Interface**

EVM soporta 4 interfaces I^2C, su conectividad puede ser revisada en la tabla 3.4:

Tabla 3.4. EVM OMAP35x Main Board Conectividad I^2I

I^2C	Interface
#1	TPS65950 (registros de propósito general)
#2	Conectores de Expansión Conector de Cámara Micron Conector de Cámara LI-3M02CM
#3	Conectores de Expansión para uso futuro. Decodificador de Video TVP5146
#4	TPS65950 (interface Smart Reflex)

Switches [42]

- **Teclado**

EVM incluye en el Main Board quince pulsadores organizados en forma de matriz (5×3), las cuales se encuentran conectadas a la interface de teclado del módulo TPS65950. En la Figura 3.14[42] se muestra el teclado (S4-S18).

- **Push Butons**

El Main Board incluye tres pulsadores que tienen funciones de encendido para la tarjeta. S1 provee un Warm Reset, S2 es un pulsador de Reset, mientras que S3 es un pulsador de encendido. En la Figura 3.14[42] se pueden observar los pulsadores (S1-S3).

- **DIP Switch**

EVM incluye un DIP Switch de ocho posiciones para el manejo de arranque del sistema llamado SW4 (ver Figura 3.11), las diferentes configuraciones del SW puede indica el modo de buteo para el procesador OMAP3530, pueden ser desde el puerto UART3, memoria MMC o NAND Micron (incluida el módulo de procesamiento). Además contiene el Switch de dos posiciones SW2 (ver Figura 3.11), con el que se selecciona la fuente de poder a utilizar, ya sea a través de Batería o Alimentación DC.

Figura. 3.14. EVM OMAP35x Main Board Push Buttons

Interfaces de Depuración [42]

EVM soporta dos interfaces de depuración JTAG y ETM (ver Figura 3.11). Los conectores de depuración se utilizan para pruebas, depuración, ejecución, rastreo y descarga de programas desde el IDE (Integrated Development Environment) a la unidad de destino.

Alimentación [42]

La alimentación de todo el sistema se realiza por medio de un adaptador de energía externa de 5V incluido con el paquete, además se tiene a opción de alimentación a través de una batería de Litio no incluida, ver Figura 3.12 DC Jack.

Periféricos [42]

A continuación se adjunta la Tabla 3.5, donde se incluyen todos los periféricos que forman parte de la plataforma Main Board y pueden ser ubicados en la Figura 11 y 12.

Tabla 3.5. EVM OMAP35X Main Board Periféricos

	Interfaces	Cant.	Obsservaciones
Video outputs	LCD y Touch Screen	1	LS037V7DW01
	DVI	1	
	CVBS	1	
	S-Video	1	
Video Inputs	Composite	1	
	CVBS	1	
	S-Video	1	
	Conector Micron Camera	1	No incluye cámara
	Conector LI-3M02CM	1	No incluye cámara
	LI/Image	1	No incluye conector ni cámara
Audio	Line In	1	
	Line Out	1	
Ethernet	RJ45 Ethernet	1	10/100 Mbps
Tarjetas de Memoria	MMC/SD/SDIO	2	MMC/SDIO2 por conectores de expansión
USB	High Speed USB OTG Interface	1	
	High speed USB Host interface	1	
Serial	UART	3	UART 1 y 2 multiplexados
	I^2C Interface	4	
Switches	Teclado 3x5	1	Matriz de 3x5
	Botones de Poder	3	
	SW4	1	Selección de Buteo
	SW2	1	Selección Alimentación
Interfaces de Dpuración	JTAG	1	
	ETM	1	
Alimentación	Interface de adaptador de Energía	1	Salida (110-5V)
	Interface Batería de Litio	1	Batería no incluida

3.4.2. OMAP Processor Module

El modulo de procesamiento (Figura 3.15[42]) consiste en la plataforma SoC OMAP3530 y una memoria externa tipo PoP micrón Micron (MT29C2G48MAKLCJI-6 IT). El módulo es adherido a la plataforma EVM Main Board a través de sus conectores centrales Board-to-Board pueden ser observados en la Figura 3.12.

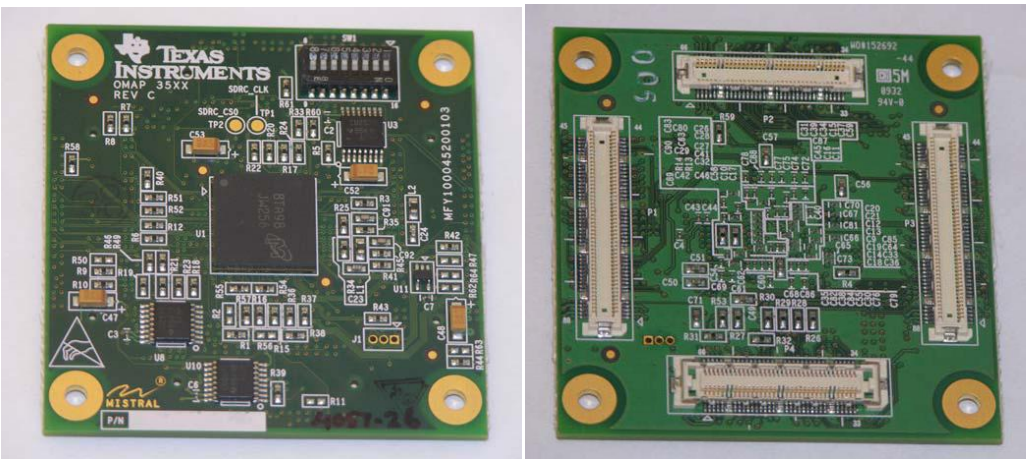


Figura. 3.15. EVM OMAP35x Processor Module (Vista Superior y Posterior)

La plataforma SoC OMAP3530 trabaja con el procesador ARM Cortex-A8 a 720MHz y el procesador C64xDSO a 520 MHz. El módulo de procesamiento incluye una memoria NAND PoP Micron de 256 MB Nand Flash/256 MB LPDDR, la cual se encuentra apilada al SoC OMAP3530 a través de su interface PoP. La memoria es montada directamente hacia los controladores de memoria externa GPMC and SDRC, en la parte superior del SoC OMAP3530. Eta memoria es utilizada principalmente para el buteo del sistema. Existe soporte de interfaces adicionales, las cuales son multiplexadas en diferentes modos a través del SW1 montado en el módulo.

3.4.3. TPS65950 Power Module

El dispositivo Integrated Power Management TPS65950 Power and Audio Companion [42] es un circuito integrado de potencia encargado de la administración de la alimentación y de la generación del clock y reset en la plataforma. Además se encarga de funciones de administración de alimentación de otros elementos como: transceptor HS USB, cargador de batería, interface de Keypad, codec de audio, reloj en tiempo real,

entradas ADC (ver Figura 3.16[42]). El módulo es adherido a la plataforma EVM Main Board a través de sus conectores Board-to-Board, se puede observar su ubicación en la Main Board Figura 3.12.

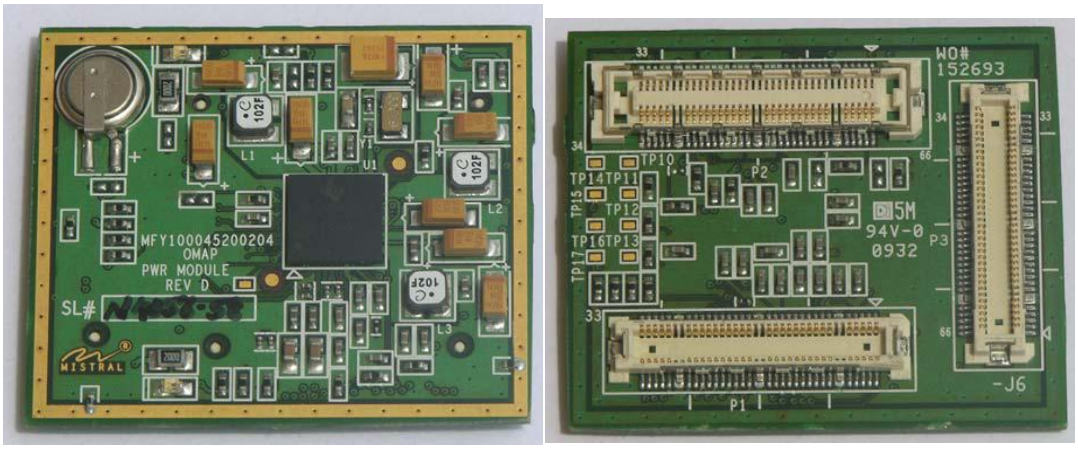


Figura. 3.16. EVM OMAP35x Power Module (Vista Superior y Posterior)

3.4.4. WL1271-based module

El modulo WL1271- Daughter Card (ver Figura 3.17[42]), es una plataforma de conectividad WLAN y Bluetooth, basada en el modulo LS Research y el dispositivo WL1271 DE Texas Instruments [42]. El dispositivo soporta los estándares Ethernet 802.11b/g/n y Bluetooth 2.0 + EDR. Posee una antena on-chip y soporta antenas externas. Su estudio detallado y aplicaciones se presentarán en la segunda parte del proyecto. El módulo es adherido a la plataforma EVM Main Board a través de su conector Board-to-Board, puede ser ubicado como GP Expansion Connector en la Figura 3.12.

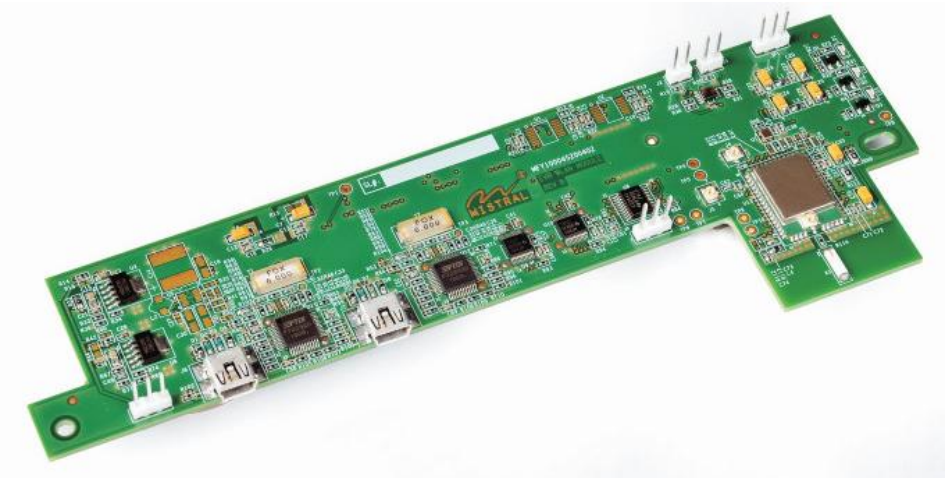


Figura. 3.17. EVM OMAP35x WL1271-based module (Vista Superior)

CAPÍTULO 4

METODOLOGÍA DE DISEÑO DE APLICACIONES

Resumen: En el presente capítulo se estudiarán los Integrated Development Environment (IDE) y System Development Kit (SDK), soportados para la tarjeta de desarrollo EVM OMAP3530. Además se analizarán las metodologías para la implementación del Sistema Operativo Android sobre dicha tarjeta. Finalmente se desarrollarán los tutoriales de los procedimientos realizados.

4.1. SOFTWARE DEVELOPMENT KIT (SDK)

Los procesadores OMAP pueden ser adaptados a diferentes soluciones tecnológicas que incluyan multimedia, procesamiento de señales, etc. Además cuenta con muchas herramientas de software que ayudan a este cometido, algunas como software específico para la plataforma y el ambiente de desarrollo (SDK) y otras como software para desarrollo de aplicaciones (IDE). El fin es lograr que los programadores puedan efectuar aplicaciones sobre un módulo de evaluación que cuente con periféricos y procesador OMAP.

Un Kit de Desarrollo de Software o SDK es un conjunto de herramientas que le permite a un programador empezar a desarrollar aplicaciones para un sistema concreto. Generalmente incluye frameworks, sistema operativo, drivers, códigos de ejemplo y notas de soporte técnico u documentación de soporte para referencia.

TI provee herramientas SDKs como un paquete de software fundamental para la evaluación y desarrollo específico para cada sistema operativo, el procesador OMAP3530, es capaz de soportar algunos sistemas operativos. Es importante recordar que el desarrollo

de aplicaciones desarrolladas para OMAP3530 pueden ser portadas hacia otro dispositivo o entre sistemas operativos.

4.1.1. SDK Linux

El SDK para Linux es llamado DVSDK (Digital Video Software Development Kit) [57], permite el desarrollo rápido de aplicaciones basadas en multimedia e incluye el sistema operativo Linux, un famework de aplicación, librería de codecs, bootloader (U-Boot/x-loader), Codec Engine, DMAI, además incluye ejemplos de codificación y decodificación de audio y video en tiempo real desde o hacia los periféricos del dispositivo.

Mediante estas herramientas se permite la programación de los procesadores GPP (Subsistema MPU) y DSP (Subsistema IVA2.2) de OMAP3530. DVSDK presenta un kernel linux libre y no comercial. Proporciona documentación, como una Guía para Inicio Rápido, que permite empezar a utilizar el sistema a través de una interfaz gráfica y permite la instalación de herramientas de desarrollo para aplicaciones.

4.1.2. SDK WinCE

El SDK WinCE es propio de Microsoft Windows Embedded CE[58] para el desarrollo de software de procesadores asociados al OMAP. Proporciona un Bootloader y un Board Support Packager (BSP) que es un conjunto de paquetes de software que incluye entre otras cosas controladores de los periféricos EVM, SDK para gráficos 3D, DVSDK para desarrollo multimedia, el cual a su vez incluye: DMAI, Codec Engine, DSP/BIOS Link, ejemplos de aplicaciones y documentación. WinCE debe ser licenciado a través de Microsoft y para poner en marcha el diseño se debe pagar por la herramienta de diseño.

4.1.2. SDK Android

TI ofrece un SDK para Android [59], llamado DevKit para las versiones v2.1 (Eclair), v2.2 (Froyo) y v2.3 (GingerBread). El kit proporciona una base de software estable y probado en varios módulos de evaluación. Incluye Kernel Linux, Bootloader (U-

Boot/x-loader), controladores y bibliotecas OpenGL para gráficas 3D, *Benchmarking*, ejemplos de aplicaciones y documentación. Se ofrece de forma gratuita sin restricciones.

TI soporta Android, a través del proyecto Rowboat[60], el cual provee de software específico para procesadores OMAP. Rowboat habilita el desarrollo de aplicaciones de una forma segura, libre y con soporte en la comunidad.

Es necesario mencionar, que aunque TI provea SDK libres para Android, Linux o WinCE; en el momento de desarrollar, tanto Linux como WinCE utilizan entornos de desarrollo pagados, lo cual dificulta la utilización de la plataforma [61]. Es por este motivo que se optó por utilizar la versión libre en desarrollo, Android.

4.2. SISTEMAS DE DESARROLLO INTEGRADO (IDE)

Un IDE (Integrated Development Environment) es un programa de computación, el cual incluye un conjunto de herramientas para desarrollo de aplicaciones en un lenguaje de programación específico, es un entorno de programación que consiste en un editor de código, compilador y depurador.

En el caso de WinCE y Linux, para el desarrollo de software en ARM, TI incluye herramientas de desarrollo como Code Sourcery G++ para Linux y Visual C++ para Windows. Para el desarrollo DSP, TI, ofrece Code Composer Studio (CCS) [61]. Para aplicaciones ARM RTOS se pueden utilizar IDEs de terceros o CCS para Linux. Todas estas soluciones de desarrollo cuestan o pueden ser utilizadas por pocos días como prueba.

El sistema operativo Android ofrece la posibilidad de desarrollo de software con entornos de programación libres. La ventaja de Android es en el momento de programar aplicaciones, ya que no es necesario preocuparse de la interacción GPP – DSP ya que es realizado mediante un framework multimedia (OpenCORE o GStreamer), del cual hablaremos más adelante.

Las aplicaciones Android son usualmente desarrolladas en lenguaje *Java* que es su lenguaje oficial. Sin embargo, además pueden utilizar otros lenguajes de programación, como es el caso de *C* o *C++* a través de la herramienta Native Development Kit (NDK);

lenguajes de tipo Script como *Python, Perl, JRuby, Lua, BeanShell, JavaScript, Tcl y shell* a través de la aplicación Scripting Layer for Android (SL4A)[62], que debe ser instalada en el celular. Android cuenta además con un entorno de desarrollo llamado Basic4android [63], que utiliza un lenguaje de programación similar a Visual Basic pero enfocado en el desarrollo de Android, el cual no es gratuito ($49 USD - $499 USD).

Para desarrollo de aplicaciones Android se utilizará lenguaje de programación Java. Los IDEs más comúnmente utilizados son Netbeans y Eclipse.

La selección del entorno de desarrollo depende mucho del programador, de cuál es su preferencia, ya que Android soporta ambos. Android proporciona un plugin[64] que activa la creación de proyectos Android para Eclipse, mientras que para Netbeans, esta no forma parte del IDE y debe ser activado de forma separada. Es por esta razón que el IDE Eclipse se utilizó para el desarrollo de aplicaciones.

4.3. ANDROID

Android fue creado por la Open Handset Alliance (OHA)[65], un grupo de más de 80 compañías de tecnología y móviles cuyo objetivo es el desarrollar estándares abiertos para dispositivos móviles.

Android es un sistema operativo tipo *Open Source*, pero más que eso, es una plataforma para dispositivos móviles que contiene un *Software Stack* cuyas capas incluyen un sistema operativo, middleware y aplicaciones básicas para el usuario. Se encuentra basada en varios proyectos de código abierto. Esto quiere decir que cualquiera puede tener acceso completo a códigos fuente y se puede colaborar de igual manera.

4.3.2. Arquitectura Android

A continuación se dará una visión global por capas de la arquitectura[64] empleada por Android. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez, los suyos propios a las capas de niveles superiores, tal como muestra la Figura 4.1[64]:

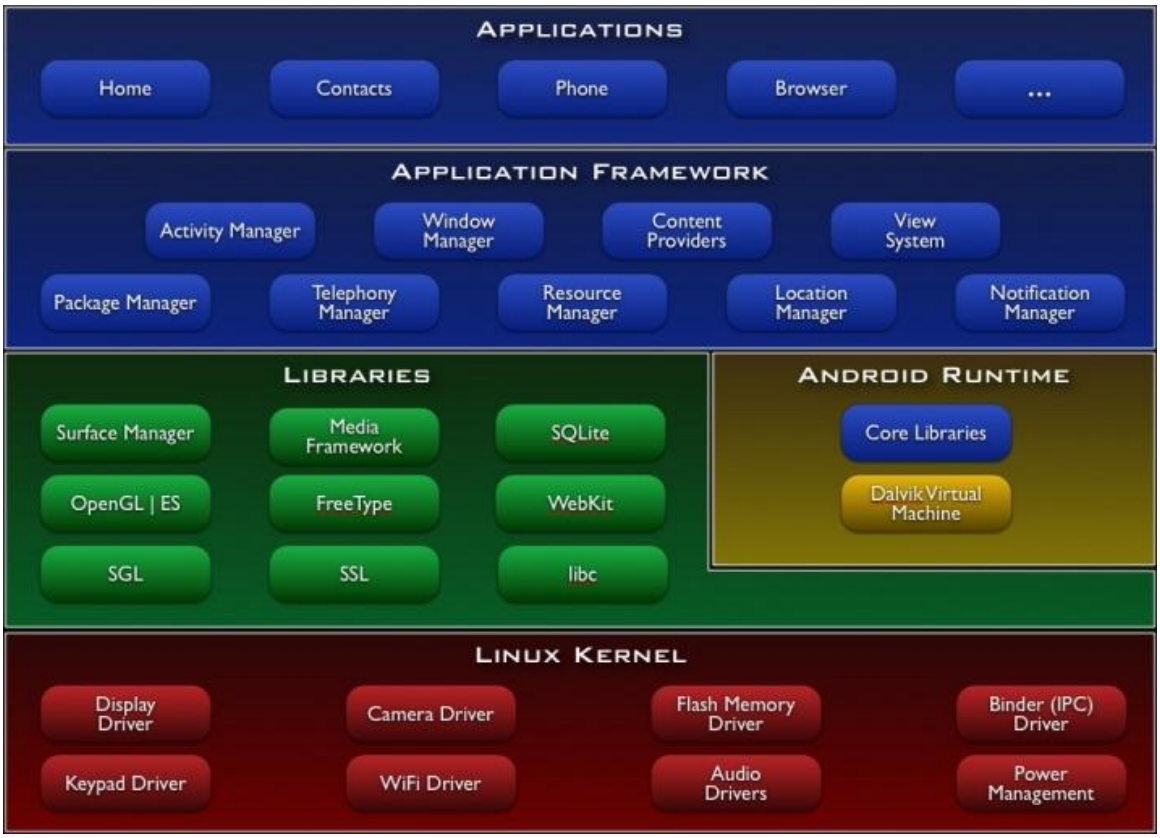


Figura. 4.1. Arquitectura Android, Software Stack Layers

Aplicaciones [64]

Esta capa contiene aplicaciones, tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente. Todas estas aplicaciones utilizan los servicios, APIs y librerías de los niveles anteriores.

Framework de Aplicaciones [64]

Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android utiliza el mismo conjunto de APIs y el mismo framework, representado por este nivel. Entre las APIs más importantes ubicadas aquí, se pueden encontrar las siguientes:

- **Activity Manager**

Conjunto de APIs que gestiona el ciclo de vida de las aplicaciones en Android.

- **Window Manager**

Gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.

- **Content Provider**

Permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.

- **View System**

Proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, *check-boxes*, tamaño de ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.

Librerías Nativas [66]

La siguiente capa le pertenece a las Librerías utilizadas por Android. Éstas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android. Entre las librerías más importantes ubicadas aquí, se pueden encontrar las siguientes:

- **Bionic Libc**

Es la versión Android de la librería Libc de los sistemas Linux, la cual ha sido modificada para sistemas embebidos. Incluye todas las cabeceras y funciones según el estándar del lenguaje C.

- **Function Libraries**

Estas librerías soportan funciones del framework, cada una específica para una función diferente, las más importantes serán detalladas a continuación:

Webkit

Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

Media Framework

Proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.). Basa su funcionamiento en la plataforma PacketVideo de OpenCore [67], o a través de GStreamer[68]. Soporta códecs de Harware/Software a través de su interface OpenMax IL. La tabla 4.1 [64] presenta los formatos multimedia soportados por Android.

Tabla 4.1. Formatos Media Soportados por Android

Tipo	Formato/Códec	Encoder	Decoder	Observaciones	Archivos soportados
Audio	AAC LC/LTP	X	X	Contenido Mono/Stereo hasta 160 kbps y velocidades de muestreo de 8 a 48 kHz	3GPP (.3gp), y MPEG-4 (.mp4, .m4a). ADTS raw AAC (.aac, decoder, ADIF no suportado, Android 3.1+).
	HE-AACv1 (AAC+)		X		
	HE-AACv2 (enhanced AAC+)		X		
	AMR-NB	X	X	4.75 a 12.2 kbps sampled @ 8kHz	3GPP (.3gp)
	AMR-WB	X	X	9 rates de 6.60 kbit/s a 23.85 kbit/s sampled @ 16kHz	3GPP (.3gp)
	FLAC		X	Mono / Stereo (no multicanal). Frecuencias de muestreo de hasta 48 kHz. Solo para Android 3.1+	FLAC (.flac)
	MP3		X	Mono / Stereo 8-320Kbps constante (CBR) o tasa de bits variable (VBR)	MP3 (.mp3)
	MIDI		X	MIDI tipo 0 y 1. DLS Version 1 y 2. XMF y Mobile XMF. Soporte para formatos ringtone RTTTL/RTX, OTA, y iMelody	Tipo 0 y 1 (.mid, .xmf, .mxmf). Además RTTTL/RTX (.rtttl, .rtx), OTA (.ota), and iMelody (.imy)
	Ogg Vorbis		X	8- and 16-bit linear PCM	Ogg (.ogg)
	PCM/WAVE		X		WAVE (.wav)
Imagen	JPEG	X	X	Base+progressive	JPEG (.jpg)
	GIF		X		GIF (.gif)
	PNG	X	X		PNG (.png)
	BMP		X		BMP (.bmp)

Video	H.263	X	X	Igual que MPG-4	3GPP (.3gp) MPEG-4 (.mp4)
	H.264 AVC	X	X	Baseline Profile (BP) 3GPP Encoder solo para (Android 3.0+)	3GPP (.3gp), MPEG-4 (.mp4), MPEG-TS (.ts, AAC solo audio, Android 3.0+)
	MPEG-4 SP		X		3GPP (.3gp)
	VP8		X	(Android 2.3.3+)	WebM (.webm)

OpenGL/SL y SGL

Representan las librerías gráficas; por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D, y SGL proporciona gráficos en 2D.

SQLite

Se encarga de la creación y gestión de bases de datos relacionales.

- **Native Servers**

A las aplicaciones no se les permite acceder a la base de funcionalidades del sistema directamente, sino que las solicitudes se las hacen a un conjunto de Servidores Nativos que a su vez se comunican con la capa Hardware Abstraction Layer (HAL) y esta con el kernel. Los más críticos son los siguientes:

Surface Manager

A través de un Surface Flinger se encarga de componer los diferentes elementos de navegación de pantalla. La capa correspondiente HAL se encarga de abstraer los contenidos juntos al Frame Buffer. Combina superficies 2D y 3D de múltiples aplicaciones.

Audio Manager

A través de un Audio Flinger se encarga de procesar múltiples tramas de audio a una señal PCM (Pulse Code Modulation), las cuales serán enviadas al Driver de Audio. Maneja el ruteo simultáneo de señales de audio de reproducción o grabación.

Media Server

Este servidor maneja todos los requerimientos multimedia de las aplicaciones Java, y las rutea a las entradas correspondientes, interactúa con los servicios Media Player, Media

Recorder y Camara Service. El Servicio Media Player a su vez, interactúa con OpenCORE, Vorbis (archivos multimedia de alerta y ringtones) o MIDI (Musical Instrument Digital Interface), dependiendo de los datos que deben ser manejados.

- **Hardware Abstraction Layer (HAL)**

La Capa de Abstracción de Hardware es la capa que se encuentra entre el Kernel y las Librerias Android. Permite que las aplicaciones sean independientes del hardware porque abstrae información acerca de tales sistemas y expone una interfaz genérica. Esta es la mayor ventaja del manejo de una arquitectura por capas (Sofware Stack), la cual permite que las aplicaciones realizadas para Android, corran sobre cualquier arquitectura de hardware. Cuando se utilice una platforma distinta, los cambios para ajustar la plataforma hacia el sistema Android se dan en las capas inferiores a HAL. Si bien Linux en si se encarga de abstraer, esta capa es necesaria ya que no se tiene un controlador estandarizado para cada interface de hardware como la iluminación de pantallas, LEDs, sensores (acelerómetros y brújulas), etc. A continuación se detallarán algunos más importantes:

AudioHardwareInterface

Es la vinculación lógica entre Audio Flinger y el controlador de Audio de la plataforma (OSS, ALSA, etc.). Esta capa abstrae las rutinas específicas del Controlador de Audio y expone una interfaz genérica al Audio Flinger en el Framework de Android. Además se encarga del ruteo de Audio hacia diferentes dispositivos de Audio.

EGLDisplaySurface

Esta representa la vinculación lógica entre Surface Fingler y Fame Buffer y es responsable de abstraer los requerimientos específicos del Frame Buffer de pipelines de Gráficos y Video.

Multimedia Accelerator

Codecs de Audio, Video, Voz e Imagen, que pueden ser integrados al Media Framework utilizando la interface OpenMax IL.

2D-3D Graphics Accelerators

Esta es la interface hacia el procesador gráfico en caso de que exista en el procesador, el hardware de procesamiento gráfico será accedido con rutinas OpenGL.

Runtime Android [64]

Al mismo nivel que las librerías de Android se encuentra ubicado el Entorno de Ejecución. Éste lo constituyen las Core Libraries, que son librerías con gran cantidad de clases Java y la máquina virtual Dalvik. La máquina virtual Dalvik es un intérprete que solo ejecuta archivos con formato ".dex" (Dalvik Exexutable), este formato es una forma eficiente de compresión de archivos, los cuales a su vez son comprimidos y almacenados en archivos ".apk" (Android Package).

Kernel Linux [66]

Android está diseñado sobre el Kernel de Linux, versión 2.6, como una capa de abstracción para el hardware. Aunque Android no incluye el set estándar de utilidades Linux, si es compatible, ya que posee características como: gran memoria, gestión de procesos, modelo de seguridad basado en permisos y modelo con controladores experimentados *Open Source*.

Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. A continuación algunos ejemplos de drivers existentes:

- **Binder**

Es un driver de kernel que actua como Backbone del mecanismo IPC (Inter-Process Communication). Este driver facilita IPC y asegura alto rendimiento a través de compartir espacios de memoria.

- **Power Manager (PM)**

Este driver constituye una fina capa de administración de energía incorporada en la parte superior de la aplicación de Linux para administración de energía. Se debe a que Linux fue concebida para desktops y servidores, mientras android fue diseñada para dispositivos móviles.

4.3.3. Android en OMAP3530

Modificaciones Arquitectura Android

Actualmente es posible encontrar la versión Android 2.3 (Gingerbread) [59], compatible con EVM OMAP3530 . Esta version hace posible el acceso a la ultima capa de la arquitectura Android antes revisada, que es la Capa de Hardware. Android en sus inicios fue concebido para smartphones, pero cuando se trata de plataformas especializadas como es el caso de las plataformas OMAP3530, fue necesario realizar ajustes para que Android sea posible en ellas y que todas sus características seas explotadas. Una red de proyectos se ha encargado de adecuar el software para la tarjeta EVM OMAP3530, pero tuvieron que tomar en cuenta algunos detalles que serán analizados a continuación:

- **Modificación de HAL [66]**

La capa de abstracción de hardware de Android fue modificada para la plataforma de hardware EVM OMAP3530. El módulo de luces fue modificado para soportar la iluminación de la pantalla y LEDs; además librerías fueron personalizadas para soporte WiFi.

- **Configuración y Generación del Kernel [66]**

Para la implementación del Kernel en la tarjeta, se recurrió a la utilizaron de parches que principalmente sirvieron para el soporte de la mayor parte de interfaces periféricos. El Kernel fue configurado para soportar particularidades de la Tarjeta EVM OMAP3530 adicionando controladores para soporte de Power Management, Networking, FrameBuffer, soporte de USB, soporte de ADB, y Binders IPC, entre otros. Aparte de éstos, se produjeron cambios en los controladores de entrada, como el teclado y la pantalla táctil, para reportar los eventos adecuados. El controlador de memoria de vídeo también se modificó para soportar doble buffer.

- **Multimedia Framework [66]**

Los codecs de audio y video que se ejecutan en el DSP fueron integrados al framework PacketVideo OpenCORE multimedia Android, en forma de plug-ins OpenMax IL. Estos utilizan el DSP-BIOS Bridge como enlace de comunicación subyacente entre los procesadores ARM y DSP.

- **OpenGL**

El apoyo a la aceleración de gráficos 3D y drivers para el procesador gráfico Imagination Technologies PowerVR SGX fue integrado en el framework OpenGL de Android.

Soporte Multimedia en OMAP3530 [69] [70]

PacketVideo (PV) es el proveedor del subsistema multimedia para Android. PacketVideo provee el framework multimedia y componentes asociados (códecs de audio y video, análisis y creación de formatos de archivos, componentes de transmisión), potenciando la experiencia multimedia para Android y haciéndola gratuita. Estos elementos son conocidos como OpenCORE. La Figura 4.2[70] muestra como se adhiere esta sistema para la utilización del subsistema IVA en OMAP.

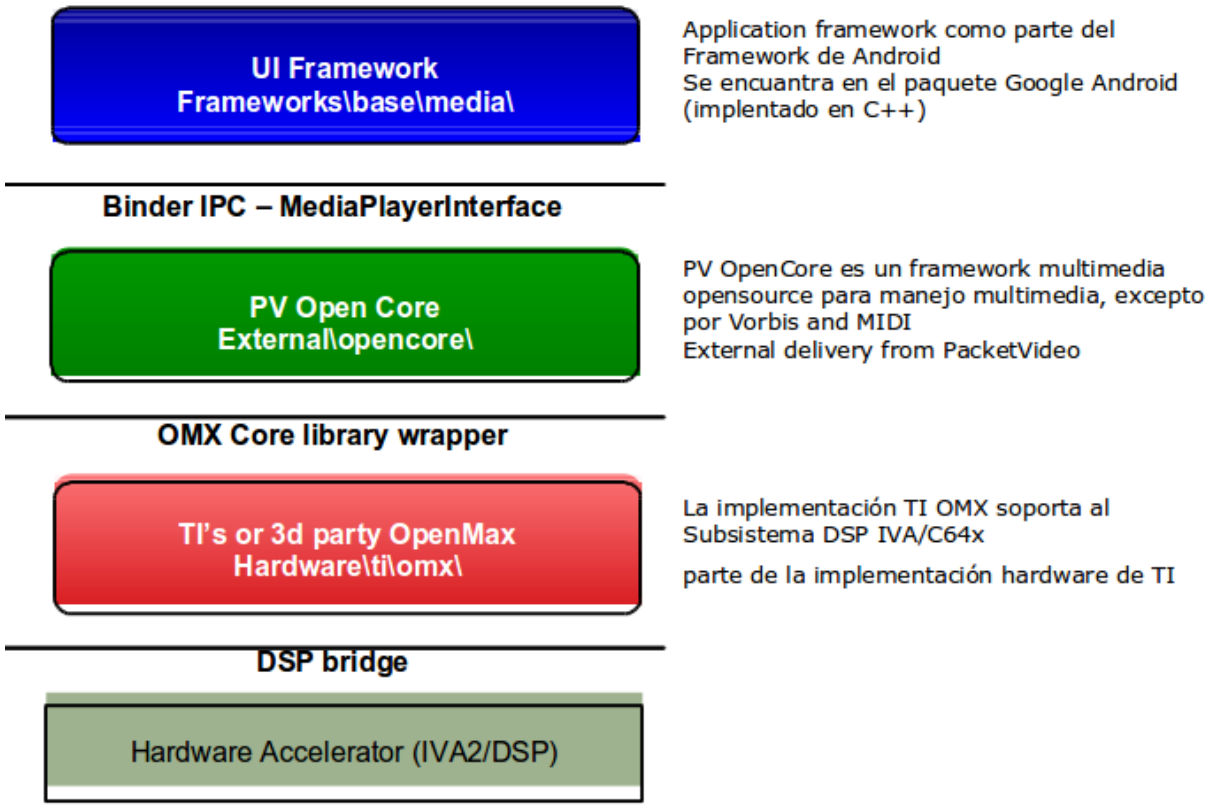


Figura. 4.2. Android Multimedia Framework en Texas Instruments

OpenCORE es un framework modular y extensible utilizado para la combinación de componentes independientes de procesamiento multimedia, tales como formatos de

archivo, codecs, componentes renderizados, componentes de flujo en protocolos y demás componentes que facilitan la aplicación de escenarios multimedia. OpenCORE soporta códecs de Hardware/Software a través de su interface OpenMAX IL. La Capa de Integración IL (Integration Layer) define la interface de comunicación con los códecs multimedia.

OpenMAX IL es una API multimedia que consta de dos segmentos, consiste en dos segmentos: Core API que se encarga del control de componentes y facilita su comunicación, y Componentes API, que son utilizados para integrar codecs de video y audio. La Figura 4.3[69] muestra el diseño OpenCORE modular que aprovecha las ventajas de la interfaz de IL OpenMAX.

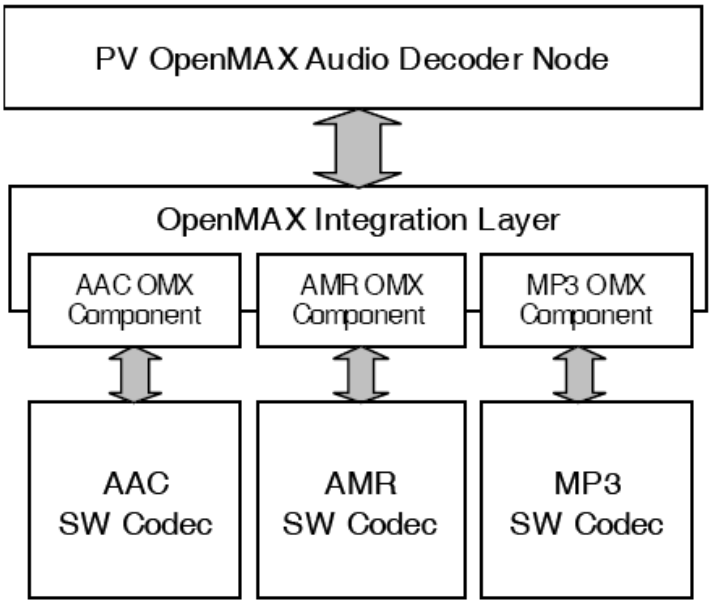


Figura. 4.3. Diseño modular OpenCORE en Android

Un componente OMX (OpenMAX) sirve como un contenedor para el códec de audio, y proporciona compatibilidad con el conjunto API OpenMAX IL. La mayoría de las APIs son asíncronas, lo que significa que el componente OMX pone en fila los comandos y los buffers de datos de entrada que recibe a través de la capa OpenMAX IL, y los procesa en un momento posterior. Cuando sea el momento, el componente OMX puede procesar uno o varios buffers de entrada de datos de la fila, con el fin de descifrar los datos asociados audio bitstream con el códec de software de audio conectado. El componente OMX envía de vuelta los buffers hacia el nodo Audio Decoder Node, a través de la capa OpenMAX IL.

La Figura 4.4. [70] representa la Arquitectura de Software Android Multimedia para plataformas OMAP, se puede observar como interactúa el famework PacketVideo OpenMax con la plataforma OMAP3530 específicamente para con el subsistema DSS y la entrada y salida de sonido. Los nodos MIO (Media I/O) se encargan de la interacción de PV OMX con el hardware específico de la plataforma. Los nodos Files son los responsables de analizar los archivos multimedia como .mp3, avi, etc. TI OMX Core contiene los componentes TI OMX, que son la interface hacia los nodos Socket DSP a través de BIOS/DSP Bridge. LCML (Linux Common Multimedia Layer) es una biblioteca compartida, cargada mientras se ejecutan los componentes OMX, y proporciona comunicación entre la aplicación (en el lado ARM) y el nodo Socket multimedia DSP.

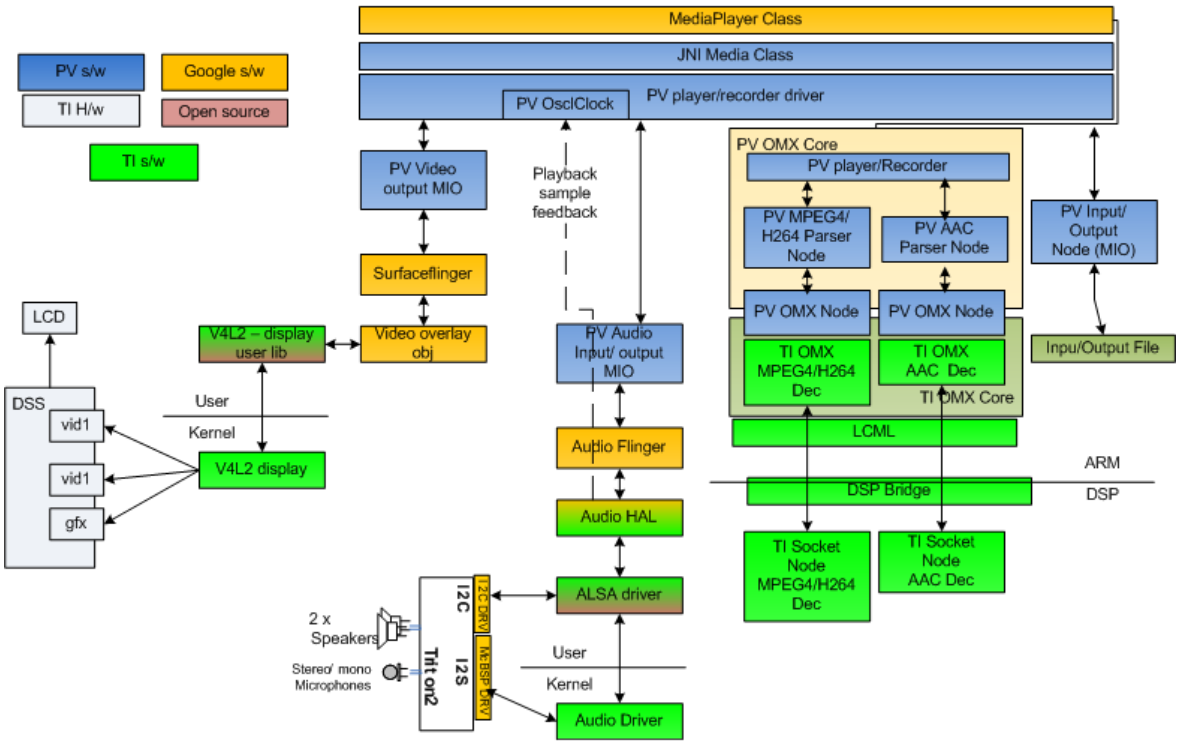


Figura. 4.4. Arquitectura de Software Android Multimedia para plataformas OMAP

- **Proyecto Rowboat**

Texas Instruments utiliza OpenMax IL para la aplicación de códecs implementados por los aceleradores de hardware en OMAP, a través del proyecto Rowboat [60]. El proyecto Rowboat brinda una base estable para portar Google Android hacia dispositivos Texas Instruments, entre los cuales constan las plataformas OMAP35x sobre la tarjeta

EVM OMAP35x, habilitando sus funciones clave en Hardware: ARM Neon, DSP, gráficos 2D/3D, aceleradores de hardware.

Rowboat soporta el procesamiento digital de señales sobre la arquitectura OMAP3530 habilitando el subsistema IVA2.2 (TMS320DMC64x +VLIN DSP) para codificación y decodificación multimedia. De esta forma se aligera la carga de procesamiento sobre el Subsistema de Procesamiento de Propósito General (ARM Cortex A8), dejándole tareas de procesamiento general y permitiendo una decodificación a mayor tasa de bits, una mayor resolución de video y audio, lo cual solamente era factible con Android corriendo solamente sobre el procesador ARM.

El soporte DSP en Rowboat se fundamenta en el DVSDK propio de TI, utilizando su software stack, el cual es necesario para trabajar eficientemente con el DSP[60]. Este esta constituido por:

- Sistema operativo en tiempo real DSP/BIOS para el procesador DSP.
- Codec Engine como el framework que permita la instalación y control de los códecs que se ejecutan en DSP desde ARM.
- DSP/Bios Link proporcionando comunicación de bajo nivel entre ARM y DSP
- DMAI el cual se ubica sobre el sistema operativo DSP/BIOS y el Codec Engine para ayudar a la rápida creación de aplicaciones sobre OMAP.
- Sistema OMX DSP para la integración de los códecs DSP de TI con el framework PacketVideo OpenCore que es proporcionado por el sitio de desarrollo de Rowboat.

El concepto principal del sistema OMX DSP es la creación de componentes framework de audio y video, para luego crear códecs basados en ellos. Algunas funciones básicas como la gestión de buffers, inicialización de puertos de entrada o salida, APIs DSP son comunes para todos los códecs, y son implementados en los componentes DSP. Los métodos comunes del componente OMX son definidos como Clase Base OMX. Esta perspectiva simplifica el proceso de creación de un códec, ya que generalmente solo se requiere heredar la Clase del componente DSP y agregar las especificaciones del códec[60].

La clase del componente DSP puede ser considerada como una implementación OMX compatible con la interface DMAI, mientras que el códec es una implementación de un códec DMAI fusionada dentro del framework OMX.

El motivo de la creación de una nueva Clase Base OMX es añadir el soporte de buffers DMAI. Esta Clase Base OMX y componentes de software del códec, utilizan memoria genérica para los buffers, pero el procesador DSP utiliza únicamente CMEM. CMEM es un API de gestión de memoria que asegura memoria físicamente contigua, posibilita creación de buffers compartidos ARM-DSP[60].

La siguiente tabla muestra los formatos multimedia compatibles con los codecs DSP[60]:

Tabla 4.2. Formatos Multimedia Soportados por Rowboat

Tipo	Formato	Encoder	Decoder	Observaciones	Archivos soportados
Audio	MPEG4 AAC Low Complexity MPEG2 AAC Low Complexity MPEG4 AAC High Efficiency		X	Contenido Mono/Stereo en cualquier combinación de velocidades de bits estándar de hasta 160 kbps y velocidades de muestreo de 8 a 48 kHz	3GPP (.3gp) MPEG-4 (.mp4) AAC audio (.aac)
	MP3		X	Mono / Stereo 8-320Kbps constante (CBR) o tasa de bits variable (VBR)	MP3 (.mp3)
Video	H.263		X	Igual que MPG-4	3GPP (.3gp) MPEG-4 (.mp4)
	H.264 AVC		X	Todas las resoluciones desde WVGA (854x480) y menores, incluyendo D1 PAL (704x576) y D1 NTSC (704x480), CIF (352x288), QCIF (176x144)	3GPP (.3gp) MPEG-4 (.mp4)
	MPEG-4		X	Decodificación de frames de hasta WVGA y resoluciones D1 a 30 fps (frame per second)	3GPP (.3gp)

Arquitectura Android en EVM OMAP 3530

La arquitectura mostrada en la Figura 4.5 incluye a la plataforma EVM OMAP 3530, tomando en cuenta las modificaciones indicadas anteriormente.

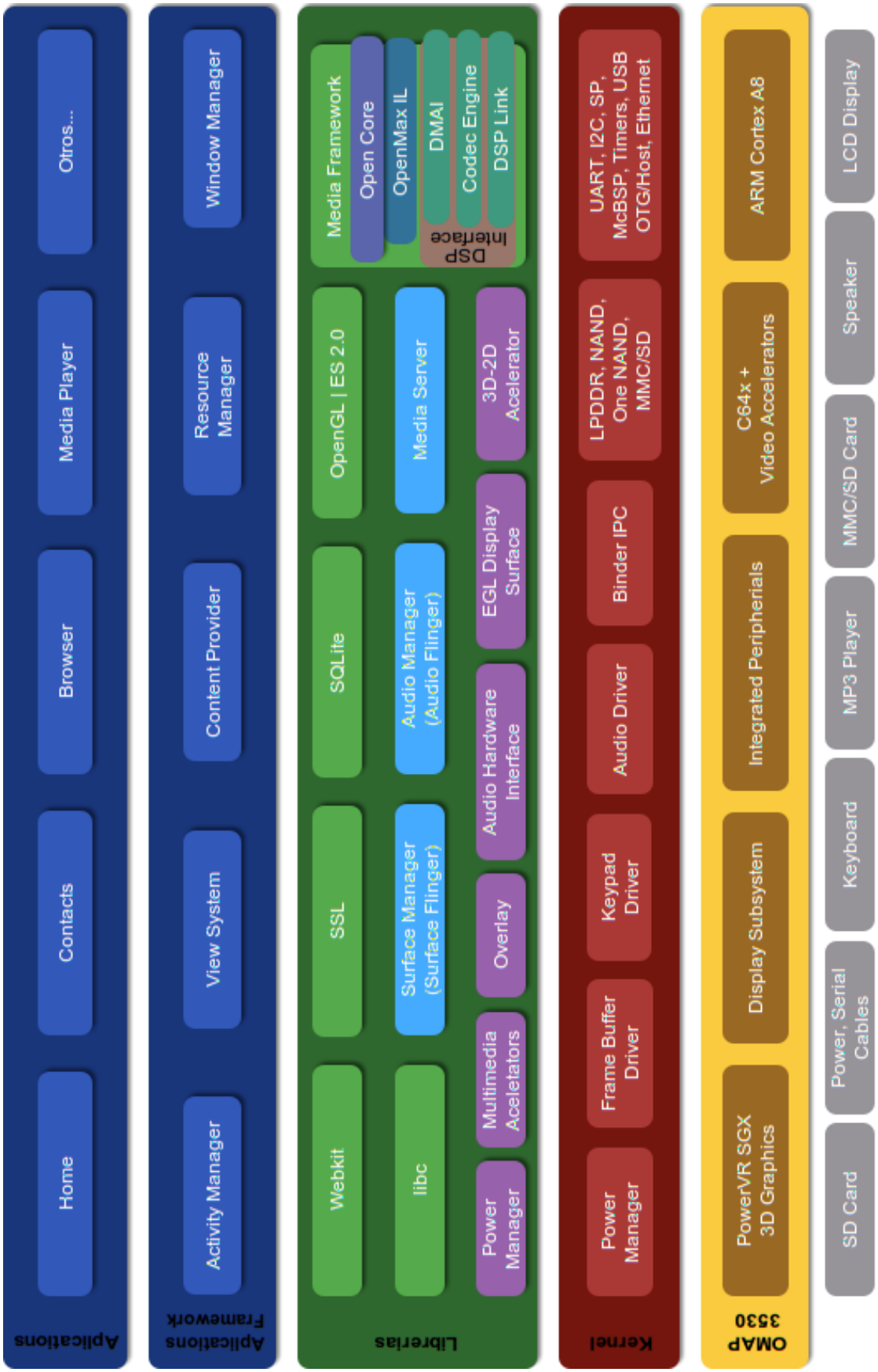


Figura. 4.5. Arquitectura Android EVM OMAP3530

4.4. TUTORIAL DE DISEÑO

A continuación se describen las configuraciones y procedimientos necesarios para tener un ambiente de desarrollo ideal, que permita implementar aplicaciones multimedia Android sobre la pataforma OMAP. EVM OMAP3530 puede correr el sistema operativo Android desde una tarjeta MMC/SD, desde su NAND o puerto UART. El presente tutorial contempla la configuración de imágenes Android y arranque sobre una tarjeta SD.

4.4.1. Configuracion de Software Embebido

Es necesario configurar software embebido a ser cargado en la plataforma, utilizado en este tutorial. Las configuraciones aquí planteadas son aplicadas para el funcionamiento de Android con los siguientes binarios:

- **Linux Kernel** de Android versión 2.6.32.
- **u-boot:** versión 2009.11
- **x-loader:** Bootstraper 1.45
- **Filesystem:** Se utilizaran dos versiones de Filesystem, esto permitirá realizar pruebas de desempeño. Aquí se los denominará Filesystem ARM y Filesystem ARM-DSP. Para las aplicaciones, pueden ser utilizados cualquiera de los dos, esto no influye en la implementación de aplicaciones.

El Filesystem ARM fue descarado de la página oficial de Texas Instruments [71], original del proyecto Rowboat, y se lo puede compilar u obtenerlo pre-compilado. El software utilizado en el proyecto será pre-compilado y listo para utilizar. El procesamiento multimedia de un Filesystem ARM no es manejado por el DSP, incluido en subsistema IVA de OMAP3530, sino que el procesador de propósito general ARM se encarga del manejo multimedia. El Filesystem ARM-DSP utilizado es la solución propuesta por el proyecto Rowboat [72] para utilizar el Subsistema de Procesamiento Digital de Señales DSP de OMAP.

El presente Tutorial comprende la configuración de los binarios del sistema operativo Android, su compilación en una tarjeta SD, la configuración de hardware para el funcionamiento de la tarjeta, configuración de u-boot, arranque del sistema, y la

configuración de herramientas para el desarrollo de aplicaciones y manipulación de la tarjeta. Los procedimientos utilizados en este tutorial fueron basados en [71], [72] y [64].

4.4.2. Símbolos del Sistema

En el Tutorial se utilizarán diferentes comandos. Los comandos utilizados serán precedidos por avisos que indican el ambiente en el que el comando debe ser escrito.

- **host$** Indica comandos que se escriben en el Terminal de la estación de trabajo host Linux.
- **OMAP3EVM #** Indica los comandos ingresados en el shell de **U-Boot**, la cual es accedida a través de comunicación serial con el puerto serial de EVM. Se utilizará un programa "Emulador de Terminal". (Sección 4.4.2 Configuración Emulador de Terminal)
- **Target#** Indica los comandos ingresados en el shell de **Android**, la cual es accedida a través de comunicación serial con el puerto serial de EVM. Se utilizará un programa "Emulador de Terminal". (Sección 4.4.2 Configuración Emulador de Terminal)

4.4.3. Requerimientos de Software para la Estación de Trabajo

Al referirse a Estación de Trabajo o *workstation* se habla de una PC HOST. Sobre esta PC se desarrollarán las aplicaciones, se instalará el software necesario para interacción con la tarjeta y se compilarán los binarios del sistema operativo. Para el desarrollo se recomienda una PC con Linux Ubuntu 8.04 o mayor. Se necesitan permisos de super usuario para algunos comandos.

Configuración Emulador de Terminal

El programa Emulador de Terminal es utilizado para la comunicación del HOST con la tarjeta EVM OMAP35X mediante interfaz de línea de comandos. Será configurado con los siguientes parámetros:

- Bits per second: 115200
- Data bits: 8

- Parity: None
- Stop bits: 1
- Flow control: none

El programa Emulador de Terminal utilizado será el programa Minicom[73]. Para instalarlo, se debe ejecutar el siguiente comando en el "Terminal" o interprete de lineas de comando de linux.

```
host$ sudo apt-get install minicom
```

Conectar el cable serial al HOST. Para ubicar el puerto colocar:

```
host$ dmesg
```

Una vez que esté instalado Minicom y el puerto serial se encuentre conectado, es necesario configurar el programa:

```
host$ minicom –s
```

Las opción utilizada (-s) inicia la configuración. Se mostrará un menú, seleccionar la opción “Serial port setup”. Luego pulsar enter. A continuación, pulsar la tecla “A” y colocar el path del puerto serial por ejemplo: */dev/ttyUSB0*, pulsar enter. Pulsar la tecla “E” y verificar que los parámetros de comunicación serial sean “115200 8N1”, si no lo son configurarlos, y pulsar enter. Pulsar nuevamente enter para salir al menú principal. Para guardar la configuración seleccionar la opción “Save setup as dfl”. Seleccionar “Exit from Minicom” para salir.

Al guardar la configuración, cada vez que se desee ingresar al Minicom colocar:

```
host$ minicom
```

4.4.4. Compilación Sistema ARM

Texas Instruments provee una distribución de Android versión 2.2., el sistema maneja multimedia desde el procesador Cortex-A8. Integra drivers SGX (gráficos 3D),

soporte Abobe Flash 10, abstracción de hardware para video overlay, aplicaciones de medición de desempeño RowboPERF y aplicaciones estándares. Este paquete incluye el kernel Linux de Android, herramientas, documentación y puede ser descargado en [71]. Los componentes de software necesarios para el sistema pueden ser compilados en una SD (min 2GB) de dos maneras, compilando las fuentes o utilizando imágenes pre-compiladas. La segunda opción será la utilizada en el tutorial.

Utilizando imágenes pre-compiladas, no es necesario instalar paquetes adicionales en al Host, salvo un editor de texto. Las imágenes pre-compiladas pueden ser descargadas en el sitio oficial de TI[71]. Descargar "OMAP35x EVM" de la sección "Pre-buit Images". Se considera que el archivo fue descargado en *~/Dowloads* (~ representa home). Mover el archivo

```
host$ cd
host$ mkdir -p Android/ARM
host$ cp Downloads/OMAP35X.tar.gz Android/ARM/
host$ cd Android/ARM/
host$ tar xvzf Android/ARM/OMAP35X.tar.gz
```

Insertar una tarjeta SD, que tenga una capacidad mínima de 2GB de memoria, en el lector de tarjetas. Ubicar la tarjeta y verificar que se encuentre montado en */dev*, ya que si se realiza el formateo de la tarjeta de manera incorrecta se puede echar a perder el disco duro.

```
host$ dmesg
host$ ls /dev
```

La carpeta de binarios pre-compilados incluye la herramienta "mkmmc-android.sh", que se encarga de formatear y crear tres particiones en la tarjeta SD: boot, rootfs y data. La partición boot será compilada con las imágenes requeridas para el booteo de la tarjeta. La partición rootfs contendrá el filesystem de Android. La partición data será utilizada como memoria externa y contendrá archivos multimedia.

En caso que la tarjeta SD se encuentre en */dev/sdc* seguir con el siguiente comando. Cuando se pregunte si quiere formatear la tarjeta SD, escribir "yes".

```
host$ sudo ./mkmmc-android.sh /dev/sdc
```

En caso que la tarjeta SD se encuentre en */dev/mmcblk0*, se debe modificar el archivo *mkmmc-android.sh* para crear correctamente las particiones. Pero primero se debe guardar un respaldo del archivo.

```
host$ cp mkmmc-android.sh mkmmc-android.sh.bak
```

Luego se debe modificar su contenido, para lo cual se utilizará la herramienta "gedit Text Editor", para acceder a la herramienta, en el panel superior, seleccionar Applications > Accessories > gedit Text Editor. Abrir el documento "mkmmc-android.sh" con "gedit", seleccionar todo y borrar. Copiar el contenido del ANEXO A.1.2. FORMATO TARJETA SD ARM, y pegarlo en el documento modificado, guardar y cerrar. En el Terminal, cuando pregunte si quiere formatear la tarjeta SD, escribir "yes".

```
host$ sudo ./mkmmc-android.sh /dev/mmcblk0
```

4.4.5. Compilación Sistema ARM-DSP

Esta sección manifiesta como compilar y ejecutar la parte del sistema operativo Android en OMAP3530, la cual activa y controla el funcionamiento del procesador embebido C64x+ DSP, para el manejo de multimedia. La solución es basada en el Proyecto Rowboat para DSP [72].

Configuraciones de Software en Host

Para la compilación del sistema en una SD, se requiere necesariamente las compilar imgenes, para lo cual es necesario obtener los siguientes paquetes, que serán instalados en el Host:

- Git 1.5.4 o más actualizada y GNU Privacy Guard.
- JDK 5.0, update 12 o más actualizada. Java 6 no es soportada, por incopatibiladas con @Override de Java.
- flex, bison, gperf, libsdl-dev, libesd0-dev, libwxgtk2.6-dev (optional), build-essential, zip, curl, minicom, tftp-server, uboot-mkimage

Estos paquetes pueden ser instalados con los siguientes comandos. Para instalar Git y otros paquetes dependientes:

```
host$ cd
host$ sudo apt-get install git-core gnupg flex bison gperf libsdl-
dev libesd0-dev libwxgtk2.6-dev build-essential zip curl
libncurses5-dev zlib1g-dev tftpd uboot-mkimage
```

Para instalar el JDK Java 5, realizar lo siguiente:

```
host$ sudo add-apt-repository "deb
http://ir.archive.ubuntu.com/ubuntu jaunty-updates main
multiverse"
host$ sudo aptitude update
host$ sudo apt-get install sun-java5-jdk
host$ sudo update-java-alternatives -s java-1.5.0-sun
```

Los usuarios de Ubuntu Intrepid (8.10), deben actualizar el paquete libreadline:

```
host$ sudo apt-get install libreadline5-dev
```

Para habilitar el manejo del DSP:

```
host$ sudo apt-get install expect
```

Para adquirir los binarios del Kernel:

```
host$ mkdir -p ~/bin
host$ curl http://android.git.kernel.org/repo >~/bin/repo
host$ chmod a+x ~/bin/repo
```

Además, se debe incluir el path en el archivo *~/.bashrc*. Para esto, se ejecutan los siguientes comandos, cambiando el **<*usuario*>** por el correcto.

```
host$ echo 'export PATH=${PATH}:/home/<usuario>/bin/repo'  >>
~/.bashrc
host$ source .bashrc
host$ mkdir rowboat-android
host$ cd rowboat-android
host$ repo init –u git://gitorious.org/rowboat/manifest.git -m
rowboat-froyo-dsp.xml
host$ repo sync
```

Esto puede tomar varios minutos dependiendo de la conexión a Internet de la estación de trabajo Host. A continuación se debe obtener el paquete TI DVSDK. Descargarlo y copiarlo en el archivo *~/rowboat-android/external/ti-dsp*, el software puede ser adquirido en [71], en la sección DVSDK_4.01 Product Downloads > OMAP3530 DVSDK, el archivo "dvsdk_omap3530-evm_4_01_00_09_setuplinux". Se considera que el archivo fue descargado en *~/Downloads*.

```
host$ cp ~/Downloads/dvsdk_omap3530-evm_4_01_00_09_setuplinux
external/ti-dsp/
```

Si el idioma local de su estación de trabajo no es el inglés, algunos scripts de instalación del paquete pueden fallar, para solucionarlo establecer la variable LANG como se muestra:

```
host$ export LANG=C
```

Para compilar Android para soportar el stack DSP, utilizar el siguiente comando:

```
host$ make TARGET_PRODUCT=omap3evm OMAPES=3.x
```

Esto compilará Android y el kernel Linux para OMAP3530, los módulos kernel TI para comunicación DSP y códecs, y además los drivers SGX.

Para crear la imágen rootfs:

```
host$ cd out/target/product/omap3evm
host$ mkdir android_rootfs
host$ cp -r root/* android_rootfs
host$ cp -r system android_rootfs
host$ ../../../../build/tools/mktarball.sh ../../../host/linux-
x86/bin/fs_get_stats android_rootfs . rootfs rootfs.tar.bz2
```

Se generará el archivo "rootfs.tar.bz2".

Las imágenes MLO y u-boot.bin pueden ser adquiridas en [74]. Al colocar la dirección web en el explorador se descargará un archivo llamado "DM37X.tar.gz". Este archivo contiene los binarios necesarios MLO y u-boot.bin Se considera que el archivo fue descargado en *~/Downloads.*

```
host$ mkdir ~/Android/ARM-DSP
host$ cp ~/Downloads/DM37X.tar.gz ~/Android/ARM-DSP/
host$ cd ~/Android/ARM-DSP/
host$ tar xvzf DM37X.tar.gz
```

Colocar el filesystem generado en la carpeta "Filesystem" y eliminar el archivo filesystem existente.

```
host$ cp ~/rowboat-android-
dsp/out/target/product/omap3evm/rootfs.tar.bz2 DM37X/Filesystem/
host$ cd DM37X/
host$ rm -r Filesystem/rootfs_dm37x.tar.bz2
```

Ahora ya se tienen las imágenes necesarias para compilar la tarjeta SD. Lo siguiente es insertar una tarjeta SD, que tenga una capacidad mínima de 2GB de memoria, en el lector de tarjetas. Ubicar la tarjeta y verificar que se encuentre en */dev*, ya que si se realiza el formateo de la tarjeta de manera incorrecta se puede echar a perder el disco duro.

```
host$ dmesg
host$ ls /dev
```

La carpeta de binarios pre-compilados incluye la herramienta "mkmmc-android.sh. En caso que la tarjeta SD se encuentre en */dev/sdc* continuar con el comando siguiente, cuando se pregunte si quiere formatear la tarjeta SD, escribir "yes".

```
host$ sudo ./mkmmc-android.sh /dev/sdc
```

En caso que la tarjeta SD se encuentre en */dev/mmcblk0*, se debe modificar el archivo. Pero primero se debe guardar un respaldo del archivo.

```
host$ cp mkmmc-android.sh mkmmc-android.sh.bak
```

Abrir el documento "mkmmc-android.sh" con la herramienta "gedit", seleccionar todo el contenido y borrar. Copiar lo indicado en el ANEXO A.1.2. FORMATO TARJETA SD ARM-DSP, y pegarlo en el documento modificado, guardar y cerrar. En el Terminal colocar, cuando pregunte si quiere formatear la tarjeta SD, poner "yes".

```
host$ sudo ./mkmmc-android.sh /dev/mmcblk0
```

4.4.6. Configuración de Hardware

El siguiente paso es configurar la plataforma en cuanto a Hardware, y que esto asegura la adecuada ubicación de switches y de cables, para que la tarjeta trabaje correctamente. Las principales configuraciones pueden ser observadas en la Figura 4.6.

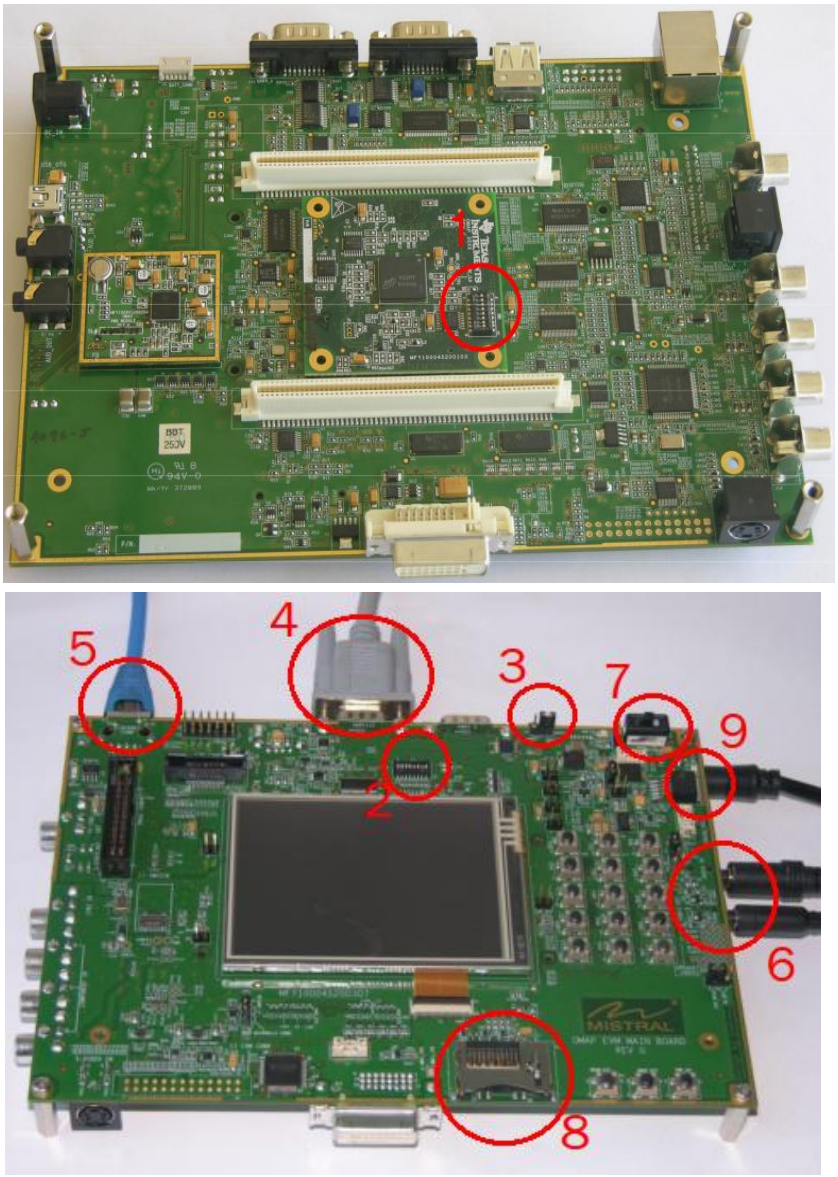
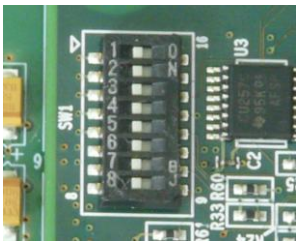
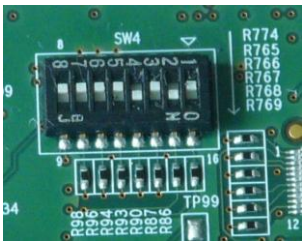


Figura. 4.6. Configuración de Hardware EVM OMAP3530

La configuración de Hardware consta de algunos pasos importantes[71] como los mostrados a continuación, esta configuración es básica, pero puede variar de acuerdo a la utilización que se le dé a la tarjeta.

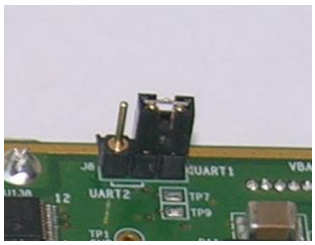


1 El SW1 se encuentra en el módulo de procesamiento, ubicado en la parte posterior de la tarjeta. SW1 debe estar como en la Figura para un correcto ruteo de señales.

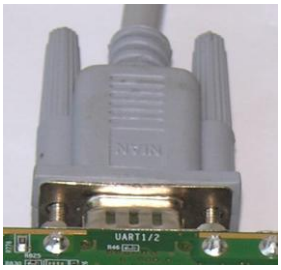


2 En la parte superior de la tarjeta, el SW4. Para buteo, la SD Card debe configurarse así: (I: on, O: off)

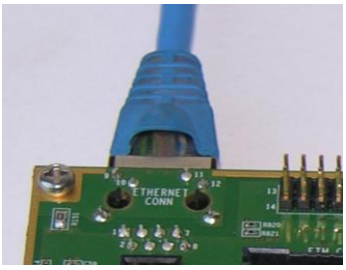
8	7	6	5	4	3	2	1
O	**O**	**O**	**O**	**I**	**I**	**I**	**O**



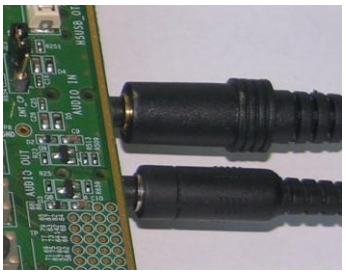
3 El puerto UART1 será utilizado. Esto es controlado por el Jumper 8, que debe ser configurado como muestra la figura



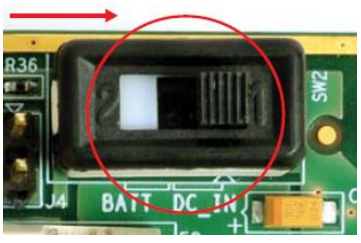
4 Colocar el conector DB9 del cable serial al puerto UART-1/2. El otro terminal debe ser conectado hacia HOST PC.



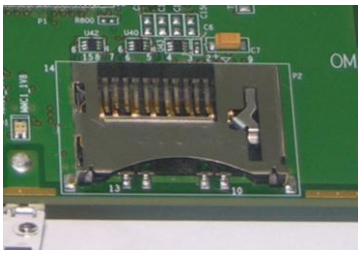
5 Conectar un cable Ethernet al puerto RJ-45 de la tarjeta. Conectar el otro extremo a un Router, Switch Ethernet o PC.



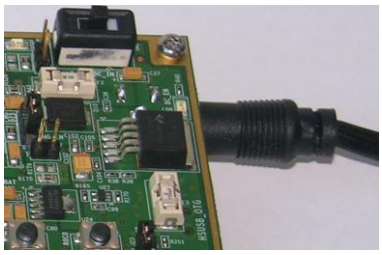
6 La entrada de audio puede ser conectada a una fuente Emisora de audio. La salida de audio puede ser conectada a un elemento Receptor de audio.



7 Colocar el SW2 en posición DC Power (lado derecho). SW2 no es un SW de on/off, debe dejarse en una posición fija.



8 Colocar la tarjeta SD (min 2G). La tarjeta SD contiene el sistema operativo y bootloaders.



9 Colocar el cable de poder en el Jack P19 al lado superior derecho de la plataforma. Esto enciende la tarjeta.

Debe existir una conexión de red entre la tarjeta y la Host PC, esta comunicación servirá para la administración de archivos, para la instalación de aplicaciones (.apk), así como también para la depuración de aplicaciones.

La comunicación puede establecerse de dos maneras, a través de una comunicación Ethernet, o por USB. En el primer caso, se debe conectar el cable Ethernet al puerto Ethernet de la PC o a la Red Local y la Tarjeta. En el de caso USB como enlace de comunicación, utilizar un cable mini-usb, para lo caual un extremo debe ser conectado hacia el puerto USB del HOST y en el lado de la tarjeta debe ser conectado al puerto USB OTG.

4.4.8. Arranque del Sistema Android

En el Terminal iniciar una sesión Minicom:

```
host$ minicom
```

Encender la Tarjeta EVM OMAP3530, en el terminal Minicom se mostrará una cuenta regresiva, pulsar cualquier tecla y se accederá al sistema *u-boot*, se deben configurar variables para el arranque del sistema desde la tarjeta SD en los escenarios ARM y ARM-DSP.

Configuración de Variables ARM [71]

Para configuración de variables si la tarjeta fue configurada para ARM colocar:

```
OMAP3EVM # setenv bootargs 'mem=128M console=tty0
console=ttyS0,115200n8 androidboot.console=ttyS0
root=/dev/mmcblk0p2 rw rootfstype=ext3 init=/init rootwait ip=off
omap_vout.vid1_static_vrfb_alloc=y mpurate=1000'
OMAP3EVM # setenv bootcmd 'mmc init; fatload mmc 0 0x8200000
boot.scr; source 0x82000000'
OMAP3EVM # saveenv
OMAP3EVM # boot
```

Configuración de Variables ARM-DSP [72]

Para configuración de variables si la tarjeta fue configurada para ARM-DSP colocar:

```
OMAP3EVM # setenv bootargs 'mem=104M@0x80000000
mem=128M@0x88000000 console=tty0 console=ttyS0,115200n8
androidboot.console=ttyS0 root=/dev/mmcblk0p2 rw rootfstype=ext3
init=/init rootwait ip=off omap_vout.vid1_static_vrfb_alloc=y
mpurate=1000'
OMAP3EVM # setenv bootcmd 'mmc init; fatload mmc 0 80800000
uImage; bootm 80800000'
OMAP3EVM # saveenv
OMAP3EVM # boot
```

Este procedimiento será necesario una sola vez, a menos que se reconfigure u-boot. Al encendier la tarjeta, no se debe pulsar ninguna tecla y el sistema arrancará por sí solo. El primer arranque del sistema tomará más tiempo que el que normalmente toma. Una vez iniciado el sistema, la pantalla de "Minicom" mostrará el Shell de Android.

El entorno de usuario será presentado sobre la pantalla LCD (Figura 4.7), ubicada sobre la tarjeta, y se utilizará el teclado de la plataforma, sus botones tienen funciones específicas para Android (Tabla 4.3)[64, 71].

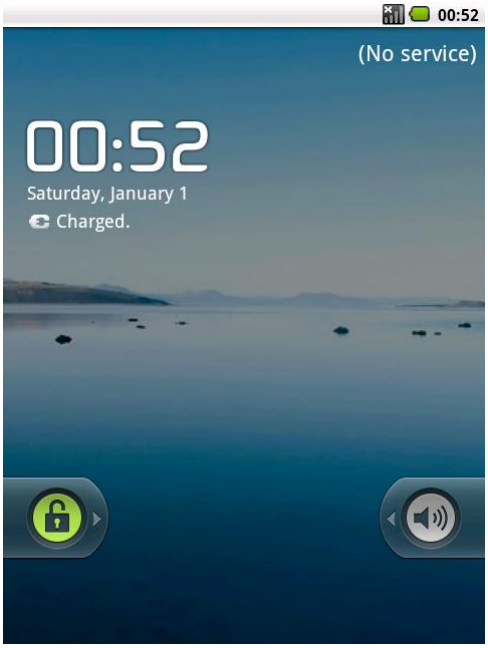


Figura. 4.7. Entorno de Usuario Android

Tabla 4.3. Teclado EVM35x

S4 Menu	S7 Back	S10 Arriba	S13 Home	S16
S5 Vol +	S8 Izq.	S11 Enter	S14 Dere.	S17
S6 Vol -	S9	S12 Abajo	S15	S18 Sleep

4.4.9. Instalación SDK Android

El SDK de Android proporciona las herramientas, ejemplos de código y documentación que se necesita para crear aplicaciones sobre equipos con Android. El procedimiento, según [64], para su instalación comprende lo siguiente:

- Preparar el equipo de desarrollo (HOST) y garantizar que se cumplen los requisitos del sistema.
- Instalar el paquete SDK.
- Instalar el plugin ADT para Eclipse.

Preparación del HOST

Los requisitos solicitados para la instalación de SDK Android, incluyen la instalación de un IDE de desarrollo, que en este caso es Eclipse y la instalación de JDK.

- **Entorno de Desarrollo Eclipse**

Para el desarrollo de aplicaciones se utilizará el IDE Eclipse, su instalación se realizará desde el Gestor de Paquetes “Synaptic Package Manager” de Linux en la estación de trabajo. Para iniciar Synaptic, pulsar en el menú System > Administration > Synaptic Package Manager. Luego en la sección “Quick Search” escribir: “eclipse”. Esto desplegará la lista de paquetes con esta palabra, buscar el paquete y seleccionarlo como se muestra en la Figura 4.8. Pulsar el botón “Apply” y se instalará Eclipse, con todas las dependencias necesarias.

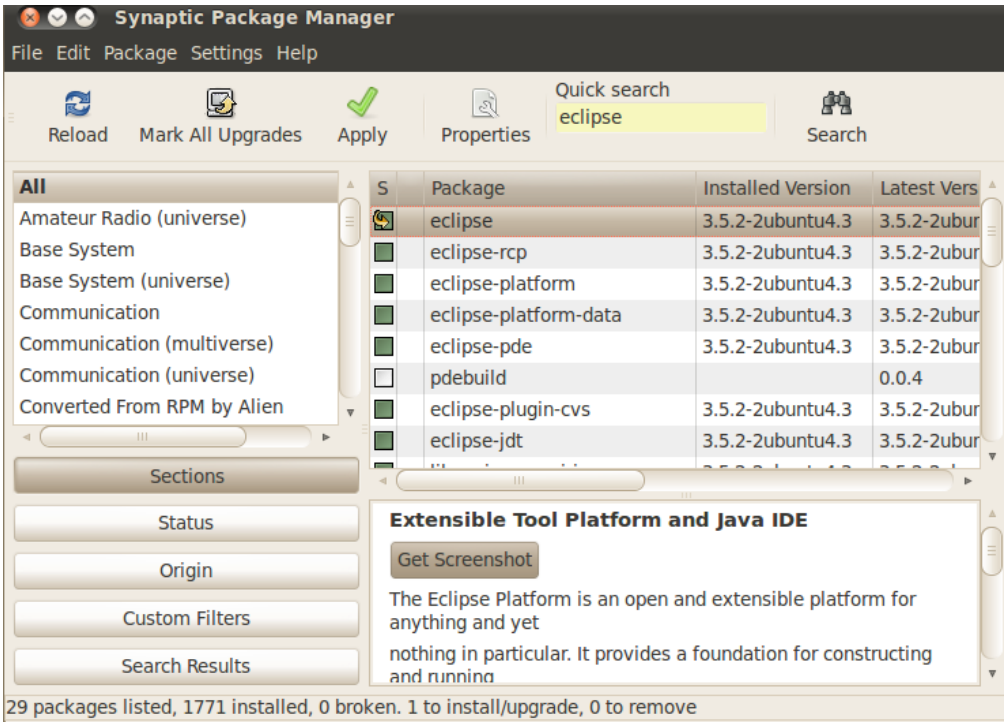


Figura. 4.8. Instalación de Eclipse usando Synaptic

- **Java Development Kit**

Java Development Kit o (JDK), es un software que provee herramientas de desarrollo para la creación de programas en java. Para la instalación del SDK Android se requiere un JDK versión 5 o 6.

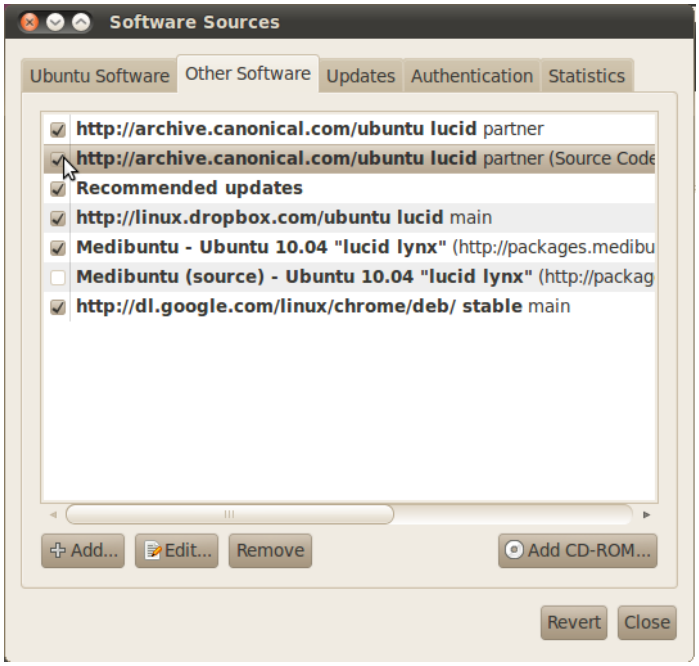


Figura. 4.9. Software Sources para JDK

Para su instalación, pulsar en el menú System > Administration > Software Sources. Seleccionar la pestaña “Other Sources” y marcar las opciones (Figura 4.9):

- deb http://archive.canonical.com/ubuntu lucid partner
- deb-src http://archive.canonical.com/ubuntu lucid partner

Iniciar Synaptic, en la sección “Quick Search” escribir: “jdk”. Esto desplegará la lista de paquetes que contengan la palabra, buscar el paquete “sun-java6-jdk”y seleccionarlo como se muestra en la Figura 4.10. Pulsar el botón “Apply” y el JDK será instalado.

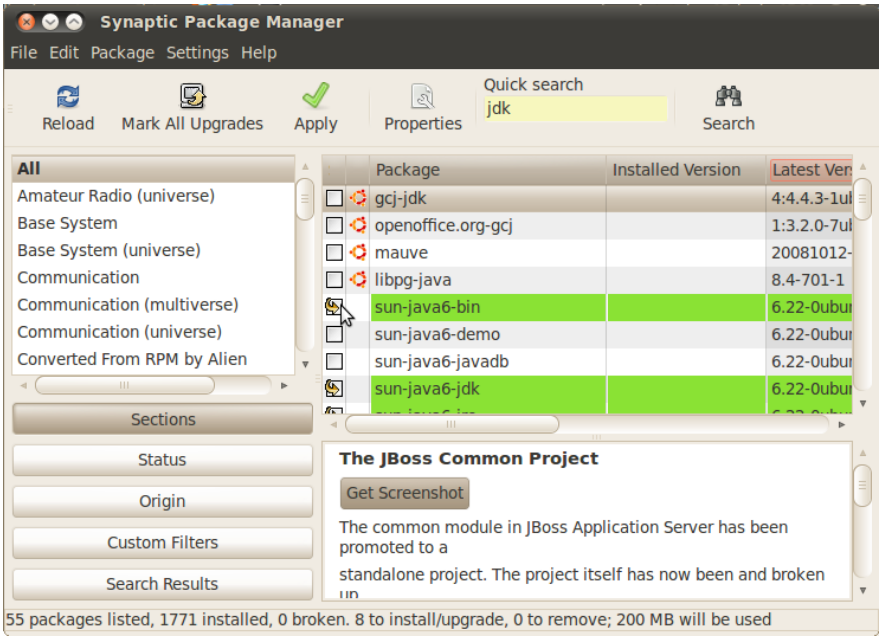


Figura. 4.10. Synaptic instalación JDK

Instalación de SDK Android [64]

El SDK debe ser descargado de la página oficial de Android en [64], en su última versión, se considera que el archivo fue descargado en *~/Downloads*. En este Tutorial se utilizó un SDK en su versión 10, pero puede descargarse una versión más actualizada.

```
host$ cd
host$ cp Downloads/android-sdk_r10-linux_x86.tgz ~
host$ tar xvzf android-sdk_r10-linux_x86.tgz
```

Se creará la carpeta *~/android-sdk-linux_x86* en Home.

ADT Android Development Tools [64]

Herramientas de desarrollo de Android (ADT, Android Development Tools) es un plugin para el IDE Eclipse que está diseñado para darle un ambiente de desarrollo integrado para la construcción de aplicaciones Android.

ADT amplía las capacidades de Eclipse para que pueda configurar rápidamente nuevos proyectos Android, crea una interfaz de usuario de aplicación, añade componentes basados en la API del Framework Android, depura las aplicaciones utilizando las herramientas de SDK de Android, y hace posible la exportación de archivos .apk con el fin de distribuir la aplicación.

- **Instalación de plugin ADT para Eclipse**

Para su instalación primero iniciar el IDE Elipse, para esto pulsar en el menú Applications > Programming > Eclipse.

Una vez abierto Eclipse, seleccionar el menú Help > Install New Software, se abrirá la ventana “Install”, seleccionar el botón “Add” de la esquina superior derecha. Aparecerá una venta de diálogo “Add Site” (Figura 4.11), escribir “ADT Plugin”en “Name” y en “Location” introducir la siguiente dirección URL:

- https://dl-ssl.google.com/android/eclipse/

Seleccionar “OK”.

Se mostrará a la pantalla “Install” nuevamente (Figura 4.12), proceder a escoger la opción “Developer Tools” y seleccionar “Next”.

En la siguiente ventana, se mostrará una lista de las herramientas a ser descargadas. Hacer clic en “Next”. En la próxima ventana se mostrará el acuerdo de Licencia, leer, aceptarla, y finalizar. Reiniciar Eclipse.

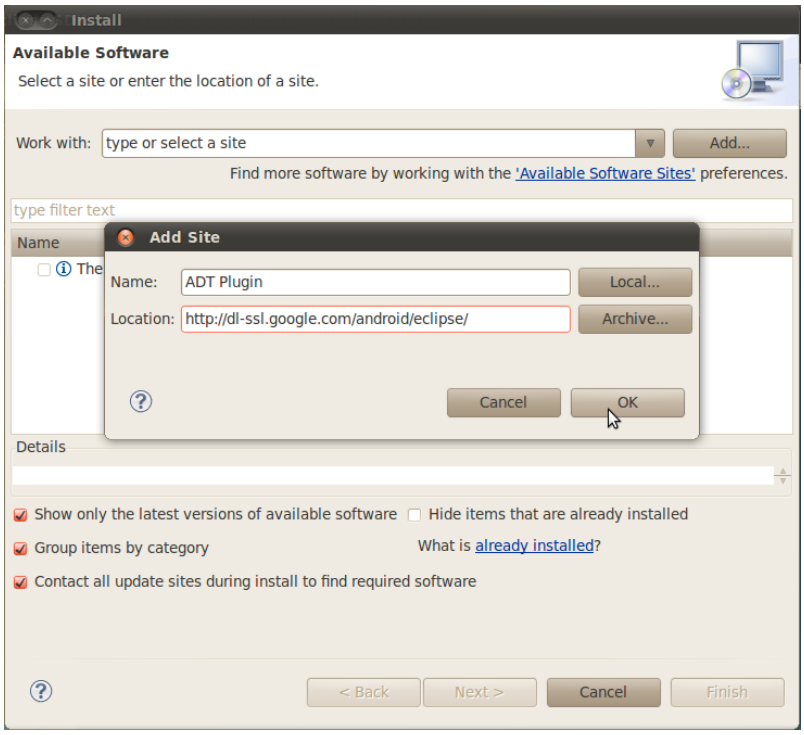


Figura. 4.11. Instalación ADT: Add Site

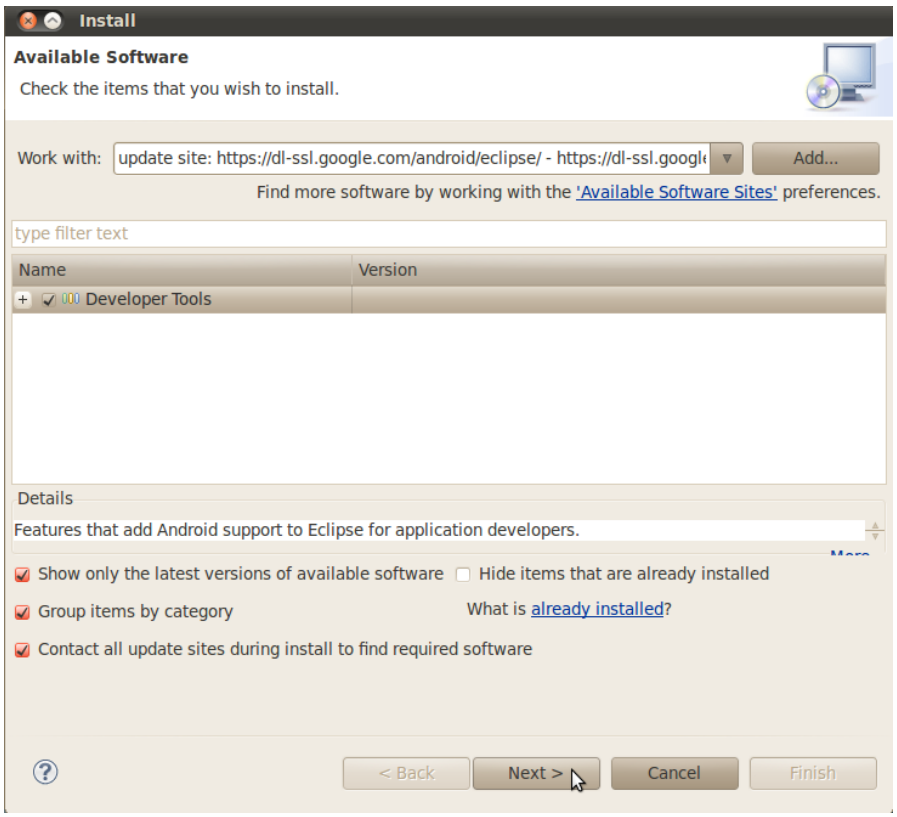


Figura. 4.12. Instalación ADT: Install

- **Configuración del plugin ADT para Eclipse**

Una vez que haya descargado e instalado correctamente el ADT como se describió anteriormente, el siguiente paso es modificar las preferencias de ADT en Eclipse para que apunte al directorio de Android SDK.

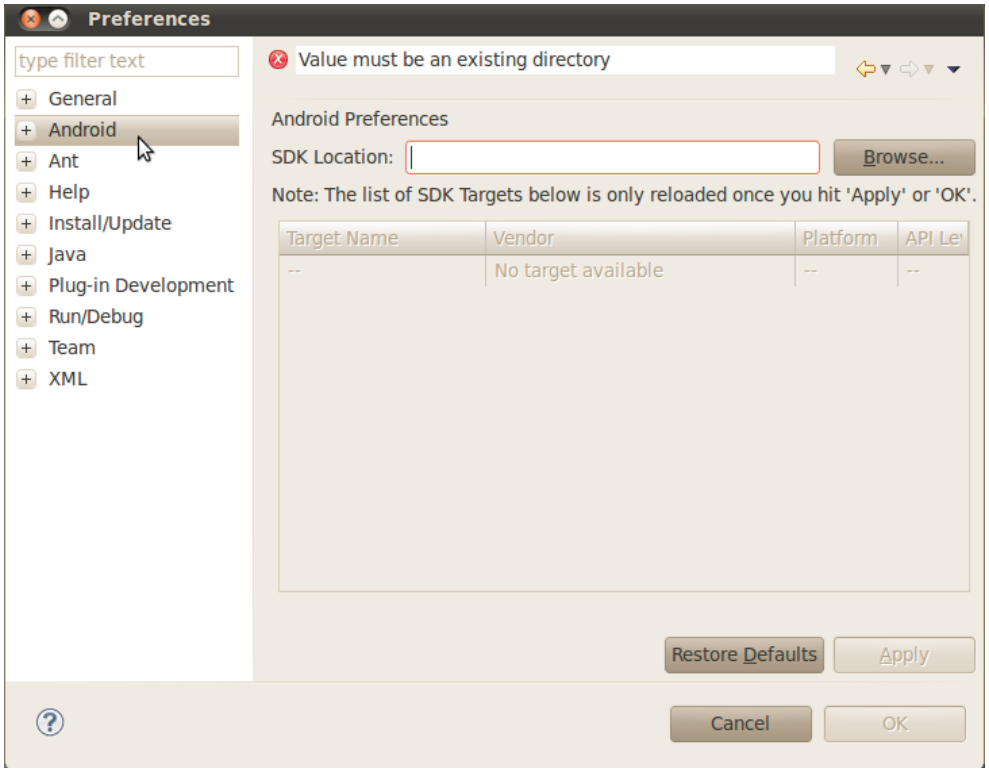


Figura. 4.13. Configuración ADT

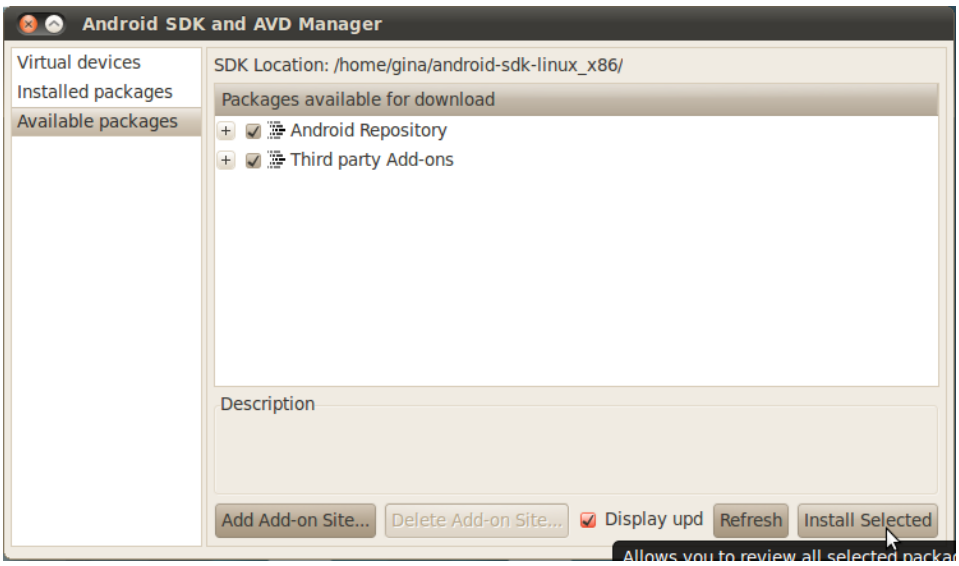


Figura. 4.14. Configuración Android SDK and AVD Manager

Seleccionar Window > Preferences para abrir el panel de preferencias. Seleccionar Android en el panel de la izquierda (Figura 4.13). Para la ubicación de SDK en el panel principal, hacer click en "Browse" y localizar el directorio SDK descargado. En el caso de este Tutorial se encontrará en el directorio *~/ android-sdk-linux_x86*. Seleccionar "Apply" y luego "OK".

Android SDK and AVD Manager

El paso final para la configuración del SDK Android es definir a herramienta SDK y Administrador AVD (Android Virtual Device). Para lo que se requiere descargar los componentes esenciales para el entorno de desarrollo.

En Eclipse, seleccionar el menú Window > Android SDK and AVD Manager, seleccionar la pestaña "Available Packages" y al seleccionar los repositorios se cargarán componentes para la actualización del SDK (Figura 4.14). Seleccionar el botón "Install Selected", se instalarán los componentes seleccionados en el entorno SDK.

ADB Android Debug Bridge [64]

ADB (Android Debug Bridge) es una herramienta que forma parte del SDK Android, permite administrar el estado de un dispositivo con Android. Además se puede utilizar para instalar una aplicación Android (.apk) en el dispositivo o enviar y obtener archivos. Forma parte del paquete SDK de Android, Se encuentra ubicada en el directorio ~/*android-sdk-linux_x86/tools/*.

Esta herramienta será utilizada para la comunicación de la tarjeta con la HOST PC, y la herramienta DDMS (Dalvik Debug Monitor Server), la cual será especificada más adelante. Pata la utilización del SDK, es necesario exportar el directorio PATH, al archivo ".bashrc", como se muestra:

```
host$ export PATH=${PATH}:~/android-sdk-linux_x86/tools/
host$ source .bashrc
```

4.4.10. Depuración de Aplicaciones

Comunicación EVM OMAP3530 – HOST PC [71]

Una vez encendida la tarjeta, es necesario configurar a la tarjeta, en modo de Depuración. Para esto, en el teclado, aplastar el botón Menu (S4), se desplegará un ventana sobre la pantalla LCD de la tarjeta, seleccionar la opción Settings > Applications > Developent y habilitar la todas las opciones ahí presentadas.

Los métodos de comunicación con la tarjeta utiliza en el Tutorial son:

- ADB sobre USB
- ADB sobre Ethernet

- **ADB sobre USB**

Asegurarse que el cable nimi-usb se encuentre conectado entre el puerto USB de la HOST PC y al puerto USB OTG de la tarjeta, y el Modo de Depuración este activo como se mostró anteriormente.

En el HOST, es necesario realizar procedimientos para detectar la tarjeta. Se debe agregar un archivo de reglas *udev* que contenga la configuración USB para cada tipo de dispositivo, en este caso EVM OMAP35x.

En el archivo de reglas, cada fabricante es identificado por un ID de vendedor único. Para identificarlo realizar lo siguiente.

```
host$ lsusb -v | grep idVendor
```

En la lista que se despliega, ubicar el dispositivo que no tenga nombre, el cual corresponde al equipo EVM OMAP35x. Para otros dispositivos, aparece el nombre claramente al lado derecho.

```
  idVendor           0x18d1
```

También es necesario conocer el nombre del ususario del HOST, para eso colocar:

```
host$ id
uid=1000(<usuario>) gid=1000(<grupo>) …
```

Se mostrá la identificaciones utilizadas en el HOST. El archivo de reglas utilizará la información del ID de fabricante que para OMAP35x siempre es “18d1” y el nombre de usuario <usuario>. Crear el archivo 51-android.rules, con la herramienta “vim”.

```
host$ sudo vim /etc/udev/rules.d/51-android.rules
```

A continuación, se mostrará una pantalla vacía, pulsar la tecla “i”, y aparecerá un cursor en la primera fila. Escribir lo siguiente, recordando reemplazar la palabra **<usuario>** por la correcta:

```
SUBSYSTEM=="usb", SYSFS{idVendor}=="18d1", MODE="0666",
OWNER="<usuario>"
```

Pulsar la tecla “Esc”, escribir “:qw” y pulsar “Enter”. Luego actualizar udev y cambiar el modo de usuario:

```
host$ sudo /etc/init.d/udev restart
host$ sudo chmod a+r /etc/udev/rules.d/51-android.rules
```

Los procedimientos realizados configuran el HOST para identificar la plataforma EVM OMAP3530 cada vez que esta sea conectada al PC, por lo tanto solo necesitan ser ejecutados una vez. Para poder utilizar el dispositivo, se debe ubicar el equipo utilizando:

```
host$ adb kill-server; adb start-server
host$ adb devices
List of devices attached
20100720   device
```

- **ADB sobre Ethernet**

Asegurarse que el cable directo Ethetnet se encuentre conectado a una red, puede ser de un Router. Si no es posible, se configurará al HOST PC para una conexión Ethernet punto a punto con la tarjeta. El procedimiento mostrado obedece a una Red Clase C, con dirección de red: 192.168.1.0, mascara de red: 255.255.255.0

Configuración Ethernet HOST PC: Este paso es necesario solamente si no se cuenta con una red a la que se encuentren asociados la HOST PC y EVM OMAP3530. Con el siguiente comando, se habilita el puerto de red Eth0 en la HOST PC, se utilizará la dirección 192.168.1.100.

```
host$ sudo ifconfig etho 192.168.1.100 netmask 255.255.255.0 up
```

Ahora se debe configurar a EVM. Para acceder al Shell de la tarjeta EVM, se utilizará la herramienta "minicom". Asegurarse que la herramienta este activa desde el encendido de la tarjeta. La dirección utilizada será 192.16.1.101. Los comandos para configurar el puerto de red en la tarjeta serán los siguientes:

```
target# ifconfig eth0 192.168.1.101 netmask 255.255.255.0 up
setprop service.adb.tcp.port 5555
target#  stop adbd
target#  start adbd
```

En la PC, activar la herramienta ADB, ligarla a la IP de la tarjeta y desplegar los dispositivos conectados, mediante:

```
host$ ADBHOST=192.168.1.101
host$ adb kill-server; adb start-server
host$ adb devices
List of devices attached
emulator-5554   device
```

4.4.11. Herramientas Android

Una vez configuradas todas las herramientas necesarias para el desarrollo de aplicaciones y el sistema se encuentre listo para el desarrollo, es preciso conocer cómo trabajar con las mismas. Android permite la instalación de aplicaciones por parte de terceros o desarrolladas por el usuario.

Instalación de Aplicaciones en Eclipse

En Eclipse es necesario solamente correr la aplicación para que este se instale en la plataforma de desarrollo, previamente reconocida por ADB.

Instalación de Aplicaciones con ADB [64]

Algunas aplicaciones existentes en el mercado, presentan sus aplicaciones en un archivo de formato .apk. La herramienta ADB, puede ser utilizada para instalar dichas aplicaciones, una vez que el equipo este activo, como se mostró en la sección anterior. El siguiente procedimiento, es utilizado para instalar cualquier aplicación .apk, en la tarjeta. Ejecutar el siguiente comando en una ventana de terminal del Host, colocando el nombre <paquete>.

```
adb install <paquete>.apk.
```

Si la instalación fue exitosa, se presentará el mensaje "SUCCESS".

Para la desinstalación de aplicaciones existen dos métodos. El primero es mediante la herramienta ADB:

```
adb uninstall <paquete>.apk.
```

El segundo puede ser realizado directamente en la interface de usuario de la tarjeta. Pulsando la Tecla "Menu", se mostrá una ventana, seleccionar **Settings > Applications > Manage application.** Ubicar el paquete a desinstalar, seleccionarlo y oprimir la opción Uninstall > OK > OK.

Dalvik Debug Monitor Server (DDMS) [64]

DDMS es una herramienta de monitoreo y depuración incluida con el SDK, que ofrece servicios de redirección de puertos, captura la imagen de la pantalla, muestra mensajes "LogCat" de registros, procesos e información del estado del dispositivo conectado.

La herramienta DDMS se encuentra integrada a Eclipse, se puede acceder a esta seleccionando el menú **Window > Open Prespective > Other**y cuando se muestre una ventana ventana, seleccionar la pestaña "DDMS", también se puede acceder mediante el ícono DDMS al lado superior derecho del la ventana del IDE Eclipse (Figura 4.15)

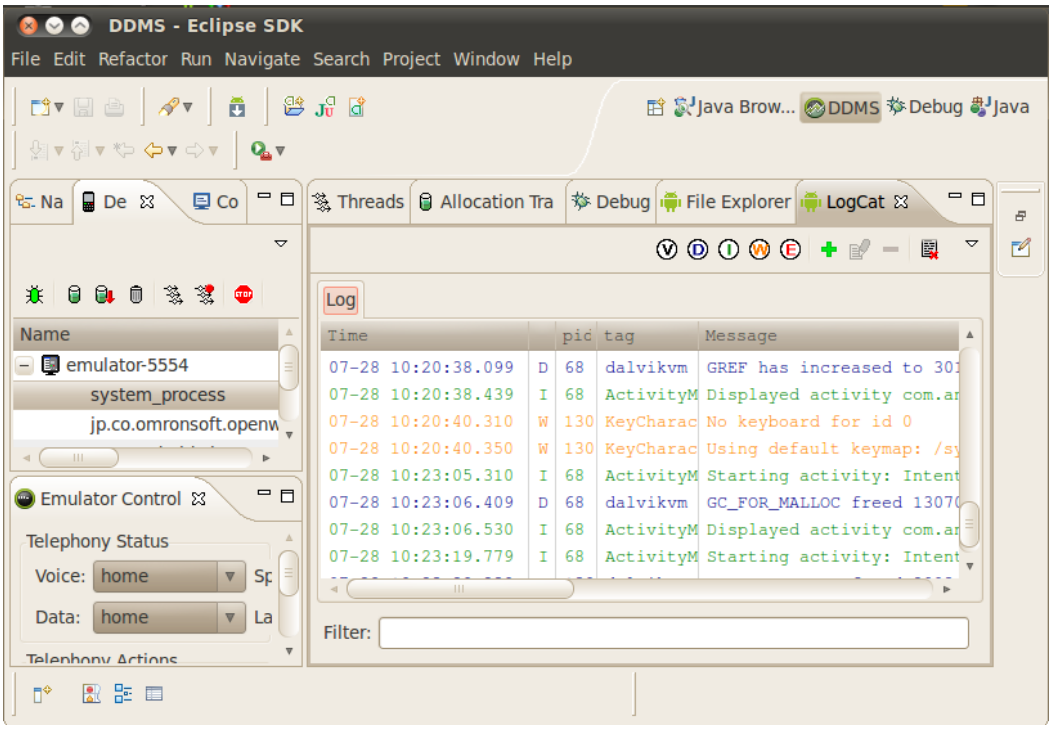


Figura. 4.15. Configuración DDMS

Transmisión de archivos a través de ADB [64]

La herramienta ADB facilita la transmisión y adquisición de archivos desde y hacia la plataforma, a través del commando “pull”y “push”. El comando debe ser utilizado de la siguente forma:

Para copiar un archivo o directorio desde la tarjeta, utilizamos:

```
host$ adb pull <dir. Remota> <dir. Local>
```

Para copiar un archivo o directorio hacia la tarjeta, utilizamos:

```
host$ adb pull <dir. Local> <dir. Remota>
```

Donde <dir. Local> es el Path del archivo o directorio en el HOST PC y <dir. Remota> es el Path del archivo o directorio en EVM OMAP35x.

Por ejemplo:

```
host$ adb push bla.mp4 /sdcard/
```

CAPÍTULO 5

DISEÑO DE LA APLICACIÓN SISTEMA DE REFERENCIA OMAP-MULTIMEDIA (SRO-M)

Resumen: Este capítulo detalla las etapas o procedimientos necesarios para el diseño y desarrollo de aplicaciones Android sobre la tarjeta EVM OMAP35x, usando como modelo un sistema de referencia (SRO-M), y empleando herramientas y metodologías desarrollo adecuadas. Como ejemplo se presentan varias implementaciones básicas.

5.1. ESPECIFICACIONES

La mejor manera de comprender la programación es a través de ejemplos, es por eso que se ha preparado un pequeño sumario de aplicaciones básicas que muestran claramente metodologías de desarrollo para aplicaciones Android, en las que se empleen las prestaciones más representativas de OMAP3530.

El Sistema de Referencia OMAP-Multimedia (SRO-M) es el nombre otorgado a la aplicación que va a ser desarrollada a continuación. La aplicación se encuentra conformada por cinco aplicaciones sencillas, a las cuales se puede acceder a través de un menú principal.

5.1.1. Especificaciones Aplicación SRO-M

La aplicación SRO-M se encuentra conformada por cinco aplicaciones ndependientes, mostradas a continuación en la Tabla 5.1:

Tabla. 5.1. Aplicaciones SRO-M

Aplicaciones SRO-M	**Características**
Conversor de Unidades	Es el primer ejemplo práctico de una actividad sencilla para transformar grados Fahrenheit a Celsius y viceversa.
3D Graphics Rendering (OpenGL)	Aplicación que presenta un Cubo 3D, utiliza las librerías OpenGL de gráficas 3D que es compatible con el subsistema acelerador gráfico POWERVR SGX de la plataforma. Describe un menú que indica el proceso para crear un cubo giratorio transparente en cada una de sus fases.
Video Player	Es un reproductor de video que muestra un video incluido en el sistema y ubicado en la tarjeta SD.
Audio Recorder/Player	Incluye dos programas destinados a la grabación de audio, el primero "SoundRecorder Android" es el método más sencillo para grabación de audio, es una simple llamada al programa predeterminado de Android para grabación de audio. El segundo es un programa “Audio Recorder/Player, mediante este se puede indicar el destino del archivo creado.
About	Despliega un mensaje para conocer más acerca del proyecto.

Dentro de estas aplicaciones se muestran diferentes métodos de programación con el objeto de cubrir una gran cantidad de opciones para el desarrollo Android, utilizando algunas prestaciones específicas del SoC OMAP3530, como es el caso de gráficas 3D, codificación y decodificación de audio y video, creación y lectura de archivos.

Se ha tratado cubrir variedad en cuanto a desarrollo se refiere y se ha implementado diversos diseños de pantalla ya sea mediante .xml o con la utilización de herramientas de diseño gráfico libres.

Antes de exponer cómo desarrollar dichas aplicaciones, es necesaria una pequeña introducción sobre los fundamentos en el desarrollo de aplicaciones Android.

5.1.2. Fundamentos de una Aplicación Android

Bloques de Construcción

La plataforma de Android proporciona diferentes componentes [64] a la hora de programar en función del objetivo de la aplicación, aunque son pocos, es necesario familiarizase con ellos ya que son muy utilizados para desarrollo:

- **Activity:** Es el componente más usado en las aplicaciones Android. Típicamente representa una pantalla individual en el terminal y presenta una interfaz gráfica al usuario. Las aplicaciones pueden definir una o más *Activities* a ser manejadas en diferentes fases de su programa, cada una de ellas es responsable de guardar su propio estado de esta forma puede ser llamado a cualquier tiempo dentro del ciclo de la aplicación.
- **Intent:** Este tipo de componente es una clase especial que utiliza Android para moverse de una pantalla a otra. Un *Intent* describe lo que una aplicación desea hacer. Cualquier *Activity* puede reutilizar funcionalidades de otros componentes con solo hacer una solicitud en la forma de *Intent*.
- **Services:** Es una tarea que no tiene interfaz gráfica, pero puede ejecutarse en “background" por un tiempo indefinido (se asemeja mucho al demonio de los sistemas Linux), sin que el usuario interactué con él directamente. Es el caso de cuando se utiliza un programa que toca música y esta sigue tocando aún cuando el usuario está utilizando otra aplicación.
- **Content Provider:** Se encarga de almacenar datos de un determinado tipo que pueden ser accedidos desde cualquier aplicación. Es la mejor manera para compartir datos globales entre aplicaciones, pueden ser contactos, imágenes, tonos, etc.
- **Broadcast receivers:** Este tipo de componentes se utilizan para recibir y reaccionar ante ciertas notificaciones broadcast. Este tipo de componentes no tienen interfaz gráfica y pueden reaccionar ante eventos como cambio de zona horaria, llamadas, nivel de batería.

Ciclo de vida de una Activity

Cada *Activity* de un programa puede estar en diferentes estados, mostrados en la Figura 5.1[64], se caracterizan porque su tiempo de vida no es controlado directamente por la aplicación, el sistema es el que decide y determina el tiempo de vida basándose en el uso y capacidades del sistema. Se puede manipular acciones para cuando se ejecuten los diferentes estados, pero aunque no se los incluya en la aplicación esto no quiere decir que no se ejecuten. Los estados serán detallados brevemente a continuación:

- **onCreate( ):** Se ejecuta cuando se crea la *Activity* por primera vez. Aquí es donde se deberían crear views, en definitiva el proceso de inicialización de la aplicación.
- **onRestart( ):** Se ejecuta cuando la aplicación se ha cerrado y se va a ejecutar nuevamente.
- **onStart( ):** Se ejecuta cuando la aplicación aparece visible para el usuario.
- **onResume( ):** Se ejecuta cuando la *Activity* puede empezar a interactuar con el usuario. Es un buen lugar para ubicar reproducción de música o animaciones.
- **onPause( ):** Se ejecuta cuando el sistema está a punto de continuar una *Activity* anterior. Se utiliza típicamente para guardar datos que no se han grabado anteriormente, parar animaciones y otras acciones que consuman CPU.
- **onStop( ):** Se ejecuta cuando la *Activity* deja de ser visible al usuario, porque otra *Activity* ha continuado. Puede ocurrir porque una nueva *Activity* ha sido creada, una *Activity* ya creada pasa a primer plano o ésta está siendo destruida.
- **onDestroy( ):** Última llamada antes de destruir la *Activity*. Puede ocurrir porque su actividad está acabando, o porque el sistema destruirá la instancia para guardar espacio.

Uso de Recursos

En el proyecto, los recursos se encuentran ubicados en el directorio ***res***, se trata de archivos que no son código, es decir texto, imagenes rasterizadas, backgrounds, colores, etc, necesarios en la aplicación. Estos deben ser agrupados de acuerdo a su formato, por ejemplo las imagenes rasterizadas (.png o .jpg) o backgrounds (.xml) deben encontrarse dentro de la carpeta ***res/drawable*** y archivos .xml que describan diseños de pantalla deben

ubicarse dentro de ***res/layout***. El compilador de recursos del sistema se encarga de comprimir y empaquetar los recursos y con estos generar una clase llamada ***R*** que contiene los identificadores de referencia que se utiliza para poder acceder a ellos en el programa.

Interface de Usuario

El diseño de interfaces de usuario en Android pueden ser diseñadas usando los métodos: Procedimental y Declarativo [75]. Procedimental o Imperativo se refiere a código creado por el programador en Java e implementado en la interfaz de usuario, por ejemplo, cuando se programa una aplicación Swing, se escribe código Java para crear y manipula todos los objetos de la interfaz de usuario como botones. El diseño Declarativo no implica introducir código, se trata de describir como lucirá la implementación. Android utiliza XML (Extensible Markup Language), que es similar al lenguaje HTML(HyperText Markup Language) para páginas web que trabaja con etiquetas. XML por ejemplo, sirve para describir como lucirá un botón, en cuanto a color, tamaño, ubicación, pero no se puede decir cuál es la acción el botón, esto se lo realizará en java.

Android permite crear interfaces de usuario en cualquiera de los estilos Procedimental y Declarativo, y combinarlos. Se puede permanecer casi en su totalidad en código Java, o se puede quedar casi en su totalidad en los descriptores XML. Si se mira la documentación para cualquier componente de la interfaz de usuario para Android, se verá tanto APIs de Java y los correspondientes atributos declarativos XML que hacen lo mismo.

Se pueden utilizar cualquiera de los dos, pero el consejo de Google es utilizar el código declarativo XML tanto como sea posible. El código XML es a menudo más corto y fácil de entender que el correspondiente código de Java, y es menos probable que cambie en futuras versiones.

Seguridades

Android proporciona seguridad basada en permisos. El acceso a ciertas operaciones es restringido y para acceder a ellas se deben especificar en un archivo llamado “Android-Manifest.xml”, cuando el programa se ejecuta este archivo le informa las restricciones del programa.

5.2. ARQUITECTURA DE HARDWARE

La Arquitectura de Hardware SRO-M describe cinco aplicaciones las cuales serán presentadas sobre la tarjeta EVM35x, su arquitectura representada por bloques se muestra en la Figura 5.1. Se puede observar que todos los componentes se encuentran intercomunicados por L4. El procesamiento de las aplicaciones SRO-M se encuentran manejadas por los subsistemas IVA 2.2 y MPU. El primero concentra su funcionamiento hacia el procesamiento digital de señales de video y audio, el segundo se encargará del procesamiento general y códigos de control. El subsistema MPU se encuentra conformado por un procesador tipo ARM, el cual es capáz de realizar procesamiento digital de señales, es por esta razón es que en el presente proyecto se utilizarán las dos formas de procesamiento digital de señales, uno por el subsistema IVA y otro por MPU.

Es necesario incluir dispositivos externos que se conecten a los periféricos mostrados en la arquitectura, dichos componentes pueden ser de tipo on-board o externos los cuales serán especificados en la siguiente sección.

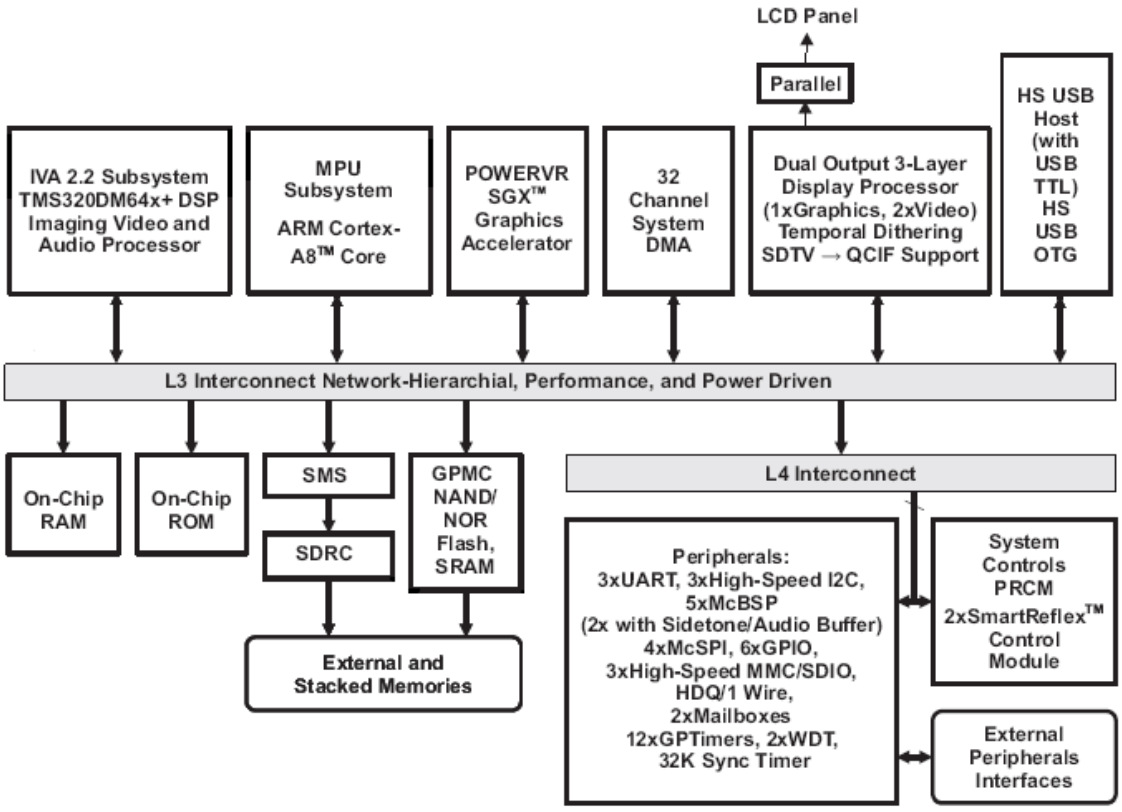


Figura. 5.1. Arquitectura de Hardware SRO-M

Componentes On-Board y Externos

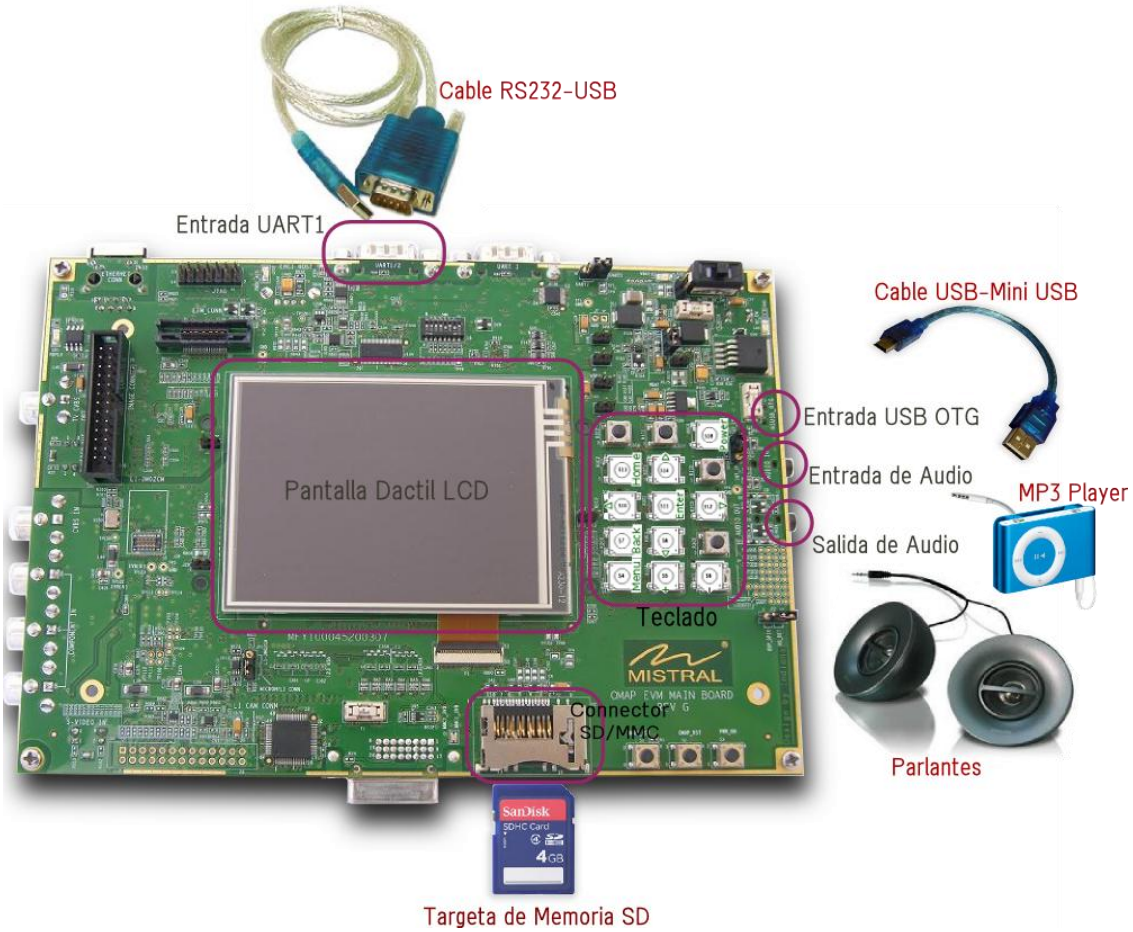


Figura. 5.2. Componentes On-Board y Externos

La distribución fisica de los componentes on-board y externos utilizados se presenta en la Figura 5.2, a continuación una pequeña descripción de cada uno:

- Elemento Receptor de Audio, Elemento Emisor de Audio y una tarjeta MMC/SD, y demás elementos para comunicación Host-PC.
- Elemento Receptor de Audio: Pueden ser audífonos o parlantes. En este caso se realizaron las pruebas utilizando parlante. Los parlantes pueden ser de cualquier tipo que requieran alimentación (*self powered*), plug 3.5mm
- Elemento Emisor de Audio: Puede ser un Reproductor de Audio Digital (MP3 Player) o en su defecto un micrófono autoalimentado, en este caso se utilizará un Reproductor de Audio digital.
- Memoria MMC/SD: Cuando se formateó de la tarjeta SD, como se muestra en el Capítulo 4, se realizan 3 particiones, una de ellas es destinada a almacenamiento

de datos, y deberá incluir un Video el cual será reproducido en la aplicación Video Player.

- Cable RS232: Necesario para la comunicación con Host-PC para inicio de boteo, el cual se conectará al Puerto UART 1. La tarjeta Incluye un cable RS232 pero se le adaptó un cable RS232-USB para computadoras que no posean puerto Serial.
- Cable de Red Directo o Cable mini USB-USB: Para comunicación mediante la Herramienta ADB, es necesario estar en red con la tarjeta. Se han probado dos opciones, a través del puerto de Red o a través del puerto USB OTG (Capítulo 4).
- Teclado: La tecla S7 (Back) del teclado es utilizada para todos los programas de SRM-O para salir del programa, no es necesario colocar ningún código para su cometido, ya que esta finaliza la *Activity* por defecto.

La tabla 5.1. muestra los elementos de hardware utilizados para el desarrollo de la aplicación SRO-M.

Tabla. 5.2. Requerimientos de Hardware SRO-M

Aplicaciones SRO-M	Elemento	Tipo
Conversor de Unidades	Pantalla Tactil LCD, Teclado	On-Board
3D Graphics Rendering	Pantalla Tactil LCD, Teclado	On-Board
Video Player	Pantalla Tactil LCD, Teclado Salida de Audio: Parlantes Memoria: Partición en Trajeta MMC/SD (min 2GB)	On-Board Externo Externo
Audio Recorder/Playback	Pantalla Tactil LCD, Teclado Salida de Audio: Parlantes Entrada de Audio: Reproductor de Audio Digital Memoria: Partición en Trajeta MMC/SD (min 2GB)	On-Board Externo Externo Externo
About	Pantalla Tactil LCD, Teclado	On-Board

5.3. ARQUITECTURA DE SOFTWARE

La arquitectura de Software SRO-M, mostrada en la Figura 5.3, esta basada en el Software Stack de Android analizado en el Capitulo 4, pero personalizado para el presente diseño. La Capa de Aplicaciones incluye a las aplicaciones SRO-M descritas en la Tabla 5.1. La capa de Framework de Aplicaciones proporciona las clases para crear aplicaciones Android. También proporciona una abstracción genérica para el acceso al hardware y periféricos, y administra la interfaz de usuario y recursos de la aplicación.

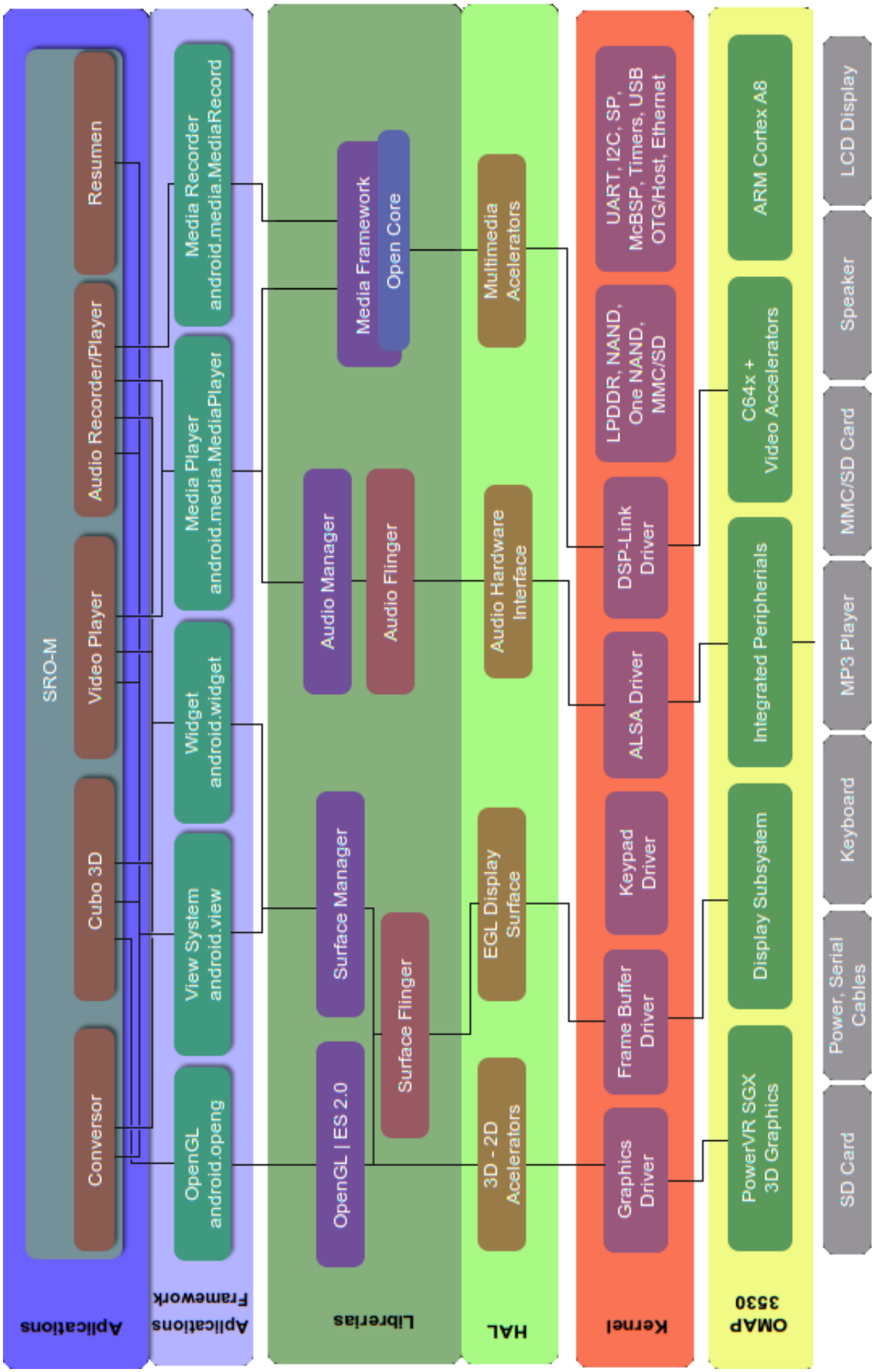


Figura. 5.3. Arquitectura de Software SRO-M

Como referencia, en la capa inferior se muestran los perifericos o componentes de hardware on-board utilizados, sin embargo esto no es parte de la arquitectura de software.

5.4. DESCRIPCIÓN DEL DISEÑO COMPLETO

A continuación se describirán metodologías de desarrollo para aplicaciones Android sobre la tarjeta EVM OMAP3530. Para lo que se utilizará el diseño del SRO-M como modelo, describiendo las *Activities* que comprenden la aplicación.

El tutorial inicia con la descripción del procedimiento para la creación de un nuevo proyecto Android, seguido por el procedimiento de la creación de un menú que contendrá el acceso para todas las aplicaciones de SRO-M, luego se definirán las demás aplicaciones.

Este tutorial se fundamenta en ejemplos de aplicaciones basadas en ejemplos encontrados en libros o en la web, los cuales han servido de base para la creación de aplicaciones básicas.

Las aplicaciones pueden ser comprobadas directamente sobre la plataforma, luego de realizados los procedimientos descritos en el Capítulo 4. Es necesario que todo el sistema esté completamente configurado; es decir, el sistema compilado sobre la plataforma, y realizadas las instalaciones y configuraciones para el funcionamiento de las herramientas de desarrollo en la estación de trabajo.

Además es necesario que la tarjeta se encuentre en modo de Depuración (Sección 4.4.9 Depuración de Aplicaciones), de esta manera cada vez que se realicen cambios en la programación de las aplicaciones, estas pueden ser probadas sobre la plataforma en ese mismo instante.

5.4.1. Creación de Aplicación SRO-M

Una vez abierto el IDE Eclipse, en el menú principal se seleccionar File > New > Project. Se mostrará un cuadro de diálogo “New Project”, seleccionar Android > Android Project, y pulsar “Next”.

Se mostrará una ventana “New Android Project" (Figura 5.4), sobre la cual se debe ingresar la información del proyecto, luego seleccionar “Finish”. Para el proyecto se consideraron los siguientes parámetros:

Project name: SRO_M
Build Target: Android 2.2
Application name: SRO_M
Package name: apps.tesis.omap
Create Activity: SRO_MActivity
Min SDK Version: 8

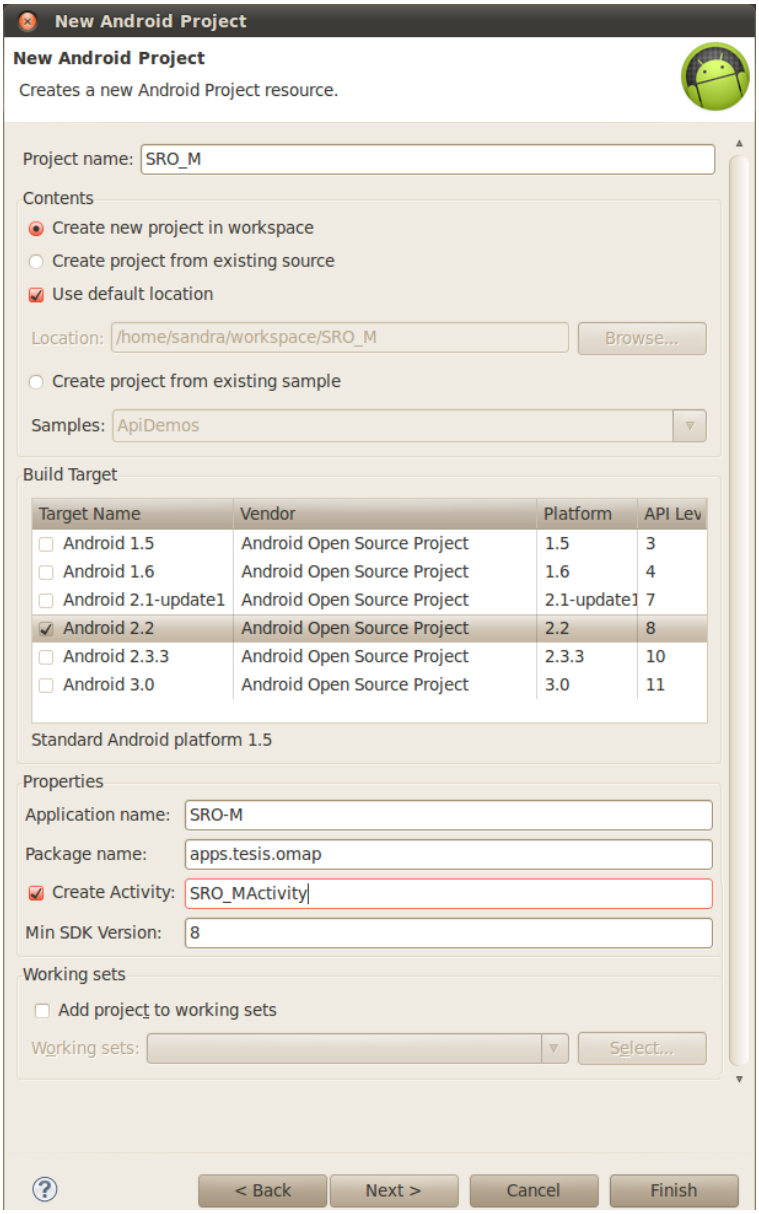


Figura. 5.4. New Android Project, aplicación SRO-M

A continuación una breve descripción de cada uno de los datos ingresados [64]:

- Project Name: Este es el nombre del proyecto, es decir el nombre del directorio que contendrá los archivos del proyecto.
- Application Name: Este es el nombre con el cual la aplicación será conocida por los usuarios.
- Package Name: Este es el nombre del *Package* (es el mismo concepto de *Package* en Java), que será utilizado al momento de generar la *Stub Activity*. El nombre utilizado debe ser único y distinto a todos aquellos intalados en el sistema.
- Create Activity: Este es el nombre de la clase que será generada por el plugin. Esta clase será una subclase de *Activity*. Una *Activity* es simplemente una Clase que puede ejecutarse y realizar una tarea.
- Min SDK Version: este valor especifica el mínimo nivel de API requerida por la aplicación.
- Otros campos: El checkbox "Use default location", permite cambiar la ubicación en el disco donde los archivos del proyecto serán generados y almacenados. Por defecto se encuentra direccionado al *Workspace* formado al momento de la instalación. "Build Target" indica para cual plataforma será compilada la aplicación.

Una vez configurados los parámetros, pulsar "Finish". El Plugin de Android generará el proyecto, e incluirá algunos archivos automáticamente. El programa contendrá el código que despliegue un texto, desde su creación, estará listo para ejecutar.

Para correr la aplicación, seleccionar el proyecto, pulsar el botón derecho del mouse y ubicar la opción Run As > Android Application. Otra forma de realizarlo, es pulsando el botón "Run" en la Barra de Herramientas y seleccionando el proyecto deseado.

Eclipse enviará una copia del programa a la tarjeta para ejecutar. Se mostrará una pantalla mostrando el texto: "Hello, World Android". La Figura 5.5. muestra como se presentará la pantalla en la tarjeta.

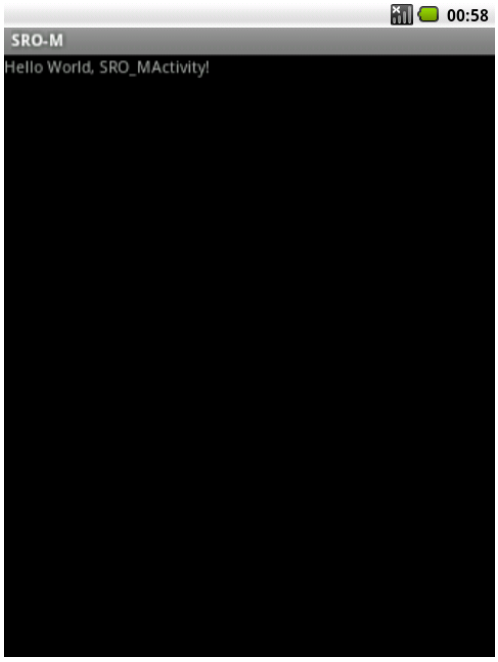


Figura. 5.5. Hello World, SRO_MActivity

Estructura del programa

Cuando se formó el proyecto, se creó una clase llamada *SRO_MActivity.java*, bajo el paquete *apps.tesis.omap*. Para acceder a la misma, dar doble clic sobre la clase, y se presentará lo siguiente:

ANEXOS, sección A.6.1. SRO_M/src/apps/tesis/omap/SRO_MActivity.java

```
package apps.tesis.omap;

import android.app.Activity;
import android.os.Bundle;

public class SRO_MActivity extends Activity {
    /** Called when the activity is first created. */

    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

            setContentView(R.layout.main);

     }
}
```

Android llama al método *onCreate()*, para que la actividad lo muestre. La llamada *setContentView()* llena los contenidos de la pantalla de la Activity con un widget de

Android. La presentación de la pantalla utilizada es el método Declarativo. *R.layout.main*, el cual es un recurso que direcciona al archivo *main.xml*, que se encuentra en *SRO_M/res/layout*. Este archivo se encarga de declarar la interface de usuario en código XML. Para formar el menú requerido debemos modificar el archivo *main.xml*. Para acceder al archivo, dar doble clic sobre él. Se mostrará la pantalla de la interface de usuario (Figura 5.6).

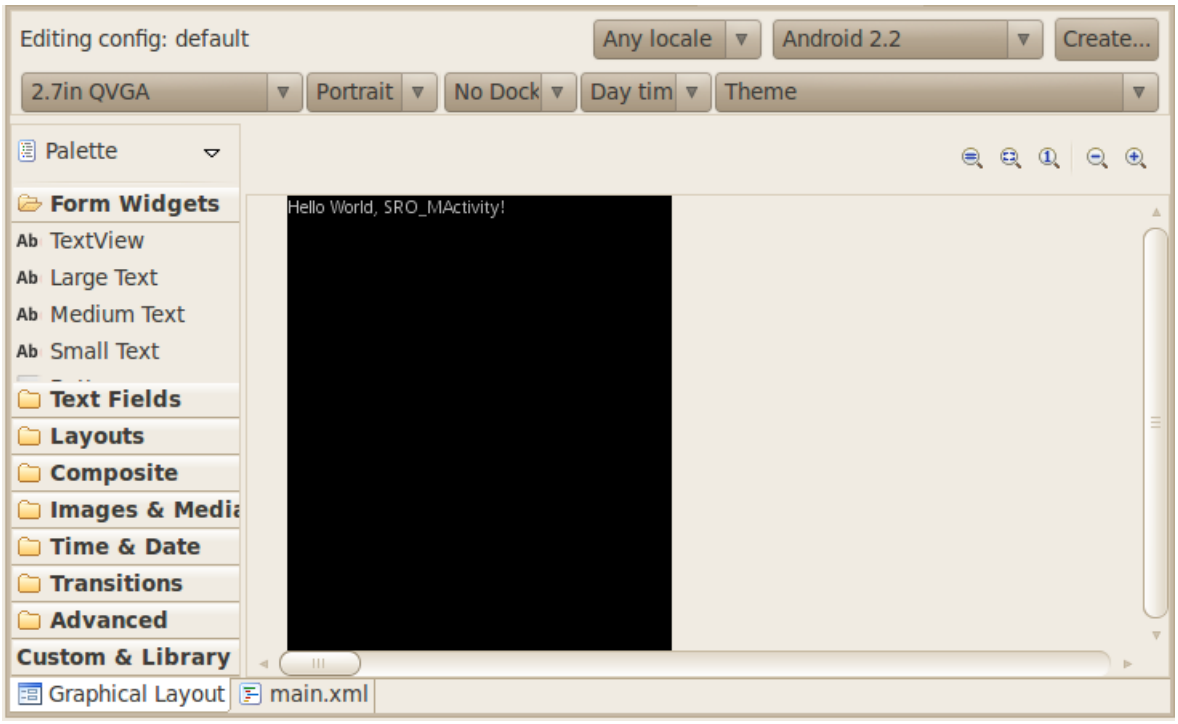


Figura. 5.6. Ventana de Recurso main.xml

Se presenta una pantalla, junto a un menú, en la parte inferior se muestran dos pestañas “Graphical Layout” y “main.xml”. Para el diseño de la interfaz de usuario por método Declarativo, Android presenta dos posibilidades, por código o de forma gráfica. Mediante la forma gráfica, simplemente se toman los widgets (Buttons, TextView, etc) del menú de la parte izquierda de la ventana y se los arrastra a la pantalla, esto genera código automáticamente.

El código puede ser examinado en la pestaña “main.xml”, el código que aparece es:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent">
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
</LinearLayout>
```

La primera línea de código debe estar presente en todos los archivos XML de Android. Indica al compilador que el archivo es de formato XML, en codificación UTF-8.

<LinearLayout

Es un contenedor para objetos y se encarga de ubicarlos sobre la pantalla como objeto rectangular. *LinearLayout* ubica a los objetos que contiene en una sola columna o fila, es el *Layout* más comúnmente utilizado. Existen muchos tipos de layouts [64]. Algunos parámetros son comunes para todos los *Layouts*:

xmlns:android="*http://schemas.android.com/apk/res/android*"

Define el namespace XML para android, solo se define una vez en todo el archivo.

android:layout_width="*fill_parent*",android:layout_height="*fill_parent*"

Define el espacio del objeto contenedor, el cual llena todo la pantalla. Pero puede ajustarse al contenido si se utiliza el parámetro *"wrap_content"*.

<TextView

TexView, se encuentra dentro de *LinearLayout*, simplemente sirve para presentar un texto. Permite asignar un texto a *TextView* utilizando una variable llamada "hello" esta puede ser encontrada en el archivo *res/values/strings.xml*.

android:text="*@string/hello*" />

El recurso asociado siempre posee el formato "@<tipo>/<nombre>", donde <tipo> es la clase de variable que se utilizará, generalmente esta se relaciona con el nombre del archivo y carpeta contenedora. Por otro lado <nombre> representa al calificativo que se le dé a la variable.

El archivo *strings.xml*, se presenta a continuación, es utilizado para declarar cadenas de texto que serán utilizadas en la aplicación, el calificativo de la variable es <nombre>. Para acceder a él hacer doble clic sobre *res/values/strings.xml*.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SRO_MActivity!</string>
    <string name="app_name">SRO-M</string>
</resources>
```

Se ha revisado la configuración de un programa muy simple, y a partir de aquí se pueden crear muchas variaciones.

Es importante resaltar que un archivo XML, no simplemente describe una interface de usuario, ya que además, puede representar atributos para fondos de pantalla, para botones, etc.

5.4.2. Menú Principal

El Menú Principal es una actividad que permitirán acceder a los programas SRO-M, esta es la primera pantalla que se verá cuando se seleccione el ícono del programa. El programa está basado en [75].

Sobre el programa creado, en el archivo *main.xml*, reemplazar el código, por el presentado en ANEXOS, Sección A.2.1. SRO_M/res/layout/main.xml) El código empleado utiliza una *TableLayout* que se encarga de distribuir botones y texto en una tabla. Se presentarán algunos errores, porque parte del código está direccionada a otros recursos *R* que aún no han sido declarados.

Es el caso del gráfico utilizado para presentar el título, se utilizó un widget *ImageView*. Como puede observarse, el gráfico asociado es de tipo “drawable”, no importa en cual carpeta *src/drawable-…* se encuentre, *R* lo reconoce como imagen.

La adición de recursos debe ser realizada sobre el archivo *res*. Para los gráficos, estos deben ser almacenados en los archivos *SRO_M/res/drawable-hdpi, SRO_M/res/drawable-mdpi* y *SRO_M/res/drawable-ldpi*. Representan los tipos de pantalla en cuanto a resolución.

Los gráficos utilizados para la aplicación se encuentran en el CD entregado con la presente, colocarlos en la carpeta *SRO_M/res/drawable-hdpi*. Para la creación de los

gráficos para los botones y título, se utilizó *Inkscape Vector Graphics[76]*, que es una herramienta gráfica libre para Ubuntu, de fácil uso y muy utilizado en la edición de imágenes. Para la creación de botones, se utilizó la herramienta Android Asset Studio[77], del proyecto "Google Project Hosting" para generación de íconos desde imágenes, que hizo posible utilizar las imágenes creadas en *Inkscape* y darles el formato de botón.

```
android:background="@drawable/background"
```

Existen recursos gráficos que no son interfaces de usuario, que pueden ser creados con XML. Este es el caso del fondo utilizado en la aplicación.

El archivo para el background debe ser generado, pulsar el botón derecho del mouse sobre la carpeta *SRO_M/res/drawable-hdpi* y seleccionar la opción New > File, se presentará una pantalla (Figura 5.7), donde indica "File Name:" colocar "Background.xml", luego seleccionar "Finish". No olvidar colocar la extensión ".xml", sino el archivo no tendrá ningún formato.

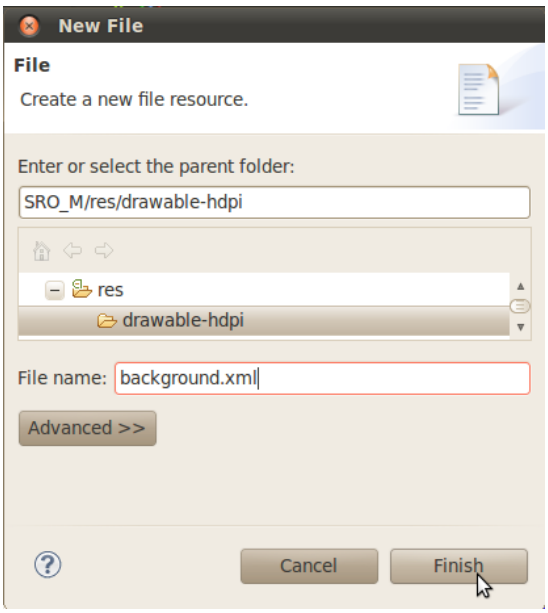


Figura. 5.7. New File XML para background

Esto generará un archivo en blanco. Colocar lo siguiente:

ANEXOS, sección A.4.1. SRO_M/res/drawable-hdpi/background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
      <gradient
```

```
            android:angle="90"
            android:startColor="@color/back_main1"
            android:centerColor="@color/back_main1"
            android:endColor="@color/back_main2"
            android:type="linear"/>
</shape>
```

"shape" facilita la creación de fondos de pantalla, se pueden formar con distintos colores, a diferentes ángulos. Algunos recursos no han sido creados todavía y esto se puede evidenciar por los errores que se presentan. Para esto, crear un archivo *SRO_M/res/values/color.xml* y definir colores como se describe a continuación..

ANEXOS, sección A.3.1. SRO_M/res/values/color.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <!-- Colores Menu -->
      <color name="back_main1">#2B70B5</color>
      <color name="back_main2">#FDFDFD</color>
      <color name="texto_main">#454B4E</color>
      <!-- Aqui pueden ir más Colores -->
</resources>
```

Para la utilización de variables gramaticales, se utilizará el recurso *String*, ubicado en *SRO_M/res/values*, incluye los caracteres a utilizar, cada uno relacionado con un nombre para poder ubicarlo. Dar doble clic en el archivo y copiar en él, lo siguiente.

ANEXOS, sección A.3.3. SRO_M/res/values/string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <!-- Strings Menu Principal -->
      <string name="app_name">SRO-M</string>
      <string name="label_t">Conversor</string>
      <string name="label_d">Cubo 3D</string>
      <string name="label_v">Video Player</string>
      <string name="label_a">Audio Rec</string>
      <string name="label_r">About</string>
</resources>
```

Los botones utilizados para la aplicación son imágenes, cuando se oprime el botón no se puede diferenciar si está seleccionado o no. Para darle formato al botón se crearon varios archivos XML, uno para cada botón, donde se iban relacionando una imagen para cuando se presione, no se presione o se enfoque un botón.

Las imágenes deben ser guardadas en *SRO_M/res/drawable-hdpi*, a continuación se muestra el código del primer botón. Para los demás, por favor referirse a ANEXOS, secciones A.4.4. SRO_M/res/drawable-hdpi/backbuttonconv.xml, A.4.5. SRO_M/res/drawable-hdpi/backbutton3d.xml, A.4.6. SRO_M/res/drawable-hdpi/backbuttonvideo.xml, A.4.7. SRO_M/res/drawable-hdpi/backbuttonaudio.xml y A.4.8. SRO_M/res/drawable-hdpi/backbuttonabout.xml.

ANEXOS, sección A.4.4. SRO-M/res/drawable-hdpi/backbuttonconv.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
      <!-- focused -->
    <item android:state_focused="true"
        android:state_pressed="false"
        android:drawable="@drawable/tempinkfoc" />
    <!-- presseed -->
    <item android:state_focused="true"
        android:state_pressed="true"
        android:drawable="@drawable/tempinkfoc" />
    <item android:state_focused="false"
        android:state_pressed="true"
              android:drawable="@drawable/tempinkfoc" />
      <!-- normal -->
    <item android:drawable="@drawable/tempink2200" />
     </selector>
```

En el archivo *main.xml*, se utiliza una sintaxis para referencia de ciertos widgets objetos, la sintaxis utilizadas es: "@+id/<nombre>", donde nombre es una variable que define un identificador, el cual puede ser utilizado para darle alguna acción al widget, por ejemplo, para que cuando se presione un botón este realice algo. Los identificadores definidos en el menú, serán recursos de *R*, estos son: R.id.boton_Conversor, R.id.boton_OpenGl, R.id.boton_Video, R.id.boton_Audio, R.id.boton_Resumen

Si en la ventana "Graphical Layout" de *main.xml*, se presenta el siguiente error: "error! Unable to resolve drawable…", pero en el código, no se muestra nada en rojo, entonces ignorarlo, al momento de correr el programa se podrán visualizar los botones.

Ahora es posible, correr el programa para visualizar el menú que se muestra en la Figura 5.8. Se puede observar que si se presionan los botones, no habrá ningún resultado.

Figura. 5.8. Menú Principal SRO-M

5.4.3. About

El primer programa SRO-M que se tratará es "About", es el último botón que aparece en el menú, su función es presentar un mensaje. Cuando el usuario desee salir de la aplicación solamente debe presionar la tecla "Back" (S4) del teclado y la ventana desaparecerá para volver al menú. Para este ejemplo se va a definir una nueva actividad que presente la ventana. Para esto, primero se debe crear su interface de usuario. Introducir al proyecto, el archivo *SRO_M/res/layout/sintesis.xml* y colocar el código presentado en ANEXOS, sección A.2.2 SRO_M/res/layout/sintesis.xml:

En el archivo *Sintesis.xml*, se ha definido un *Layout* tipo *ScrollView* que permitirá mover la pantalla para mostrar todo el contenido. Se debe definir además variables *String*, para lo cual se deben agregar variables a las ya existentes.

ANEXOS, sección A.3.3 SRO_M/res/values/string.xml . . .

```
<!-- Strings Menu Sintesis -->
<string name="titulo_resumen">About</string>
<string name="texto_resumen">Aplicaciones SRO-M ... </string>
```

Debe definirse la *Activity* que presente esta pantalla. Cada Programa de SRO-M será contenido en un paquete, dentro se hallarán las actividades que lo componen.

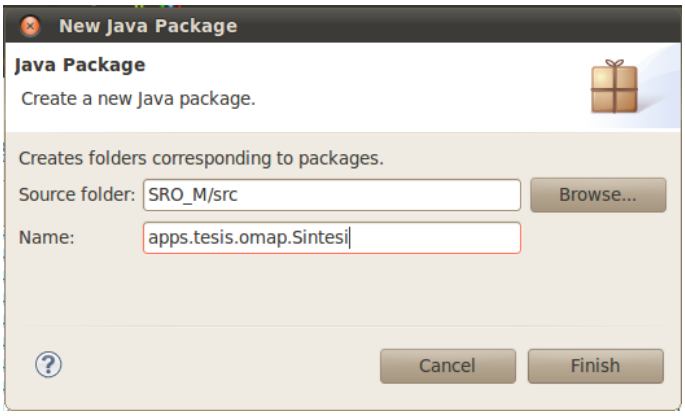


Figura. 5.9. New Java Package

Para definir un nuevo paquete, seleccionar *apps.tesis.omap*, presionar el botón derecho del mouse y seleccionar New > Package, se mostrará una ventana (Figura 5.9), en "Name" colocar "apps.tesis.omap.Sintesis", luego seleccionar "Finish" y se creará un paquete nuevo, dentro del cual se puede ubicar la nueva clase. Este procedimiento no es obligatorio, solamente se lo ha realizado para dar un mejor orden a los programas a realizar.

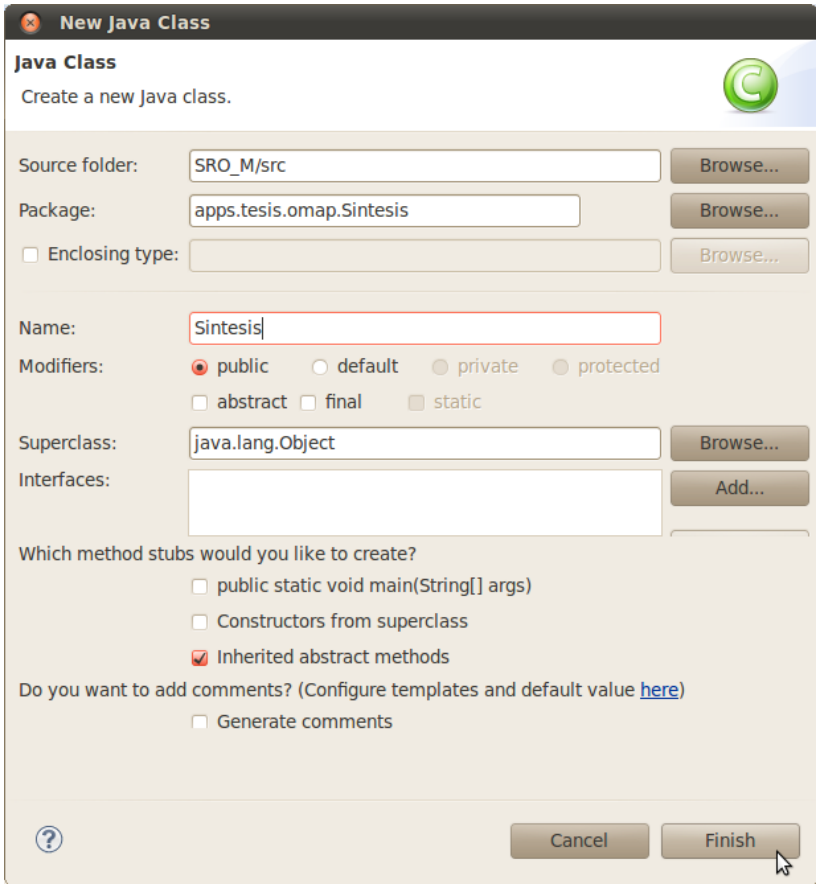


Figura. 5.10. New Java Class

Para crear una nueva clase, pulsar el botón derecho del mouse sobre *apps.tesis.omap.Sintesis*, seleccionar New > Class, se presentará una ventana (Figura 5.10). En "Name" colocar "Sintesis", luego seleccionar "Finish". Se generará la Clase, editarla con lo siguiente:

ANEXOS, sección A.6.2. SRO_M/ src/apps/tesis/omap/Sintesis/Sintesis.java

```
package apps.tesis.omap.Sintesis;

import android.app.Activity;
import android.os.Bundle;
import apps.tesis.omap.R;

public class Sintesis extends Activity{
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.resumen);
	}
}
```

import apps.tesis.omap.R;

Debe ser colocada en todas las clases que se encuentren en un paquete diferente a "apps.tesis.omap", para implortar otras librerías solamente es necesario pulsar las teclas crtl+shift+"o" y se agregarán automáticamente.

En la clase *SRO_M/src/apps/tesis/omap/SRO_MActivity.java*, en el método *onCreate()*, agregar código para llamar al método *findViewById()* para buscar un widget, dado su ID de recurso. Tambien agregar el método *setOnClicListener()*, para informar al sistema cual objeto ha sido accionado en la pantalla.

ANEXOS, sección A.6.1. SRO_M/ src/apps/tesis/omap/SRO-MActivity.java

```
...
@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		// Clic listeners para los botones
		View botonHola = findViewById(R.id.boton_Conversor);
		botonHola.setOnClicListener(this);
		View botonOpenGl = findViewById(R.id.boton_OpenGl);
		botonOpenGl.setOnClicListener(this);
		View botonVideo = findViewById(R.id.boton_Video);
		botonVideo.setOnClicListener(this);
		View botonAudio = findViewById(R.id.boton_Audio);
		botonAudio.setOnClicListener(this);
```

```
            View botonResumen = findViewById(R.id.boton_Sintesis);
            botonResumen.setOnClicListener(this);
      }
```

Hacer lo mismo para todos los botones, nótese que las constantes fueron creadas automáticamente por el plug-in en *R.java*.

El método *setOnClicListener()* necesita implementar la interface java *OnClicListener*. Así que para hacer que la clase *SRO_MActivity.java* implemente la interface se crea el método *onClic()*. Para iniciar una *Activity* en Android primero se debe crear una instancia de la clase *Intent*. Las *Activities* serán llamadas dependiendo de cual botón ha sido seleccionado, en este caso será llamada la *Activity* de la Clase *Sintesis.java*.

ANEXOS, sección A.6.1. SRO_M/src/apps/tesis/omap/SRO_MActivity.java

```
import apps.tesis.omap.Sintesis.Sintesis;
...
public class SRO_MActivity extends Activity implements OnClicListener {
      ...
      @Override
      public void onClic(View arg0) {
            // TODO Auto-generated method stub
            switch (arg0.getId()) {
            // Resumen de programas
            case R.id.boton_Sintesis:
                  Intent i = new Intent(this, Sintesis.class);
                  startActivity(i);
                  break;
            // Mas botones pueden ir aqui…
            }
      }
}
```

import apps.tesis.omap.Sintesis.Sintesis;

Una vez finalizado es necesario importar la clase *Sintesis.java*, ya que esta pertenece a un paquete diferente que *SRO_MActivity.java*,.Además se debe importar las demás librerías, pulsar las teclas crtl+shift+“o” y se agregarán automáticamente.

Si se corre el programa y se presiona en el botón “About”, se producirá un error, esto se debe por la falata de definición de la *Activty* en el *AndroidManifest.xml*. Abrir el archivo, seleccionar la pestaña *AndroidManifest.xml* ubicada en la parte inferior, y agregar la *Activity*, luego de la anterior *Activity* declarada “.SRO_MActivity”.

Además, aquí se puede dar estilo a la ventana agregandole un tema. El prefijo "@android", indica que el recurso es definido por Android, no por el programa.

ANEXOS, sección A.5.1. SRO_M/AndroidManifest.xml

```
<!-- Activity de clase Sintesis -->
<activity android:name=".Sintesis.Sintesis"
          android:label="@string/label_r"
          android:theme="@android:style/Theme.Dialog">
      <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.SAMPLE_CODE" />
      </intent-filter>
</activity>
```

Si se corre el programa, esta vez funcionará y se mostrará el mensaje (Figura 5.11.).

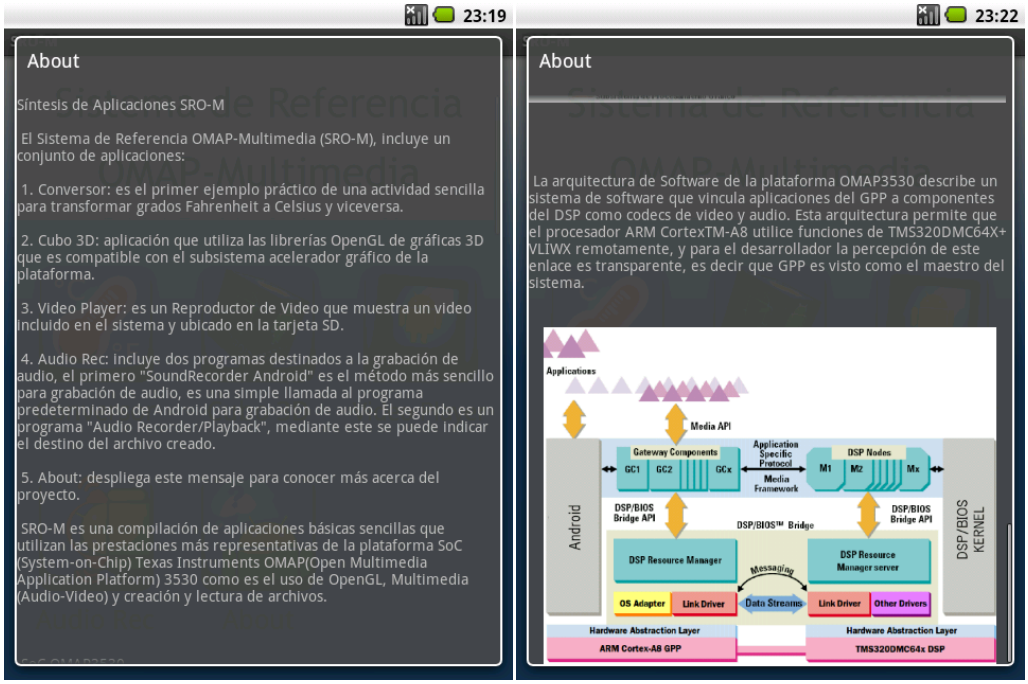


Figura. 5.11. Aplicación About

5.4.4. Conversor de Unidades

El siguiente programa es un ejemplo práctico de una actividad sencilla, se trata de un Conversor de Unidades de Temperatura, para la conversión de grados Fahrenheit a Celsius y viceversa. El programa está basado en el ejemplo de [78]. Para acceder al mismo se debe presionar el primer botón del menú. Cuando el usuario desee salir de la aplicación solamente debe presionar la tecla "Back" (S4) del teclado y la ventana desaparecerá para volver al menú.

El programa consiste en un *EditText* en el que se ingresan números, y *RadioButtons* para seleccionar a que unidad de temperatura se desea transformar. Se le han agregado *Listeners* para que cuando se seleccione un *RadioButton* o cuando se escriba en el *EditTex*, se realice la conversión y esta sea presentada en un *TextView*.

El ejemplo será definido sobre una nueva actividad que presente la ventana. Para esto, primero se debe crear su interface de usuario. Crear el archivo *SRO_M/res/layout/conversor.xml*, cuyo código puede ser encontrado en ANEXOS, sección A.2.3 SRO_M/res/layout/conversor.xml.

El interface de usuario está compuesto por un *LinearLayout*, que contine un *TableLayout* para la ubicación de un *EditText* con ID “EditTemp” y un *TextView* con ID “ViewGrados”, Tambien contiene otro *LinearLayout* para la ubicación del *RadioGroup*, que contiene dos *RadioButton* con ID “RadioC” y “RadioF”. Al relacionar los recursos a un ID, estos pueden ser creados por el plug-in en archivo *R*. Al crear el archivo *conversor.xml*, se muestran varios errores, esto es porque hay que definir alguno recursos faltantes.

```
android:background="@drawable/backgroundtemp"
```

Se aprecia que se direcciona a un fondo, este código contiene un error ya que hay que definir este archivo. Este no es una imagen rasterizada, es un archivo que define un fondo de color. Crear un archivo en *SRO-M/res/drawable-hdpi/backgroundtemp.xml* y escribir lo siguiente:

ANEXOS, sección A.4.2 SRO_M/res/drawable-hdpi/backgroundtemp.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
  xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
            android:angle="90"
            android:startColor="@color/back_conv1"
            android:centerColor="@color/back_conv2"
            android:endColor="@color/back_conv3"
            android:type="linear"/>
</shape>
```

Este archivo representa un gradiente con colores que deben ser definidos en el archivo *SRO-M/res/values/color.xml*, para esto agregar lo siguiente al archivo.

ANEXOS, sección A.3.1 SRO_M/res/values/color.xml

```
<!-- Colores Conversor -->
<color name="back_conv1">#DCDDA4</color>
<color name="back_conv2">#558063</color>
<color name="back_conv3">#002F3A</color>
<color name="texto_conv">#FFFFBA</color>
```

Además es necesario agregar más variables de texto al archivo *string.xml*

ANEXOS, sección A.3.3 SRO_M/res/values/string.xml

```
<!-- String Menu Conversor -->
<string name="label_1"> Conversor de Unidades</string>
<string name="app_conv">Conversor \nFahrenheit - Celsius</string>
<string name="celsius">Celsius</string>
<string name="fahrenheit">Fahrenheit</string>
<string name="c">  °C</string>
<string name="f">  °F</string>
<string name="unid">Unidades  </string>
<string name="edit_temp">0</string>
<string name="resp_temp">"-17.777779"</string>
```

Una vez realizadas las modificaciones, es necesario crear la *Activity*, para manejar la interface. Crear un paquete nuevo: *SRO_M/src/apps/tesis/omap/Conversor*, y dentro de este crear la clase *ConversorTemperatura.java*, y colocar lo siguiente:

ANEXOS, sección A.6.3.

SRO_M/src/apps/tesis/omap/Conversor/ConversorTemperatura.java

```
package apps.tesis.omap.Conversor;

import apps.tesis.omap.R;

public class ConversorTemperatura extends Activity {
	/** Called when the activity is first created. */
	float Valor;
	EditText text;
	TextView respuesta;
	TextView deGrad;
	TextView aGrad;
	RadioGroup grados;
	RadioButton celsiusButton;
	RadioButton fahrenheitButton;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.conversor);

		text = (EditText) findViewById(R.id.EditTemp);
```

```
            respuesta = (TextView) findViewById(R.id.ViewGrados);
            celsiusButton = (RadioButton) findViewById(R.id.RadioC);
            fahrenheitButton = (RadioButton) findViewById(R.id.RadioF);
            grados = (RadioGroup) findViewById(R.id.radioGroup1);
            deGrad = (TextView) findViewById(R.id.deGrados);
            aGrad = (TextView) findViewById(R.id.aGrados);
}
```

Primero es necesario declarar las variables a utilizar que se van a asociar con el recurso *R*, los *TextView* "deGrad" y "aGrad", son utilizados para informar al usuario en qué sentido va la conversión. En el método *onCreate()*, agregar código para llamar al método *findViewById()* para buscar un *view*, dado su ID de recurso y relacionarlo a las variables declaradas.

Se debe importar las demás librerías, pulsar las teclas crtl+shift+"o" y se agregarán automáticamente.

Agregar las siguientes funciones para realizar el cálculo de conversión de Fahrenheit a Celcius y viceversa. La función *Conversor()* será llamada para el cálculo.

ANEXOS, sección A.6.3.

SRO_M/src/apps/tesis/omap/Conversor/ConversorTemperatura.java

```
// Conversor de Temperatura
public void Conversor() {
      if (text.getText().length() == 0) {
            respuesta.setText("0");
            text.setText("0");
            return;
      }
      // Transformar a flotante numero ingresado
      Valor = Float.parseFloat(text.getText().toString());
      // Calculos
      if (celsiusButton.isChecked()) {
            respuesta.setText(String.valueOf(FahrenheitACelsius(Valor)));
            deGrad.setText(R.string.f);
            aGrad.setText(R.string.c);
      } else {
            respuesta.setText(String.valueOf(CelsiusAFahrenheit(Valor)));
            deGrad.setText(R.string.c);
            aGrad.setText(R.string.f);
      }
}
// A celsius
private float FahrenheitACelsius(float fahrenheit) {
      return ((fahrenheit - 32) * 5 / 9);
}
// A fahrenheit
private float CelsiusAFahrenheit(float celsius) {
```

```
        return ((celsius * 9) / 5) + 32;
}
```

A continuación del código, en *OnCreate()*, agregar *Listeners* que intervengan cada vez que exista alguna acción en el *RadioGroup* "Grados", o en *EditText* "text". Cuando el sistema sea notificado por el *Listener*, se llamará a la función *Conversor( )* para realizar el cálculo matemático de lo que se encuentre en el *EditText* "Respuesta".

ANEXOS, sección A.6.3.

SRO_M/src/apps/tesis/omap/Conversor/ConversorTemperatura.java

```
...
@Override
public void onCreate(Bundle savedInstanceState) {
...
      // Listener RadioButton
      grados.setOnCheckedChangeListener(new OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup arg0, int arg1) {
                  // TODO Auto-generated method stub
                  Conversor();
            }
      });
      // Listener EditText
      text.addTextChangedListener(new TextWatcher() {
            @Override
            public void afterTextChanged(Editable arg0) {
                  // TODO Auto-generated method stub
                  Conversor();
            }
            @Override
            public void beforeTextChanged(CharSequence arg0, int arg1,
                        int arg2, int arg3) {
                  // TODO Auto-generated method stub
            }
            @Override
            public void onTextChanged(CharSequence arg0, int arg1,
                        int arg2,int arg3) {
                  // TODO Auto-generated method stub
            }
      });
}
```

Una vez creada la *Activity* del Conversor de Temperatura, es necesario relacionarla con el botón correspondiente en el menú principal. Es por eso que hay agregar código en el archivo *SRO_MActivity.java*. En el método *onClic()*, se debe relacionar la selección del botón con la *Activity* utilizando la clase *Intent*.

ANEXOS, sección A.6.1. SRO_M/src/apps/tesis/omap/SRO-MActivity.java

```
import apps.tesis.omap.Conversor.ConversorTemperatura;

...// Aplicación Conversor
       case R.id.boton_Conversor:
              Intent m = new Intent(this, ConversorTemperatura.class);
              startActivity(m);
              break;
```

No olvidar importar la Clase *ConversorTemperatura.java*, para que el paquete sea reconocido. Ahora solo queda agregar la *Activty* al archivo *AndroidManifest.xml*, para que pueda ser activada. Agregar el siguiente código.

ANEXOS, sección A.5.1. SRO_M/AndroidManifest.xml

```
<!-- Activity de clase Conversor -->
<activity android:name=".Conversor.ConversorTemperatura"
          android:label="@string/label_1">
      <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.SAMPLE_CODE" />
      </intent-filter>
</activity>
```

Correr el programa, seleccionar el botón “Conversor” y se presentará la ventana “Conversor de Unidades”, Su funcionamiento se muestra en la Figura 5.12

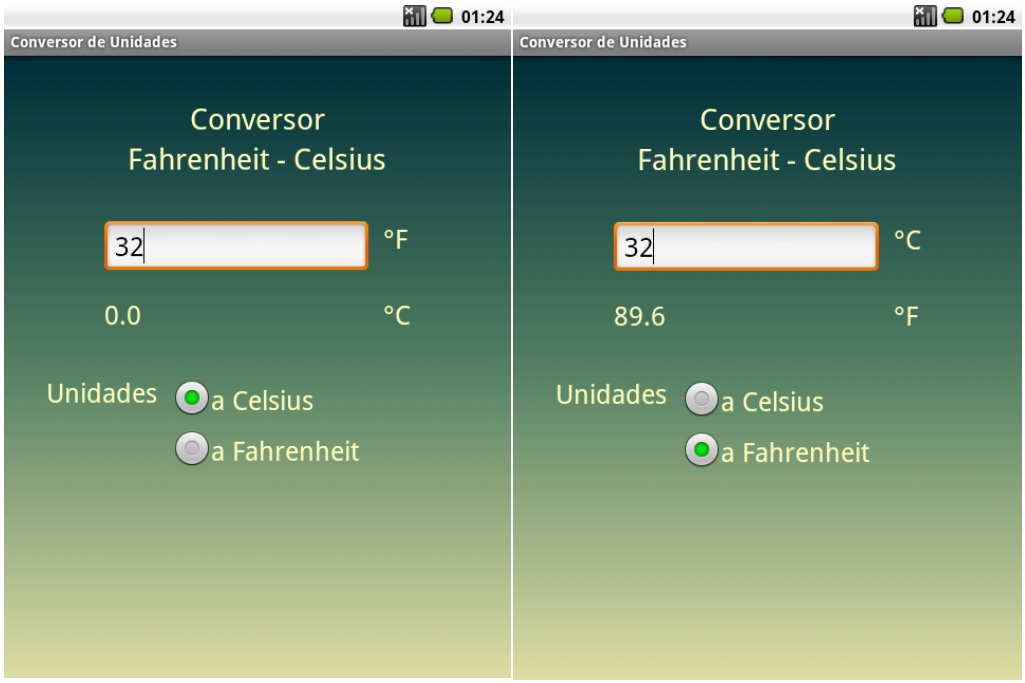


Figura. 5.12. Aplicación Conversor de Unidades

5.4.5. Cubo 3D (Graphics Rendering)

Android provee una librería de gráficos en tres dimensiones basados en el estándar OpenGL ES, implementando conceptos como rendero de imágenes, iluminación, aplicación de texturas. El procesamiento gráfico en la plataforma EVM OMAP3530 con su procesador grafico POWERVR SGX, soporta OpenGL ES. Por esta razón se decidió presentar un programa que utilice gráficos 3D.

La siguiente es una aplicación que presenta el procedimiento para la creación de un cubo 3D giratorio parcialmente transparente. El programa se basa en un ejemplo del libro [75]. Su metodología de diseño consiste en crear un cubo, iluminarlo, hacerlo girar, añadirle textura, y hacerlo transparente. Para el mejor entendimiento de las fases para su construcción, se ha creado un menú que muestra este procedimiento.

Para acceder al mismo se debe presionar el botón “Cubo”. Se presentará una lista, al seleccionar la opción se verá en cubo correspondiente, para regresar a menú presionar la tecla “Back” (S4). Cuando el usuario desee salir de la aplicación, ubicarse en el menú “3D Graphics Rendering (OpenGL)” y presionar la tecla “Back” (S4) del teclado y la ventana desaparecerá para volver al menú principal.

Menú de Selección

Crear el paquete *apps.tesis.omap.OpenGL*. Antes de la creación de la *Activity* para la implementación de OpenGL en el programa, es preciso crear un menú que permita visualizar el proceso de creación del cubo, permitiendo el acceso a las diferentes partes del procedimiento. Para esto es necesario definir una nueva clase dentro del paquete OpenGL: *mainOpenGL.java*, como muestra el código a continuación. La definición del menú contempla una interfaz de usuario combinando ambos métodos.

ANEXOS, sección A.6.4. SRO_M/src/apps/tesis/omap/OpenGL/mainOpenGL.java

```
package apps.tesis.omap.OpenGL;

import apps.tesis.omap.R;

public class mainOpenGl extends ListActivity {
  Context context = this;
```

```
  public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);

       // View de un Menu tipo Lista
       String[] lista = getResources().getStringArray(R.array.ArrayCubo);
       setListAdapter(new ArrayAdapter<String>(this,
                   R.layout.maingl, lista));
       ListView lv = getListView();
       lv.setTextFilterEnabled(true);
  }
}
```

Se debe importar las demás librerías, pulsar las teclas crtl+shift+“o” y se agregarán automáticamente. La clase *mainOpenGL.java*, contiene una *Activity* tipo *ListActivity*. La idea principal de una *ActivityList* es la creación de un *layout* que permita representar una lista con información, en este caso se mostrará una lista del proceso de creación del Cubo. Relaciona dos recursos que son: un arreglo *string* y un *layout* para el fondo de la ventana. Estos recursos son archivos en XML. Primero se creará el arreglo, para esto es necesario crear un nuevo archivo *array.xml*, que será ubicado en la carpeta *SRO_M/res/values/*. El archivo será un recurso con identificación “ArrayCubo”.

ANEXOS, sección A.3.2. SRO_M/res/values/array.xml

```
<resources>
      <string-array name="ArrayCubo">
            <item>@string/uno</item>
            <item>@string/dos</item>
            <item>@string/tres</item>
            <item>@string/cuatro</item>
            <item>@string/cinco</item>
      </string-array>
</resources>
```

Agregar las variables *String* relacionadas al archivo *SRO_M/res/values/string.xml*

ANEXOS, sección A.3.3. SRO_M/res/values/string.xml

```
      <!-- Strings Menu OpenGL -->
      <string name="label_2"> 3D Graphics Rendering (OpenGL)</string>
      <string name="uno">  Cubo sin sombra</string>
      <string name="dos">  Cubo Iluminado</string>
      <string name="tres">  Cubo Rotatorio</string>
      <string name="cuatro">  Cubo Texturado</string>
      <string name="cinco">  Cubo Transparente</string>
```

Proseguir con la creación del *Layout*, crear el archivo *maingl.xml* y ubicarlo en la carpeta *SRO_M/res/layout/*.

ANEXOS, sección A.2.4 SRO_M/res/layout/maingl.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:gravity="center|left"
       android:padding="15dp"
       android:textSize="28sp"
       android:background="@color/back_maingl1">
</TextView>
```

Agregar, el color para el fondo del *Layout*, agregar el siguiente código en el archivo *SRO_M/res/values/color.xml.*

ANEXOS, sección A.3.1. SRO_M/res/values/color.xml

```
       <!-- Colores Cubo3D -->
       <color name="back_maingl1">#374F67</color>
```

Una vez creada la *Activity* del Menú del Cubo, es necesario relacionarla con el botón correspondiente en el menú principal. Es por eso que hay agregar código en el archivo *SRO_MActivity.java*. En el método *onClic()*, se debe relacionar la selección del botón con la *Activity* utilizando la clase *Intent*.

ANEXOS, sección A.6.1. SRO_M/src/apps/tesis/omap/SRO-MActivity.java

```
import apps.tesis.omap.OpenGL.mainOpenGL;

...
// Aplicación OpenGL
             case R.id.boton_OpenGl:
                    Intent j = new Intent(this, mainOpenGl.class);
                    startActivity(j);
                    break;
```

No olvidar importar la Clase *mainOpenGL.java*, para que el paquete sea reconocido, además no olvidar incluir la *Activity*, en *AndroidManifest.xml*, y hacer que se muestre como una ventana de diálogo aplicando el tema.

ANEXOS, sección A.5.1. SRO_M/AndroidManifest.xml

```
<!-- Activity de clase OpenGL -->
  <activity android:name=".OpenGL.mainOpenGl"
```

```
            android:label="@string/label_2"
            android:theme="@android:style/Theme.Dialog">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.SAMPLE_CODE" />
        </intent-filter>
    </activity>
```

Si se corre el programa en este momento, se presentará un menú como en la Figura 5.13.

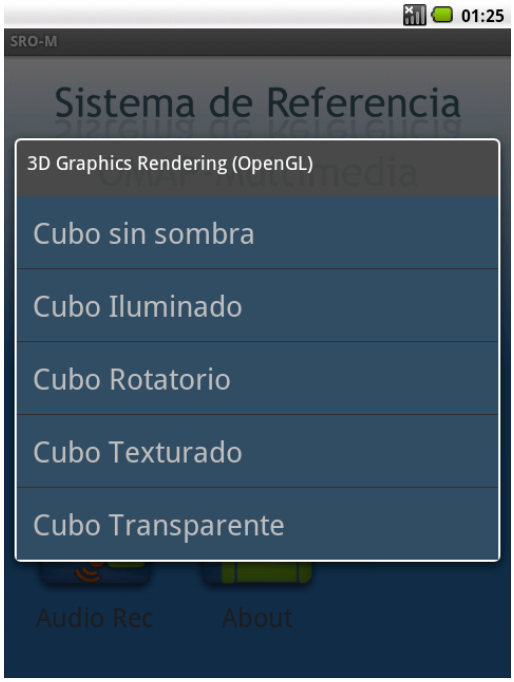


Figura. 5.13. Aplicación Cubo 3D: Menú

Ahora se debe incluir un *Listener*, para que cuando una de las filas del arreglo sea accionada, se presente la imagen del cubo solicitada. Agregar el siguiente código en método *onCreate()*, de la clase *mainOpenGL.java*.

ANEXOS, sección A.6.4. SRO_M/src/apps/tesis/omap/OpenGL/mainOpenGL.java

```
public void onCreate(Bundle savedInstanceState) {
// . . .
// Listener Lista
      lv.setOnItemClicListener(new OnItemClicListener() {
            public void onItemClic(AdapterView<?> parent, View view,
                        int posicion, long id) {
                  // Cuando se selecciona se muestra el título
                  Toast.makeText(getApplicationContext(),
                        ((TextView) view).getText(),
                        Toast.LENGTH_SHORT).show();
                  try {
```

```
                    // Envia a Intent OpenGL cual item fue seleccionado
                    Intent intent = new Intent(context, OpenGL.class);
                    Bundle bundle = new Bundle();
                    bundle.putInt("opcion", posicion);
                    intent.putExtras(bundle);
                    startActivity(intent);
                } catch (Exception e) {
                    System.out.println("Posición: " + posicion);
                }
            }
        });
```

Cuando se implementa el *Listener* sobre el arreglo, se puede obtener la posición de la fila seleccionada empezando desde cero, y se la almacena en "posición". Esta posición puede ser enviada hacia otros *Activities*, por medio de *Intent*.

```
Intent intent = new Intent(context, OpenGL.class);
```

Un *Intent* será utilizado para moverse a la nueva ventana que presentará el cubo.

```
intent.putExtras(bundle);
```

Además, podrá enviarle la posición de la fila seleccionada con la función "putExtras".

Toast.*makeText*(getApplicationContext(),((TextView) view).getText(),
Toast.*LENGTH_SHORT*).show();

"Toast" se emplea para mostrar mensajes, por un periodo de tiempo, este se mantendrá aunque se cambie de *Activity*. Aquí se mostrará cual fue la opción seleccionada.

Las funciones antes descritas, presentarán una nueva ventana para la aplicación de OpenGL. Bajo el paquete *apps.tesis.omap.OpenGL*, crear la clase *OpenGL.java* para contener la *Activity* del proyecto. Colocar lo siguiente:

ANEXOS, sección A.6.5. SRO_M/src/apps/tesis/omap/OpenGL/OpenGL.java

```
package apps.tesis.omap.OpenGL;

import android.app.Activity;
import android.os.Bundle;

public class OpenGL extends Activity {
    GLView view;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Obtiene extras del Menu Lista
        Bundle extras = getIntent().getExtras();
```

```
            int estado = extras.getInt("opcion");
            view = new GLView(this, estado);
            setContentView(view);
      }
      @Override
      protected void onPause() {
            super.onPause();
            view.onPause();
      }
      @Override
      protected void onResume() {
            super.onResume();
            view.onResume();
      }
}
```

View será referenciada a una nueva clase *GLView*, la cual será definida más adelante.

```
Bundle extras = getIntent().getExtras();
int estado = extras.getInt("opcion");
```

La variable "estado", esta le indicará cuál posición fue seleccionada. Para obtener una variable de otra *Activity*, utilizar estas funciones. Es necesario obtener "extras" en el método *onCreate()*, ya que es cuando se inicia la *Activity*.

Se sobrescribirán los métodos *onPause( )* y *OnResume( )* configurados por defecto, para que llamen a los métodos del mismo nombre que en *View*.

No olvidar incluir la *Activity*, en *AndroidManifest.xml*, para que se muestre.

ANEXOS, sección A.5.1. SRO_M/AndroidManifest.xml

```
<activity android:name=".OpenGL.OpenGL"
          android:label="@string/label_2">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.SAMPLE_CODE" />
    </intent-filter>
</activity>
```

La clase que definirá la pantalla presentada debe ser especificada, crear la clase *GLview.java*, bajo el paquete *apps.tesis.omap.OpenGL*, y colocar el siguiente código.

ANEXOS, sección A.6.6. SRO_M/src/apps/tesis/omap/OpenGL/GLView.java

```
package apps.tesis.omap.OpenGL;
```

```
import android.content.Context;
import android.opengl.GLSurfaceView;

class GLView extends GLSurfaceView {
	private final GLRenderer renderer;
	int selec;

	GLView(Context context, int i) {
		super(context);
		renderer = new GLRenderer(context, i);
		setRenderer(renderer);
	}
}
```

GLSurfaceView es una clase que simplifica el uso de OpenGL en Android, provee la conexión para adherir OpenGL ES a la pantalla y al ciclo de vida de la *Activity*. Se encarga de seleccionar los formatos buffer, pixel y frame apropiados, y maneja renderizado por separado, habilitando una animación más continua. *GLView* debe ser extendida a un *GLSurfaceView* y definir un *Renderer* como *View*. La clase *GLRenderer()* también debe aceptar la variable de ubicación de la fila de la lista antes desplegada.

Preparación de Pantalla

Cuando se utiliza OpenGL, se deben definir clases de renderizado diferentes para cada dibujo a representar. Cada clase es responsable por inicializar y dibujar sobre la pantalla completa.

Crear la clase *GLRenderer.java* dentro del paquete *apps.tesis.omap.OpenGL* y colocar lo siguiente:

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
package apps.tesis.omap.OpenGL;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

import android.content.Context;
import android.opengl.GLSurfaceView;
import android.opengl.GLU;
import apps.tesis.omap.R;

class GLRenderer implements GLSurfaceView.Renderer {
	private final Context context;
	private int opcion;
```

```
        GLRenderer(Context context, int i) {
                this.context = context;
                opcion = i;
        }

        public void onSurfaceCreated(GL10 gl, EGLConfig config) {
                // ...
        }

        public void onSurfaceChanged(GL10 gl, int width, int height) {
                // ...
        }

        public void onDrawFrame(GL10 gl) {
                // ...
        }
}
```

Al inicializar la clase *GLRenderer*.java, se tomó en cuenta incluir un valor entero como parámetro "opcion", este indica cual fila fue seleccionada. La clase *GLRenderer* implementa la interface *GLSurfaceView.Render*, la cual contiene tres métodos.

El método *onSurfaceCreated* es llamado cuando la superficie de OpenGL es creada o recreada. Agregar lo siguiente:

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
        public void onSurfaceCreated(GL10 gl, EGLConfig config) {
                // Cubo
                // Opciones OpenGL
                gl.glEnable(GL10.GL_DEPTH_TEST);
                gl.glDepthFunc(GL10.GL_LEQUAL);
                gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        }
```

Considerar que el desempeño del sistema depende de cuantas opciones se encuentren habilitadas.

gl.glEnable(GL10.*GL_DEPTH_TEST*);

gl.glEnable permite habilitar opciones para OpenGL, las cuales pueden activarse con *glEnable( )* o desactivarse *glDisable( )*. La opción *GL_DEPTH_TEST* permite ejecutar comparaciones de profundidad, y actualizar el buffer de profundidad.

El método *onSurfaceChanged( )* es llamado una vez que la superficie fue creada y es llamada nuevamente siempre que el tamaño de la superficie cambie. Agregar el código siguiente:

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
public void onSurfaceChanged(GL10 gl, int width, int height) {
	// Define el campo de vision
	gl.glViewport(0, 0, width, height);
	gl.glMatrixMode(GL10.GL_PROJECTION);
	gl.glLoadIdentity();
	float ratio = (float) width / height;
	GLU.gluPerspective(gl, 45.0f, ratio, 1, 100f);
}
```

Aquí se ha configurado, el plano de visualización y se ha establecido algunas opciones OpenGL.

GLU.*gluPerspective*(gl, 45.0f, ratio, 1, 100f);

Los dos últimos argumentos de la función representan la distancia hacia del ojo hacia los planos de dibujo, uno más cercano y uno más lejano, juntos proponen la dimensión de visualización total de un elemento dibujado en tres dimensiones.

Renderización se refiere al proceso de generar una imagen 3D, a partir de una en 2D. Una imagen 3D puede ser formada por estructuras polinomiales, comportamiento de luces, textura y materiales, animaciones, etc [75]. En el programa, el método *onDrawFrame()* es llamado una y otra vez para dibujar una imagen renderizada creada por *GLSurfaceView*. Agregar lo siguiente.

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
public void onDrawFrame(GL10 gl) {
	// Pantalla en negro
	gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
	// Posicion del modelo
	gl.glMatrixMode(GL10.GL_MODELVIEW);
	gl.glLoadIdentity();
	gl.glTranslatef(0, 0, -3.0f);
}
```

Para comenzar se establece la pantalla de color negro, se inicializan los buffers de color y profundidad, siempre ambos. Además se conFigura la posición inicial para el resto de los comandos de dibujo.

Correr el programa y seleccionar cualquiera de las opciones, se mostrará una pantalla negra (Figura 5.14). Aquí será la base donde se presentarán las imágenes 3D. Notar que además se muestra durante un periodo corto de tiempo, el mensaje de la fila seleccionada. El método dibujará la pantalla una y otra vez, esto tendrá más sentido cuando se genere una Figura animada sobre la pantalla.

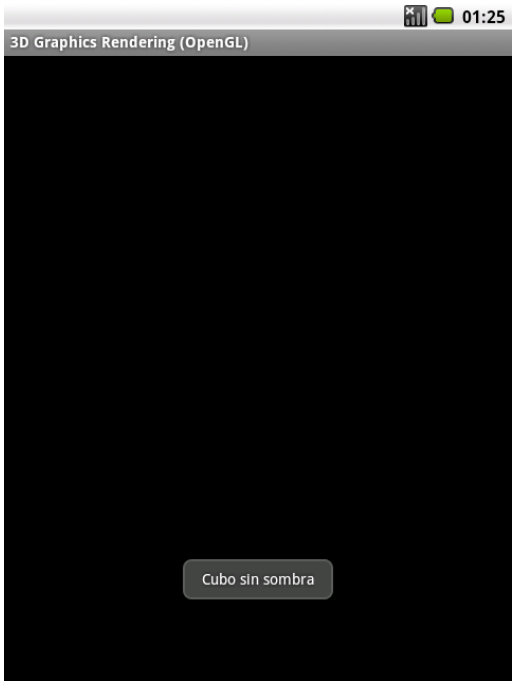


Figura. 5.14. Aplicación Cubo 3D: Pantalla en negro inicial

Construcción de un Modelo

Es hora de dibujar algo sobre la pantalla. Dependiendo de la complejidad de los objetos a dibujar, generalmente se utilizarán herramientas de diseño gráfico y se importará el diseño hacia el programa. En este programa se definirá un modelo simple, un cubo. Para esto necesitamos crear una nueva clase llamada *GLCube* y colocar el código:

ANEXOS, sección A.6.8. SRO_M/src/apps/tesis/omap/OpenGL/GLCube.java

```
package apps.tesis.omap.OpenGL;


class GLCube {
	private final IntBuffer mVertexBuffer;

	public GLCube() {
		int one = 65536;
		int half = one / 2;
		int vertices[] = {
			// Vista Frontal
			-half, -half, half, half, -half, half,
			-half, half, half, half, half, half,
			// Vista Posterior
			-half, -half, -half, -half, half, -half,
			half, -half, -half, half, half, -half,
			// Vista Lateral Izquierda
			-half, -half, half, -half, half, half,
			-half, -half, -half, -half, half, -half,
			// Vista Lateral Derecha
			half, -half, -half, half, half, -half,
			half, -half, half, half, half, half,
			// Vista Superior
			-half, half, half, half, half, half,
			-half, half, -half, half, half, -half,
			// Vista Inferior
			-half, -half, half, -half, -half, -half,
			half, -half, half, half, -half, -half, };
		ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
		vbb.order(ByteOrder.nativeOrder());
		mVertexBuffer = vbb.asIntBuffer();
		mVertexBuffer.put(vertices);
		mVertexBuffer.position(0);

	public void draw(GL10 gl) {
		gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);

		gl.glColor4f(1, 1, 1, 1);
		gl.glNormal3f(0, 0, 1);
		gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 4);
		gl.glNormal3f(0, 0, -1);
		gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 4, 4);

		gl.glColor4f(1, 1, 1, 1);
		gl.glNormal3f(-1, 0, 0);
		gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 8, 4);
		gl.glNormal3f(1, 0, 0);
		gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 12, 4);

		gl.glColor4f(1, 1, 1, 1);
		gl.glNormal3f(0, 1, 0);
		gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 16, 4);
		gl.glNormal3f(0, -1, 0);
		gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 20, 4);
	}

}
```

Se debe importar librerías, pulsar las teclas crtl+shift+“o” y se agregarán automáticamente. El arreglo *vertices[]*, define las esquinas del cubo. Cada cara del cubo es un cuadrado, que se compone de dos triángulos llamados *triangle strips*, que son dibujados en la función *draw()*.

```
gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 4);
```

En el modo *triangle strips*, se definen dos puntos iniciales y los puntos definidos consecutivamente definen triángulos con los dos puntos anteriores.

```
gl.glNormal3f(0, 0, 1);
```

Cada punto tiene tres coordenadas en el espacio (*x*,*y*,*z*). Los ejes *x* e *y* tienen su dirección hacia la derecha o hacia arriba respectivamente. El eje *z*, apunta hacia afuera de la pantalla.

El método *draw( )*, utiliza el buffer *vertex* creado por el constructor y dibuja seis diferentes pares de triángulos, seis por los seis lados del cubo. Ahora podemos llamarla en la clase *GLRenderer.java*.

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
// Clase que dibuja el cubo
private final GLCube cube = new GLCube();
public void onDrawFrame(GL10 gl) {
      // ...
      // Dibujar el cubo
      cube.draw(gl);
}
```

Si se corre el programa en este momento y se selecciona cualquiera de las opciones, se presentará un cuadrado, que es el cubo dibujado de frente como muestra la Figura 5.15.

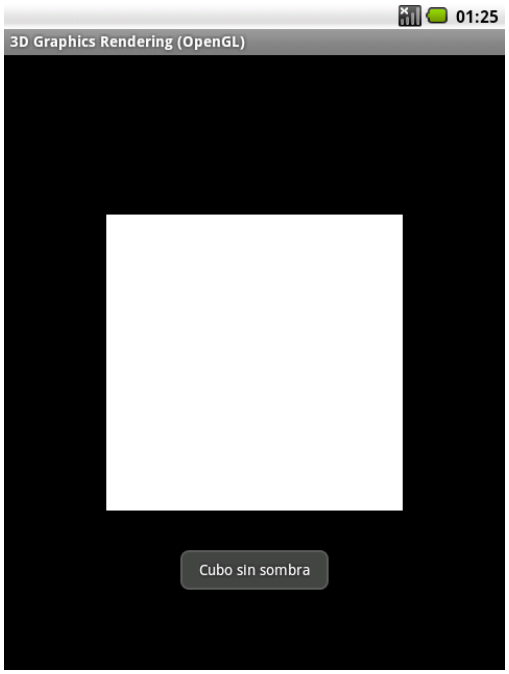


Figura. 5.15. Aplicación Cubo 3D: Cubo sin sombra

Iluminación del Modelo

La iluminación es utilizada para crear realismo a los modelos dibujados. OpenGL permite definir ocho fuentes de luz en la escena. Para la iluminación hace falta dos elementos, la luz y un algo para iluminar. Las librerías de gráficos 3D soportan tres tipos de iluminación[75]:

- Ambient: es un brillo general que la luz contribuye a la escena entera. Incluso los ubicados a lo lejos.
- Diffuse: luz suave directiva, como la que se obtiene de una luz fluorecente.
- Specular: luz brillante, que viene desde dos fuentes de luz. En combinación con materiales brillantes, le añade realismo a la escena.

Si se define una sola fuente de luz, esta puede ofrecer estos tres tipos de iluminación. Estos valores se incluyen en una ecuación de iluminación que determina el color y brillo de cada pixel de la pantalla. La iluminación debe ser definida en el método *GLRenderer.onSurfaceCreated( ).* Agregar lo siguiente.

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
       // ...
       // Luz
       if (opcion >= 1) {
           // Definir iluminacion
           float lightAmbient[] = new float[] { 0.2f, 0.2f, 0.2f, 1 }
           float lightDiffuse[] = new float[] { 1, 1, 1, 1 };
           float[] lightPos = new float[] { 1, 1, 1, 1 };
           gl.glEnable(GL10.GL_LIGHTING);
           gl.glEnable(GL10.GL_LIGHT0);
           gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient, 0);
           gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);
           gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 2);
      } }
```

La iluminación del cubo es la segunda opción del menú principal, por lo que se incluye la condición *if( )*, para separar este procedimiento.

```
float lightDiffuse[] = new float[] { 1, 1, 1, 1 };
```

En el código se define una fuente de luz en la posición *(1,1,1)*. Esta es una luz omnidireccional que tiene un componente difuso y obscuro. En este ejemplo no se utilizó iluminación espectacular.

A continuación se configurarán los materiales que conforman el cubo. La luz se refleja de forma diferente en distintos materiales, como metal, plástico o papel. Para simular como la luz reacciona sobre algún material, agregar código en el método *GLRenderer.onSurfaceCreated( )*.

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
       // ...
       // Luz
       if (opcion >= 1) {
              // De que esta hecho el cubo
              float matAmbient[] = new float[] { 1, 1, 1, 1 };
              float matDiffuse[] = new float[] { 1, 1, 1, 1 };
              gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT,
                           matAmbient, 0);
              gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE,
                           matDiffuse, 0);

      }
}
```

Ahora se puede correr el programa, al seleccionar la segunda opción del menú, se desplegará un cuadrado iluminado, como se muestra la Figura 5.16. Puede observarse que la parte superior derecha del cubo es ligeramente más brillante.

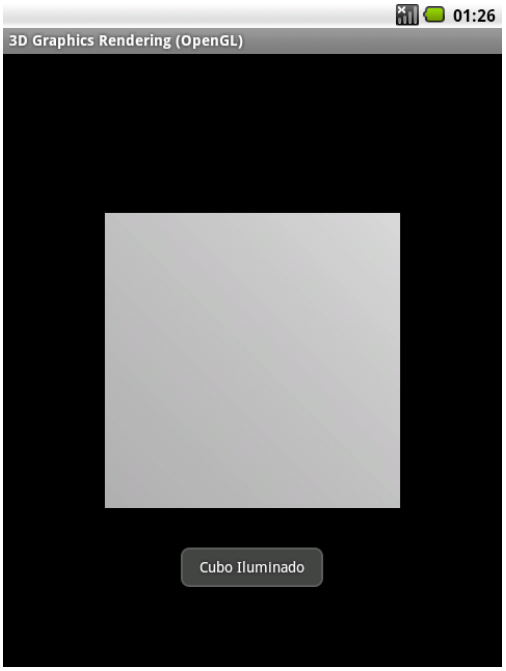


Figura. 5.16. Aplicación Cubo 3D: Cubo Iluminado

Movimiento

La siguiente parte del proceso es darle movilidad al cubo, de esta manera se podrá observar, que efectivamente se trata de un cubo. Para realizar esto, es necesario agregar código en los métodos *onSurfaceCreated( )* y *onDrawFrame( )* en la clase *GLRenderer.java*. Ubicar la condición *if( )*, para el caso de la opción "Cubo Iuminado".

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
// Movimiento del cubo
long startTime;
long fpsStartTime;
long numFrames;
// ...
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
      // ...
      if (opcion >= 1) {
            // ...
            // Movimiento
            if (opcion >= 2) {
```

```
                    startTime = System.currentTimeMillis();
                    fpsStartTime = startTime;
                    numFrames = 0;
                }
            }
        }

        public void onDrawFrame(GL10 gl) {
            // ...
            // Giro
            if (opcion >= 2) {
                // Set rotation angle based on the time
                long elapsed = System.currentTimeMillis() - startTime;
                gl.glRotatef(elapsed * (30f / 1000f), 0, 1, 0);
                gl.glRotatef(elapsed * (15f / 1000f), 1, 0, 0);
            }
            // Dibujar el cubo
            cube.draw(gl);
        }
```

El código generado produce un giro sobre el cubo, cada cierto tiempo en un lazo. Específicamente, gira 30 grados sobre su eje *x*, y 15 grados sobre su eje *y*. Al correr el programa y seleccionar, la opción tercera, se mostrará la siguiente figura:

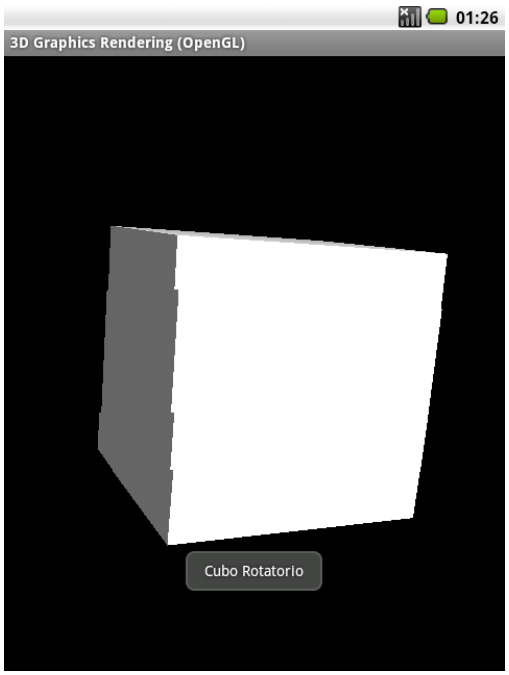


Figura. 5.17. Aplicación Cubo 3D: Cubo Rotatorio

Textura

Para que una imagen se vea más real, se le agrega textura, que no es más que colocar una imagen sobre la superficie del objeto. En este caso, debemos agregar unas coordinadas de textura, en la clase SRO_M/src/apps/tesis/omap/OpenGL/GLCube.java

ANEXOS, sección A.6.8. SRO_M/src/apps/tesis/omap/OpenGL/GLCube.java

```
class GLCube {
   private final IntBuffer mTextureBuffer;
   // ...
    public GLCube() {
    // ...
      int texCoords[] = {
                    // Vista Frontal
                    0, one, one, one, 0, 0, one, 0,
                    // Vista Posterior
                    one, one, one, 0, 0, one, 0, 0,
                    // Vista Lateral Izquierda
                    one, one, one, 0, 0, one, 0, 0,
                    // Vista Lateral Derecha
                    one, one, one, 0, 0, one, 0, 0,
                    // Vista Superior
                    one, 0, 0, 0, one, one, 0, one,
                    // Vista Inferior
                    0, 0, 0, one, one, 0, one, one, };

      ByteBuffer tbb = ByteBuffer.allocateDirect(texCoords.length * 4);
      tbb.order(ByteOrder.nativeOrder());
      mTextureBuffer = tbb.asIntBuffer();
      mTextureBuffer.put(texCoords);
       mTextureBuffer.position(0);
   }

   static void loadTexture(GL10 gl, Context context, int resource) {
       Bitmap bmp = BitmapFactory.decodeResource(context.getResources(),
                    resource);
       GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bmp, 0);
       gl.glTexParameterx(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MIN_FILTER,
                    GL10.GL_LINEAR);
       gl.glTexParameterx(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MAG_FILTER,
                    GL10.GL_LINEAR);
       bmp.recycle();
    }
}
```

Una vez declaradas, OpenGL debe utilizar las coordenadas, esto es realizado en el método *draw( )*. Agregar lo siguiente:

ANEXOS, sección A.6.8. SRO_M/src/apps/tesis/omap/OpenGL/GLCube.java

```
public void draw(GL10 gl) {
       gl.glEnable(GL10.GL_TEXTURE_2D);
       gl.glTexCoordPointer(2, GL10.GL_FIXED, 0, mTextureBuffer);
```

```
        // ...
}
```

Finalmente se debe llamar al método *loadTexture( )* en *GLRenderer.java*. Ubicar la condición *if( )*, para el caso de la opción "Cubo Texturado". Agregar el siguiente código al método *onSurfaceCreated( )*:

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
       // ...
       if (opcion >= 1) {
              // ...
              if (opcion >= 2) {
                 // Cubo con Textura
                 if (opcion >= 3) {
                     // Habilitar texturas
                     gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
                     gl.glEnable(GL10.GL_TEXTURE_2D);
                     // Carga textura
                     GLCube.loadTexture(gl, context, R.drawable.logoespe1);
                 }
}}}
```

El código utilizado, habilita texturas y coordinadas para aplicar la textura y luego llama al método *loadTexture( )*, le envía el contexto de la *Activiy* y la identificación del recurso de la imagen a utilizar como textura.

R.drawable.*logoespe1*

La imagen PNG de 128x128 px (Figura 5.18), que debe ser agregada al archivo *SRO_M/res/drawable-mdpi*, la imagen se encuentra dentro del CD entregado con la presente, pero puede utilizarse cualquier otro que cumpla con el tamaño, solamente hacer coincidir el nombre. El tamaño de la imagen puede ser diferente, solamente debe ser de forma cuadrada para que ajuste de mejor manera sobre el cuadrado.

Figura. 5.18. Imagen PNG 128x128 px

Cuando se corra el programa y se seleccione la opción cuatro “Cubo Texturado”, se mostrará lo siguiente (Figura 5.19)

Figura. 5.19. Aplicación Cubo 3D: Cubo Texturado

Transparencias

El siguiente código hace posible que el cubo texturado se muestre parcialmente transparente.

ANEXOS, sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
      // ...
      if (opcion >= 1) {
            // ...
            if (opcion >= 2) {
                  // Cubo con Textura
                  if (opcion >= 3) {
                     // Cubo Transparente
                     if (opcion >= 4) {
                         gl.glDisable(GL10.GL_DEPTH_TEST);
                         gl.glEnable(GL10.GL_BLEND);
                         gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE);
                     }
                  }
            }
       }}
```

Para acceder a este punto de la configuración se debe agregar un nuevo *if( )*, cuando la opción sea mayor igual que cuatro, que es la ubicación de la última fila de la matriz de menú, se presentará el cubo texturado parcialmente transparente.

```
gl.glDisable(GL10.GL_DEPTH_TEST);
```

Al desactivar la opción *GL_DEPTH_TEST* se permite la observación de la parte del objeto obscurecido en la profundad, así como la parte que se encuentre en primer plano.

```
gl.glEnable(GL10.GL_BLEND);
```

Esta opción permite que la opacidad de los objetos se base en una transparencia. El efecto que tiene es que las caras posteriores del cubo aparecerán a través de las caras frontales.

Cuando se corra el programa y se seleccione la última opción del menú, se mostrará lo siguiente (Figura 5.20)

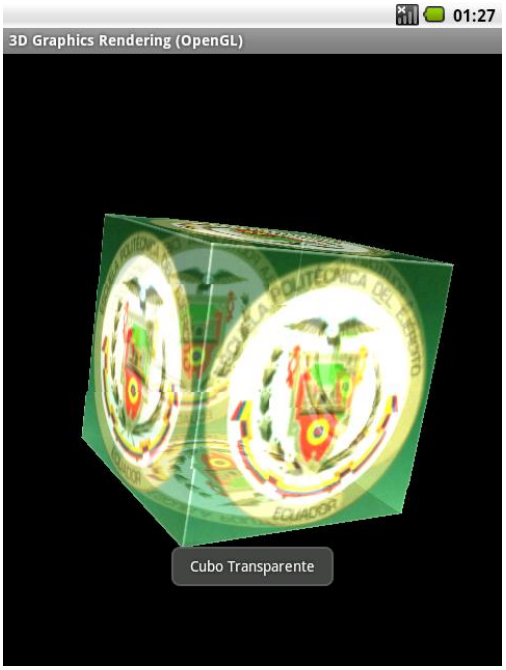


Figura. 5.20. Aplicación Cubo 3D: Cubo Transparente

5.4.6. Video Player

Los programas que se tratarán a partir de ahora, serán destinados a trabajar con audio y vedio. El siguiente programa se trata de un reproductor de video. El programa está

basado en los ejemplos de [79] y [75]. Para acceder al mismo se debe presionar el tercer botón del menú "Video Player". Cuando el usuario desee salir de la aplicación solamente debe presionar la tecla "Back" (S4) del teclado y la ventana desaparecerá para volver al menú.

Existe una manera sencilla de reproducir videos a través de la clase *VideoView*, que utiliza un *view* o una superficie sobre la cual se proyectará el video. Android, además ofrece la forma de controlar la reproducción que puede ser incluida en el *view*.

Primero es necesario definir un layout, crear el archivo *video.xml*, en la carpeta *SRO_M/res/layout* y escribir lo siguiente:

ANEXOS, sección A.2.5 SRO_M/res/layout/video.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent">
       <VideoView
              android:id="@+id/video"
              android:layout_height="wrap_content"
              android:layout_width="wrap_content"
              android:layout_gravity="center" />
</FrameLayout>
```

A continuación crear el paquete *apps.tesis.omap. Video*, crear la clase *RepVideo.java*, y escribir lo siguiente:

ANEXOS, sección A.6.9. SRO_M/src/apps/tesis/omap/Video/RepVideo.java

```
package apps.tesis.omap.Video;

import apps.tesis.omap.R;

public class RepVideo extends Activity {
       private String videoPath;

       @Override
       public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.video);

              // Inicializar Video View
              VideoView video = (VideoView) findViewById(R.id.video);

              // Origen achivo a mostrar
              videoPath = "/sdcard/Video/big_buck_bunny_cif_mpeg4_aac.mp4";
```

```
            video.setVideoPath(videoPath);
        }
}
```

Se debe importar las demás librerías, pulsar las teclas crtl+shift+“o” y se agregarán automáticamente.

video.setVideoPath(videoPath);

El método *setVideoPath( )* abre el archivo y cambia su tamaño en la pantalla, manteniendo la relación del formato original.

Se puede añadir un controlador que permita un acceso a cualquier momento del video, este será construido en java y añadido a la pantalla, este es un ejemplo de cómo crear una interfaz grafica por método Procedimental (Sección 5.1.2).

ANEXOS, sección A.6.9. SRO_M/src/apps/tesis/omap/Video/RepVideo.java

```
// ...
// Agregar Media Controller
MediaController mediacontroller = new MediaController(this);
mediacontroller.setAnchorView(video);
video.setMediaController(mediacontroller);
if (video.canSeekForward())
      video.seekTo(video.getDuration() / 2);
```

Luego de declarar el controlador, es momento de iniciar el video. También se puede agregar un “Toast” para que muestre la dirección del archivo. Bajo el código escrito anteriormente, agregar lo siguiente.

ANEXOS, sección A.6.9. SRO_M/src/apps/tesis/omap/Video/RepVideo.java

```
// ...
// Inicio de Video
video.start();

// Indica mensaje de origen de video
Toast.makeText(this, "Video en Reproducción:\n" + videoPath,
            Toast.LENGTH_LONG).show();
```

Asegurarse que la tarjeta SD contenga el video al cual se está direccionando. Si no se encuentra, se lo puede agregar de dos maneras, colocando el archivo manualmente en la tarjeta SD, o utilizando comandos.

```
videoPath = "/sdcard/Video/big_buck_bunny_cif_mpeg4_aac.mp4";
```

El archivo se encuentra dentro de una carpeta llamada “Audio”. Cuando se utilice comandos, la carpeta puede ser creada utilizando la conexión serial con Minicom:

```
target#  mkdir /sdcard/Video
```

O se puede utilizar la herramienta ADB:

```
host$ adb shell
#mkdir /sdcard/Video
```

Para agregar el video, se puede utilizar la herramienta ADB de Android. El video utilizado es parte de las imágenes pre-compiladas, de las que se hablo en el Capítulo 4. En el Terminal ubicarse sobre la carpeta contenedora del video y escribir:

```
host$ adb push big_buck_bunny_cif_mpeg4_aac.mp4 /sdcard/Video/
```

Antes de correr el programa, no olvidar relacionar la *Activity* con el botón correspondiente en el menú principal. Además se debe importar la Clase *Video.RepVideo.java*, para que el paquete sea reconocido:

ANEXOS, sección A.6.1. SRO_M/src/apps/tesis/omap/SRO-MActivity.java

```
import apps.tesis.omap.Video.RepVideo;
// ...
// Aplicación Reproductor de Video
            case R.id.boton_Video:
                  Intent l = new Intent(this, RepVideo.class);
                  startActivity(l);
                  break;
```

Direccionar la *Activity* en el archivo en *AndroidManifest.xml*

ANEXOS, sección A.5.1. SRO_M/AndroidManifest.xml

```
<!-- Activity de clase Reproductor de Video -->
<activity android:name=".Video.RepVideo"
          android:label="@string/label_3">
       <intent-filter>
          <action android:name="android.intent.action.MAIN" />
```

```
        <category android:name="android.intent.category.SAMPLE_CODE" />
      </intent-filter>
</activity>
```

Añadir la variable *string* correspondiente a la ventana que presenta el video.

ANEXOS, sección A.3.3. SRO_M/res/values/string.xml

```
    <!-- Strings Video -->
         <string name="label_3"> Video Player</string>
```

Al correr el programa, se mostrará una ventana con el video y un mensaje que indicará el path seleccionado (Figura 5.21). El Control se mostrará cada vez de se toque la pantalla, y permitirá manipular el avance del video.

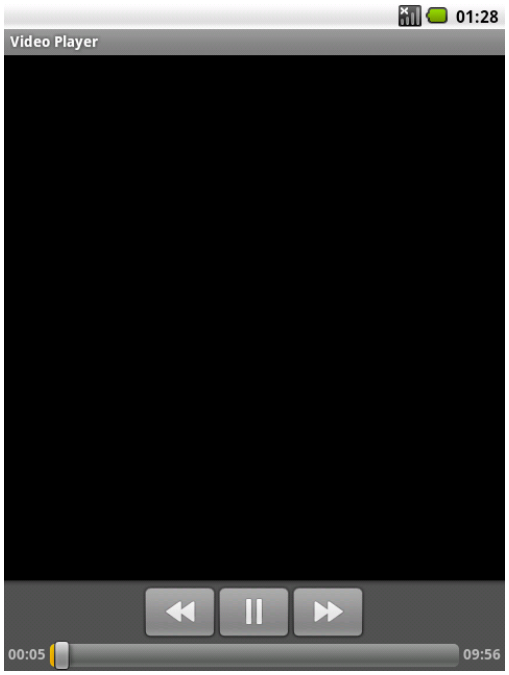


Figura. 5.21. Aplicación Video Player

5.4.7. Audio Rec

A continuación se describen dos clases de programas destinados a la grabación y reproducción de audio: SoundRecorder Audio y Audio Recorder/Player. En el menú, seleccionar la opción “Audio Rec”, se desplegarán dos opciones de aplicaciones para la grabación de audio. Los programas están basados en los ejemplos de [79], en donde se

describen dos maneras para grabación de audio que son utilizando el programa predefinido por Android y utilizando la clase *MediaRecorder*.

Para que el programa funcione, verificar que se encuentre conectados los parlantes y el elemento Emisor de Audio (MP3 Player), en los puertos de salida y entrada de audio respectivamente.

Menú Audio Rec

Primero se realizará el menú de selección, para esto se utilizará un diálogo de alerta que será desplegado. Sobre la clase principal *SRO_M/src/apps/tesis/omap/SRO-MActivity.java*, se debe añadir la siguiente función:

ANEXOS, sección A.6.1. SRO_M/src/apps/tesis/omap/SRO-MActivity.java

```
// Función Selección de Aplicación Audio
private void SelecAudio() {
   new AlertDialog.Builder(this).setTitle(R.string.label_a).setItems(
      R.array.ArrayAudio, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialoginterface, int i) {
                  Audio(i);
            }
      }).show(); }
```

El código presentado, se basa en la construcción de una ventana que se aparece sobre el menú principal, el contenido se encuentra definido por un arreglo de opciones.Se define una nueva variable “i”, la cual contiene la fila seleccionada.

R.array.*ArrayAudio*

El arreglo utilizado debe ser definido en el archivo construido anteriormente, el cual se ubicaba en *SRO_M/res/values/array.xml*, es necesario agregar el siguiente código:

ANEXOS, sección A.3.2. SRO_M/res/values/array.xml

```
<string-array name="ArrayAudio">
      <item>@string/sound_android</item>
      <item>@string/sound_tesis</item>
</string-array>
```

Además es necesario definir las variables *string* utilizadas, además agregaremos todas la variables tipo *string* utilizadas por los dos programas a tratar.

ANEXOS, sección A.3.3. SRO_M/res/values/string.xml

```
    <!-- Strings Menu Audio Principal-->
    <string name="label_4"> Audio Recorder/Playback</string>
    <string name="sound_android">SoundRecorder Android</string>
    <string name="sound_tesis">Audio Recorder/Player</string>
    <!-- Strings Menu Nombre -->
    <string name="titulo_nombre">Ingrese un nombre y ubicación al archivo:
</string>
    <string name="ext_nombre">.3gp</string>
    <string name="boton_nombre">Guardar</string>
    <!-- Strings Menu Audio -->
    <string name="play_inicio">00:00</string>
    <string name="estado_inicial">...</string>
```

Ahora es necesario asociar el botón “Audio Rec” del menú principal, a la función del menú de audio, en la función *onClick* agregar:

ANEXOS, sección A.6.1. SRO_M/src/apps/tesis/omap/SRO-MActivity.java

```
            // Aplicación Grabador de Audio
            case R.id.boton_Audio:
                  SelecAudio();
                  break;
```

Si se corre la aplicación y seleccionamos el botón, se presentará el siguiente menú:

Figura. 5.22. Aplicación Audio Rec: Menú

Cuando una aplicación utiliza alguna forma de grabación, se deben definir los permisos necesarios, en este caso para grabación de audio. Sobre el archivo *AndroidManifest.xml*., antes de la definición de la aplicación se debe colocar:

ANEXOS, sección A.5.1. SRO_M/AndroidManifest.xml

```
<uses-permission android:name="android.permission.RECORD_AUDIO"></uses-
permission>
```

Sound Recorder

"SoundRecorder Android" es el método más sencillo para grabación de audio, es una simple llamada al programa predeterminado de Android para grabación de audio, realizado a través de un *Intent*.

Utilizando la variable "i", que será obtenida al seleccionar las opciones del menú, se construye una función llamada *Audio( )*, donde se seleccionan los *Intents* a utilizar. Si i = 0, se utilizará grabador de audio predeterminado para dispositivos Android.

ANEXOS, sección A.6.1. SRO_M/ src/apps/tesis/omap/SRO-MActivity.java

```
	// Seleccion ejecución programas de Audio
private void Audio(int i) {
	if (i == 0) {
		// Ejecuta aplicación SoundRecorder propia de Android
		Intent intent = new Intent(
				MediaStore.Audio.Media.RECORD_SOUND_ACTION);
		startActivityForResult(intent, 1);
	} else {
		// Aplicación para Grabación de Audio ...
	}
}
```

Cuando se corra el programa y se seleccione la opción *SoundRecorder*, se presentará la siguiente pantalla (Figura 5.23).

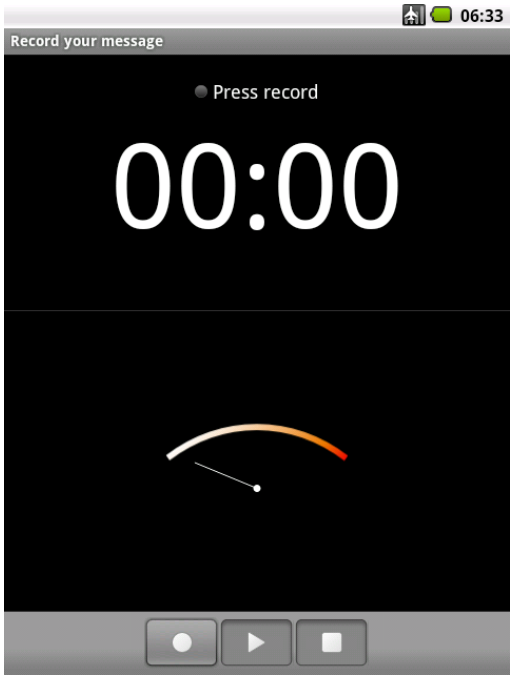


Figura. 5.23. Aplicación Audio Rec: Sound Recorder Android

Presionar el botón grabar, se mostrará la Figura 5.24, donde se indicará el tiempo de grabación y la intensidad de sonido.

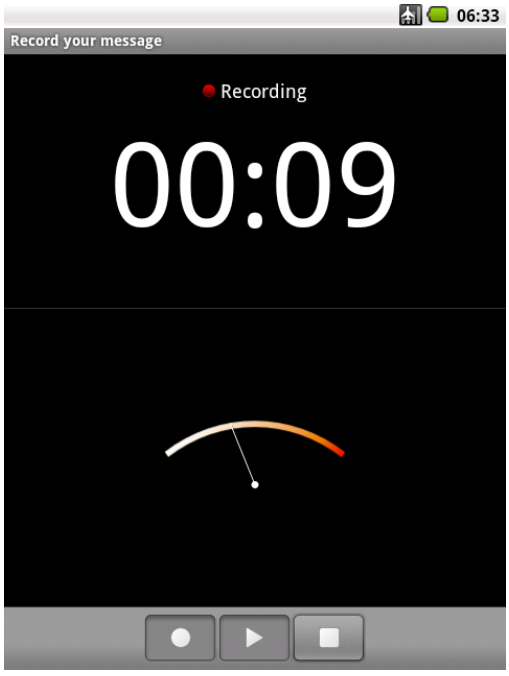


Figura. 5.24. Aplicación Audio Rec: Sound Recorder Android - Recording

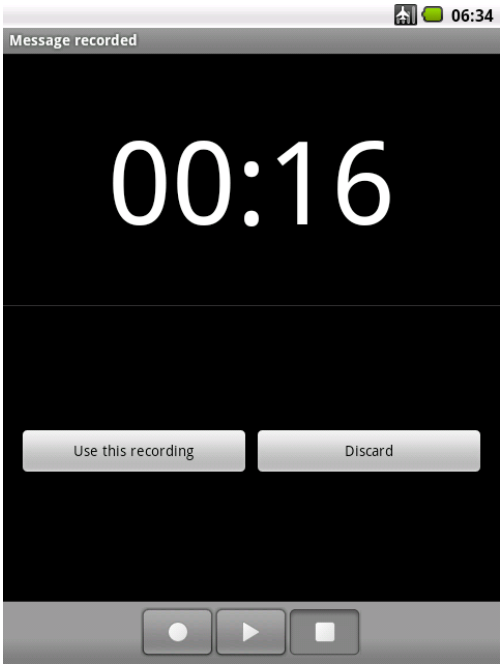


Figura. 5.25. Aplicación Audio Rec: Sound Recorder Android - Recorded

Al presionar Stop, se mostrará la Figura 5.25. Esta ventana indica el tiempo total de grabación, y además muestra dos botones con las opciones: "Use this recording" y "Discard". Al seleccionar la primera opción, se utilizará la grabación y se almacenará automáticamente en la tarjeta SD, con el nombre de la fecha de su creación. Al seleccionar la segunda opción, el archivo será descartado.

Audio Recorder/Player

La segunda opción en el menú de grabación lleva a una aplicación para grabación y reproducción de audio en el que se tiene más control sobre las definiciones y ubicación de la grabación. La aplicación consiste en dos clases *Nombre.java* y *AudioRec.java*. *Nombre.java* se trata de una *Activity* para el ingreso del nombre del archivo de audio a crear. Y *AudioRec.java* se trata del programa en sí, estas clases intercambian información gracias a la utilización de *Intents*.

Primero debemos colocar el *intent* que llamará a la clase Nombre. Para esto, en la función *Audio( )*, de la clase *SRO-MActivity.java*, escribir:

ANEXOS, sección A.6.1. SRO_M/ src/apps/tesis/omap/SRO-MActivity.java

```
import apps.tesis.omap.Audio.Nombre;
...
```

```
	// Seleccion ejecución programas de Audio
	private void Audio(int i) {
...
		} else {
			// Aplicación para Grabación de Audio
			Intent b = new Intent(this, Nombre.class);
			startActivity(b);
		}
	}
```

No olvidar importar la clase *Nombre.java*, la cual será llamada para iniciar la aplicación. Notar que la clase será definida bajo un paquete llamado "Audio", el cual debe ser creado. Luego de creado el paquete, definir la clase *Nombre.java*, bajo el paquete *apps.tesis.omap.Audio*, la cual debe contener lo siguiente:

ANEXOS, sección A.6.10. SRO_M/src/apps/tesis/omap/Audio/Nombre.java

```
package apps.tesis.omap.Audio;

import apps.tesis.omap.R;

public class Nombre extends Activity {
	private EditText text;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.nombre);
		text = (EditText) findViewById(R.id.nombreAudio);

		// Poner el cursor al principio del texto editable
		text.setSelection(1);
		text.extendSelection(0);
		text.setSelection(0);
	}
}
```

Se debe importar las demás librerías, pulsar las teclas crtl+shift+"o" y se agregarán automáticamente. La interface gráfica debe ser definida, crear el archivo nombre.xml en la carpeta layout, para el código favor referirse a ANEXOS, sección A.2.6. SRO_M/res/layout/nombre.xml.

El código del archivo *nombre.xml* define un editor de texto, donde se escribirá el nombre del archivo y un botón para guardar el nombre del archivo. Es necesario definir los colores, y el background, que en este caso es una imagen. La imagen del background puede ser encontrada en el CD, o puede ser cualquier otra. Debe guardarse en *drawable-mdpi*,

bajo el nombre de *background_audio_nomb.png*. Además incluir los colores restantes, agregando lo siguiente en el archivo *color.xml*.

ANEXOS, sección A.3.1. SRO_M/res/values/color.xml

```
<!-- Colores Audio -->
<color name="back_audio1">#F8EBE8</color>
<color name="back_audio2">#012529</color>
<color name="texto_audio1">#394D4A</color>
<color name="texto_audio2">#050505</color>
<color name="texto_audio3">#67C1F4</color>
```

Incluir la *Activity* en el archivo *AndroidManifest.xml* y colocarle el tema tipo diálogo.

ANEXOS, sección A.5.1. SRO_M/AndroidManifest.xml

```
<!-- Activity de clase Grabador de Audio -->
<activity android:name=".Audio.Nombre"
          android:label="@string/label_4"
          android:theme="@android:style/Theme.Dialog">
     <intent-filter>
         <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.SAMPLE_CODE" />
     </intent-filter>
</activity>
```

Al correr el programa, y seleccionar la opción “Audio Recorder/Player” del menú “Audio Rec”, se mostrará la Figura 5.26.

Figura. 5.26. Aplicación Audio Rec: Audio Recorder/Player - Ventana Ingreso nombre

Al momento no se ha definido ningún método que maneje el botón “Guardar”, por lo que esta *Activity* no hará nada más que presentarnos como lucirá. A continuación se definen nuevos métodos sobre la clase *Nombre.java.*

En el archivo *nombre.xml*, se identifican los recursos: editor de texto y botón, bajo las identificaciones *nombreAudio* y *botonNombre* respectivamente, serán utilizados como recursos por *Nombre.java* para adquirir la ruta y nombre donde se guardará el archivo a crear y pasará la información a una nueva actividad que se encargue de la grabación y reproducción de audio. A continuación, en la clase *Nombre.java*, se define un *listener*, agregar lo siguiente:

ANEXOS, sección A.6.10. SRO_M/src/apps/tesis/omap/Audio/Nombre.java

```
// Listener boton guardar
public void myClickHandler(View view) {
	switch (view.getId()) {
	case R.id.botonNombre:
		// Longitud > de .3gp
		if (text.getText().length() <= 4) {
			Toast.makeText(this,
				"Ingrese un nombre y ubicación al archivo:",
					Toast.LENGTH_LONG).show();
			return;
		}
		// Inicio Intent AudioRec, envio de extras
		Intent intent = new Intent(Nombre.this, AudioRec.class);
		Bundle bundle = new Bundle();
		bundle.putString("nomb", text.getText().toString());
		intent.putExtras(bundle);
		finish();
		startActivity(intent);
	}
}
```

El código verifica que el archivo contenga la extensión deseada, en este caso es “.3gp”, de no ser así, se desplegará un “Toast” indicando que se ingrese un nombre nuevamente. Cuando se verifique el nombre, se iniciará la clase *AudioRec.java*, para lo cual se utiliza un *Intent*, y se le agregan *Extras* con la información del nombre del archivo. Antes de iniciar la nueva *Activity AudioRec*, se finaliza la *Activity* actual mediante el método *finish( ).*

La clase *Nombre.java* solamente pretende dar un nombre a un archivo de audio que aún no ha sido establecido, la clase *AudioRec.java* se encargará de grabar las señales de

audio adquiridas desde la entrada de audio presente en la tarjeta y reproducirlas por la salida de audio, como se muestra en la Figura 5.2

Crear la clase *AudioRec.java* bajo el paquete *apps.tesis.omap.Audio* y modificarla como se mostrará a continuación. Cuando se requiere adquirir *Extras* de una *Activity* estos deben ser obtenidos en el método *OnCreate( )* siempre.

ANEXOS, sección A.6.11. SRO_M/src/apps/tesis/omap/Audio/AudioRec.java

```
package apps.tesis.omap.Audio;

import apps.tesis.omap.R;

public class AudioRec extends Activity {
	private MediaRecorder mediarecorder = null;
	private MediaPlayer mediaplayer = null;
	private int i; // bandera play/rec
	private String archivo = null;
	private ImageButton bro;
	private ImageButton br;
	private ImageButton bpo;
	private ImageButton bp;
	private ImageButton bso;
	private ImageButton bs;
	private TextView text;
	private TextView playI;
	private TextView estado;
	private Chronometer chronoRec;
	private Chronometer chronoPlay;
	private ProgressBar progreso;
	private int posicionActual = 0;
	private int total;
	private Handler mHandler = new Handler();

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.audio);
		// Obtener extras de Intent Nombre
		Bundle extras = getIntent().getExtras();
		if (extras != null) {
			archivo = extras.getString("nomb");
			// validación ubicación archivo
			if (!archivo.startsWith("/")) {
				archivo = "/" + archivo;
			}
		}
		bro = (ImageButton) findViewById(R.id.b_rec_on);
		br = (ImageButton) findViewById(R.id.b_rec);
		bpo = (ImageButton) findViewById(R.id.b_play_on);
		bp = (ImageButton) findViewById(R.id.b_play);
		bso = (ImageButton) findViewById(R.id.b_stop_on);
		bs = (ImageButton) findViewById(R.id.b_stop);
		text = (TextView) findViewById(R.id.TextNombre);
```

```
            estado = (TextView) findViewById(R.id.Estado);
            playI = (TextView) findViewById(R.id.PlayInicio);
            chronoRec = (Chronometer) findViewById(R.id.ChronoRec);
            chronoPlay = (Chronometer) findViewById(R.id.ChronoPlay);
            progreso = (ProgressBar) findViewById(R.id.BarraProgreso);
      }
}
```

No olvidar importar las demás librerías presionando las teclas crtl+shift+“o”, hacer esto siempre que se agregue nuevo código. Se observa que la información enviada por la *Activity Nombre*, es obtenida y almacenada en la variable *archivo*. Se verifica que sea un destinación válida, de no serlo, será grabada en el archivo raíz.

Se utilizan varios recursos definidos en la interfaz de usuario *audio.xml*, la cual debe ser creada en la carpeta *layout*. El código definido para la interface puede ser encontrado en ANEXOS, sección A.2.7. SRO_M/res/layout/audio.xml.

Los colores utilizados por el *layout*, fueron definidos anteriormente, pero hace falta definir el fondo de pantalla utilizado, para esto, crear el archivo backgroundaudio.xml y ubicarlo bajo SRO_M/res/drawable-hdpi. El diseño descrito es de tipo radial y define un gradiente en forma circular, en el archivo colocar lo siguiente:

ANEXOS, sección A.4.3. SRO_M/res/drawable-hdpi/backgroundaudio.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
      <gradient
            android:startColor="@color/back_audio1"
            android:endColor="@color/back_audio2"
            android:type="radial"
            android:gradientRadius="360"
            android:centerX="0.25"
            android:centerY="0.22"/>
</shape>
```

La Figura 5.27 presenta los recursos utilizados por la *Activity AudioRec.java*, muestra como está distribuida la ventana en cuanto a la información que exhibirá en la *Activity*, se encuentran relacionando cada recurso a una variable. En el gráfico se especifica el nombre de dicha variable, y su respectivo recurso:

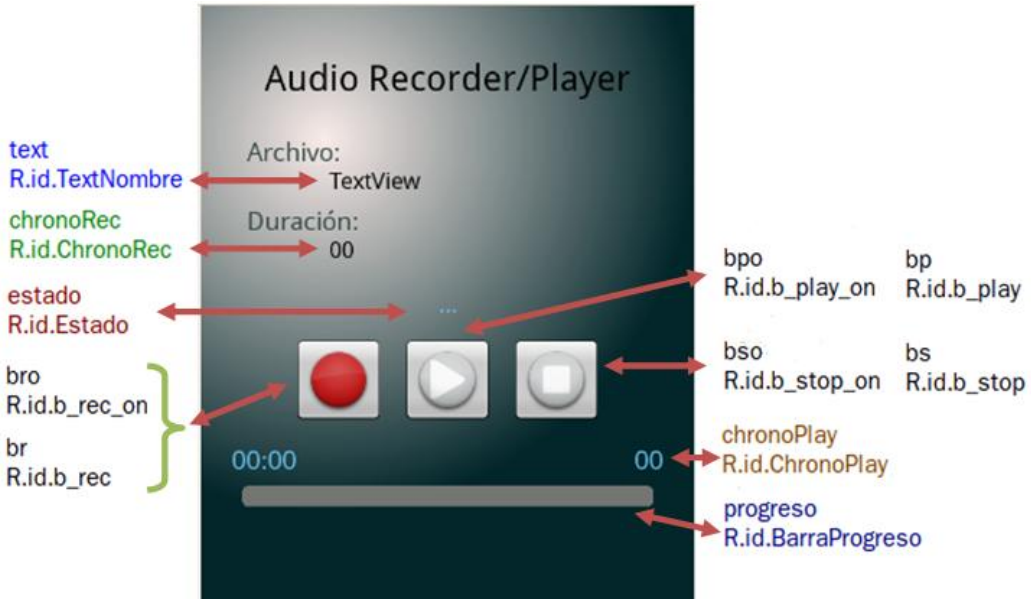


Figura. 5.27. Aplicación Audio Rec: Audio Recorder/Player - Asignación de variables

La interfaz de usuario *SRO_M/res/layout/audio.xml* describe botones, *TextViews* para presentar información del archivo y el estado del programa, además asigna dos cronómetros para mostrar la duración, y una barra de progreso. En realidad el programa está compuesto por seis botones tipo imagen, los cuales utilizan un layout *RelativeLayout* para ubicarlos uno sobre otro. Esta es otra forma de establecer la acción de un botón controlada desde una clase java. Estos son imágenes de botones seleccionados y no seleccionados para las acciones Play, Rec y Stop.

Sobre la pantalla, los recursos son presentados según lo maneje la clase *AudioRec.java*, Para lo cual se pretende crear método que hagan visibles o invisibles a los recursos que así lo ameriten. Para lo cual, en la clase *AudioRec.java*, se necesita agregar el siguiente código:

ANEXOS, sección A.6.11. SRO_M/src/apps/tesis/omap/Audio/AudioRec.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
...
// Layout
BotonInicio();
text.setText(archivo);
SinBarra();
}
```

Esta es el layout inicial, para lo cual se llama a la función *BotonInicio( )*, se coloca el path del archivo, y se desaparece la barra de progreso. Para este y todos los estados de layout definidos por la clase, es necesario agregar algunas funciones:

ANEXOS, sección A.6.11. SRO_M/src/apps/tesis/omap/Audio/AudioRec.java

```
public class AudioRec extends Activity {
	...
	// Estado diferente a play: Ocultar Barra de Progreso
	private void SinBarra() {
		progreso.setVisibility(ProgressBar.GONE);
		chronoPlay.setVisibility(Chronometer.INVISIBLE);
		playI.setText("");
	}
	// Layout de Botones
	private void BotonStop() {
		bro.setVisibility(View.VISIBLE);
		br.setVisibility(View.GONE);
		bpo.setVisibility(View.VISIBLE);
		bp.setVisibility(View.GONE);
		bs.setVisibility(View.VISIBLE);
		bso.setVisibility(View.GONE);
		}
	private void BotonPlayRec() {
		br.setVisibility(View.VISIBLE);
		bro.setVisibility(View.GONE);
		bp.setVisibility(View.VISIBLE);
		bpo.setVisibility(View.GONE);
		bso.setVisibility(View.VISIBLE);
		bs.setVisibility(View.GONE);
	}
	private void BotonInicio() {
		bro.setVisibility(View.VISIBLE);
		br.setVisibility(View.GONE);
		bp.setVisibility(View.VISIBLE);
		bpo.setVisibility(View.GONE);
		bs.setVisibility(View.VISIBLE);
		bso.setVisibility(View.GONE);
	}
}
```

El programa es manejado por tres funciones para definir estados, el estado *Rec* cuando se grabe audio, *Play* cuando se reproduzca audio y *Stop* para detener el estado de cualquiera de las anteriores. Los estados descritos son manejados por los botones definidos en la interface de usuario, y para utilizarlos es necesario agregar *listeners*, las cuales llamarán a las funciones.

ANEXOS, sección A.6.11. SRO_M/src/apps/tesis/omap/Audio/AudioRec.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
...
	// OnClick Listener de Boton de Grabación
```

```
        ImageButton botonRec = (ImageButton) findViewById(R.id.b_rec_on);
        botonRec.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                        rec(archivo);
                }
        });
        // OnClick Listener de Boton Stop
        ImageButton botonStop = (ImageButton) findViewById(R.id.b_stop_on);
        botonStop.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                        stop();
                }
        });
        // OnClick Listener de Boton Play
        ImageButton botonPlay = (ImageButton) findViewById(R.id.b_play_on);
        botonPlay.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                        play(archivo);
                }
        });

}
```

Primero se definirá la función *rec( )*, para lo cual se debe agregar el siguiente código.

ANEXOS, sección A.6.11. SRO_M/src/apps/tesis/omap/Audio/AudioRec.java

```
public class AudioRec extends Activity {
        ...

// Función para Grabación de Audio
        private void rec(String filePath) {
                i = 1;
                // layout
                BotonPlayRec();
                estado.setText("Grabando...");
                SinBarra();
                // MediaRecorder
                try {
                // Archivo donde grabar
                File mediafile = new File(filePath);
                if (mediafile.exists()) {
                        mediafile.delete();
                }
                mediafile = null;
                // Inicializa al media recorder
                mediarecorder = new MediaRecorder();
                // ConFigura al media recorder
mediarecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
mediarecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
mediarecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
                mediarecorder.setOutputFile(filePath);
                // Prepara al media recorder
                mediarecorder.prepare();
                // Inicia al media recorder
                mediarecorder.start();
                // Cronometro Grabacion
```

```
            chronoRec.setBase(SystemClock.elapsedRealtime());
            chronoRec.start();
            } catch (Exception e) {   e.printStackTrace();}
    }
}
```

mediarecorder.setOutputFormat(MediaRecorder.OutputFormat.*THREE_GPP*);

El código establece cual será el formato de salida, en este caso es *THREE_GPP*. Los formatos soportados se encuentran el las Tabla 4.1 y Tabla 4.2, los cuales se aplican para multimedia maejada por ARM y DSP respectivamente.

Se inicializarán las variables para el *layout*. La variable *i=1* es utilizada como una bandera que permita a la función *stop( )*, sin detener la grabación o la reproducción del audio. Además se definirá un nuevo archivo con el path *archivo*.

Android provee un estándar para la grabación de audio y video, a través de la clase *MediaRecorder*, que pasa por varios estados durante su ciclo de vida. Los estados son los siguientes [64]:

- Initialize: Inicializa la clase *MediaRecorder*.
- Inicialized: *MediaRecorder* se encuentra lista para utilizar.
- Data Source Configured: La fuente de donde proviene la media, estará lista.
- Prepared: Cuando *MediaRecorder* está lista para realizar grabaciones
- Recording: Cuando la grabación está en marcha.
- Realeased: Libera los recursos.

Al inicializar *MediaPlayer* se definen cuales serán las configuraciones de grabación. Se define que la fuente de audio. Se utilizó 3GP como formato de salida, ya que para la versón de Android Froyo, es la única que puede ser utilzada para grabación. Se definió una codificación por defecto, para que sea elegida por la plataforma, ya que puede variar de acuerdo a si el sistema es manejado por DSP o no.

La función que detendrá la grabación en curso será definida a continuación. La función *stop( )* actúa de acuerdo a una bandera definida enteriormente por la función *rec( ).*

ANEXOS, sección A.6.11. SRO_M/src/apps/tesis/omap/Audio/AudioRec.java

```
public class AudioRec extends Activity {
      ...


     // Función Stop Recorder y Play
      private void stop() {
            // Layout
            BotonStop();
            estado.setText("...");
            SinBarra();

            // Grabacion stop
            if (i == 1) {
                  mediarecorder.stop();
                  mediarecorder.release(); // reset();
                  mediarecorder = null;
                  chronoRec.stop();
            }
     }
```

Finalmente se presenta la función *play( )*, que será definida a continuación. Colocar el siguiente código a continuación.

ANEXOS, sección A.6.11. SRO_M/src/apps/tesis/omap/Audio/AudioRec.java

```
public class AudioRec extends Activity {
      ...

      // Funcion para Reproducir Audio Grabado
      private void play(String file) {
            // Layout
            BotonPlayRec();
            i = 2;
            try {
            // Inicializa al media player
            mediaplayer = new MediaPlayer();
            // Listener de termino de media player
            MediaPlayer.OnCompletionListener listener = new
MediaPlayer.OnCompletionListener() {
                  @Override
                  public void onCompletion(MediaPlayer mediaplayer) {
                        mediaplayer.release();
                        BotonStop();
                        chronoPlay.stop();
                        estado.setText("...");
                        SinBarra();
                  }
            };

            // ConFigura al media player
            mediaplayer.setOnCompletionListener(listener);
            mediaplayer.setDataSource(file);
            // Prepara al media player
            mediaplayer.prepare();
            // Inicia al media player
```

```
            mediaplayer.start();
            //layout
            estado.setText("Reproduciendo...");
            // Cronometro
            chronoPlay.setVisibility(Chronometer.VISIBLE);
            chronoPlay.setBase(SystemClock.elapsedRealtime());
            chronoPlay.start();
            playI.setText("00:00");
            // Barra de Progreso
            progreso.setVisibility(ProgressBar.VISIBLE);
           progreso.setProgress(0);
           progreso.setMax(mediaplayer.getDuration());
           total = mediaplayer.getDuration();
            new Thread(new Runnable() {
              public void run() {
                while (mediaplayer!=null && posicionActual < total){
                  try {
                   posicionActual = mediaplayer.getCurrentPosition();
                   } catch (Exception e){
                      return;
                   }
                  // Actualizacion de barra de progreso
                  mHandler.post(new Runnable() {
                  public void run() {
                           progreso .setProgress(posicionActual);
                   }
                   });
                 }
              }
           }).start();
            } catch (Exception e) {   e.printStackTrace();}
      }
}
```

La función recibe una variable contniendo el path donde leer el archivo. Android proporciona un estándar para la reproducción de archivos de audio y video, utilizando la clase *MediaPlayer*, y durante su ciclo de vida pasa por varios estados [64]:

- Idle: *MediaPlayer* es instanciado.
- Initialized: La fuente de media es establecida.
- Preparing: Se prepara la fuente de la media a reproducir.
- Prepared: *MediaPlayer* se encuantra preparada para la reproducción.
- Stated: La reproducción está en curso.
- Paused: Se realizó una pausa en la reproducción.
- Playback Complete: La reproducción termina.
- Stopped: *Media Player* no se encuentra preparado para reproducir el archivo.
- End: *MediaPlayer* no se encuentra asociado a ningún recurso y es liberado.

MediaPlayer.OnCompletionListener listener

Fue definida para tomar acción cuando la reproducción de audio terminara, de esta forma se pusieron atributos al layout y se liberó a MediaPlayer.

La Barra de Estado fue utilizada para visualizar el estado de la reproducción, obteniendo es estado de reproducción actual. Además fue definida la variable *i=2* para permitir la utilización del botón Stop. A la función *stop( )*, se le agregó lo siguiente:

ANEXOS, sección A.6.11. SRO_M/src/apps/tesis/omap/Audio/AudioRec.java

```
public class AudioRec extends Activity {
      ...
      // Función Stop Recorder y Play
      private void stop() {
            ...
            // Play stop
            if (i == 2) {
                  mediaplayer.stop();
                  mediaplayer = null;
                  chronoPlay.stop();
            }
      }
```

Ahora es necesario agregar la *Activity* al archivo *AndroidManifest.xml*

ANEXOS, sección A.5.1. SRO_M/AndroidManifest.xml

```
    <activity android:name=".Audio.AudioRec"
              android:label="@string/sound_tesis">
       <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.SAMPLE_CODE" />
       </intent-filter>
   </activity>
```

Correr el programa, y seleccionar la opción “Audio Recorder/Player” del menú “Audio Rec”. Se mostrará (Figura 5.28) la ventana sobre la cual se debe ingresar un path para el archivo, en esta caso se utilizó “/sdcard/Audio/audio.3gp”, cuando se seleccione el botón Guardar, se mostrará la ventana de al lado. La información mostrada en “Archivo:” es el path de la unicación donde será creado y guardado el archivo.

Figura. 5.28. Aplicación Audio Rec: Audio Recorder/Player - Inicio

Como se muestra en la Figura 5.29. Cuando se seleccione el botón “Rec”, empezará a grabar, se mostrará el estado “Grabando”, y el cronómetro indicará cuanto tiempo ha transcurrido. Al accionar el botón “Stop”, se mostrará la ventana de al lado, indicando el path del archivo y su duración.

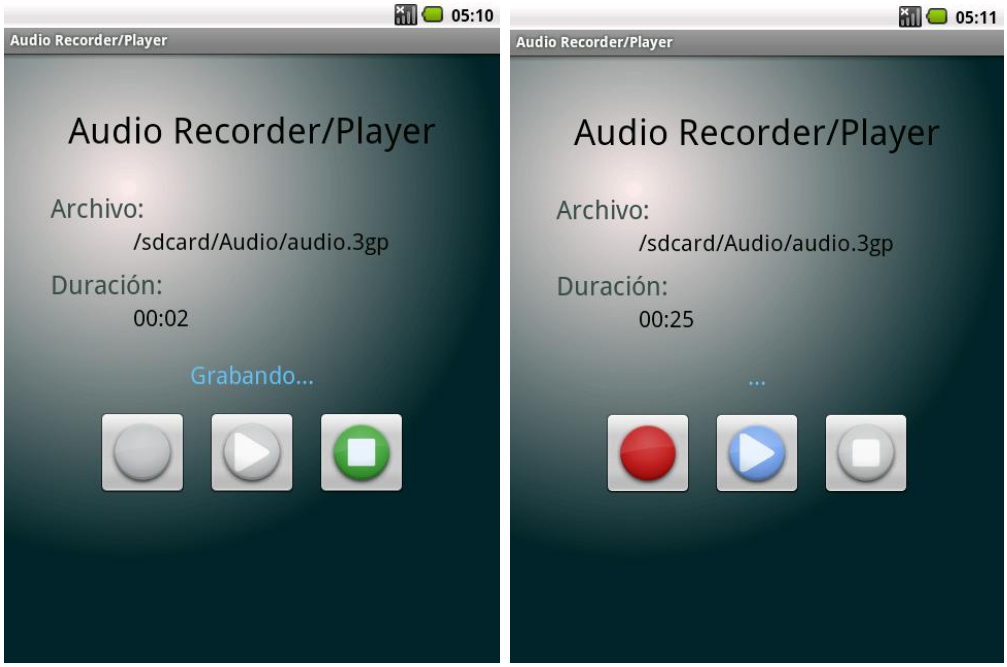


Figura. 5.29. Aplicación Audio Rec: Audio Recorder/Player - Grabación

Los botónes “Rec” y “Play” serán visualizados, cuando se accione el botón “Rec”se regresará a la Figura 5.29, donde se podrá sobre escribir el archivo y volver a grabar audio. Si se selecciona el botón “Play” se presentará la Figura 5.30, que corresponde a la reproducción del archivo; se muestra la información del archivo, y se visualiza el tiempo transcurrido de la reproducción en la barra de estado y el el cronómetro inferior. Una vez terminada la reproducción de audio o cuando se presione el botón “Stop”, se mostrará la pantalla derecha.

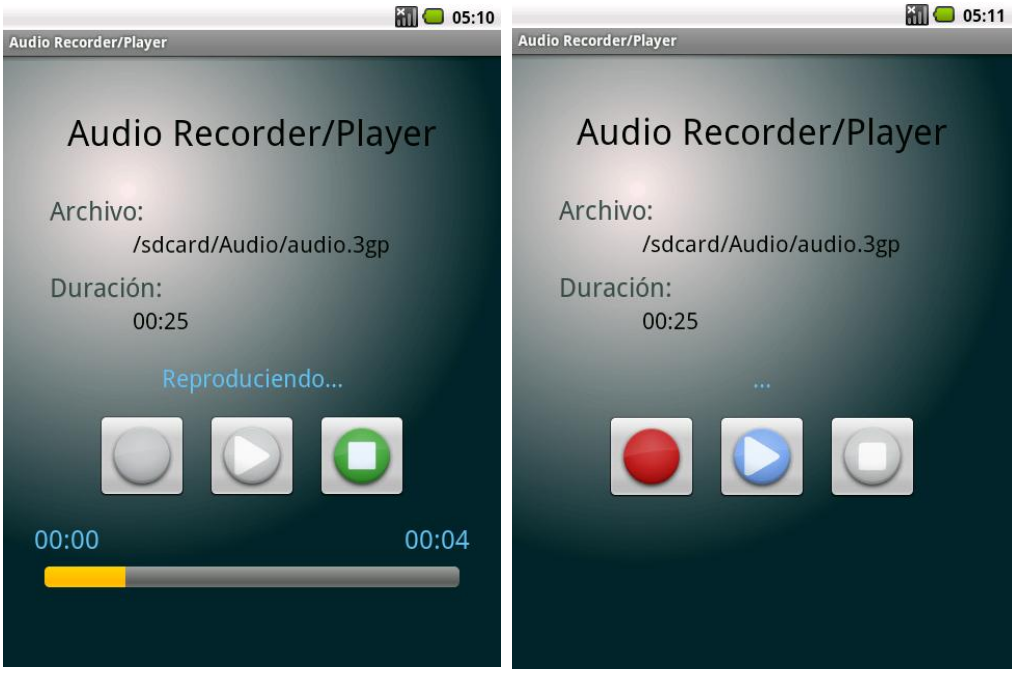


Figura. 5.30. Aplicación Audio Rec: Audio Recorder/Player - Reproducción

5.4. RESUMEN DEL CAPÍTULO

Las aplicaciones tratadas en la presente, involucran ciertos aspectos sobresalientes sobre Android y sobre la plataforma EVM OMAP3530. Fueron realizadas pensando motrar las capacidades que brindan las plataformas embebidas cuando son aplicadas sobre Android. Presentan aplicaciones simples que pueden ser base para aplicaciones más complejas, inclyen el manejo de multimedia, tanto de grabación como de reproducción de audio y video. Incluye un pequeño programa utilizando las librerías OpenGL, mostrando el proceso de desarrollo de una figura básica 3D, pero las oportuidades son infinitas.

El Módulo de Evaluación permite desarrollar aplicaciones de bajo consumo de potencia, tales como: Reproductores multimedia portátiles, Dispositivos de navegación, Consolas portátiles, Ordenadores single board, Terminales de datos, Punto de venta/servicio, Radio definido por software, Aplicaciones médicas, Controladores de medios de comunicación, y muchos otros productos que requieren de baja potencia y alto rendimiento. Y ahora que se mostró un pequeño tutorial de cómo realizar aplicaciones básicas, es posible que en el futuro el DEEE pueda realizar alguna de las aplicaciones enumeradas.

CAPÍTULO 6

PRUEBAS Y RESULTADOS

Resumen: En el presente capítulo se realizará un análisis cualitativo, comparativo y de desempeño sobre las aplicaciones realizadas. El análisis será enfocado principalmente en el desempeño multimedia y procesamiento gráfico. El desempeño de la plataforma EVM OMAP3530 será comparado entre dos dispositivos de similares características. La información recolectada permitirá valorar las ventajas de la utilizacion de plataformas OMAP y Sistemas de Desarrollo Android para el prototipo rápido de aplicaciones embebidas.

6.1. PRUEBAS DE APLICACIÓN

En el Capítulo 4 y 5, se busco crear un sistema de referencia que brinde un soporte para futuras aplicaciones. Con la plataforma adquirida EVM OMAP3530. Es necesario recordar que el sistema efectuado, SRO-M, se basaba en un compendio de aplicaciones enfocadas, en su mayoría, al manejo multimedia y gráficas 3D. En el presente Capítulo se busca poner a prueba dichas aplicaciones, para conocer cómo influyen en el desempeño de la tarjeta y como se comportan frente a plataformas similares.

Al referirse a plataformas similares se debe considerar que este es un proyecto novel en el país y talvez en latinoamérica, por lo que ni el departamento DEEE u otra institución conocida cuantan con con plataformas de desarrollo SoC de las mismas características que EVM OMAP3530, por lo que se optó por utilizar dispositivos comerciales como un Smartphone y Tablet Android que utilizaban multiprocesamiento a través de sistemas SoC

ya instalados de fabrica y con poco nivel de configurabilidad de hardware e incluso software.

6.1.1. Contextos de Prueba

Se han planteado pruebas de aplicación sobre dos contextos. 1) evaluación de desempeño multimedia OMAP con diferentes configuraciones internas y 2) comparando el desempeño de las aplicaciones multimedia sobre la tarjeta OMAP y de las aplicaciones multimedia sobre otras plataformas existentes.

1) Desempeño OMAP

La plataforma SoC OMAP3530 se encuentra compuesta por un procesador ARM Cortex-A8 como procesador de propósito general GPP y con el Procesador TMS320DMC64x + VLIM DSP para procesamiento multimedia. Idealmente los dos trabajan juntos a través del DSP/BIOS Link y protocolos IPC. . Sin embargo, el procesador ARM tiene también la posibilidad de encargarse de procesamiento multimedia por su cuenta. En consecuencia, estos son precisamente los escenarios que se plantean, uno en el que las señales multimedia son procesadas solamente por el GPP y otro en el que existe comunicación entre procesadores mediante protocolos IPC, habilitando al DSP a través de DVSDK.

En el Capítulo 4 se habló sobre la creación de ambientes de desarrollo para Android, para lo cual, se planteó las configuraciones de trabajo de la plataforma SoC OMAP3530. Se utilizó información del proyecto ROWBOAT, el cual ofrece la creación de un Filesystem para el manejo multimedia a través del DSP y el GPP. Ambas configuraciones presentadas serán evaluadas en el presente capítulo.

El sistema operativo Android se encuentra presente en el procesador de propósito general GPP ARM, así que las pruebas buscarán analizar cómo influye la carga de procesamiento multimedia en el desempeño del procesador ARM.

2) OMAP y Otros Dispositivos a Evaluar

Se vio necesario realizar pruebas de desempeño para comparar el rendimiento de OMAP3530 frente a otras plataformas para el desarrollo de sistemas embebidos. Al no encontrar plataformas semejantes en el medio, se eligió utilizar los siguientes dispositivos comerciales de similares características funcionales: Smartphone HTC Wildfire A3333 [80] y Tablet Acer Iconia Tab A500 [81]. La Tabla 6.1 muestra un cuadro comparativo de las principales características de los dispositivos presentados, comparándolos con la tarjeta EVM OMAP3530:

Tabla. 6.3. Tabla Comparativa - Características Generales de Dispositivos

Modelo	Chipset	OS	Version Kernel	Tamaño	Display	Teclado	Formatos de Audio	Formatos de Video
Acer Iconia Tab A500	Nvidia Tegra 250 SoC	Google Android 3.0 Honeycomb	2.6.36.3	260 × 177 × 13.3 mm	10.1"	Pantalla sensible al tacto, multitouch	MP3, WMA, AMR, OGG, AAC, AAC+, WAV	MPEG-4, H.263, H.264, Xvid
HTC Wildfire A3333	Qualcomm MSM7225	Google Android 2.1 Eclair	2.6.29-4266b2e1	106.8 × 60.4 × 12.2 mm	3.2"	Pantalla sensible al tacto	AAC, AMR, AWB, QCP, MP3, WMA, WAV, MIDI	AVI, MPEG-4, H.263, H.263, WMV
EVM OMAP3530	OMAP3530	Google Android 2.2 Froyo	2.6.32 a0393957@ swubn01 #2	203.2 × 143.51 mm	3.7”	Pantalla sensible al tacto	ACC, AMR FLAC, MP3, MIDI, Ogg, PCM/WAVE	BMP, H.263, H.264, MPEG-4, VP8

A continuación se presenta la Tabla 6.2, que es un cuadro comparativo de los procesadores utilizados por los dispositivos y su comparación con OMAP3530.

Tabla. 6.4. Tabla Comparativa - Procesadores y Memoria

Modelo	Chipset	GPP	DSP	GPU	RAM	Memoria Interna
Acer Iconia Tab A500	Nvidia Tegra 250	ARM Cortex-A9 (dual-core) 1 GHz	Procesador embebido de Audio y Video de Alta Definición	NVIDIA ULP GeForce	1 GB	16GB micro-SD
HTC Wildfire A3333	Qualcomm MSM7225	ARM1136EJ-S 528 MHz	QDSP5000 122 MHz	Adreno 200	384 MB	512 MB
EVM OMAP3530	OMAP3530	ARM Cortex-A8 720 MHz	TMS320C64x+TM 520 MHz	POWERVR SGX530	64 KB	GPP 16KB DSP 32KB

Iconia Tab A500, es una tablet propio de Acer que incluye un SoC multiprocesador NVIDIA Tegra2, su arquitectura se compone de dos unidades de procesamiento ARM

Cortex-A9, e incluyen un procesador embebido de Audio y Video de Alta Definición. Para el procesamiento gráfico incluye el procesador GeForce Ultra Low Power (ULP) [81].

Figura. 6.31. Iconia Tab A500

HTC Wildfire, es un smartphone creado por la compañía HTC Corporation. Basa su funcionamiento en el SoC Qualcomm MSM7225, el cual se compone de varios procesadores, uno de propósito general ARM1136EJ-S, uno DSP para manejo multimedia QDSP5000 y dos procesadores Baseband (BBP, Baseband Radio Processor), para manejo de telefonía celular. Incluye un procesador gráfico Adreno 200 [80].

Figura. 6.32. HTC Wildfire A3333

6.1.2. Descripción de Escenarios y Parametros de Evaluación

Para evaluar el desempeño de la plataforma OMAP3530 se han planteado tres escenarios: A, B, y C. En el Escenario A se evaluará la incidencia del desempeño multimedia sobre el procesador de propósito general, cuando se reproducen videos a

diferentes resoluciones. El escenario B se basará en el análisis de desempeño multimedia sobre el procesador de propósito general, al reproducir videos con distintos FPS (Frames per Second). Finalmente el escenario C evaluará el procesamiento gráfico mediante la obtención de los FPS necesarios para generar una imagen.

El escenario A y B serán llevados a cabo sobre la aplicación Reproductor de Video, el escenario C se realizará sobre la aplicación Cubo 3D. Los parámetros de evaluación necesarios para efectuar las pruebas antes descritas serán los siguientes:

Escenario A

Obtención de carga en el CPU para reproducción de video sobre distintas resoluciones. Se realizará sobre la plataforma EVM OMAP3530 a través de procesamiento multimedia mediante el procesador ARM (GPP) y subsistema IVA (DSP); sobre Tablet Acer Iconia A500 y sobre Smartphone HTC Wildfire. A continuación se presenta la Tabla 6.3 con los parámetros fijos y variables escogidos para la evaluación.

Tabla. 6.5. Parámetros Escenario A

Parámetros Fijos	**Formato**	MP4
	Parámetros de Video	Codec: MPEG-4 FPS: 30 Bitrate: 1264 Kbps
	Parámetros Audio	Codec: AAC Frecuencia: 44.1 KHz Bitrate: 128 Kbps Canales: 2
Parámetros Variables	**Resoluciones de Video**	SQCIF: 128 x 96 QQVGA: 160 x 120 QCIF: 176 x 144 256 x 192 QVGA: 320 x 240 CIF: 352 x 288 512x384 VGA: 640 x 480 PAL: 768 x 576 SVGA: 800 x 600 832 x 624 XGA: 1024 x 768

Escenario B

Adquisición de carga en el CPU para reproducción de video a diferentes tasas de FPS. Se analizará sobre la plataforma EVM OMAP3530, a través de procesamiento multimedia mediante el procesador ARM (GPP) y el subsistema IVA (DSP); sobre Tablet Acer Iconia A500, y sobre Smartphone HTC Wildfire. A continuación se presenta la Tabla 6.4 con los parámetros fijos y variables escogidos para la evaluación.

Tabla. 6.6. Parámetros Escenario B

Parámetros Fijos	**Formato**	MP4
	Parámetros de Video	Codec: MPEG-4 FPS: 30 Bitrate: 1264 Kbps Resolución: VGA 640 x 480
	Parámetros Audio	Codec: AAC Frecuencia: 44.1 KHz Bitrate: 128 Kbps Canales: 2
Parámetros Variables	**Frames per Second**	6.25 FPS 10 FPS 15 FPS 30 FPS

Escenario C

Se obtendrán los FPS utilizados en el programa Cubo 3D. Se realizará sobre la plataforma EVM OMAP3530, sobre Tablet Acer Iconia A500 y sobre Smartphone HTC Wildfire. El análisis de carga de procesamiento DSP o GPP no influye en para esta prueba ya que el sistema gráfico es manejado por el procesador gráfico, en cada uno de los casos.

6.1.3. Procedimiento

El análisis requerido considera dos parámetros que serán los que identifiquen el desempeño de las diferentes configuraciones. Estos parámetros son: 1) consumo de recursos en el procesador de propósito general y 2) FPS.

1) Consumo de Recursos GPP

Es importante analizar la carga sobre el procesador GPP ya que este es el encargado de la mayoría de procesos sobre la plataforma. El procesador GPP también es llamado procesador Host ya que maneja el resto de procesos, y es importante que tenga una baja carga en multimedia para que pueda encargarse de procesos paralelos con alto desempeño.

El análisis se realizó en el programa "Reproductor de Video" del SRO-M. Se efectuaron análisis sobre la carga consumida por la reproducción de contenido multimedia, en este caso video, cuando existían diferentes resoluciones de video y distintos FPS de un video. La conversión de video se llevó a cabo, utilizando la herramienta de edición de video "Free Video Converter" y se modificaron las características revisadas en la sección 6.1.2. Los videos fueron colocados en la tarjeta MMC/SD externa, en la dirección: */sdcard/Test*. Además se modificó el programa en java, para que presente un menú de selección de los videos, cuyo código fuente se encuentra en ANEXOS, Sección A.7. Pruebas SRO-M Video Player.

Se observó que la carga se mostraba en el proceso */system/bin/mediaserver*, así que este fue el enfoque del estudio. La toma de muestras fue realizado de la siguiente manera: fueron tomas 30 muestras por segundo, y se realizaron 5 toma de muestras. Con esto fue posible obtener un solo promedio y acercarse a un valor verdadero. El procedimiento fue realizado para cada video.

El análisis de carga será realizado sobre el CPU ARM, para esto se utilizó la herramienta *top* [82], que es una herramienta que monitorea los recursos del sistema, pertenece al paquete procps, que se encarga de la recolección de utilidades que proveen información de procesos utilizando el filesystem ***/proc***. La herramienta *top* provee estadísticas de tareas manejadas por el kernel de forma dinámica y en tiempo real. Este comando muestra el porcentaje de utilización del CPU y de memoria utilizada en cada proceso. Este programa puede ejecutarse como un proceso background mientras que corre la aplicación analizada.

Para acceder al Shell de la plataforma se utilizó la herramienta ADB, configurada en el capítulo 3, mediante el siguiente comando, ejecutado en el Terminal de la máquina de trabajo:

```
adb shell top –d 1 –n 30 | grep  /system/bin/mediaserver
```

La opción *–d* indica que las tomas serán tomadas a cada segundo, la opción *–n* limita el número de muestras, que en este caso es 30, y se imprimirá solamente las estadísticas del proceso */system/bin/mediaserver* con la opción *grep*. Es importante considerar que el proceso *top* también consume recursos del sistema, pero como pudo observarse, este no era mayor a un 5% [82].

b) Adquisición de FPS

Frames per Second, tiene relación con la ilusión de movimiento que produce la sucesión de imágenes, está definido por la rapidez con que se puede actualizar la pantalla, es medido contando el número de pantallas o frames mostrados por segundo. El ojo humano es capaz de distinguir la actualización de la pantalla cuando esta se encuentra a una tasa baja de FPS. Se dice que una tasa aceptable se encuentra en un rango de 15 a 30 FPS, pero en el caso de video juegos y de aplicaciones gráficas computarizadas, este puede llegar a ser de hasta 60 FPS [75].

Fueron necesarias ciertas modificaciones en el programa “Cubo 3D” del SRO-M, para medir los FPS de la animación. Se precisó incluir un bucle que contabilice el número de frames dibujados en un periodo de 10 segundos y que presente un promedio final. Esto fue realizado 12 veces para obtener un resultado verdadero, en el proceso fueron ignoradas las dos primeras muestras para que los valores analizados sean constantes. El código de la aplicación Cubo 3D, fue modificado, en su clase *GLRender.java*, a la cual se le agregó código para contar FPS, favor referirse a ANEXOS, Sección A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java.

El bucle [75] fue añadido al método *onDrawFrame( )*, que es el encargado de presentar el gráfico. Los resultados fueron presentados en un TAG sobre el LogCat del Monitor de Depuración DDMS.

6.2. RESULTADOS

6.2.1. Análisis de Arquitecturas

Antes de analizar los resultados de las pruebas realizadas, es necesario establecer las diferencias entre los procesadores utilizados, es decir que ventajas y desventajas presentan sus arquitecturas al momento del procesamiento multimedia y procesamiento gráfico.

Los procesadores ARM [33] son los más utilizados en la industria de los sistemas embebidos. Los tres equipos analizados incluyen, en su configuración, un procesador ARM. La familia de procesadores ARM11 pertenecen a los procesadores Clásicos de ARM, los cuales fueron especialmente desarrollados para dispositivos con alta eficiencia, rendimiento y bajo precio. La familia de procesadores ARM Cortex-A, son enfocados hacia aplicaciones de alto rendimiento y son utilizados para desarrollo de smartphones, Notebooks, eBook Readers, TV Digital, etc.

El procesador ARM1136EJ-S [33] no soporta Tecnología Neon, ni instrucciones Thumb-2. Neon posibilita instrucciones SIMD para ejecución paralela de operaciones, y ARM1136EJ-S utiliza Thumb que incorpora un conjunto de de instrucciones adicional de 16 bits, Thumb-2 adiciona un set de instrucciones a 32 bits para dar al conjunto de instrucciones mayor amplitud y haciendo posible la ejecución de instrucciones de tamaño variable. Estas características son esenciales en el desempeño final de aplicaciones y es lo que lo diferencia de los procesadores de la serie Cortex-A.

La arquitectura ARM Cortex-A9 [33] fue desarrollada a partir del procesador ARM Cortex A8, implementando algunas mejoras, así como el procesamiento dinámico, mediante la reordenacoón de instrucciones, operación de procesamiento a mayor frecuencia (1GHz) y Multiprocesamiento Simétrico (SMP Symmetrical Multiprocessing) de doble núcleo, incrementando el rendimiento de procesamiento.

En cuanto a procesamiento gráfico, podemos observar que los equipos estudiados utilizan procesadores distintos. Podemos decir que todos tienen en común el soporte de OpenGL ES 2.0, y cada uno está equipado para procesar datos de imagen como pixeles y vertexs, pero ofrecen diferente desempeño.

6.2.2. Presentación y Análisis de Resultados

Los resultados representan el consumo de recursos sobre el procesador de propósito general. En el caso de la Tarjeta EVM OMAP3530 ARM es el GPP. Para evitar complicaciones, cuando la carga de multimedia sea manejada solamente por el procesador ARM, se lo denominará OMAP-ARM y cuando el procesamiento multimedia sea manejado por el DSP, se lo denominará OMAP-DSP.

Escenario A

El primer escenario representa al consumo de Recursos de procesamiento sobre el procesador de propósito general, para diferentes Resoluciones de Video. Los resultados se encuentran plasmados en la tabla 6.5 y gráfico 6.3.

Tabla. 6.7. Resultado Porcentaje de Utilizacion de CPU a diferentes Resoluciones de Video

Resolución	OMAP-ARM	OMAP-DSP	ACER	HTC
128x96	8.4	8.0	10.0	26.9
160x120	9.6	8.1	10.2	26.3
176x144	10.7	8.1	10.4	33.8
256x192	15.6	8.2	11.1	49.0
320x240	19.1	7.5	11.1	64.7
352x288	25.5	8.7	12.5	74.8
512x384	33.9	8.6	12.8	86.8
640x480	41.8	7.8	12.4	90.4
768x576	55.5		13.8	
800x600	57.6		13.6	
832x624	65.0		14.0	
854x480	66.0		13.4	
960x720	66.1		14.4	
1024x768	66.9		14.2	

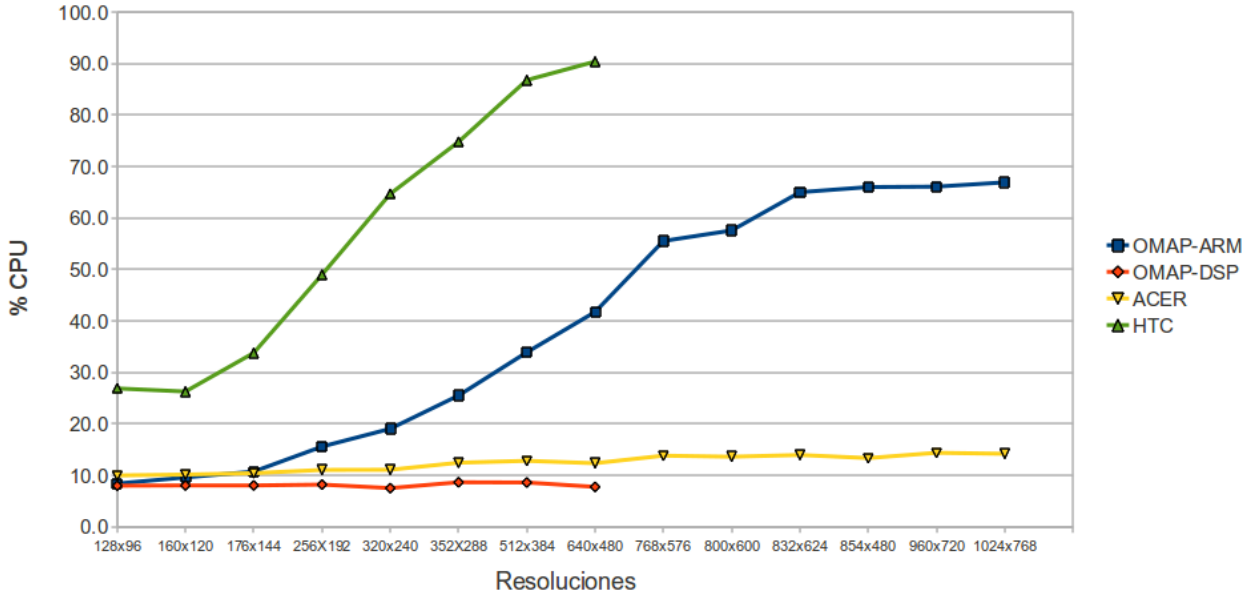


Figura. 6.33. Resultado Porcentaje de Utilizacion de CPU a diferentes Resoluciones de Video

Primero se debe analizar el desempeño OMAP, de la Tabla 6.5 se observa que la carga OMAP-DSP es un valor constante, no representa ni un 10% del consumo total de CPU, y además se observa que las resoluciones de video soportados por OMAP-DSP solamente llegan hasta una resolución VGA (640 x 480). A diferencia de OMAP-DSP, OMAP-ARM soporta un mayor rango de resoluciones de video, pero se observa además que el consumo de CPU incrementa hasta 80% para conseguirlo.

Para evaluar el desempeño de OMAP frente a otras soluciones SoC, se referirá a la Figura 6.3. Se puede observar que HTC representa la mayor carga para su procesador llegando hasta un 90% del consumo total, y al igual que OMAP-DSP soporta resoluciones de video hasta VGA (640 x 480). Su consumo de recursos es seguido OMAP-ARM. Acer presenta un incremento de carga muy ligero de un 4%, soportando todos los rangos de resolución utilizados. OMAP-DSP presenta el menor consumo de recursos de las cinco, su desventaja se centra en el limitado rango de resoluciones soportadas.

Escenario B

El segundo escenario representa al consumo de Recursos de procesamiento sobre el procesador de propósito general, para diferentes FPS para un video de resolución y demás parámetros de Video fijos. A continuación los resultados.

Tabla. 6.8. Resultado Porcentaje de Utilizacion de CPU a diferentes FPS de Video

FPS	OMAP-ARM	OMAP-DSP	Acer	HTC
6.25	19.3	4.4	9.6	55.0
10	25.1	5.0	10.5	75.0
15	29.7	5.0	11.2	86.0
25	37.5	7.0	11.9	89.0
30	41.8	7.8	12.4	90.4

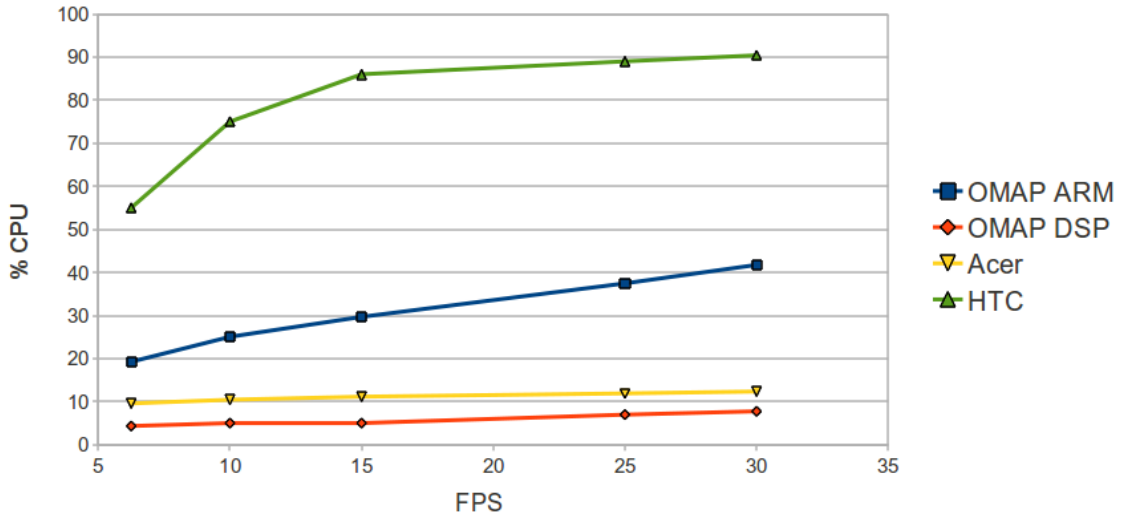


Figura. 6.34. Resultado Porcentaje de Utilizacion de CPU a diferentes FPS de Video

La Figura 6.4 representa el consumo de recursos del procesador para las plataformas. Casi todas muestran incrementos lineales conforme se incrementa los FPS del video. Para OMAP se observa nuevamente la relación entre el manejo multimedia en sus dos configuraciones. Su relación es de 5 a 1 aproximadamente, por lo que resulta más adecuado el manejo multimedia a través de la configuración OMAP-DSP.

Con respecto a las demás plataformas, se concluye que OMAP-DSP representa un menor consumo de CPU, seguido por la tablet Acer y OMAP-ARM. Respecto a HTC, se observa un incremento de carga de alrededor de un 45%, que es un valor considerable para el procesador.

Es importante analizar la incidencia que tiene el uso de procesadores DSP sobre el procesador de propósito general GPP, ya que se acaba de comprobar que influye directamente sobre la carga de procesamiento. Puede ser decisivo para el desempeño de la

tarjeta cuando además se sumen otros procesos que puedan correr simultáneamente con aplicaciones multimedia.

Escenario C

El tercer escenario representa la velocidad de actualización gráfica sobre la pantalla LCD de las plataformas experimentadas, mediante la medición de los FPS, es decir capacidad de actualización de los cuadros por segundo de los gráficos presentados, al renderizar la imagen. La Tabla 6.7 y Figura 6.5 indican sus resultados:

Tabla. 6.9. Resultado FPS en Procesamiento de Gráficos

Cubos	FPS OMAP	FPS HTC	FPS Acer
Cubo Básico	58.4	67.7	61.0
Cubo Iuminado	58.4	67.5	61.0
Cubo con Movimiento	58.4	67.3	61.0
Cubo con Textura	58.4	36.9	61.0
Cubo Transparente	58.4	37.5	61.0

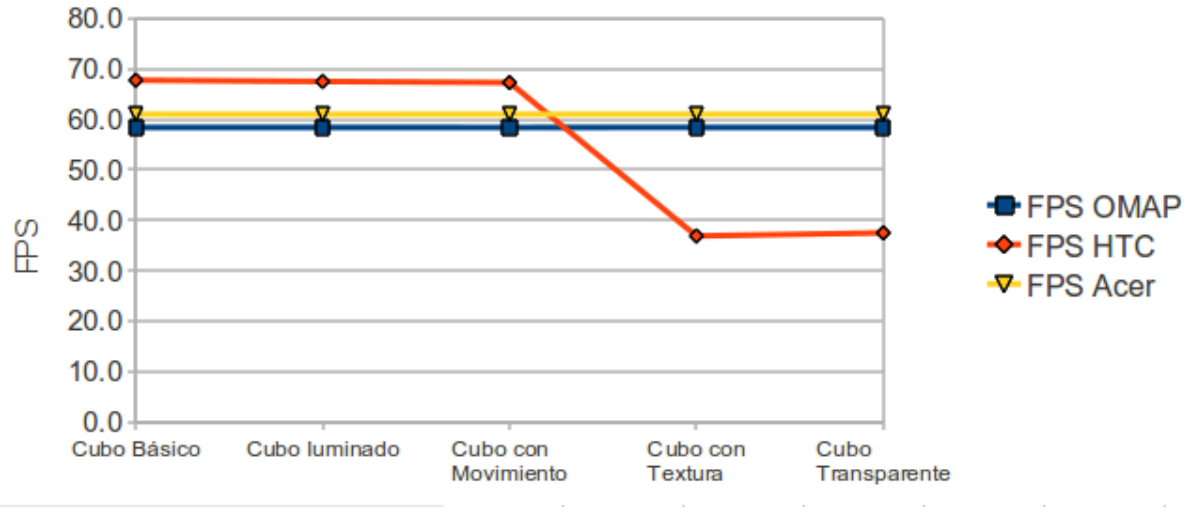


Figura. 6.35. Resultado FPS gráficos

No fue necesario analizar la tasa de FPS para OMAP-ARM y OMAP-DSP, ya que inciden de igual manera para las dos. El procesamiento de imágenes es manejado por el Procesador Gráfico en el SoC OMAP y de las otras plataformas. Se observa que tanto OMAP como Acer tienden a presentar FPS constantes, a diferencia de HTC que disminuye

la tasa de FPS al colocar textura y transparencia al cubo. Esto quiere decir que su desempeño disminuye al aumentar el procesamiento de instrucciones gráficas. Realizando un promedio de tasa de FPS se determinó que HTC presentaba un FPS de 55.4, es decir tiene el menor desempeño, seguido por OMAP y Acer.

Es necesario aclarar que los resultados presentados en el presente trabajo, son bastante lógicos ya que no es posible comparar las capacidades que tendrían las plataformas evaluadas, porque en cierta manera el tablet y el smartphone son dispositivos de propósito general, mientras que el EVM OMAP3530 utilizado es una plataforma de desarrollo (evaluación) que permite alta configurabilidad, mientras que los otros fueron optimizados ya para un mercado especifico.

CAPÍTULO 7

CONCLUSIONES Y RECOMENDACIONES

Resumen: A continuación se muestran algunas conclusiones y recomendaciones finales.

7.1. CONCLUSIONES

A través del presente trabajo se logró desarrollar un Sistema de Referencia para el desarrollo rápido y robusto de aplicaciones multimedia el cual servirá como modelo para futuras aplicaciones, empleando la Plataforma SoC TI OMAP EVM OMAP3530.

Se estudió, analizó y documento la teoría encontrada tras el diseño de circuitos integrados. Su evolución hacia los System-onChip, a través del análisis de metodologías de diseño, y cómo se solventaron problemas de costo y tiempo de desarrollo. Se analizaron los System-on-Chip, y cómo representan la mayor evolución en el diseño de circuitos integrados, implementando la reutilización de bloques de propiedad intelectual, basando su diseño en plataformas, su diseño busca la optimización de desempeño, consumo de energía y time-to-market. El diseño con plataformas reduce aun más el tiempo de desarrollo al reunir un conjunto de funcionalidades y arquitecturas comunes para varios sistemas embebidos y reuniendo un conjunto de funcionalidades en arquitecturas comunes.

Esta tendencia de diseño fue estudiada a través de las plataformas SoC OMAP de propiedad de Texas Instruments, específicamente la versión OMAP3530, debido a su flexibilidad para la evaluación de aplicaciones orientadas hacia multimedia. Es necesario aclarar que en el proyecto, el desarrollo de una aplicación no fue el objetivo principal. Por el contrario, el trabajo se ha orientado a un nivel más avanzado que es los conceptos,

metodologías de desarrollo actuales de sistemas embebidos, y la arquitectura existente en dichas plataformas

Fueron analizadas las preferencias del mercado en cuanto a tarjetas de evaluación que incluyan plataformas SoC OMAP3530. El resultado fue la adquisición por parte del DEEE de la tarjeta de desarrollo fabricada por Mistral Solutions Inc., llamada EVM OMAP35x, sobre la cual se ha desarrollado el proyecto y tesis.

Se realizó un análisis de la arquitectura de las plataformas OMAP3530. Además, se pudieron diferenciar importantes características de diseño SoC basado en plataformas, como es el diseño jerárquico, considerando a la arquitectura software, tan crítica como la de hardware. Las plataformas TI OMAP implementan la reutilización IP como parte de su infraestructura.

El Sistema de Referencia realizado incluyó las metodologías para el desarrollo rápido y económico de sistemas embebidos multimedia de alto desempeño orientados a diversas aplicaciones, empleando Plataformas OMAP, interfaces de última generación y código abierto Android.

Se revisaron los procedimientos necesarios para obtener un ambiente de desarrollo ideal que permita implementar aplicaciones multimedia Android. Dichos procedimientos incluyen la configuración de los binarios del sistema operativo Android, su compilación en una tarjeta SD, la configuración de hardware para el funcionamiento de la tarjeta, configuración de u-boot, arranque del sistema, y la configuración de herramientas para el desarrollo de aplicaciones y manipulación de la tarjeta.

Para la realización de pruebas, se presentaron dos configuraciones, una en la cual el procesamiento digital de señales era controlado solamente por el procesador de propósito general GPP y la segunda mediante la utilización del procesador DSP para el manejo de imágenes, video y audio.

Esta tesis y proyecto de iniciación científica presenta como uno de sus resultados el desarrollo del Sistema de Referencia SRO-M y documentación respectiva. Este incluye

cinco aplicaciones multimedia pequeñas que muestran la versatilidad de la programación de Android y la aplicación del leguaje de programación Java. Las aplicaciones tratadas en la presente involucran ciertos aspectos sobresalientes sobre Android y sobre la plataforma EVM OMAP3530. Estas fueron realizadas pensando en mostrar las capacidades que brindan las plataformas embebidas cuando son aplicadas sobre Android.

Fueron realizadas pruebas de desempeño, para la evaluación de las ventajas de esta experiencia frente a otras soluciones de desarrollo de Sistemas Embebidos y SoC. Se realizó análisis cualitativo, comparativo y de desempeño sobre las aplicaciones realizadas. El análisis fue enfocado sobre el desempeño multimedia y de procesamiento gráfico. Para ello, se tomaron en cuenta los siguientes escenarios sobre distintas configuraciones de la plataforma OMAP3530, a través de procesamiento multimedia mediante el procesador ARM (GPP) y mediante el subsistema IVA (DSP). Además, se realizaron pruebas de desempeño sobre los dispositivos Tablet Acer Iconia A500 y Smartphone HTC Wildfire.

Las pruebas realizadas inciden sobre el procesamiento multimedia en Carga de CPU sobre los distintos escenarios planteados, y se pudo observar que para OMAP, el manejo de multimedia influye directamente sobre el desempeño del CPU. De hecho, se nota una gran mejora cuando el procesamiento multimedia es manejado por su procesador dedicado DSP. Cuando el análisis consideró otras plataformas, como es el caso de la Tablet Acer Iconia, la cual incluye una de las más nuevas tecnologías que existen actualmente. En este caso fue importante observar que su desempeño no estuvo lejos del manejado por OMAP3530 en su configuración DSP, lo que indica que OMAP3530 es una gran opción para desarrollo. Además, se analizó el desempeño gráfico y su incidencia sobre la Carga del CPU y los FPS resultantes. Se pudo observar que en cuanto a consumo de CPU, OMAP es el que menos consume sus recursos. En cuanto FPS, la carga se encontraba sobre el procesador gráfico y como resultado, Acer dio los mejores resultados.

Es necesario aclarar que los resultados presentados en el presente trabajo, son bastante lógicos ya que no es posible comparar las capacidades que tendrían las plataformas evaluadas, porque en cierta manera el tablet y el smartphone son dispositivos de propósito general, mientras que el EVM OMAP3530 utilizado es una plataforma de desarrollo (evaluación) que permite alta configurabilidad, mientras que los otros fueron optimizados ya para un mercado especifico.

7.2. RECOMENDACIONES

Luego de haber trabajado Plataformas SoC OMAP y con Sistemas de Desarrollo Android, es necesario mencionar las siguientes recomendaciones.

Uno de las contribuciones mas importantes y a la vez complicadas fue el estudiar y descubrir como configurar la plataforma, la selección e instalación del kernel y Filesystem. Sin embargo, una vez superada esta etapa el proceso de desarrollo de aplicaciones fluye con mayor rapidez. Una de las razones es el concepto de capas de software, entre las que esta Java. Por esta razon, se recomienda utilizar binarios pre-compilados, ya que es una ahorro de tiempo, el cual puede ser dedicado al diseño y programación de aplicaciones.

Se recomienda utilizar el IDE Eclipse, ya que en este se puede incluir las herramientas exclusivas para Android, ahorrando tiempo y aumentando efectividad en el diseño. Además de dicha herramiento, encontramos las herramientas SDK, las cuales facilitaron el desarrollo de binarios y aportaron con ejemplos.

Al DEEE, se recomienda invertir en tecnología como la analizada en el presente trabajo, por la ventaja que constituyen para el desarrollo dentro del país. Aprovechando el potencial que representa este tipo de plataformas en innumerables aplicaciones. Estas pueden ser utilizadas para crear modelos de negocios que involucren un cambio de perspectiva para la industria ecuatoriana, para ofrecer capacitacion a las industrias, o para incluirse en programas de posgrado o laboratorios de pregrado.

Cabe recalcar que no es necesario invertir mucho capital ya que existen en el mercado plataformas de desarrollo económicas, las cuales incluyen menor número de periféricos que pueden ser utilizados para aplicaciones específicas.

ANEXOS

A.1.

ARCHIVOS DE CONFIGURACIÓN PARA FORMATO DE TARJETA SD

A.1.1. FORMATO TARJETA SD ARM

Modificación al archive ~/Android/ARM/mkmmc-android.sh

```
#!/bin/bash

EXPECTED_ARGS=1
if [ $# == $EXPECTED_ARGS ]
then
      echo "Assuming Default Locations for Prebuilt Images"
      $0  $1  Boot_Images/MLO  Boot_Images/u-boot.bin  Boot_Images/uImage
Boot_Images/boot.scr Filesystem/rootfs* Media_Clips
      exit
fi

if [[ -z $1 || -z $2 || -z $3 || -z $4 ]]
then
      echo "mkmmc-android Usage:"
      echo "      mkmmc-android  <device>  <MLO>  <u-boot.bin>  <uImage>
<boot.scr> <rootfs tar.bz2 > <Optional Media_Clips>"
      echo "      Example: mkmmc-android /dev/sdc MLO u-boot.bin uImage
boot.scr rootfs.tar.bz2 Media_Clips"
      exit
fi

if ! [[ -e $2 ]]
then
      echo "Incorrect MLO location!"
      exit
fi

if ! [[ -e $3 ]]
then
      echo "Incorrect u-boot.bin location!"
      exit
fi

if ! [[ -e $4 ]]
then
      echo "Incorrect uImage location!"
      exit
fi
```

```
if ! [[ -e $5 ]]
then
      echo "Incorrect boot.scr location!"
      exit
fi

if ! [[ -e $6 ]]
then
      echo "Incorrect rootfs location!"
      exit
fi

echo "All data on "$1" now will be destroyed! Continue? [y/n]"
read ans
if ! [ $ans == 'y' ]
then
      exit
fi

echo "[Unmounting all existing partitions on the device ]"

umount $1*

echo "[Partitioning $1...]"

DRIVE=$1
dd if=/dev/zero of=$DRIVE bs=1024 count=1024 &>/dev/null

SIZE=`fdisk -l $DRIVE | grep Disk | awk '{print $5}'`

echo DISK SIZE - $SIZE bytes

CYLINDERS=`echo $SIZE/255/63/512 | bc`

echo CYLINDERS - $CYLINDERS
{
echo ,9,0x0C,*
echo ,$(expr $CYLINDERS / 2),,-
echo ,,0x0C,-
} | sfdisk -D -H 255 -S 63 -C $CYLINDERS $DRIVE &> /dev/null

echo "[Making filesystems...]"

mkfs.vfat -F 32 -n boot "$1"p1 &> /dev/null
mkfs.ext3 -L rootfs "$1"p2 &> /dev/null
mkfs.vfat -F 32 -n data "$1"p3 &> /dev/null

echo "[Copying files...]"

mount "$1"p1 /mnt
cp $2 /mnt/MLO
cp $3 /mnt/u-boot.bin
cp $4 /mnt/uImage
cp $5 /mnt/boot.scr
umount "$1"p1

mount "$1"p2 /mnt
tar jxvf $6 -C /mnt &> /dev/null
chmod 755 /mnt
umount "$1"p2
```

```
if [ "$7" ]
then
      echo "[Copying all clips to data partition]"
      mount "$1"p3 /mnt
      cp -r $7/* /mnt/
      umount "$1"p3
fi

echo "[Done]"
```

A.1.2. FORMATO TARJETA SD ARM-DSP

Modificación al archive ~/Android/ARM/mkmmc-android.sh

```
#!/bin/bash

EXPECTED_ARGS=1
if [ $# == $EXPECTED_ARGS ]
then
      echo "Assuming Default Locations for Prebuilt Images"
      $0  $1  Boot_Images/MLO  Boot_Images/u-boot.bin  Boot_Images/uImage
Boot_Images/boot.scr Filesystem/rootfs* Media_Clips START_HERE
      exit
fi

if [[ -z $1 || -z $2 || -z $3 || -z $4 ]]
then
      echo "mkmmc-android Usage:"
      echo "      mkmmc-android  <device>  <MLO>  <u-boot.bin>  <uImage>
<boot.scr> <rootfs tar.bz2 > <Optional Media_Clips> <Optional START_HERE
folder>"
      echo "      Example: mkmmc-android /dev/sdc MLO u-boot.bin uImage
boot.scr rootfs.tar.bz2 Media_Clips START_HERE"
      exit
fi

if ! [[ -e $2 ]]
then
      echo "Incorrect MLO location!"
      exit
fi

if ! [[ -e $3 ]]
then
      echo "Incorrect u-boot.bin location!"
      exit
fi

if ! [[ -e $4 ]]
then
      echo "Incorrect uImage location!"
      exit
```

```
fi

if ! [[ -e $5 ]]
then
      echo "Incorrect boot.scr location!"
      exit
fi

if ! [[ -e $6 ]]
then
      echo "Incorrect rootfs location!"
      exit
fi

echo "All data on "$1" now will be destroyed! Continue? [y/n]"
read ans
if ! [ $ans == 'y' ]
then
      exit
fi

echo "[Unmounting all existing partitions on the device ]"

umount $1*

echo "[Partitioning $1...]"

DRIVE=$1
dd if=/dev/zero of=$DRIVE bs=1024 count=1024 &>/dev/null

SIZE=`fdisk -l $DRIVE | grep Disk | awk '{print $5}'`

echo DISK SIZE - $SIZE bytes

CYLINDERS=`echo $SIZE/255/63/512 | bc`

echo CYLINDERS - $CYLINDERS
{
echo ,9,0x0C,*
echo ,$(expr $CYLINDERS / 2),,-
echo ,,0x0C,-
} | sfdisk -D -H 255 -S 63 -C $CYLINDERS $DRIVE &> /dev/null

echo "[Making filesystems...]"

mkfs.vfat -F 32 -n boot "$1"p1 &> /dev/null
mkfs.ext3 -L rootfs "$1"p2 &> /dev/null
mkfs.vfat -F 32 -n data "$1"p3 &> /dev/null

echo "[Copying files...]"

mount "$1"p1 /mnt
cp $2 /mnt/MLO
cp $3 /mnt/u-boot.bin
cp $4 /mnt/uImage
cp $5 /mnt/boot.scr
if [ "$8" ]
then
        echo "[Copying START_HERE floder to boot partition]"
        cp -r $8 /mnt/START_HERE
fi
```

```
umount "$1"p1

mount "$1"p2 /mnt
tar jxvf $6 -C /mnt &> /dev/null
chmod 755 /mnt
umount "$1"p2

if [ "$7" ]
then
      echo "[Copying all clips to data partition]"
      mount "$1"p3 /mnt
      cp -r $7/* /mnt/
      umount "$1"p3
fi

echo "[Done]"
```

A.2.

SRO_M/res/layout XML

A.2.1. SRO_M/res/layout/main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:background="@drawable/background"
5      android:layout_width="fill_parent"
6      android:layout_height="fill_parent">
7      <ImageView
8      android:id="@+id/TituloMain"
9      android:src="@drawable/titulo"
10        android:layout_height="wrap_content"
11        android:layout_width="fill_parent"
12        android:gravity="center"
13        android:padding="30px"/>
14     <LinearLayout
15        android:orientation="vertical"
16        android:layout_width="wrap_content"
17        android:layout_height="fill_parent"
18        android:layout_gravity="center">
19        <TableLayout
20           android:id="@+id/tableLayout1"
21           android:layout_width="wrap_content"
22        android:layout_height="fill_parent">
23           <TableRow
24              android:id="@+id/tableRow0"
25              android:layout_height="wrap_content"
26              android:layout_width="fill_parent">
27              <Button
28                 android:id="@+id/boton_Conversor"
29                 android:layout_width="wrap_content"
30                 android:layout_height="wrap_content"
31                 android:background="@drawable/backbuttonconv"
32                 android:gravity="center"/>
33              <Button
34                 android:id="@+id/boton_OpenGl"
35                 android:layout_width="wrap_content"
36                 android:layout_height="wrap_content"
37                 android:background="@drawable/backbutton3d"
38                 android:layout_marginLeft="20px"
39                 android:layout_marginRight="20px"
40                 android:gravity="center"/>
41              <Button
42                 android:id="@+id/boton_Video"
```

```
43              android:layout_width="wrap_content"
44              android:layout_height="wrap_content"
45              android:background="@drawable/backbuttonvideo"
46              android:gravity="center"/>
47          </TableRow>
48          <TableRow
49              android:id="@+id/tableRow1"
50              android:layout_width="wrap_content"
51              android:layout_height="wrap_content"
52              android:paddingBottom="30px">
53              <TextView
54                  android:layout_width="wrap_content"
55                  android:layout_height="wrap_content"
56                  android:textColor="@color/texto_main"
57                  android:text="@string/label_t"
58                  android:textSize="25sp"
59                  android:gravity="center"/>
60              <TextView
61                  android:layout_width="wrap_content"
62                  android:layout_height="wrap_content"
63                  android:textColor="@color/texto_main"
64                  android:text="@string/label_d"
65                  android:textSize="25sp"
66                  android:gravity="center"
67                  android:paddingRight="20px"
68                  android:paddingLeft="20px"/>
69              <TextView
70                  android:layout_width="wrap_content"
71                  android:layout_height="wrap_content"
72                  android:textColor="@color/texto_main"
73                  android:text="@string/label_v"
74                  android:textSize="23sp"
75                  android:gravity="center"/>
76          </TableRow>
77          <TableRow
78              android:id="@+id/tableRow2"
79              android:layout_width="wrap_content"
80              android:layout_height="wrap_content">
81              <Button
82                  android:id="@+id/boton_Audio"
83                  android:layout_width="wrap_content"
84                  android:layout_height="wrap_content"
85                  android:background="@drawable/backbuttonaudio"
86                  android:gravity="center"/>
87              <Button
88                  android:id="@+id/boton_Sintesis"
89                  android:layout_width="wrap_content"
90                  android:layout_height="wrap_content"
91                  android:background="@drawable/backbuttonabout"
92                  android:layout_marginLeft="20px"
93                  android:layout_marginRight="20px"
94                  android:gravity="center"/>
95          </TableRow>
96          <TableRow
97              android:id="@+id/tableRow3"
98              android:layout_width="wrap_content"
99              android:layout_height="wrap_content">
```

```
100             <TextView
101                android:layout_width="wrap_content"
102                android:layout_height="wrap_content"
103                android:textColor="@color/texto_main"
104                android:text="@string/label_a"
105                android:textSize="25sp"
106                android:gravity="center"/>
107             <TextView
108                android:layout_width="wrap_content"
109                android:layout_height="wrap_content"
110                android:textColor="@color/texto_main"
111                android:text="@string/label_r"
112                android:textSize="25sp"
113                android:gravity="center"/>
114          </TableRow>
115       </TableLayout>
116    </LinearLayout>
117 </LinearLayout>
```

A.2.2. SRO_M/res/layout/sintesis.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent">
6     <LinearLayout
7        android:id="@+id/Layout1"
8        android:layout_width="fill_parent"
9     android:layout_height="fill_parent"
10       android:orientation="vertical">
11       <TextView
12          android:id="@+id/resumen_content"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:text="@string/texto_resumen" />
16       <ImageView
17          android:layout_width="fill_parent"
18          android:layout_height="wrap_content"
19          android:src="@drawable/arquitectura_hw"
20          android:layout_gravity="center" />
21       <TextView
22          android:id="@+id/resumen_content"
23          android:layout_width="wrap_content"
24          android:layout_height="wrap_content"
25          android:text="@string/texto_resumenSW" />
26       <ImageView
27          android:layout_width="wrap_content"
28          android:layout_height="wrap_content"
29          android:src="@drawable/arquitectura_sw"
```

```
            android:layout_gravity="center"/>
        </LinearLayout>
</ScrollView>
```

A.2.3. SRO_M/res/layout/conversor.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/backgroundtemp"
    android:orientation="vertical"
    android:padding="40px">
    <!-- Titulo -->
    <TextView
        android:text="@string/app_conv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="28sp"
        android:textColor="@color/texto_conv"
        android:gravity="center"/>
    <TableLayout
        android:id="@+id/tableLayoutConversor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:padding="30px" >
        <TableRow
            android:id="@+id/tableRow1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:padding="10px" >
        <!-- Editor de Temp -->
            <EditText
            android:id="@+id/EditTemp"
            android:text="@string/edit_temp"
            android:inputType="numberDecimal|numberSigned"
            android:layout_width="250px"
            android:layout_height="wrap_content"
            android:textSize="24.5sp" />
        <!-- Etiqueta de Temp inicial -->
        <TextView
            android:id="@+id/deGrados"
                android:text="@string/f"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
                android:textSize="24.5sp"
                android:textColor="@color/texto_conv"  />
        </TableRow>
```

```
44        <TableRow
45           android:id="@+id/tableRow2"
46           android:layout_width="fill_parent"
47           android:layout_height="wrap_content"
48           android:padding="10px">
49           <!-- Respuesta de Temp -->
50           <TextView
51           android:id="@+id/ViewGrados"
52              android:text="@string/resp_temp"
53           android:layout_width="250px"
54           android:layout_height="fill_parent"
55              android:textSize="24.5sp"
56              android:textColor="@color/texto_conv"  />
57        <!-- Etiqueta de Temp final -->
58           <TextView
59           android:id="@+id/aGrados"
60              android:text="@string/c"
61           android:layout_width="wrap_content"
62           android:layout_height="fill_parent"
63              android:textSize="24.5sp"
64              android:textColor="@color/texto_conv"  />
65        </TableRow>
66     </TableLayout>
67      <LinearLayout
68        android:orientation="horizontal"
69        android:layout_height="wrap_content"
70        android:layout_width="fill_parent"
71        android:layout_gravity="center" >
72        <TextView
73              android:text="@string/unid"
74           android:layout_width="wrap_content"
75           android:layout_height="wrap_content"
76              android:textSize="24.5sp"
77              android:textColor="@color/texto_conv"  />
78           <!-- Radiogroup Seleccion temp -->
79           <RadioGroup
80              android:layout_height="wrap_content"
81           android:id="@+id/radioGroup1"
82           android:layout_width="match_parent">
83        <RadioButton
84           android:layout_width="fill_parent"
85           android:id="@+id/RadioC"
86           android:layout_height="wrap_content"
87                 android:checked="true"
88                 android:text="@string/celsius"
89                 android:textColor="@color/texto_conv"
90                 android:textSize="24.5sp"/>
91        <RadioButton
92           android:layout_width="fill_parent"
93           android:id="@+id/RadioF"
94           android:layout_height="wrap_content"
95           android:text="@string/fahrenheit"
```

```
            android:textColor="@color/texto_conv"
            android:textSize="24.5sp"/>
        </RadioGroup>
    </LinearLayout>
</LinearLayout>
```

A.2.4. SRO_M/res/layout/maingl.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center|left"
    android:padding="15dp"
    android:textSize="28sp"
    android:background="@color/back_maingl1">
</TextView>
```

A.2.5. SRO_M/res/layout/video.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <VideoView
        android:id="@+id/video"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center" />
</FrameLayout>
```

A.2.6. SRO_M/res/layout/nombre.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:background="@drawable/background_audio_nomb"
    android:padding="35px">
```

```
    <TextView
       android:id="@+id/titulo_audio"
    android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="@string/titulo_nombre"
       android:textColor="@color/texto_audio3"
       android:textSize="25sp"
       android:gravity="center"/>
    <LinearLayout
    android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:padding="30px">
      <EditText
       android:id="@+id/nombreAudio"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="@string/ext_nombre"
       android:singleLine="true"

    </LinearLayout>
    <Button
    android:id="@+id/botonNombre"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/boton_nombre"
    android:onClick="myClickHandler"
       android:textSize="21.5sp"
    android:layout_gravity="center"
    android:padding="15px"/>
</LinearLayout>
```

A.2.7. SRO_M/res/layout/audio.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@drawable/backgroundaudio"
   android:padding="30px"
   android:gravity="top|center_horizontal">
   <!-- Titulo -->
   <TextView
      android:id="@+id/titulo_audio"
      android:text="@string/sound_tesis"
   android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
15          android:textColor="@color/texto_audio1"
16          android:gravity="center"
17      android:textSize="35sp"
18      android:padding="20px"/>
19      <!-- Info Track -->
20      <LinearLayout
21          android:id="@+id/layoutInfoTrack"
22          android:orientation="vertical"
23          android:padding="15px"
24      android:gravity="center"
25          android:layout_width="fill_parent"
26          android:layout_height="wrap_content">
27      <!-- NOMBRE -->
28          <TextView
29              android:text="Archivo: "
30          android:layout_width="fill_parent"
31              android:layout_height="wrap_content"
32              android:textColor="@color/texto_audio1"
33              android:gravity="left"
34          android:textSize="25sp"/>
35          <TextView
36          android:text="TextView"
37          android:id="@+id/TextNombre"
38          android:layout_width="fill_parent"
39          android:layout_height="wrap_content"
40          android:textSize="22sp"
41          android:textColor="@color/texto_audio2"
42          android:paddingLeft="80px"/>
43          <!-- DURACION -->
44       <TextView
45              android:text="Duración: "
46          android:layout_width="fill_parent"
47              android:layout_height="wrap_content"
48              android:textColor="@color/texto_audio1"
49              android:gravity="left"
50          android:textSize="25sp"
51          android:paddingTop="10px"/>
52      <Chronometer
53          android:text="Chronometer"
54          android:id="@+id/ChronoRec"
55          android:layout_width="fill_parent"
56          android:layout_height="wrap_content"
57          android:textSize="22sp"
58          android:textColor="@color/texto_audio2"
59          android:paddingLeft="80px"/>
60      </LinearLayout>
61      <!-- Mensaje estado -->
62       <TextView
63          android:id="@+id/Estado"
64          android:text="@string/estado_inicial"
65      android:layout_width="fill_parent"
66          android:layout_height="wrap_content"
```

```
67      android:textColor="@color/texto_audio3"
68    android:textSize="23sp"
69      android:gravity="center"
70    android:padding="10px"/>
71    <!-- Botones rec play stop -->
72     <LinearLayout
73      android:id="@+id/LayoutBotones"
74    android:orientation="horizontal"
75    android:layout_width="wrap_content"
76    android:layout_height="wrap_content">
77    <!-- Record -->
78    <RelativeLayout
79          android:layout_width="wrap_content"
80          android:layout_height="wrap_content"
81          android:padding="10px">
82       <ImageButton
83             android:id="@+id/b_rec"
84             android:layout_width="wrap_content"
85             android:layout_height="wrap_content"
86             android:src="@drawable/record"/>
87          <ImageButton
88             android:id="@+id/b_rec_on"
89             android:layout_width="wrap_content"
90             android:layout_height="wrap_content"
91             android:src="@drawable/record_on"
92             android:layout_above="@+id/b_rec"/>
93       </RelativeLayout>
94       <!-- Play -->
95       <RelativeLayout
96          android:layout_width="wrap_content"
97          android:layout_height="wrap_content"
98          android:padding="10px">
99          <ImageButton
100            android:id="@+id/b_play_on"
101            android:layout_width="wrap_content"
102            android:layout_height="wrap_content"
103            android:src="@drawable/play_on"/>
104         <ImageButton
105            android:id="@+id/b_play"
106            android:layout_width="wrap_content"
107            android:layout_height="wrap_content"
108            android:src="@drawable/play"
109            android:layout_above="@+id/b_play_on"/>
110      </RelativeLayout>
111   <!-- Stop -->
112      <RelativeLayout
113         android:layout_width="wrap_content"
114         android:layout_height="wrap_content"
115         android:padding="10px">
116         <ImageButton
117            android:id="@+id/b_stop_on"
118            android:layout_width="wrap_content"
```

```
119             android:layout_height="wrap_content"
120             android:src="@drawable/stop_on"/>
121          <ImageButton
122             android:id="@+id/b_stop"
123             android:layout_width="wrap_content"
124             android:layout_height="wrap_content"
125             android:src="@drawable/stop"
126             android:layout_above="@+id/b_stop_on"/>
127       </RelativeLayout>
128    </LinearLayout >
129    <!-- barra de progreso -->
130    <TableLayout
131    android:layout_width="fill_parent"
132    android:id="@+id/layoutBarraProgreso"
133    android:layout_height="wrap_content"
134    android:stretchColumns="1"
135    android:paddingTop="10px">
136    <TableRow
137       android:id="@+id/tableRow1"
138       android:layout_width="fill_parent"
139       android:layout_height="wrap_content">
140       <TextView
141          android:id="@+id/PlayInicio"
142          android:text="@string/play_inicio"
143          android:layout_width="wrap_content"
144          android:layout_height="wrap_content"
145          android:textSize="25sp"
146          android:textColor="@color/texto_audio3"/>
147       <Chronometer
148          android:id="@+id/ChronoPlay"
149          android:layout_width="wrap_content"
150          android:layout_height="wrap_content"
151          android:textSize="25sp"
152          android:textColor="@color/texto_audio3"
153          android:gravity="right"/>
154    </TableRow>
155    </TableLayout>
156    <ProgressBar android:id="@+id/BarraProgreso"
157       style="@android:style/Widget.ProgressBar.Horizontal"
158       android:layout_width="fill_parent"
159       android:layout_height="wrap_content"
160       android:padding="10px"/>
161 </LinearLayout>
```

A.3.

SRO_M/res/values XML

A.3.1. SRO_M/res/values/color.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <!-- Colores Menu -->
4      <color name="back_main1">#2B70B5</color>
5      <color name="back_main2">#FDFDFD</color>
6      <color name="texto_main">#454B4E</color>
7
8      <!-- Colores Conversor -->
9      <color name="back_conv1">#DCDDA4</color>
10     <color name="back_conv2">#558063</color>
11     <color name="back_conv3">#002F3A</color>
12     <color name="texto_conv">#FFFFBA</color>
13
14     <!-- Colores Cubo3D -->
15     <color name="back_maingl1">#374F67</color>
16
17     <!-- Colores Audio -->
18     <color name="back_audio1">#F8EBE8</color>
19     <color name="back_audio2">#012529</color>
20     <color name="texto_audio1">#394D4A</color>
21     <color name="texto_audio2">#050505</color>
22     <color name="texto_audio3">#67C1F4</color>
23 </resources>
```

A.3.2. SRO_M/res/values/array.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <string-array name="ArrayCubo">
4          <item>@string/uno</item>
5          <item>@string/dos</item>
6          <item>@string/tres</item>
7          <item>@string/cuatro</item>
8          <item>@string/cinco</item>
9      </string-array>
10     <string-array name="ArrayAudio">
11         <item>@string/sound_android</item>
```

```
12          <item>@string/sound_tesis</item>
13      </string-array>
14  </resources>
```

A.3.3. SRO_M/res/values/string.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3       <!-- Strings Menu Principal -->
4       <string name="app_name">SRO-M</string>
5       <string name="label_t">Conversor</string>
6       <string name="label_d">Cubo 3D</string>
7       <string name="label_v">Video Player</string>
8       <string name="label_a">Audio Rec</string>
9       <string name="label_r">About</string>
10      <!-- String Menu Conversor -->
11      <string name="label_1"> Conversor de Unidades</string>
12      <string name="app_conv">Conversor \nFahrenheit - Celsius</string>
13      <string name="celsius">Celsius</string>
14      <string name="fahrenheit">Fahrenheit</string>
15      <string name="c">  °C</string>
16      <string name="f">  °F</string>
17      <string name="unid">Unidades  </string>
18      <string name="edit_temp">0</string>
19      <string name="resp_temp">"-17.777779"</string>
20      <!-- Strings Menu OpenGL -->
21      <string name="label_2"> 3D Graphics Rendering (OpenGL)</string>
22      <string name="uno">  Cubo sin sombra</string>
23      <string name="dos">  Cubo Iluminado</string>
24      <string name="tres">  Cubo Rotatorio</string>
25      <string name="cuatro">  Cubo Texturado</string>
26      <string name="cinco">  Cubo Transparente</string>
27      <!-- Strings Video -->
28      <string name="label_3"> Video Player</string>
29      <!-- Strings Menu Audio -->
30      <string name="label_4"> Audio Recorder/Playback</string>
31      <string name="sound_android">SoundRecorder Android</string>
32      <string name="sound_tesis">Audio Recorder/Player</string>
33      <string name="play_inicio">00:00</string>
34      <string name="estado_inicial">...</string>
35      <!-- Strings Menu Nombre -->
36      <string name="titulo_nombre">Ingrese un nombre y ubicación al archivo: </string>
37      <string name="ext_nombre">.3gp</string>
38      <string name="boton_nombre">Guardar</string>
39      <!-- Strings Menu Resumen -->
40      <string name="titulo_resumen">About</string>
        <string name="texto_resumen">Aplicaciones SRO-M \n\n El Sistema de Referencia OMAP-
41  Multimedia (SRO-M), incluye un conjunto de aplicaciones:
```

\n\n 1. Conversor: es el primer ejemplo práctico de una actividad sencilla para transformar grados
42 Fahrenheit a Celsius y viceversa.
\n\n 2. Cubo 3D: aplicación que utiliza las librerías OpenGL de gráficas 3D que es compatible con el
43 subsistema acelerador gráfico de la plataforma.
\n\n 3. Video Player: es un Reproductor de Video que muestra un video incluido en el sistema y
44 ubicado en la tarjeta SD.
\n\n 4. Audio Rec: incluye dos programas destinados a la grabación de audio, el primero \"SoundRecorder Android\" es el método más sencillo para grabación de audio, es una simple llamada al programa predeterminado de Android para grabación de audio. El segundo es un programa \"Audio Recorder/Playback\", mediante este se puede indicar el destino del archivo
45 creado.
46 \n\n 5. About: despliega este mensaje para conocer más acerca del proyecto.
\n\n SRO-M es una compilación de aplicaciones básicas sencillas que utilizan las prestaciones más representativas de la plataforma SoC (System-on-Chip) Texas Instruments OMAP(Open Multimedia Application Platform) 3530 como es el uso de OpenGL, Multimedia (Audio-Video) y creación y
47 lectura de archivos.
48 \n\n\n\n SoC OMAP3530
\n\n System-on-Chip representa la mayor evolución en el diseño de circuitos integrados, y fue posible gracias al avance de la tecnología, que permitió la integración de la mayoría de componentes y subsistemas electrónicos en un solo CI, su diseño se encuentra integrado como un solo sistema. SoC implementa la reutilización de bloques de propiedad intelectual, además basa su diseño en plataformas, las cuales son muy utilizadas en la actualidad pues permiten reunir un conjunto de funcionalidades en arquitecturas comunes, las cuales pueden soportar una variedad de
49 aplicaciones.
\n\n El desarrollo de este proyecto se basa en el estudio de una plataforma SoC desarrollada por Texas Instruments llamada OMAP, la cual es una plataforma de aplicaciones que permite que las funciones multimedia sean incluidas en Smartphones 2.5G - 3.5G y PDAs. Estas funciones incluyen video, mensajería, navegación web, video conferencia, juegos, comercio móvil y otras tareas computacionalmente exigentes, es utilizada en el mercado por muchos teléfonos móviles que
50 soportan diversos Sistemas Operativos.
\n\n Se optó por Android como sistema operativo ya que es libre, tiene gran acogida para dispositivos móviles en la actualidad y además debido a la gran abstracción que representa en el desarrollo de aplicaciones, una de sus mayores ventajas es que puede ser desarrollado en varios
51 lenguajes de desarrollo como Java.
\n\n Exactamente se estudia la plataforma OMAP3530, sobre la plataforma de aplicaciones EVM35x de Mistral Solutions, que es una plataforma de pruebas que consta de los periféricos necesarios para el desarrollo de aplicaciones. La arquitectura de HW del OMAP3530 se encuentra conformada por un subsistema de procesamiento que integrados procesadores uno de Propósito General GPP ARM® CortexTM-A8 y uno de Señales Digitales DSP TMS320DMC64X+ VLIWX, un subsistema de Memoria, sistema de interconexiones, subsistema de aceleración gráfica POWERVR SGX, Procesador
52 multimedia ISP, gestión de energía y periféricos. \n\n
53 </string>
54 <string name="texto_resumenSW">\n\n
La arquitectura de Software de la plataforma OMAP3530 describe un sistema de software que vincula aplicaciones del GPP a componentes del DSP como codecs de video y audio. Esta arquitectura permite que el procesador ARM CortexTM-A8 utilice funciones de TMS320DMC64X+ VLIWX remotamente, y para el desarrollador la percepción de este enlace es transparente, es decir
55 que GPP es visto como el maestro del sistema.\n\n
56 </string>
57 </resources>

A.4.

SRO_M/res/drawable-hdpi XML

A.4.1. SRO_M/res/drawable-hdpi/background.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <shape xmlns:android="http://schemas.android.com/apk/res/android">
3      <gradient
4          android:angle="90"
5          android:startColor="@color/back_main1"
6          android:centerColor="@color/back_main1"
7          android:endColor="@color/back_main2"
8          android:type="linear"
9      />
10 </shape>
```

A.4.2. SRO_M/res/drawable-hdpi/backgroundtemp.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <shape
3   xmlns:android="http://schemas.android.com/apk/res/android">
4     <gradient
5          android:angle="90"
6          android:startColor="@color/back_conv1"
7          android:centerColor="@color/back_conv2"
8          android:endColor="@color/back_conv3"
9          android:type="linear"/>
10 </shape>
```

A.4.3. SRO_M/res/drawable-hdpi/backgroundaudio.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <shape xmlns:android="http://schemas.android.com/apk/res/android">
3      <gradient
4          android:startColor="@color/back_audio1"
5          android:endColor="@color/back_audio2"
```

```
        android:type="radial"
        android:gradientRadius="360"
        android:centerX="0.25"
        android:centerY="0.22"/>
</shape>
```

A.4.4. SRO_M/res/drawable-hdpi/backbuttonconv.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
   <!-- focused -->
   <item android:state_focused="true"
     android:state_pressed="false"
     android:drawable="@drawable/tempinkfoc" />
   <!-- presseed -->
   <item android:state_focused="true"
     android:state_pressed="true"
     android:drawable="@drawable/tempinkfoc" />
   <item android:state_focused="false"
     android:state_pressed="true"
     android:drawable="@drawable/tempinkfoc" />
   <!-- normal -->
   <item android:drawable="@drawable/tempink2200" />
</selector>
```

A.4.5. SRO_M/res/drawable-hdpi/backbutton3d.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
   <!-- focused -->
   <item android:state_focused="true"
     android:state_pressed="false"
     android:drawable="@drawable/open3dinkfoc" />
   <!-- presseed -->
   <item android:state_focused="true"
     android:state_pressed="true"
     android:drawable="@drawable/open3dinkfoc" />
   <item android:state_focused="false"
     android:state_pressed="true"
     android:drawable="@drawable/open3dinkfoc" />
   <!-- normal -->
   <item android:drawable="@drawable/open3dink200" />
</selector>
```

A.4.6. SRO_M/res/drawable-hdpi/backbuttonvideo.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
   <selector
2  xmlns:android="http://schemas.android.com/apk/res/android">
3     <!-- focused -->
4     <item android:state_focused="true"
5       android:state_pressed="false"
6       android:drawable="@drawable/videoinkfoc" />
7     <!-- presseed -->
8     <item android:state_focused="true"
9       android:state_pressed="true"
10      android:drawable="@drawable/videoinkfoc" />
11    <item android:state_focused="false"
12      android:state_pressed="true"
13      android:drawable="@drawable/videoinkfoc" />
14    <!-- normal -->
15    <item android:drawable="@drawable/videoink2200" />
16 </selector>
```

A.4.7. SRO_M/res/drawable-hdpi/backbuttonaudio.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <selector xmlns:android="http://schemas.android.com/apk/res/android">
3     <!-- focused -->
4     <item android:state_focused="true"
5       android:state_pressed="false"
6       android:drawable="@drawable/audioinkfoc" />
7     <!-- presseed -->
8     <item android:state_focused="true"
9       android:state_pressed="true"
10      android:drawable="@drawable/audioinkfoc" />
11    <item android:state_focused="false"
12      android:state_pressed="true"
13      android:drawable="@drawable/audioinkfoc" />
14    <!-- normal -->
15    <item android:drawable="@drawable/audioink3200" />
16 </selector>
```

A.4.8. SRO_M/res/drawable-hdpi/backbuttonabout.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <selector xmlns:android="http://schemas.android.com/apk/res/android">
3      <!-- focused -->
4      <item android:state_focused="true"
5        android:state_pressed="false"
6        android:drawable="@drawable/aboutinkfoc" />
7      <!-- presseed -->
8      <item android:state_focused="true"
9        android:state_pressed="true"
10       android:drawable="@drawable/aboutinkfoc" />
11     <item android:state_focused="false"
12       android:state_pressed="true"
13       android:drawable="@drawable/aboutinkfoc" />
14     <!-- normal -->
15     <item android:drawable="@drawable/aboutink20" />
16 </selector>
```

A.5.

SRO_M/AndroidManifest.xml

A.5.1. SRO_M/AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="apps.tesis.omap"
4      android:versionCode="1"
5      android:versionName="1.0">
6    <uses-sdk android:minSdkVersion="8" />
7    <uses-permission android:name="android.permission.RECORD_AUDIO"></uses-permission>
8
9    <application android:icon="@drawable/mistral_omap" android:label="@string/app_name">
10     <activity android:name=".SRO_MActivity"
11           android:label="@string/app_name">
12       <intent-filter>
13         <action android:name="android.intent.action.MAIN" />
14         <category android:name="android.intent.category.LAUNCHER" />
15       </intent-filter>
16     </activity>
17
18     <!-- Activity de clase Sintesis -->
19      <activity android:name=".Sintesis.Sintesis"
20            android:label="@string/label_r"
21            android:theme="@android:style/Theme.Dialog">
22        <intent-filter>
23          <action android:name="android.intent.action.MAIN" />
24          <category android:name="android.intent.category.SAMPLE_CODE" />
25        </intent-filter>
26     </activity>
27
28     <!-- Activity de clase Conversor -->
29     <activity android:name=".Conversor.ConversorTemperatura"
30       android:label="@string/label_1">
31   <intent-filter>
32           <action android:name="android.intent.action.MAIN" />
33           <category android:name="android.intent.category.SAMPLE_CODE" />
34       </intent-filter>
35     </activity>
36
37     <!-- Activity de clase OpenGL -->
38     <activity android:name=".OpenGL.mainOpenGl"
39           android:label="@string/label_2"
```

```
40              android:theme="@android:style/Theme.Dialog">
41          <intent-filter>
42              <action android:name="android.intent.action.MAIN" />
43              <category android:name="android.intent.category.SAMPLE_CODE" />
44          </intent-filter>
45      </activity>
46
47      <activity android:name=".OpenGL.OpenGL"
48        android:label="@string/label_2">
49          <intent-filter>
50              <action android:name="android.intent.action.MAIN" />
51              <category android:name="android.intent.category.SAMPLE_CODE" />
52          </intent-filter>
53       </activity>
54
55       <!-- Activity de clase Reproductor de Video -->
56      <activity android:name=".Video.RepVideo"
57              android:label="@string/label_3">
58          <intent-filter>
59              <action android:name="android.intent.action.MAIN" />
60              <category android:name="android.intent.category.SAMPLE_CODE" />
61          </intent-filter>
62      </activity>
63
64      <!-- Activity de clase Grabador de Audio -->
65      <activity android:name=".Audio.Nombre"
66              android:label="@string/sound_tesis"
67              android:theme="@android:style/Theme.Dialog">
68          <intent-filter>
69              <action android:name="android.intent.action.MAIN" />
70              <category android:name="android.intent.category.SAMPLE_CODE" />
71          </intent-filter>
72      </activity>
73      <activity android:name=".Audio.AudioRec"
74              android:label="@string/sound_tesis">
75          <intent-filter>
76              <action android:name="android.intent.action.MAIN" />
77              <category android:name="android.intent.category.SAMPLE_CODE" />
78          </intent-filter>
79      </activity>
80    </application>
81  </manifest>
```

A.6.

SRO_M/src/apps/tesis/omap Java

A.6.1. SRO_M/src/apps/tesis/omap/SRO_MActivity.java

```
1  package apps.tesis.omap;
2
3  import android.app.Activity;
4  import android.app.AlertDialog;
5  import android.content.DialogInterface;
6  import android.content.Intent;
7  import android.os.Bundle;
8  import android.provider.MediaStore;
9  import android.view.View;
10 import android.view.View.OnClickListener;
11 import apps.tesis.omap.Sintesis.Sintesis;
12 import apps.tesis.omap.Conversor.ConversorTemperatura;
13 import apps.tesis.omap.OpenGL.mainOpenGl;
14 import apps.tesis.omap.Video.RepVideo;
15 import apps.tesis.omap.Audio.Nombre;
16
17 public class SRO_MActivity extends Activity implements OnClickListener {
18         /** Called when the activity is first created. */
19
20         @Override
21         public void onCreate(Bundle savedInstanceState) {
22                 super.onCreate(savedInstanceState);
23                 setContentView(R.layout.main);
24                 // Click listeners para los botones
25                 View botonHola = findViewById(R.id.boton_Conversor);
26                 botonHola.setOnClickListener(this);
27                 View botonOpenGl = findViewById(R.id.boton_OpenGl);
28                 botonOpenGl.setOnClickListener(this);
29                 View botonVideo = findViewById(R.id.boton_Video);
30                 botonVideo.setOnClickListener(this);
31                 View botonAudio = findViewById(R.id.boton_Audio);
32                 botonAudio.setOnClickListener(this);
33                 View botonResumen = findViewById(R.id.boton_Resumen);
34                 botonResumen.setOnClickListener(this);
35         }
36
37         @Override
```

```
38      public void onClick(View arg0) {
39          // TODO Auto-generated method stub
40          switch (arg0.getId()) {
41          // Resumen de programas
42          case R.id.boton_Resumen:
43              Intent i = new Intent(this, Sintesis.class);
44              startActivity(i);
45              break;
46          // Resumen de programas
47          case R.id.boton_Conversor:
48              Intent m = new Intent(this, ConversorTemperatura.class);
49              startActivity(m);
50              break;
51          // Aplicación OpenGL
52          case R.id.boton_OpenGl:
53              Intent j = new Intent(this, mainOpenGl.class);
54              startActivity(j);
55              break;
56          // Aplicación Reproductor de Video
57          case R.id.boton_Video:
58              Intent l = new Intent(this, RepVideo.class);
59              startActivity(l);
60              break;
61          // Aplicación Grabador de Audio
62          case R.id.boton_Audio:
63              SelecAudio();
64              break;
65          }
66      }
67
68      // Función Selección de Aplicación Audio
69      private void SelecAudio() {
70          new AlertDialog.Builder(this).setTitle(R.string.label_a).setItems(
                    R.array.ArrayAudio, new
71                  DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialoginterface, int
72                      i) {
73                              Audio(i);
74                      }
75                  }).show();
76      }
77
78      // Seleccion ejecución programas de Audio
79      private void Audio(int i) {
80          if (i == 0) {
81              // Ejecuta aplicación SoundRecorder propia de Android
82              Intent intent = new Intent(
83                      MediaStore.Audio.Media.RECORD_SOUND_ACTION);
84              startActivityForResult(intent, 1);
85          } else {
86              // Aplicación para Grabación de Audio
87              Intent b = new Intent(this, Nombre.class);
```

```
                    startActivity(b);
                }
            }
        }
```

A.6.2. SRO_M/src/apps/tesis/omap/Sintesis/Sintesis.java

```
package apps.tesis.omap.Sintesis;

import android.app.Activity;
import android.os.Bundle;
import apps.tesis.omap.R;

public class Sintesis extends Activity{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sintesis);
    }
}
```

A.6.3. SRO_M/src/apps/tesis/omap/Conversor/ConversorTemperatura.java

```
package apps.tesis.omap.Conversor;

import android.app.Activity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.RadioGroup.OnCheckedChangeListener;
import apps.tesis.omap.R;

public class ConversorTemperatura extends Activity {
    /** Called when the activity is first created. */
    float Valor;
    EditText text;
    TextView respuesta;
    TextView deGrad;
    TextView aGrad;
    RadioGroup grados;
    RadioButton celsiusButton;
    RadioButton fahrenheitButton;

```

```
25      @Override
26      public void onCreate(Bundle savedInstanceState) {
27          super.onCreate(savedInstanceState);
28          setContentView(R.layout.conversor);
29
30          text = (EditText) findViewById(R.id.EditTemp);
31          respuesta = (TextView) findViewById(R.id.ViewGrados);
32          celsiusButton = (RadioButton) findViewById(R.id.RadioC);
33          fahrenheitButton = (RadioButton) findViewById(R.id.RadioF);
34          grados = (RadioGroup) findViewById(R.id.radioGroup1);
35          deGrad = (TextView) findViewById(R.id.deGrados);
36          aGrad = (TextView) findViewById(R.id.aGrados);
37
38          // Listener RadioButton
            grados.setOnCheckedChangeListener(new OnCheckedChangeListener()
39          {
40              @Override
41              public void onCheckedChanged(RadioGroup arg0, int arg1) {
42                  // TODO Auto-generated method stub
43                  Conversor();
44              }
45          });
46          // Listener EditText
47          text.addTextChangedListener(new TextWatcher() {
48              @Override
49              public void afterTextChanged(Editable arg0) {
50                  // TODO Auto-generated method stub
51                  Conversor();
52              }
53              @Override
54              public void beforeTextChanged(CharSequence arg0, int arg1,
55                      int arg2, int arg3) {
56                  // TODO Auto-generated method stub
57              }
58              @Override
59              public void onTextChanged(CharSequence arg0, int arg1, int arg2,
60                      int arg3) {
61                  // TODO Auto-generated method stub
62              }
63          });
64      }
65      // Conversor de Temperatura
66      public void Conversor() {
67          if (text.getText().length() == 0) {
68              respuesta.setText("0");
69              text.setText("0");
70              return;
71          }
72          // Transformar a flotante numero ingresado
73          Valor = Float.parseFloat(text.getText().toString());
74          // Calculos
75          if (celsiusButton.isChecked()) {
```

```
76                  respuesta.setText(String.valueOf(FahrenheitACelsius(Valor)));
77                  deGrad.setText(R.string.f);
78                  aGrad.setText(R.string.c);
79              } else {
80                  respuesta.setText(String.valueOf(CelsiusAFahrenheit(Valor)));
81                  deGrad.setText(R.string.c);
82                  aGrad.setText(R.string.f);
83              }
84          }
85          // A celsius
86          private float FahrenheitACelsius(float fahrenheit) {
87              return ((fahrenheit - 32) * 5 / 9);
88          }
89          // A fahrenheit
90          private float CelsiusAFahrenheit(float celsius) {
91              return ((celsius * 9) / 5) + 32;
92          }
93      }
```

A.6.4. SRO_M/src/apps/tesis/omap/OpenGL/mainOpenGL.java

```
1   package apps.tesis.omap.OpenGL;
2
3   import android.app.ListActivity;
4   import android.content.Context;
5   import android.content.Intent;
6   import android.os.Bundle;
7   import android.view.View;
8   import android.widget.AdapterView;
9   import android.widget.ArrayAdapter;
10  import android.widget.ListView;
11  import android.widget.TextView;
12  import android.widget.Toast;
13  import android.widget.AdapterView.OnItemClickListener;
14  import apps.tesis.omap.R;
15
16  public class mainOpenGl extends ListActivity {
17      Context context = this;
18
19      public void onCreate(Bundle savedInstanceState) {
20          super.onCreate(savedInstanceState);
21
22          // View de un Menu tipo Lista
23          String[] lista = getResources().getStringArray(R.array.ArrayCubo);
24          setListAdapter(new ArrayAdapter<String>(this,
25                  R.layout.maingl, lista));
26          ListView lv = getListView();
27          lv.setTextFilterEnabled(true);
28
29          // Listener Lista
```

```
30          lv.setOnItemClickListener(new OnItemClickListener() {
31              public void onItemClick(AdapterView<?> parent, View view,
32                      int posicion, long id) {
33                  // Cuando se selecciona se muestra el título
34                  Toast.makeText(getApplicationContext(),
35
36
37                  try {
38                      // Envia a Intent OpenGL cual item fue seleccionado
39                      Intent intent = new Intent(context, OpenGL.class);
40                      Bundle bundle = new Bundle();
41                      bundle.putInt("opcion", posicion);
42                      intent.putExtras(bundle);
43                      startActivity(intent);
44                  } catch (Exception e) {
45                      System.out.println("Posición: " + posicion);
46                  }
47              }
48          });
49      }
50  }
```

A.6.5. SRO_M/src/apps/tesis/omap/OpenGL/OpenGL.java

```
1   package apps.tesis.omap.OpenGL;
2
3   import android.app.Activity;
4   import android.os.Bundle;
5
6   public class OpenGL extends Activity {
7       GLView view;
8       @Override
        public void onCreate(Bundle savedInstanceState)
9       {
10          super.onCreate(savedInstanceState);
11          // Obtiene extras del Menu Lista
12          Bundle extras = getIntent().getExtras();
13          int estado = extras.getInt("opcion");
14          view = new GLView(this, estado);
15          setContentView(view);
16      }
17      @Override
18      protected void onPause() {
19          super.onPause();
20          view.onPause();
21      }
22      @Override
23      protected void onResume() {
24          super.onResume();
25          view.onResume();
```

```
    }
}
```

A.6.6. SRO_M/src/apps/tesis/omap/OpenGL/GLView.java

```
package apps.tesis.omap.OpenGL;

import android.content.Context;
import android.opengl.GLSurfaceView;

class GLView extends GLSurfaceView {
    private final GLRenderer renderer;
    int selec;

    GLView(Context context, int i) {
        super(context);
        renderer = new GLRenderer(context, i);
        setRenderer(renderer);
    }
}
```

A.6.7. SRO_M/src/apps/tesis/omap/OpenGL/GLRenderer.java

```
package apps.tesis.omap.OpenGL;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

import android.content.Context;
import android.opengl.GLSurfaceView;
import android.opengl.GLU;
import android.util.Log;
import apps.tesis.omap.R;

class GLRenderer implements GLSurfaceView.Renderer {
    private static final String TAG = "GLRenderer";
    private final Context context;
    private int opcion;
    private int CONT = 0;
    private float total = 0;
    // Movimiento del cubo
    long startTime;
    long fpsStartTime;
    long numFrames;
    // Clase que dibuja el cubo
    private final GLCube cube = new GLCube();
    GLRenderer(Context context, int i) {
```

```
25                  this.context = context;
26                  opcion = i;
27      }
28      public void onSurfaceCreated(GL10 gl, EGLConfig config) {
29                  // Cubo
30                  // Opciones OpenGL
31                  gl.glEnable(GL10.GL_DEPTH_TEST);
32                  gl.glDepthFunc(GL10.GL_LEQUAL);
33                  gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
34                  // Luz
35                  if (opcion >= 1) {
36                      // Definir iluminacion
37                      float lightAmbient[] = new float[] { 0.2f, 0.2f, 0.2f, 1 };
38                      float lightDiffuse[] = new float[] { 1, 1, 1, 1 };
39                      float[] lightPos = new float[] { 1, 1, 1, 1 };
40                      gl.glEnable(GL10.GL_LIGHTING);
41                      gl.glEnable(GL10.GL_LIGHT0);
42                      gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient, 0);
43                      gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);
44                      gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 2);
45                      // De que esta hecho el cubo
46                      float matAmbient[] = new float[] { 1, 1, 1, 1 };
47                      float matDiffuse[] = new float[] { 1, 1, 1, 1 };
48                      gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT,
49                              matAmbient, 0);
50                      gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE,
51                              matDiffuse, 0);
52                      // Movimiento
53                      if (opcion >= 2) {
54                          startTime = System.currentTimeMillis();
55                          fpsStartTime = startTime;
56                          numFrames = 0;
57                          // Cubo con Textura
58                          if (opcion >= 3) {
59                              // Habilitar texturas
60                              gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
61                              gl.glEnable(GL10.GL_TEXTURE_2D);
62                              // Carga textura
63                              GLCube.loadTexture(gl, context, R.drawable.logoespe1);
64                              // Cubo Transparente
65                              if (opcion >= 4) {
66                                  gl.glDisable(GL10.GL_DEPTH_TEST);
67                                  gl.glEnable(GL10.GL_BLEND);
68                                  gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE);
69                              }
70                          }
71                      }
72                  }
73      }
74
75      public void onSurfaceChanged(GL10 gl, int width, int height) {
76                  // Define el campo de vision
```

```
77                      gl.glViewport(0, 0, width, height);
78                      gl.glMatrixMode(GL10.GL_PROJECTION);
79                      gl.glLoadIdentity();
80                      float ratio = (float) width / height;
81                      GLU.gluPerspective(gl, 45.0f, ratio, 1, 100f);
82      }
83
84      public void onDrawFrame(GL10 gl) {
85                      // Pantalla en negro
86                      gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
87                      // Posicion del modelo
88                      gl.glMatrixMode(GL10.GL_MODELVIEW);
89                      gl.glLoadIdentity();
90                      gl.glTranslatef(0, 0, -3.0f);
91                      // Giro
92                      if (opcion >= 2) {
93                         // Set rotation angle based on the time
94                         long elapsed = System.currentTimeMillis() - startTime;
95                         gl.glRotatef(elapsed * (30f / 1000f), 0, 1, 0);
96                         gl.glRotatef(elapsed * (15f / 1000f), 1, 0, 0);
97                      }
98                      // Dibujar el cubo
99                      cube.draw(gl);
100
101                     // número de FPS
102         numFrames++;
103         long fpsElapsed = System.currentTimeMillis() - fpsStartTime;
104         if (fpsElapsed > 10 * 1000) { // c/5seg
105           float fps = (numFrames * 1000.0F) / fpsElapsed;
106           CONT++;
107
108           if (CONT >= 3 & CONT <= 12){
109                    total = total + fps/10;
110           }
111           Log.d(TAG, CONT +" FPS " + fps  + " Total " + total);
112           fpsStartTime = System.currentTimeMillis();
113           numFrames = 0;
114         }
115                     }
116 }
```

A.6.8. SRO_M/src/apps/tesis/omap/OpenGL/GLCube.java

```
1  package apps.tesis.omap.OpenGL;
2
3  import javax.microedition.khronos.egl.EGLConfig;
4  import javax.microedition.khronos.opengles.GL10;
5
6  import android.content.Context;
```

```
7  import android.opengl.GLSurfaceView;
8  import android.opengl.GLU;
9  import apps.tesis.omap.R;
10
11 class GLRenderer implements GLSurfaceView.Renderer {
12     private final Context context;
13     private int opcion;
14
15     // Movimiento del cubo
16     long startTime;
17     long fpsStartTime;
18     long numFrames;
19
20     // Clase que dibuja el cubo
21     private final GLCube cube = new GLCube();
22
23     GLRenderer(Context context, int i) {
24         this.context = context;
25         opcion = i;
26     }
27     public void onSurfaceCreated(GL10 gl, EGLConfig config) {
28         // Cubo
29         // Opciones OpenGL
30         gl.glEnable(GL10.GL_DEPTH_TEST);
31         gl.glDepthFunc(GL10.GL_LEQUAL);
32         gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
33         // Luz
34         if (opcion >= 1) {
35             // Definir iluminacion
36             float lightAmbient[] = new float[] { 0.2f, 0.2f, 0.2f, 1 };
37             float lightDiffuse[] = new float[] { 1, 1, 1, 1 };
38             float[] lightPos = new float[] { 1, 1, 1, 1 };
39             gl.glEnable(GL10.GL_LIGHTING);
40             gl.glEnable(GL10.GL_LIGHT0);
41             gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient, 0);
42             gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);
43             gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 2);
44             // De que esta hecho el cubo
45             float matAmbient[] = new float[] { 1, 1, 1, 1 };
46             float matDiffuse[] = new float[] { 1, 1, 1, 1 };
47             gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT,
48                     matAmbient, 0);
49             gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE,
50                     matDiffuse, 0);
51
52             // Giro
53             if (opcion >= 2) {
54                 // Movimiento
55                 startTime = System.currentTimeMillis();
56                 fpsStartTime = startTime;
57                 numFrames = 0;
58
```

```
59              // Cubo con Textura
60              if (opcion >= 3) {
61                 // Habilitar texturas
62                 gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
63                 gl.glEnable(GL10.GL_TEXTURE_2D);
64                 // Carga textura de un bitmap
65                 GLCube.loadTexture(gl, context, R.drawable.logoespe1);
66
67                 // Cubo Transparente
68                 if (opcion >= 4) {
69                    gl.glDisable(GL10.GL_DEPTH_TEST);
70                    gl.glEnable(GL10.GL_BLEND);
71                    gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE);
72                 }
73              }
74           }
75        }
76     }
77
78     public void onSurfaceChanged(GL10 gl, int width, int height) {
79        // Define el campo de vision
80        gl.glViewport(0, 0, width, height);
81        gl.glMatrixMode(GL10.GL_PROJECTION);
82        gl.glLoadIdentity();
83        float ratio = (float) width / height;
84        GLU.gluPerspective(gl, 45.0f, ratio, 1, 100f);
85     }
86
87     public void onDrawFrame(GL10 gl) {
88        // Pantalla en negro
89        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
90        // Posicion del modelo
91        gl.glMatrixMode(GL10.GL_MODELVIEW);
92        gl.glLoadIdentity();
93        gl.glTranslatef(0, 0, -3.0f);
94        // Giro
95        if (opcion >= 2) {
96           // Set rotation angle based on the time
97           long elapsed = System.currentTimeMillis() - startTime;
98           gl.glRotatef(elapsed * (30f / 1000f), 0, 1, 0);
99           gl.glRotatef(elapsed * (15f / 1000f), 1, 0, 0);
100       }
101       // Dibujar el cubo
102       cube.draw(gl);
103       }
104 }
```

A.6.9. SRO_M/src/apps/tesis/omap/Video/RepVideo.java

```
1  package apps.tesis.omap.Video;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.widget.MediaController;
6  import android.widget.Toast;
7  import android.widget.VideoView;
8  import apps.tesis.omap.R;
9
10 public class RepVideo extends Activity {
11     private String videoPath;
12
13     @Override
14     public void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.video);
17
18         // Inicializar Video View
19         VideoView video = (VideoView) findViewById(R.id.video);
20
21         // Origen achivo a mostrar
22         videoPath = "/sdcard/Video/big_buck_bunny_cif_mpeg4_aac.mp4";
23
24         video.setVideoPath(videoPath);
25
26         // Agregar Media Controller
27         MediaController mediacontroller = new MediaController(this);
28         mediacontroller.setAnchorView(video);
29         video.setMediaController(mediacontroller);
30         if (video.canSeekForward())
31            video.seekTo(video.getDuration() / 2);
32
33         // Inicio de Video
34         video.start();
35
36         // Indica mensaje de origen de video
37         Toast.makeText(this, "Video en Reproducción:\n" + videoPath,
38
39     }
40 }
```

A.6.10. SRO_M/src/apps/tesis/omap/Audio/Nombre.java

```
1  package apps.tesis.omap.Audio;
2
3  import android.app.Activity;
4  import android.content.Intent;
5  import android.os.Bundle;
6  import android.view.View;
```

```
import android.widget.EditText;
import android.widget.Toast;
import apps.tesis.omap.R;

public class Nombre extends Activity {
   private EditText text;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.nombre);
      text = (EditText) findViewById(R.id.nombreAudio);

      // Poner el cursor al principio del texto editable
      text.setSelection(1);
      text.extendSelection(0);
      text.setSelection(0);
   }
   // Listener boton guardar
   public void myClickHandler(View view) {
      switch (view.getId()) {
      case R.id.botonNombre:
         // Longitud > de .3gp
         if (text.getText().length() <= 4) {
            Toast.makeText(this,
                  Ingrese un nombre y ubicación al archivo:,
                  Toast.LENGTH_LONG).show();
            return;
         }
         // Inicio Intent AudioRec, envio de extras
         Intent intent = new Intent(Nombre.this, AudioRec.class);
         Bundle bundle = new Bundle();
         bundle.putString("nomb", text.getText().toString());
         intent.putExtras(bundle);
         finish();
         startActivity(intent);
      }
   }
}
```

A.6.11. SRO_M/src/apps/tesis/omap/Audio/AudioRec.java

```
package apps.tesis.omap.Audio;

import java.io.File;

import android.app.Activity;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
```

```
import android.os.Bundle;
import android.os.Handler;
import android.os.SystemClock;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Chronometer;
import android.widget.ImageButton;
import android.widget.ProgressBar;
import android.widget.TextView;
import apps.tesis.omap.R;

public class AudioRec extends Activity {
    private MediaRecorder mediarecorder = null;
    private MediaPlayer mediaplayer = null;
    private int i; // bandera play/rec
    private String archivo = null; //ejemplo: "/sdcard/Audio/audioOmap.3gpp";
    private ImageButton bro;
    private ImageButton br;
    private ImageButton bpo;
    private ImageButton bp;
    private ImageButton bso;
    private ImageButton bs;
    private TextView text;
    private TextView playI;
    private TextView estado;
    private Chronometer chronoRec;
    private Chronometer chronoPlay;
    private ProgressBar progreso;
    private int posicionActual = 0;
    private int total;
    private Handler mHandler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.audio);
        // Obtener extras de Intent Nombre
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            archivo = extras.getString("nomb");
            // validación ubicación archivo
            if (!archivo.startsWith("/")) {
                archivo = "/" + archivo;
            }
        }
        bro = (ImageButton) findViewById(R.id.b_rec_on);
        br = (ImageButton) findViewById(R.id.b_rec);
        bpo = (ImageButton) findViewById(R.id.b_play_on);
        bp = (ImageButton) findViewById(R.id.b_play);
        bso = (ImageButton) findViewById(R.id.b_stop_on);
        bs = (ImageButton) findViewById(R.id.b_stop);
        text = (TextView) findViewById(R.id.TextNombre);
```

```
60          estado = (TextView) findViewById(R.id.Estado);
61          playI = (TextView) findViewById(R.id.PlayInicio);
62          chronoRec = (Chronometer) findViewById(R.id.ChronoRec);
63          chronoPlay = (Chronometer) findViewById(R.id.ChronoPlay);
64          progreso = (ProgressBar) findViewById(R.id.BarraProgreso);
65          // Layout
66          BotonInicio();
67          text.setText(archivo);
68          SinBarra();
69          // OnClick Listener de Boton de Grabación
70          ImageButton botonRec = (ImageButton) findViewById(R.id.b_rec_on);
71          botonRec.setOnClickListener(new OnClickListener() {
72              public void onClick(View v) {
73                  rec(archivo);
74              }
75          });
76          // OnClick Listener de Boton Stop
77          ImageButton botonStop = (ImageButton) findViewById(R.id.b_stop_on);
78          botonStop.setOnClickListener(new OnClickListener() {
79              public void onClick(View v) {
80                  stop();
81              }
82          });
83          // OnClick Listener de Boton Play
84          ImageButton botonPlay = (ImageButton) findViewById(R.id.b_play_on);
85          botonPlay.setOnClickListener(new OnClickListener() {
86              public void onClick(View v) {
87                  play(archivo);
88              }
89          });
90      }
91      // Función para Grabación de Audio
92      private void rec(String filePath) {
93          i = 1;
94          // layout
95          BotonPlayRec();
96          estado.setText("Grabando...");
97          SinBarra();
98          // MediaRecorder
99          try {
100             // Archivo donde grabar
101             File mediafile = new File(filePath);
102             if (mediafile.exists()) {
103                 mediafile.delete();
104             }
105             mediafile = null;
106             // Inicializa al media recorder
107             mediarecorder = new MediaRecorder();
108             // ConFigura al media recorder
109             mediarecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
110             mediarecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
111             mediarecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
```

```
            mediarecorder.setOutputFile(filePath);
            // Prepara al media recorder
            mediarecorder.prepare();
            // Inicia al media recorder
            mediarecorder.start();
            // Cronometro Grabacion
            chronoRec.setBase(SystemClock.elapsedRealtime());
            chronoRec.start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    // Funcion para Reproducir Audio Grabado
    private void play(String file) {
        // Layout
        BotonPlayRec();
        i = 2;
        try {
            // Inicializa al media player
            mediaplayer = new MediaPlayer();
            // Listener de termino de media player
            MediaPlayer.OnCompletionListener listener = new MediaPlayer.OnCompletionListener() {
                @Override
                public void onCompletion(MediaPlayer mediaplayer) {
                    mediaplayer.release();
                    BotonStop();
                    chronoPlay.stop();
                    estado.setText("...");
                    SinBarra();
                }
            };
            // ConFigura al media player
            mediaplayer.setOnCompletionListener(listener);
            mediaplayer.setDataSource(file);
            // Prepara al media player
            mediaplayer.prepare();
            // Inicia al media player
            mediaplayer.start();
            // layout
            estado.setText("Reproduciendo...");
            // Cronometro
            chronoPlay.setVisibility(Chronometer.VISIBLE);
            chronoPlay.setBase(SystemClock.elapsedRealtime());
            chronoPlay.start();
            playI.setText("00:00");
            // Barra de Progreso
            progreso.setVisibility(ProgressBar.VISIBLE);
            progreso.setProgress(0);
            progreso.setMax(mediaplayer.getDuration());
            total = mediaplayer.getDuration();
            new Thread(new Runnable() {
                public void run() {
```

```
164                 while (mediaplayer != null && posicionActual < total) {
165                     try {
166                         posicionActual = mediaplayer.getCurrentPosition();
167                     } catch (Exception e) {
168                         return;
169                     }
170                     // Actualizacion de barra de progreso
171                     mHandler.post(new Runnable() {
172                         public void run() {
173                             progreso.setProgress(posicionActual);
174                         }
175                     });
176                 }
177             }
178         }).start();
179     } catch (Exception e) {
180         e.printStackTrace();
181     }
182 }
183 // Función Stop Recorder y Play
184 private void stop() {
185     // Layout
186     BotonStop();
187     estado.setText("...");
188     SinBarra();
189     // Grabacion stop
190     if (i == 1) {
191         mediarecorder.stop();
192         mediarecorder.release(); // reset();
193         mediarecorder = null;
194         chronoRec.stop();
195     }
196     // Play stop
197     if (i == 2) {
198         mediaplayer.stop();
199         mediaplayer = null;
200         chronoPlay.stop();
201     }
202 }
203 // Estado diferente a play: Ocultar Barra de Progreso
204 private void SinBarra() {
205     progreso.setVisibility(ProgressBar.GONE);
206     chronoPlay.setVisibility(Chronometer.INVISIBLE);
207     playl.setText("");
208 }
209 // Layout de Botones
210 private void BotonStop() {
211     bro.setVisibility(View.VISIBLE);
212     br.setVisibility(View.GONE);
213     bpo.setVisibility(View.VISIBLE);
214     bp.setVisibility(View.GONE);
215     bs.setVisibility(View.VISIBLE);
```

```
216             bso.setVisibility(View.GONE);
217             }
218         private void BotonPlayRec() {
219             br.setVisibility(View.VISIBLE);
220             bro.setVisibility(View.GONE);
221             bp.setVisibility(View.VISIBLE);
222             bpo.setVisibility(View.GONE);
223             bso.setVisibility(View.VISIBLE);
224             bs.setVisibility(View.GONE);
225         }
226         private void BotonInicio() {
227             bro.setVisibility(View.VISIBLE);
228             br.setVisibility(View.GONE);
229             bp.setVisibility(View.VISIBLE);
230             bpo.setVisibility(View.GONE);
231             bs.setVisibility(View.VISIBLE);
232             bso.setVisibility(View.GONE);
233         }
234     }
```

A.7.

Pruebas SRO-M Video Player

A.7.1. TestVideo2_1/src/apps/tesis/omap/video/v2_1/TestVideo2_1Activity.java

```
1  package test.tesis.omap.video.v2_1;
2
3  import android.app.ListActivity;
4  import android.content.Context;
5  import android.content.Intent;
6  import android.os.Bundle;
7  import android.view.View;
8  import android.widget.AdapterView;
9  import android.widget.ArrayAdapter;
10 import android.widget.ListView;
11 import android.widget.TextView;
12 import android.widget.AdapterView.OnItemClickListener;
13
14 public class TestVideo2_1Activity extends ListActivity {
15    Context context = this;
16    public void onCreate(Bundle savedInstanceState) {
17       super.onCreate(savedInstanceState);
18       // View de un Menu tipo Lista
19       String[] lista = getResources().getStringArray(R.array.ArrayVideo);
20       setListAdapter(new ArrayAdapter<String>(this, R.layout.main, lista));
21       ListView lv = getListView();
22       lv.setTextFilterEnabled(true);
23       // Listener Lista
24       lv.setOnItemClickListener(new OnItemClickListener() {
25          public void onItemClick(AdapterView<?> parent, View view,
26                int position, long id) {
27             try {
28                // Envia a Intent OpenGL cual item fue seleccionado
29                Intent intent = new Intent(context, VideosPlay.class);
30                //Bundle bundle = new Bundle();
31                //bundle.putInt("opcion", position);
32                //intent.putExtras(bundle);
33                Bundle bundle2 = new Bundle();
                  bundle2.putString("SDVideo", ((TextView)
34                view).getText().toString());
35                intent.putExtras(bundle2);
36                startActivity(intent);
```

```
37                } catch (Exception e) {
38                    System.out.println("Posición: " + position);
39                }
40            }
41        });
42    }
43 }
```

A.7.2. TestVideo2_1/src/apps/tesis/omap/video/v2_1/VideosPlay.java

```
1  package test.tesis.omap.video.v2_1;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.widget.MediaController;
6  import android.widget.Toast;
7  import android.widget.VideoView;
8
9  public class VideosPlay extends Activity {
10    private String videoPath;
11    @Override
12    public void onCreate(Bundle savedInstanceState) {
13        super.onCreate(savedInstanceState);
14        // Obtiene extras del Menu Lista
15       Bundle extras = getIntent().getExtras();
16       videoPath = extras.getString("SDVideo");
17        setContentView(R.layout.video);
18        VideoView video = (VideoView) findViewById(R.id.video);
19        video.setVideoPath(videoPath);
20        // Agregar Media Controller
21        MediaController mediacontroller = new MediaController(this);
22        mediacontroller.setAnchorView(video);
23        video.setMediaController(mediacontroller);
24        if (video.canSeekForward())
25            video.seekTo(video.getDuration()/2);
26        // Inicio de Video
27        video.start();
28        // Indica mensaje de origen de video
29        Toast.makeText(this, "Video en Reproducción:\n" + videoPath,
30                Toast.LENGTH_LONG).show();
31    }
32 }
```

A.7.3. TestVideo2_1/res/layout/main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
   <TextView
2  xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:gravity="center|left"
6     android:padding="15dp"
7     android:textSize="28sp"
8     android:textColor="#F8EBE8">
9  </TextView>
```

A.7.4. TestVideo2_1/res/layout/video.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <FrameLayout
3  xmlns:android="http://schemas.android.com/apk/res/android"
4  android:layout_width="fill_parent"
5  android:layout_height="fill_parent">
6     <VideoView
7        android:id="@+id/video"
8        android:layout_height="wrap_content"
9        android:layout_width="wrap_content"
10       android:layout_gravity="center" />
11 </FrameLayout>
```

A.7.5. TestVideo2_1/res/values/strings.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3     <string name="hello">Hello World, TestAudioActivity!"</string>
4     <string name="app_name">"TestVideos</string>
5     <!-- Path Videos en SD Card -->
6     <string name="SQCIF">"/sdcard/OMAP/1omapsqcif.mp4"</string>
7     <string name="QQVGA">"/sdcard/OMAP/2omapqqvga.mp4"</string>
8     <string name="QCIF">"/sdcard/OMAP/3omapqcif.mp4"</string>
9     <string name="A">"/sdcard/OMAP/4omapA.mp4"</string>
10    <string name="QVGA">"/sdcard/OMAP/5omapqvga.mp4"</string>
11    <string name="CIF">"/sdcard/OMAP/6omapcif.mp4"</string>
12    <string name="B">"/sdcard/OMAP/7omapB.mp4"</string>
13    <string name="VGA">"/sdcard/OMAP/8omapvga.mp4"</string>
14    <string name="CIF4">"/sdcard/OMAP/9omap4cif.mp4"</string>
15    <string name="PAL">"/sdcard/OMAP/10omappal.mp4"</string>
```

```
    <string name="SVGA">"/sdcard/OMAP/11omapsvga.mp4"</string>
    <string name="C">"/sdcard/OMAP/12omapC.mp4"</string>
    <string name="WVGA">"/sdcard/OMAP/13omapwvga.mp4"</string>
    <string name="D">"/sdcard/OMAP/14omapD.mp4"</string>
    <string name="XGA">"/sdcard/OMAP/15omapxga.mp4"</string>
    <string name="SXGA">"/sdcard/OMAP/16omapsxga.mp4"</string>
    <string name="CIF16">"/sdcard/OMAP/17omap16cif.mp4"</string>
    <string name="FPS:6_5">"/sdcard/OMAP/fps6_5omap.mp4"</string>
    <string name="FPS:10">"/sdcard/OMAP/fps10omap.mp4"</string>
    <string name="FPS:15">"/sdcard/OMAP/fps15omap.mp4"</string>
    <string name="FPS:25">"/sdcard/OMAP/fps25omap.mp4"</string>
    <string name="FPS:30">"/sdcard/OMAP/omapvgaFPS30.mp4"</string>
</resources>
```

A.7.6. TestVideo2_1/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="ArrayVideo">
        <item>@string/SQCIF</item>
        <item>@string/QQVGA</item>
        <item>@string/QCIF</item>
        <item>@string/A</item>
        <item>@string/QVGA</item>
        <item>@string/CIF</item>
        <item>@string/B</item>
        <item>@string/VGA</item>
        <item>@string/CIF4</item>
        <item>@string/PAL</item>
        <item>@string/SVGA</item>
        <item>@string/C</item>
        <item>@string/WVGA</item>
        <item>@string/D</item>
        <item>@string/XGA</item>
        <item>@string/SXGA</item>
        <item>@string/CIF16</item>
        <item>@string/FPS:6_5</item>
        <item>@string/FPS:10</item>
        <item>@string/FPS:15</item>
        <item>@string/FPS:25</item>
        <item>@string/FPS:30</item>
    </string-array>
</resources>
```

REFERENCIAS

1. Moralejo, M. and E. Pascual, *Circuitos Integrados*, in *La Electrónica*. 1973, Salvat Editores S.A.: Barcelona.
2. Haviland, D. *The History of Integrated Circuit*. 2004 [cited; Available from: http://www.icknowledge.com/history/history.html.
3. ASOCAE. *Historia de la Electrónica*. Electrónica [cited; Available from: http://www.natureduca.com/tecno_electro_hist05.php.
4. Calculator, D. *The history of Calculators and Computers*. [cited; Available from: http://www.diycalculator.com/cool.shtml#Hist.
5. Bakshi, U. and A. Godse, *Linear Integrated Circuits*. 2009: Technical Publications Pune.
6. Kalipa, U. (2006) *The evolution of Microprocessors from 1971 to the Present*. **Volume**,
7. Intel. *Intel Executive Biography Gordon E. Moore*. Intel Corporation [cited; Available from: http://www.intel.com/pressroom/kits/bios/moore.htm?iid=tech_mooreslaw+body_bio.
8. Tuya, M. (2000) *Todo lo que quiso saber sobre la Ley de Moore y nunca se atrevió a preguntar*. Bacuia Knowledge Center **Volume**,
9. Moore, G., *Cramming more components onto integrated circuits*, in *Electronics Magazine* 1965.
10. Fernandez, R. and F. Pérez. *Moore y la Ley de Moore*. Universidad de Oviedo [cited; Available from: http://petra.euitio.uniovi.es/~arrai/historia/trilobytes/5-Moore%20y%20la%20ley%20de%20Moore/Moore.htm.
11. ICKnolage, *Evolución de la Ley de Moore*, in *IExponential Trends in Integrated Circuit Industry*. 2004.
12. Collins, B., *In defence of Moore's Law.* PC Pro, 2010.
13. Juniper, F. (2009) *IDF 2009: Intel continues to pursue Moore's Law*. CNET Asia **Volume**,
14. Rashinkar, P., P. Paterson, and L. Singh, *System-On-A-Chip verification: methodology and techniques*, ed. K.A. Publishers. 2000.
15. Martin, G., *The History of the SoC Revoluction*, in *Winning the SoC Revolution: Experiences in Real Design*, G. Martin and H. Chang, Editors. 2003, Kluwer Academic Publishers: Massachusetts.
16. ITRS, *Potencial Design Complexity and Design Productivity*. 1999, Semiconductor Industry Association International Technology Roadmap for Semiconductors.
17. Lockwood, J.W., *Reconfigurable System On Chip Design*, W. University, Editor. 2002.
18. Piqué, R., *Sistemas Electrónicos: de la placa al chip*, in *Técnica Industrial*. 2006, Universidad de Sevilla: Sevilla.
19. Chang, H., et al., *Surviving the SOC Revolution*. 1999, Dordrecht: KLUWER ACADEMIC PUBLISHERS.
20. Shefer, D., *Non Recurring Engineering*. ScanDisk, 2005.

21. Haase, J., *Virtual Components - From Research to Business*, in *Virtual Components Design and Reuse*, R. Seepold and N. Martinez, Editors. 2001, Kluwer Academic Publishers: Boston.
22. Martin, G., *System-on-Chip Design*, in *Embedded Systems Handbook*, R. Zurawski, Editor. 2006, Industrial Communication Technology Handbook: Boca Raton London New York.
23. Nekoogar, F. and F. Nekoogar, *From ASICs to SOCs: A Practical Approach*. 2003, New Jersey: Prentice Hall Modern Semiconductor Design.
24. Cumming, P., *The TI OMAP Platform Approach to SoC*, in *Winning the SoC Revolution: Experiences in Real Design*, G. Martin and H. Chang, Editors. 2003, Kluwer Academic Publishers: Massachusetts.
25. Ferrari, A. and A. Sangiovanni-Vincentelli, *System Design: Traditional Concepts and New Paradigms*, in *Proceedings of the 1999 International Conference on Computer Design (ICCD)*, IEEE, Editor. 1999: Austin.
26. *MIPI ALLIANCE*. 2011 [cited]; Available from: http://www.mipi.org.
27. ARM. *Company Profile*. 2010 [cited 2011; Available from: http://www.arm.com/about/company-profile/index.php.
28. TI, *Wireless Terminals Solutions Guide*. Texas Instruments Incorporated. 2006, USA.
29. TI. *Wireless Handset Solutions: OMAP™ Applications Processors*. Texas Instruments Incorporated 1995-2011 [cited; Available from: http://focus.ti.com/general/docs/wtbu/wtbugencontent.tsp?templateId=6123&navigationId=11988&contentId=4638.
30. TI. *OMAP35x Applications Processors*. 1995-2011 [cited 2011]; Available from: http://focus.ti.com/dsp/docs/dspcontent.tsp?contentId=53403.
31. TI. *OMAP35X Aplications Processors Product Bulletin*. 2009 [cited Available from: http://www.ti.com/omap35x.
32. ARM. *Cortex-A8 Processor*. 2011 [cited 2011]; Available from: http://www.arm.com/products/processors/cortex-a/cortex-a8.php.
33. ARM. *Processors*. 2011 [cited 2011]; Available from: http://www.arm.com/products/processors/index.php.
34. TI, *Technical Reference Manual*, ed. T. Instruments. 2010.
35. ImaginationTechnologies. *SGX Graphics IP Core Family*. 2011 [cited 2011 2011]; Available from: http://www.imgtec.com/PowerVR/sgx.asp
36. KhronosGroup. *OpenGL ES Implementations, Tutorials and Sample Code*. KhronosGroup 2010 [cited 2011]; Available from: http://www.khronos.org/developers/resources/opengles/
37. Hawkins, K., *OMAP35x Processors Inspire New Aplications with unprecedented Performance at Handheld Power Levels*, in *TI Developer Conference*, T. Instruments, Editor. 2008: Beijing.
38. eLinux.org-EmbeddedLinuxWiki. *Computer-on-Module*. 2011 [cited 2011]; Available from: http://elinux.org/Computer-on-Module.
39. TI, *Linux Embedded System Design Workshop Student Guide*, ed. T. Instruments. 2006-2010, Dallas: Texas Instruments Incorporated.
40. LOGIC, *OMAP35x Develpoment Kit Quick Start*. Logic PD, ed. L. PD.
41. MistralSolutions, *OMAP35x Evaluation Module Hardware User Guide*. Texas Instruments, ed. T. Instruments. 2009, Bangalore.
42. MistralSolutions. *OMAP35x Evaluation Module Hardware User Guide*. 2009 [cited 2010]; Rev. 1.5. 2009:[Available from: http://mistralsolutions.com/pes-

support/support-downloads/omap3-evm-documentation-a-software-/tmdxevm3530.html.

43. Karunaratne, M., A. Sagahyroon, and A. Weerakkody, *An Advanced Library Format for ASIC Design*, in *IEEE*. 2004, CCGEI IEEE: Niagara Falls.
44. TI. *OMAP35x Applications Processor Technical Reference Manual*. 2011 2011 [cited 2011 Literature Number: SPRUF98I]; Available from: http://focus.ti.com/lit/ug/spruf98p/spruf98p.pdf.
45. Zulki, K., *Ball Grid Array Soldering*, in *Assembly Magazine*. 2008, bnp Webside: BNP Media.
46. ARM. *Cortex-M3 Debug and Trace*. 2011 [cited 2011]; Available from: http://www.keil.com/coresight/.
47. TI. *Codec Engine*. Texas Instruments Embedded Processors Wiki November 2010 [cited 2011]; Available from: http://processors.wiki.ti.com/index.php?title=Category:Codec_Engine.
48. Sonics, I. *SONICS On-Chip Communications Networks for Advanced SoCs*. 2011 [cited 2011]; Available from: http://www.sonicsinc.com/.
49. Microsoft. *Direct3D Mobile*. 2011 [cited 2011]; Available from: http://msdn.microsoft.com/en-us/library/aa452478.aspx.
50. KhronosGroup. *OpenVG - The Standard for Vector Graphics Acceleration*. KhronosGroup 2011 [cited 2011]; Available from: http://www.khronos.org/openvg/.
51. TI. *OMAP™ Applications Processors: SmartReflex™ Technologies*. Texas Instruments Incorporated 2011 [cited 2011]; Available from: http://focus.ti.com/general/docs/wtbu/wtbugencontent.tsp?contentId=4609&navigationId=12032&templateId=6123.
52. TI, *Programming Guide for DSP/BIOSTM Bridge*. Texas Instruments Incorporated, ed. T.I. Incorporated. 2008.
53. TI. *DSP/BIOS LINK WebEx Presentations*. Texas Instruments Incorporated 2011 [cited 2011]; Available from: http://focus.ti.com/docs/training/catalog/events/event.jhtml?sku=WEB108007.
54. Helmig, J., *White Paper: Developing Core Software Technologies for TI's OMAPTM Platform*. Texas Instruments Incorporated, ed. T.I. Incorporated. Vol. SWPY006. August 2002.
55. TI. *eXpressDSP Algorithm Standard – xDAIS Developer's Kit and xDM*. Texas Instruments Incorporated 1995-2011 [cited 2011]; Available from: http://focus.ti.com/docs/toolsw/folders/print/tmdxdaisxdm.html.
56. TI. *Operating Systems (OS/RTOS)*. Texas Instruments Incorporated 1995-2011 [cited 2011]; Available from: http://focus.ti.com/dsp/docs/dspfindtoolswbytooltype.tsp?sectionId=3&tabId=2095&toolTypeId=5&familyId=44.
57. TI. *Linux Digital Video Software Development Kit (DVSDK) for OMAP3530/3525 Digital Media Processors*. 1995-2011 [cited 2011]; Available from: http://focus.ti.com/docs/toolsw/folders/print/linuxdvsdk-omap3530.html.
58. TI. *Windows Embedded CE Software Development Kit (SDK) for Cortex-A8-based Processors*. 1995-2011 [cited 2011]; Available from: http://focus.ti.com/docs/toolsw/folders/print/wincesdk-a8.html.
59. TI. *Android Development Kit for Sitara Microprocessors*. 1995-2011 [cited 2011]; Available from: http://focus.ti.com/docs/toolsw/folders/print/androidsdk-sitara.html.

60. Google. *Rowboat Android for Texas Instruments Devices (Sitara, Davinci and Integra)*. Google Project Hosting 2011 [cited 2011]; Available from: http://code.google.com/p/rowboat/.
61. TI. *Software & Tools for DaVinci™- and/or Integra™-based processors*. 1995-2011 [cited 2011]; Available from: http://focus.ti.com/dsp/docs/dspcontent.tsp?contentId=72126.
62. Google. *android-scripting Scripting Layer for Android brings scripting languages to Android.* . Google Project Hosting 2011 [cited 2011]; Available from: http://code.google.com/p/android-scripting/.
63. Basic4android. *Basic4android*. Anywhere Software 2005-2011 [cited 2011]; Available from: http://www.basic4ppc.com/.
64. Android. *Android Developers*. Google 2011 [cited 2011]; Available from: http://developer.android.com/index.htmlAndroid Developers.
65. *Android Overview | Open Handset Alliance.* [cited 2011]; Available from: http://www.openhandsetalliance.com/android_overview.html.
66. Sasidharan, A., *WHITE PAPER, What Does Android Expect?* Mistral Solutions.
67. PV. *PacketVideo Embedded Software for Multimedia Services*. PacketVideo Corporation 2008 [cited 2011]; Available from: http://www.packetvideo.com/products/index.html.
68. GStreamer, *GStreamer Open Source Miltimedia Framework.* 2011.
69. TAPIA, J., et al., *INTRODUCTION TO THE OPENCORE AUDIO COMPONENTS USED IN THE ANDROID PLATFORM*. PacketVideo Corporation, San Diego CA, USA.
70. Gottardo, D., *Android Multimedia Overview*, in *System SW Architecture at Texas Instruments* February 11th 2009: France.
71. TI. *TI_Android_DevKit 02_00_00 Product Download Page*. Texas Instruments Incorporated 2010 [cited; Available from: http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/02_00_00/index_FDS.html.
72. TI. *DSP stack integration Android for Texas Instruments Devices (Sitara, Davinci and Integra)*. Powered by Google Project Hosting 2011 [cited 2011]; Available from: http://code.google.com/p/rowboat/wiki/DSP#Building_and_Testing_DSP_stack.
73. die.net. *minicom(1) - Linux man page*. [cited; Available from: http://linux.die.net/man/1/minicom.
74. TI. *DM37X.tar.gz*. [cited 2011]; Available from: http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/02_02_00/exports/DM37X.tar.gz.
75. Burnette, E., *Hello, Android Introducing Google's Mobile Development Platform*, ed. T.P. Bookshelf. 2010, Raleigh, North Carolina.
76. SOURCEFORGE.NET. *Inkscape Vector Graphics*. [cited 2011]; Available from: http://inkscape.org/?lang=es.
77. Google. *Android Asset Studio*. Android Asset Studio [cited 2011]; Available from: http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html.
78. Vogel, L. *Android Development Tutorial*. Revision 0.2 - 7.8 2011 [cited 2011]; Available from: http://www.vogella.de/articles/Android/article.html#first.
79. Zigurd Mednieks, L.D., G. Blake Meike, Masumi Nakamura, *Programming Android Java Programming for the New Generation of Mobile Devices*. O'Reilly Media, ed. O.R. Media. July 2011.

80. HTC. *HTC Wildfire*. 2011 [cited 2011]; Available from: http://www.htc.com/europe/product/wildfire/specification.html.
81. Acer. *ICONIA TAB A500*. 2011 [cited 2011]; Available from: http://acer.us/ac/en/US/content/iconia-tab-a500.
82. SURCEFORGE.NET. *Procps - The /proc file system utilities*. 2009 [cited 2011]; Available from: http://procps.sourceforge.net/.

FECHA DE ENTREGA

El presente proyecto de grado fue entregado a la fecha.

Sangolquí, _____ 2011.

Realizado por:

_____

Sandra Elizabeth Argüello Jácome

_____

Ing. Gonzalo Olmedo, PH.D.

Director Carrera Ingeniería en Electrónica y Telecomunicaciones