

*Airplay SDK can help to remove limitations experienced by content developers as they balance between the number of platforms supported with the resources available*

# Overcoming Ecosystem Challenges for Digital Home Content

By Tim Closs, CTO, Ideaworks Labs

Ubiquitous device connectivity, and advances in device performance, are changing the landscape for content developers in the digital home. However, there are still some serious challenges to address. Ideaworks Labs has been working with ARM for many years to address these challenges through its Airplay Software Development Kit (SDK). This article will address how the Airplay SDK can help to remove limitations experienced by content developers as they balance between the number of platforms supported with the resources available.

An increasing number of devices in the digital home are becoming 'content aware' - not only do they have the technical capability to run rich media; they also have the connectivity and distribution channels to deliver that media to the device.

While this presents exciting opportunities for content developers on ARM-based platforms, there are considerable technical, economic and ecosystem challenges still to be overcome. This article outlines some of those challenges, and describes how Ideaworks Labs has worked with ARM to provide Airplay SDK, a unique software platform and developer SDK to help accelerate this emerging ecosystem.

The main technical challenge is supporting the range of performance points and execution environments, across an increasingly diverse set of devices. In terms of hardware, we have CPUs ranging from ARM9 up to ARM Cortex A8, with and without hardware graphics accelerators, which differ widely in their performance characteristics. In terms of software platforms, we have 'open' (primarily Linux-based) operating systems on hardware not powerful enough to support web runtimes; and we have 'closed' (primarily OEM-proprietary) operating systems without support for standard execution environments. This 'fragmentation' of hardware and soft-

ware platforms presents a problem for distribution channel owners and content developers alike.

This leads to one of the main economic challenges, which is the production cost of supporting content across all devices. As the smartphone content industry has demonstrated, the cost of 'porting' content across a wide range of devices can be disproportionately high, often higher than developing the content in the first place, and developers will not pay this price until the revenue potential of each platform is both large and proven.

Finally, a significant ecosystem challenge for the device manufacturers or channel owners is that of providing a content SDK to the development community, and encouraging its use. As outlined above, many of these device platforms are not suitable to support existing complex operating systems (Android, Symbian, etc.) which already come with an experienced development community. Instead, they are running

Linux-based or proprietary operating systems which have little or no awareness within the content development community. The device manufacturers or channel owners then face a vicious circle; they need a convincing body of premium content with which to launch the channel, yet the development community will not risk



investing in porting content to a new device (or SDK) until the channel is proven and profitable.

Ideaworks Labs has been working with this problem set for over 7 years, and has developed a revolutionary product, Airplay SDK, that unifies the device landscape and transforms the economics of content development. Airplay SDK allows developers to compile their application once only, to native ARM CPU instructions. The compiled ARM code can be tested and debugged at source code or disassembly level, within a standard desktop environment. An integrated deployment tool then provides a one-click process for deploying the single ARM binary to all supported operating systems and devices. This includes iPhone, Android, Symbian, Windows Mobile, BREW, mobile and embedded Linux, and closed RTOS platforms.

Let's look at how Airplay's approach transforms the ecosystem for applications development in the digital home.

Firstly, Airplay compiles to native ARM code (and supports use of ARM's best-in-class RVCT compiler) and does not use any kind of "virtual machine" during the compilation, installation or execution phase. This results in the fastest possible application code on each device. Together with the world's fastest embedded 2D and 3D software renderer, this means that important classes of digital home device (for example, high-resolution TVs running powerful ARM CPUs but without hardware graphics acceleration) can now be addressed with high-quality content (sourced from another platform, for example iPhone) for the first time. By default, Airplay SDK produces an ARM binary that will work on all ARM CPUs from ARM9 and above; it is easy for developers to create separate Stock Keeping Units (SKU) optimized for different ARM CPU instruction sets. These SKUs then become CPU-specific, but are still entirely portable across operating systems and devices.

This leads on to the second key point; Airplay developers do not need to think about different operating systems or device inconsistencies. For example, an Airplay developer never "builds for Android" or "builds for Linux" – they simply build for ARM. The Airplay APIs make it easy to write a single application that can accommodate different device form factors; for example screen sizes, keypad or keyboard type, touchscreen and accelerometer. Airplay's advanced UI framework provides dynamic layout management and seamless handling of aspects such as text entry, which typically are burdens to implement well across different input form factors such as keypad, keyboard and touchscreen. Furthermore, the Airplay runtime deals with firmware-specific and device-specific anomalies and bugs, meaning that the developer doesn't have to. Ideaworks Labs operates a sophisticated process for procuring and testing new devices (running over 500 unit tests and system tests on each device, including manual tests such as physical screen rotation, sending and receiving SMS and phone calls, etc.) and adjusts the Airplay runtime if required to make new devices behave as expected. The device is then "certified" as Airplay-compliant.

The economic consequences of this approach for developers are dramatic. In the smartphone sphere alone, content producers are



currently spending more on "porting" between operating systems and devices than they spend on the core application development itself, sometimes as much as 2 to 3 times more. A production budget must cover the cost of the initial development and the subsequent porting. As porting efforts and costs increase, this reduces the amount spent on initial development, meaning features and quality are heavily compromised. Furthermore, entire classes of devices will be considered too costly to port to, and so the addressable market for the application becomes limited. Airplay SDK improves this situation by removing the barriers between operating systems and between devices, effectively presenting a single "super-device" to the developer. Porting costs between operating systems are reduced to zero, and porting costs between devices are reduced to simply accommodating the standard range of hardware form factors.

For the owner of the content distribution channel, this is all a great relief. The channel owner might be the device manufacturer, the digital service provider, a media rights owner – or all three. Regardless of the category, the distribution channel must enable content across all device types at the end of whatever network the device accesses. Again, this requires crossing both hardware and software platform boundaries. Many digital home devices are based on software platforms (for example, embedded Linux variants) without suitable tools for content development, and with little or no awareness within the development community. The channel owner could create a bespoke SDK for these platforms – but will developers invest in using it? Not until the channel is proven commercially; but that requires "priming the pump" to overcome the content developers inertia. Airplay SDK solves the channel owner's conundrum by providing a development environment that delivers high-quality content not only to their devices, but also to existing commercially-proven platforms. So the developer's risk is removed, and the channel owner still gets content; a win-win proposition..

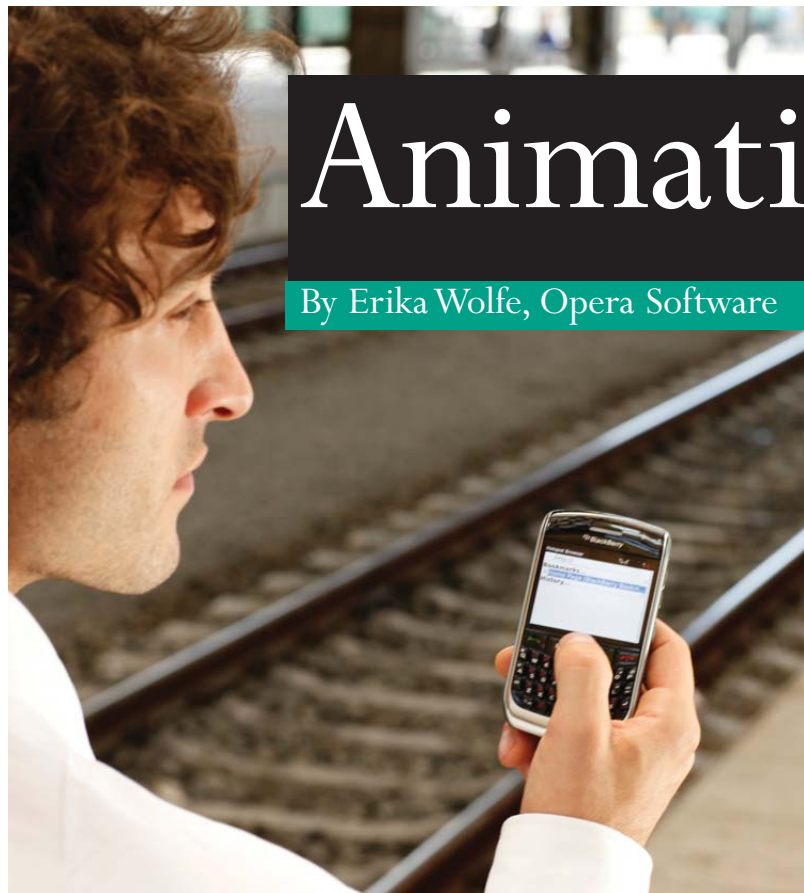
Ideaworks Labs has been successfully licensing Airplay SDK to some of the world's biggest games and application publishers for the last 2 years, including EA Mobile, Nokia, Konami and Capcom. Airplay-based content has made its way across all smartphone platforms and is now starting to appear on set-top boxes and digital TVs. Ideaworks Labs is working closely with manufacturers of digital home devices to ensure that new platforms and devices are supported.

At the end of October this year, Ideaworks Labs will, for the first time, make Airplay SDK available to the wide applications development community, with a licensing model that scales all the way down to \$99 per seat for "indie" developers. This will be a huge step forward in democratizing applications development across the increasing variety of device types, whether they are carried in one's pocket or reside in the digital home.

**END**

# Animating the Conn

By Erika Wolfe, Opera Software



# ected Home and Life

In the home media space, Opera provides a web runtime Software Development Kit (SDK), that enables ARM licensees to develop their own browser and their own widget engine.

## Connected Home - Connected Life

**A** ARM and Opera Software share a similar role in bringing the connected home concept to life. As technology providers, both companies develop complex behind-the-scenes solutions for running and connecting a whole host of devices, both outside and within the home, with the ultimate aim of creating seamless interoperability and connectivity for end users.

## A Day in the Life of a Connected User

Today the traditional connected home uses PCs or laptops to connected to the web. However ARM, Opera and other partners recognize that the flow of desired content to devices already extends far beyond the PC. It is expanding beyond the connected home into a completely connected lifestyle with mobile phones, netbooks, DTVs, STBs and even white goods.

Your whole connected day could include:

### **Samsung phone**

You wake up to go to work and immediately check your mobile phone to find out the weather, using the idle-screen weather widget you have installed. With Opera Mobile and Opera Widgets, this is a simple part of the morning routine.

### **Sharp Mebius NJ70A netbook**

At work, you power up your netbook, read your e-mail, and start searching for information you will need for a meeting later that day. Once you get the information you need, you create a presentation for the meeting. You synchronize this information easily to your mobile phone for use later at home with the Opera Link syncing tool, which is downloadable for the Sharp netbook.

### **Kindle eBook Reader**

Crystal ball gazing - at home, while cooking dinner you look up recipes directly from your home Internet tablet or access the recipe file from another device in your home network where you have previously stored a favorite recipe, downloaded from the Internet. Opera's SDK enables UI/Web interfaces that are user friendly and intuitive, making content accessible from every device in the home.

### **Sony Bravia TV (home)**

Winding down in the evening, you settle in to watch the show you earlier downloaded, while also keeping an eye on the latest news headlines using TV widgets. With Opera's SDK powering the UI and TV Web browsing and Opera Widgets giving you access to the information you want most - you have everything you need at your fingertips.

### **Nintendo DSi (home)**

Deciding to turn in for the night but not ready to sleep, you grab your Web-enabled Nintendo DSi and prepare for some fun. The Opera SDK-powered Web browser lets you go online and play interactive games or just browse the Web.

# ARM and Qt Development Frameworks

## “Staying Tuned”

*New Models for 21st Century Television and Internet Video Viewing Demand a Broader Approach to Flexible Application Delivery at the User Interface Level*

By Dilip Kenchamma, Product Line Manager, Nokia Qt Development Frameworks

Qt is a cross platform C++ application framework that today is available on several embedded and mobile platforms as well as devices, which use Linux, Mac OS X and Windows operating environments. The inherent cross platform nature of the Qt technology proposition is playing a crucial role now, as home and mobile electronic devices constantly spawn new form factors. While the physical ‘shape of the box’ may be changing, people demand a consistent user experience that they know and recognize as they spend more time using increasingly sophisticated pieces of equipment.

As technologies converge, OEMs and individual software programmers everywhere are looking for enabling technologies to take advantage of the proliferation of new digital television models. In this space, the Qt framework already has a strong track record of working with video interface technologies such as the Asus touch-screen video communications device. Similar devices driven by the power of ARM processors can now also benefit from the inherent flexibility and advanced features of Qt Version 4.6.

Using Qt's powerful application and user interface (UI) framework matched with the strength of ARM processors, OEMs now have an attractive option for designing products within the ARM architecture. Web services on television sets and television programs on ‘connected’ IPTV devices, are on the rise. Within this scenario we also see the convergence of multiple devices (TV, phone, laptop) with similar UI/GUI demands made on them, as users access content sometimes using multiple screens. Providing users with a consistent experience across all devices is key to market success - having a flexible cross platform framework enables this interoperability to be realized from the start.

Driving this effort forward is a community of empowered developers who are already active in writing applications for the new generation of digital TV's. In many cases, these developers are also adopters of the open source contribution model that, over time, will ultimately lead to better code enhancements for all developers. Certainly this was the case with initially the Linux kernel and is also true as we move upwards in the software stack hierarchy to include WebKit - and now Qt.

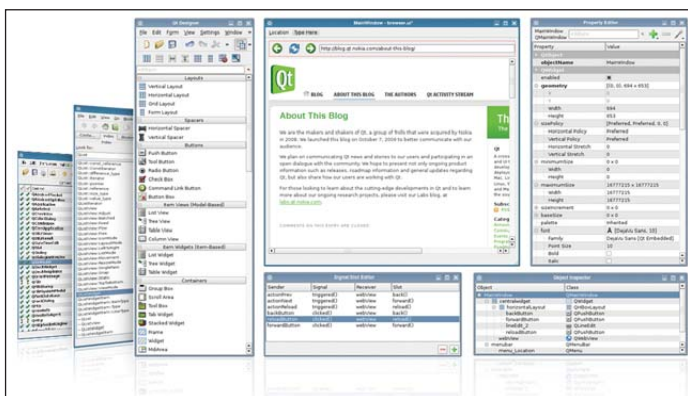
Industry analysts and commentators alike are agreed upon the need for open source-inspired ingenuity to drive and underpin the next tier of digital TV development.

Certainly the thought of anyone developing their own kernel for embedded devices or set top boxes from scratch, rather than using open source for example, would be unusual. Developers using Linux have long found themselves able to benefit from the wealth of options available from the open source community. As we see further commoditization in this space it makes sense that we should see increasing standardization of software components, including the user interface layer.

Further, regarding some Asian markets, there is already a community

of home original device manufacturers (ODM), who have worked in parallel with their software partners, to bring the latest breed of TV technologies to market. This work itself is happening in parallel with the type of innovations seen at Ovi.com, Nokia's converging Internet services environment, which is another excellent example of a company in this space utilizing converging methods for accessing media/data content.

On-line digital delivery removes geographic boundaries for distribution of ondemand and broadcast content, allowing consumption of



any media anywhere, anytime. Browsing and discovering content in this explosion of media requires very powerful user interfaces and an entirely new, open/standard software stack. It requires hardware acceleration, HTML5 support, 3D animated UIs, etc. Such a new STB/TV system requires high performance, excellent power management, multiple choices in terms of chipset selection and a truly solid and stable hardware/software platform, with support and on-going development. Qt and ARM can deliver this and are currently working together towards a shared vision for home media and entertainment.

With the power of ARM processors driving the engine room to empower Qtbased GUIs particularly in the home media segment, it's important to understand, that from the media service provider's point of view, flexibility in defining and tuning the user experience is the most important factor. Qt's HTML5 support enables an upgrade of the entire user experience without a firmware upgrade when necessary. This means that an application can be built using familiar web architectures and tools, so that the net result is a more personalized and engaging end user experience.

What is also important is that from the OEM's perspective, Qt's HTML5 support provides a standard container for the ever-expanding list of content stores. Also, Qt has proven animated UI technologies over DirectFB and Flash plugin support in QtWebKit, all in a single integrated package.

One of the aims of Qt is to provide native GUI performance with the flexibility and choice of web based media delivery. In combination with ARM-based processors, this enables device makers, content distributors and their vendors to create home media devices that engage, entertain and delight the consumer.

ARM itself fosters a large network of partners, the ARM Connected Community, to bring together systems specialists, design support professionals as well as software and training providers to enable a complete solution for products based on the ARM Architecture. Qt similarly benefits from this connected community as the implementations for digital television start to break ground, in new as yet comparatively untouched areas.

As the OEMs now start to work more closely with digital TV technology, they have greater exposure to both Lesser General Public License (LGPL) licensed offerings with experienced corporate foundations such as Nokia Qt Development Frameworks and the wider open source community itself.

If the combination of established brands such as Qt and ARM processors means OEMs are more likely to build in open interoperability as a core consideration in their product roadmaps, then there is the strong likelihood that everyone will gain in terms of product usability and price. We could all be headed for a bright, open, tuned in and totally digital future.

END

# The latest ARM design news, features, products, webcasts and more are available 24/7 on the new IQMagazineonline!

Bookmark it today, and visit often!  
[www.iqmagazineonline.com](http://www.iqmagazineonline.com)

*Looking at the design of  
an internet enabled device for  
the home using ARM Cortex  
powered SoCs*

# Internet Enabled Devices for Your Home

By Murali Babu Muthukrishnan, Ittiam Systems Pvt. Ltd.

The Internet is now available to everyone, enabling a variety of use cases from news gathering and content sharing to instant messaging. Such activities are primarily enabled by PCs and smartphones while at home the connected TV is starting to take on this role.

In this article we look at the design of an internet enabled device for the home using ARM Cortex powered SoCs. The Cortex-A8 with NEON is the latest processor from ARM which packs a punch on multimedia performance. A detailed analysis of the performance requirements is presented for a multimedia device, capable of playing several audio and video formats suitable for the multitude of formats available on the Internet.

Connecting to the Internet today is still closely tied to our desktops and portable computers. Even mobile phones, with increasing connectivity, are becoming personal Internet navigation devices, their small form factor and screen sizes make them not always the best route for accessing/sharing videos and photos on the Web.

The TV, however, still dominates in the home, thanks to the appeal of its large screen. With the arrival of picture and video sharing sites like YouTube, Flickr and others, consumers are looking beyond their laptops and smartphones to enjoy content. As these sites become increasingly popular, consumers will look to bring them into the family room where the TV currently dominates. In such a context a Home Internet Device (HID) is needed, acting as an IP set top box, bringing Web content or User Generated Content (UGC) to the living room.

## Product Conceptualization and Design

### System Requirements

A system built to bring Internet content to a TV at home will need to be capable of handling the following system requirements:

- **Rich Web Experience on a TV screen:** The HID will leverage the rich content available on the Web to bring an enhanced Web experience to the home user. The HID will connect to favorite menus such as a video listing on YouTube or subscribing to a feed in Yahoo, Google or CNN. This can open up 'on demand' services with users being able to receive services like their favorite Tweets to their TV.
- **Home Networking of Media:** The HID will be able to aggregate and be used as a home server for all media content on the Web, as well as within home. This could mean being able to stream a video from the Web or watch your child performing in a school video, streamed straight to your TV, from your home PC server.
- **Rich Interactive Gaming:** Rich interactive gaming with 3D graphics will be on the TV instead of your home PC, enabling a rich user experience through one interface.

# EMBEDDED QUALITY, VERIFIED RELIABILITY

– Backed by scalable OS and 30 years proven track record –

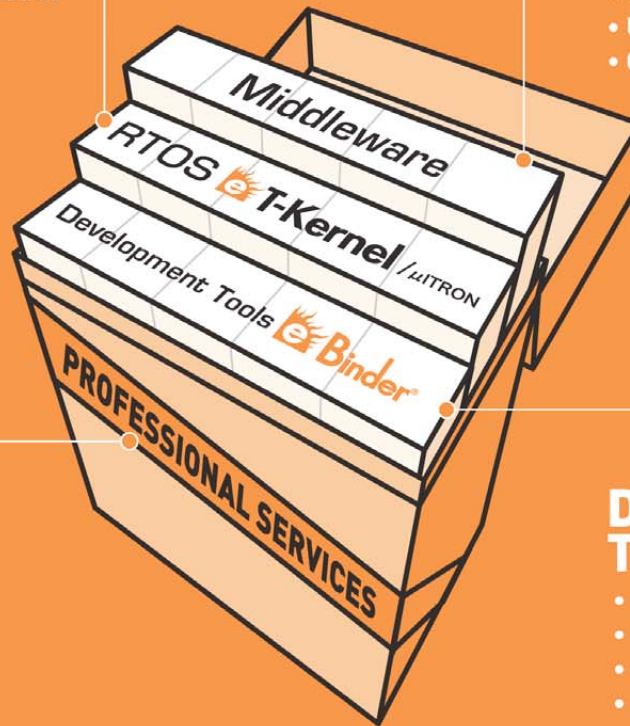
## Real-time Operating System

- Scalable profiles
- Process model/Thread model
- POSIX-compliant
- Memory protection
- Multi-core support

## Middleware

- File system
- TCP/IP protocol stack
- USB host/device stack
- GUI development tool

ALL-IN-ONE  
PACKAGE



## Professional Services

- Customer support
- Porting
- Customization
- Training

## Development Tools

- Configuration tool
- Build tool
- C/C++ Compiler
- Multiprogramming tool
- System analysis tool

Supported Architecture: ARM7, ARM9/9E, ARM11, ARM11 MPCore, Cortex A-8, Cortex-A9, Cortex A-9 MPCore, MIPS, SH

## The Proven Solution for Embedded Software Development

**LOOKING TO ACCELERATE YOUR COMPETITIVE EDGE?** eCROS eliminates the time-consuming need to build software platforms, so you can focus on what really matters: application development. This all-in-one suite combines the RTOS, middleware, and development tools with comprehensive professional services. Our scalable RTOS in particular enables you to reuse software assets for effective product line development. Our Memory Protection and unique Memory Partitioning technology assures system reliability and high quality.

Backed by 30+ years of expertise—and proven track record in car navigation systems, consumer electronics, factory automation devices, and much more—eCROS is the best way to develop high-quality application software and maximize reliability of your system.

web search:  or go to [www.esol.co.jp](http://www.esol.co.jp)



eSOL is a member of the  
ARM Connected Community

eSOL Co., Ltd. Embedded Products Division Tel: +81 3-5302-1360 E-mail: [ep-info@esol.co.jp](mailto:ep-info@esol.co.jp)

- **Digital Picture Frame:** HID will act as a Digital Picture Frame (DPF), synchronizing with content on the Web to bring you the latest pictures uploaded by family and friends.

- **Instant Messaging and Video Communication:** The HID will also connect to an IP network and act as a video communication device for home connected users, enabling discussions via applications like Skype™ or Windows Messenger™.



including codecs and other functions such as Digital Rights Management (DRM) to be able to watch pay-per-view as well as user generated content. Additionally, the Media player should be capable of supporting JPEG playback to provide DPF functionality with all the necessary transitions and effects.

- **Streaming Server:** As the HID will also provide home server functionality for devices on the home network, streaming server support would be a useful addition.

- **Web Content Support:** As Web support is the primary use for such device, a fully W3C compliant HTML/CSS standalone and embeddable browser is a key requirement. Flash support would make the device a compelling buy and would provide a rich Web experience for the user.

- **Graphics Support:** Open GL ES 2.x support will be required to bring games alive, as well as enhance the Web experience on the device. Graphics support will also be required to bring the DPF function alive.

- **Web support:** Web browsing support in the form of Web toolkits and support for subscribing to the feeds and/or download and play sites, like peer-to-peer content distribution, would be a key requirement to bring a rich consumer Web experience. JavaScript support in software and, if possible, in hardware, will allow for the seamless delivery of Web applications.

### Connectivity Requirements

- **Internet Connectivity:** Given that the primary requirement of this device is to connect to the Internet and bring web content to TV, the device should have an Ethernet connection. Additionally, wireless connectivity (802.11 a/b/g/n) will be desirable, making the HID wireless enabled.

- **Interface Connectivity:** The HID is primarily aimed at playing Internet content on TV, having an interface to a SD/MMC card will enable local playback of content on the device. USB connectivity will add the ability to synchronize content with your mobile or other devices and can also serve as a mass storage device interface.

- **Display Connectivity:** Since the key purpose of this device is to bring Internet content to the TV, display connectivity in the form of High Definition Multimedia Interface (HDMI), is a must. Additional connectivity, through component video and S-Video outputs would help maintain compatibility with non-HDMI enabled TVs. Since the display medium will be a TV, no LCD connectivity is envisioned, lowering the overall system cost. TV decoder and encoder chips will be required to make sure that regular NTSC out and S-Video is available, providing the additional ability to record from analog TV or cable sources.

- **Audio Connectivity:** Stereo outputs are a minimum requirement for TV. Additional line in connectivity helps in providing stereo recording of TV or cable content. A Microphone is also a useful addition for recording functions.

- **Remote Control:** IrDA interface is a requirement, providing remote control access to the device. This will also be the primary interface for entering text inputs for browsing the Internet.

- **Camera Interface:** Such an interface will enable video communication through the device. As it is connected to the Internet, live video can be shared during a VoIP-based communication using the camera.

### Software Requirements

- **Media Playback:** The main objective of this device is to connect to the Internet, bringing Web content, video and music to the home. System requirements therefore need to include a Media player, a streaming client capable of connecting to the Internet. The Streaming Media player needs to be capable of supporting peer-to-peer content distribution such as YouTube, Hulu, or BBC iPlayer. Other content delivery mechanisms will also need to be supported,

### Choosing the Processor Sub System

Given the above system and software requirements, a powerful processor capable of handling the complex software as well as the web requirements is needed. The latest ARM Cortex™ family of processors equipped with vector coprocessors in the form of NEON™ have the power and scalability to handle these complex software requirements. The Cortex family of processors, with their deep pipelines, are capable of clocking higher speeds, making them ideal for handling such demanding applications.

With the increasing popularity of third party software available on ARM processors, and the ease of bringing web applications to ARM processors, the ARM Cortex family of processors are ideal for building these Internet devices on. One of the processor families available in the market suitable for such devices is Texas Instruments' OMAP series of processors, notably the OMAP3515 processor composed of a Cortex-A8 + NEON processor clocked at 600 MHz coupled with a graphics coprocessor in the form of the powerful PowerVR SGXTM core fits the bill well for creating a HID.

**Processing power and flexibility:** The Cortex-A8 + NEON processor is capable of handling both multimedia and non-multimedia applications. The NEON powered Cortex-A8 processor is capable of decoding video and audio streams.

- **Clock speed of 600 MHz:** it can support most video standards including H.264, MPEG-4, and VC-1 at standard resolution (SD) (720x480 at 30 frames per second, 720x576 at 25 frames per sec-

ond) within 75% loading of the processor; leaving the remaining 25% of the processing power free for other functions such as audio decode and video playback.

- **Additional features:** Aside from the powerful NEON vector processor, the Cortex-A8 core on the OMAP3515 is packed with features such as TrustZone®, designed to secure consumer products, Thumb®-2 for higher performance at lower code density and, Jazelle® RCT execution environment architecture for accelerating Java-based application support.

- **Strong Graphics Capabilities:** Powerful graphics support on the device make UI applications rendering faster and easier, especially with OpenGL ES 1.1 and 2.0 and OpenVG 1.0 support. 16Kbytes of L1 - Instruction Cache and 16Kbytes of L1 - Data cache, with a large 256Kbytes of unified L2 cache, helps speed up performance of all multimedia as well as non-multimedia algorithms. Built-in display subsystem support on the OMAP3515 includes 24-bit RGB output, HD resolution output, Composite NTSC/PAL video support and resizing of output images from factor 1/8- to 4- & 8-bit alpha blending.

- **Broad Connectivity:** A removable media interface, in the form of MMC/SD/SDIO card support, allows for storage of media as required in the system and connectivity requirements listed in sections above. Connectivity requirements envisioned on a HID are sat-

isfied by the Serial and UART connections for remote IrDA including the USB OTG support. Additionally, the device can decode JPEG images, required for the DPF.

- **Eco-friendly system:** The powerful Cortex-A8+NEON core can run one CIF channel encode and decode simultaneously within 400 MHz (of H.263 or MPEG-4 video) to provide video communication features. Comprehensive power reset and clock management help reduce power consumption leading to a 'greener' design. An architecture diagram of the device is shown below in figure 1.

### Software Architecture

Given the above OMAP3515 system block diagram and the features available on the device, we now need to build a software architecture which will equip the HID with the necessary software to be able to connect to the Internet and bring Web content to our home TV.

The first step in the software architecture is to evaluate the Operating System (OS) choices. There are several choices for OS that could be considered, but given the proliferation of Linux as a royalty free OS and the availability of Linux ports on devices, it seems an obvious choice. Google Android, for example, already has a strong developer community and is quickly gaining in popularity. It also has the advantage of coming bundled with Google applications and a framework for Media Player.

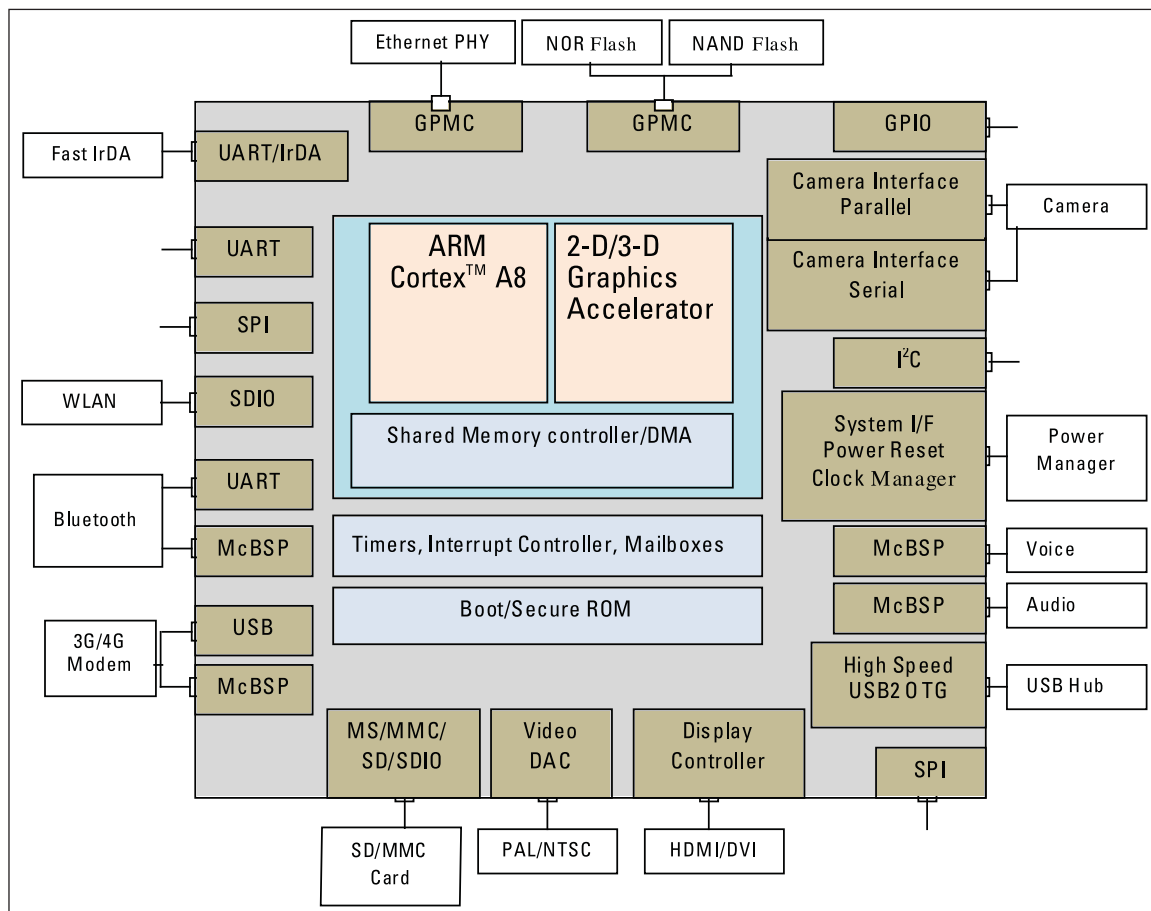


Figure 1: OMAP3515 based HID System Block Diagram

Drivers will need to be supported by the chosen connectivity devices. This includes the video out or frame buffer driver for video display onto the TV through a HDMI/DVI connection or through the NTSC/PAL. In addition, WLAN, WiFi, and Bluetooth drivers will be needed for connectivity as well as USB and MMC/SD card drivers for mass storage support. An image interface and a camera driver will also be required for video communication applications. Other standard embedded support such as bootloader, NOR/NAND flash support are taken as read.

On the middleware side, given the key requirements of bringing YouTube and other internet content to TV, a Multimedia Engine capable of supporting playback of Web content along

with support for other functionalities like video communication, DPF functionality is essential.

Ittiam Media System SDK (MSS) is a proven multimedia engine, bundled with codecs and components. This has been verified on several TI OMAP platforms for different multimedia applications including Portable Media Player & Recorders, Digital Video Recorders, Broadcast delivery, Transcoders, Streaming clients/servers, Digital Media Adapters and Network Media Players. The Ittiam MSS has a well defined set of media Application Programming Interfaces (API) that covers many functions of a multimedia application and also has a strong portfolio of codecs and components including parsers, composers, streaming subsystems, DRM amongst many, and can integrate into several peripherals seamlessly.

This small footprint highly portable MSS is designed for embedded multimedia, ensuring high performance abstraction layers that minimize the porting effort required to move to a new platform. With its SDK type approach it makes the application development and integration to other middleware and application components, like DLNA clients or streaming protocols, simple and easy.

The Media System SDK is also capable of supporting voice and video calls over SIP. Audio and video codecs optimized on the Cortex-A8+NEON, providing the required decoding and encoding capabilities to support any of the above mentioned multimedia applications. These codecs have been optimized for performance to

ensure a rich user experience in terms of both frame rates and resolutions in bringing internet content to TV.

Additionally, the device should support software extensibility which will enable software upgrades on the device to support the ever growing demands of the Internet. As shown in figure 2, a software stack built in a layered form helps to tie in all the applications on the device. Highly optimized and efficient codecs and middleware like DRM, file parsers, RTP, RTSP, HTTP subsystems are connected to a framework such as Ittiam's Media System Framework, which supports all multimedia features including playback of audio and video streams either from a file, or through a streaming system like RTP/RTSP based system or a progressive HTTP download. The MSS also supports JPEG display with transitions effects and can also run a simultaneous slide show with audio. Here again the JPEG playback can be from a progressive HTTP download or through a RTP/RTSP system. DRM support in the MSS coupled with Web applications allow users to purchase content online and play it either through download or through a streaming system.

The APIs provided by the MSS can be used across several applications to enable media playback, archival and streaming. MSS can also stream the content from a local server to other devices within the home through the streaming server or act as a home streaming player with the DLNA client application on the device. Adobe-like UI enhancements and other Web applications can be accelerated using the built-in support for Open GL in hardware through well defined Open GL APIs. Additionally, the MSS along with the camera and microphone inputs can be used to drive a video communication

application which can be either VoIP based or on a local dedicated home network within an apartment or community complex.

The ecosystem of the OMAP3515 device, coupled with the ease of programmability for the Cortex-A8 processor core and software system can enable a rich applications canvas for web connected devices.

### Conclusions

In this article we have covered the system design of a Home Internet Device which can be used as a companion device to access your favorite content on the Web for the TV. We have presented a complete programmable platform based system and software architecture design. The advantage of this design is that it is simple, yet scalable to handle the ever-changing demands of Internet multimedia. The programmability and power of the Cortex-A8 processor makes the design scalable to handle new media technologies as well as applications overtime.

END

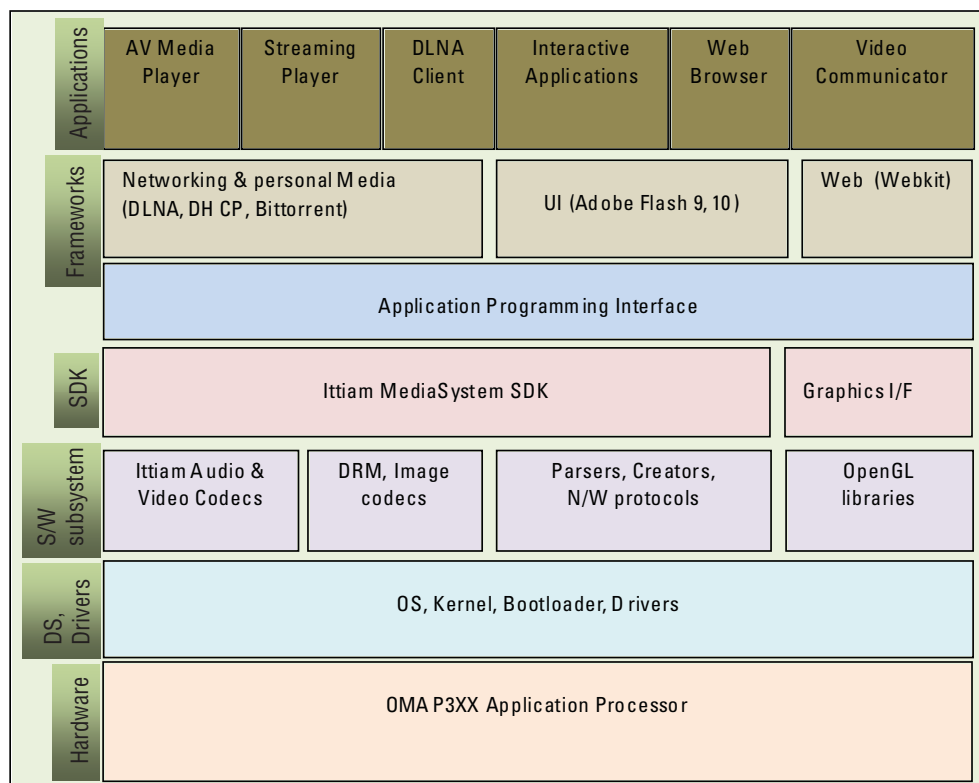


Figure 2: Software Architecture

# The proven Virtual Platform leader

It's like hardware...  
before RTL is available!

## What is a Virtual Platform?

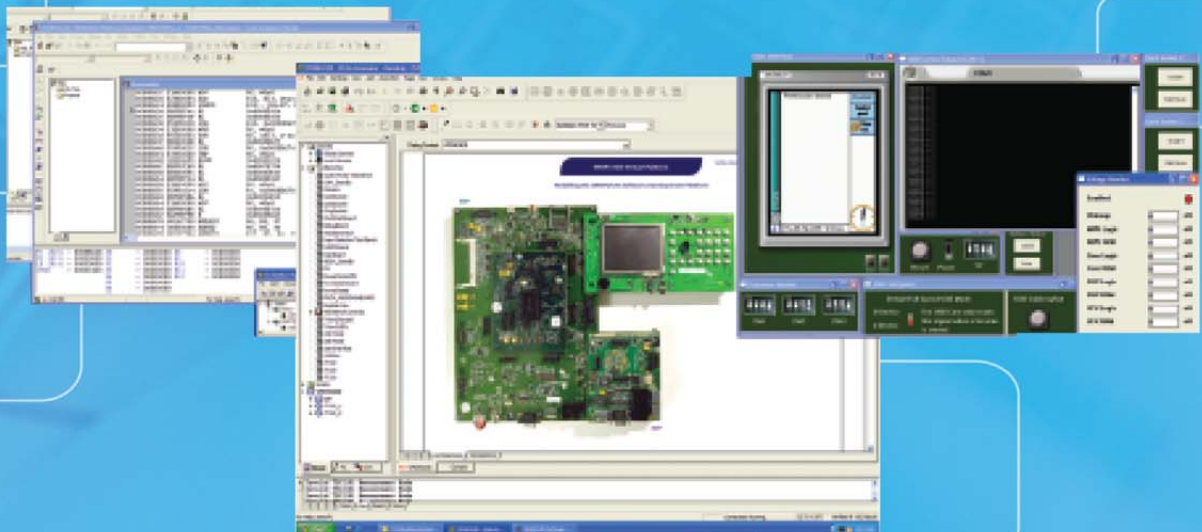
- Transaction-level model of a complete system
- Fully functional model – runs unmodified binary images
- High performance model – boots OS in seconds

## What can I do with Virtual Platforms?

- Shorten design cycle by 9 to 12 months
- Integrate software and hardware continuously
- Reduce project risk by performing system validation before hardware is available

## Proven Success!

- Over 50 commercial platforms in the hands of software developers
- Models the most complex mobile platforms
- TI OMAP1<sup>®</sup>, OMAP2<sup>®</sup>, OMAP3<sup>®</sup>; Freescale i.MX, MXC; Intel<sup>®</sup> XScale<sup>™</sup>, Marvell PXA3xx



See how virtual platforms are a solution for addressing slipping schedules in software-dominated chip designs. Virtual platforms are an effective development environment for drivers, middleware, OS and application software. They allow pre-silicon software development and also link effectively to verification and power analysis.

Download virtual platform white papers at [www.synopsys.com/whitepapers](http://www.synopsys.com/whitepapers)