

Open source

Handhelds

— a broadcaster-led innovation for BTH services

**François Lefebvre (Project Leader), Jean-Michel Bouffard
and Pascal Charest**

Communications Research Centre, Canada

Emerging *Broadcasting to Handhelds* (BTH) technologies could be used to convey much more than the usual audio or video programming. For a long time now, broadcasters have imagined and standardized many new multimedia and data applications which, deplorably, did not succeed in the market.

In the first part of this article, we suggest that the *open source handhelds* which have become prominent as a consequence of recent technological trends, could also bring the emergence of broadcaster-led applications on mobile devices. In the second part, we will introduce the Openmokast project and describe how the CRC was able to produce, with very limited resources, the first open mobile-phone prototype, capable of receiving and presenting live broadcasting services.

There is a growing enthusiasm today for BTH services as presented in EBU Technical Review by Weck & Wilson [1]. These services can either use broadcast standards such as DAB/DMB and ATSC-M/H or the standards proposed by the mobile telecommunications industry such as DVB-H or MediaFLO. These technologies provide efficient delivery mechanisms for up-to-date information, popular media and valuable data services to mobile users. In the present interest for rich media convergence, BTH and mobile technologies have complementary features. BTH infrastructures promise to transmit large volumes of valuable content in one-to-many communications while wireless telecommunications networks could provide the channels for one-to-one exchanges.

This interest has led to the development of many new standards for BTH services in the context of DAB: MOT transport, Broadcast Web Site (BWS), SlideShow, DLS, TopNews, EPG and TPEG. But only a few of these standards have actually been implemented on commercial receivers. For example, BWS, which could be used to provide very attractive information services, is generally not supported on current receivers. In the remainder of this article, we will refer to those unsuccessful applications as the *missing apps*.

The stagnation of BTH technological advances could be explained by the innovative and competitive wireless communications ecosystem that is thriving today. Several kinds of new wireless communications technologies are emerging in the quest to reach mobile users wherever they are with a maximum throughput.

It appears as if BTH is standing at a juncture between broadcasters and mobile network operators (MNOs). Their business models are in conflict. Broadcasters are naturally inclined to pursue and

extend their current free-to-air (FTA) services which are monetized by public funding, licence fees and advertising. MNOs, on the other hand, plan to deploy BTH services to generate new revenue streams through cable-like subscriptions and pay-per-view models.

The Internet is also challenging the traditional broadcasting model. It has led to rapid innovation cycles that produce direct and disruptive benefits to end-users. For example, webcasting, peer-to-peer streaming and podcasting all represent new and attractive alternatives to broadcasting.

We suggest here that one of the key limiting factors for broadcaster-led innovation lies in the broadcasters' limited control over the implementation of their standards into receivers. The market for broadcast receivers is horizontal. Implementing broadcaster-led standards into mobile phones is an even bigger challenge because these devices are part of a vertical market. That is, MNOs have a firm control over which feature sets get implemented into "their" devices. Understandably, they will likely promote their own feature sets before those of broadcasters. As a consequence, we can expect that BTH innovations that are broadcaster-led are less likely to be implemented into mobile phones. Von Hippel's theory [2] would suggest that as empowered "users" of the mobile phone technology, MNOs stand in a much better position to innovate and compete.

The following section will introduce emerging trends that could create new opportunities for broadcaster-led innovations in the BTH space. Please see *Appendix A: "Anatomy of a Handheld"* for an overview of the components and terminology that will be referred to throughout the remainder of this article.

NOTE: Throughout this paper, the term *handhelds* is used to refer to generic pocket-sized computing devices which may provide connectivity to any kind of networks. On the other hand, mobile phones are a specialized category of handhelds that connect to MNOs' infrastructures.

Trends towards broadcaster-led innovation

Specialized hardware goes generic

The manufacturing of handhelds is expensive. Mass-production is imperative to reach profitability. This results in steep barriers to entry for newcomers in the field. Fortunately, the relentless push of Moore's law has permitted the implementation of generic, compact and powerful integrated circuits. These components can be produced affordably and run on little power. This benefits smart phones too.

Inside today's mobile devices, specialized hardware components are increasingly being replaced by generic ones. Flexible application processors (APs) can be re-used in the design of many types of devices, thereby lowering the overall production costs associated with specific applications.

Functionality goes software

In early-generation mobile phones, user applications were performed by low-level software loaded directly on the device's permanent storage. Only basic tasks were supported: dialler, phonebook, device settings or ringtone selection. There was no room for new applications. Nowadays, generic and powerful APs coupled with large storage capacities have led to better capabilities. This has also significantly raised the importance of software in handhelds.

As a consequence, functionality features – based on software – present much lower barriers to entry than for hardware because of the fundamental properties of software:

- it can be duplicated at no cost;
- it can be distributed instantly and at no cost;
- it can be developed with low-cost or free tools;
- it can be modified, fixed and enhanced with no impact on the manufacturing chain.

Software goes open

The software industry is experiencing changes as open source software (OSS) is gradually entering the domain. With OSS, the global level of collaboration adds significant value to the software development chain. Individual programmers and organizations work at innovating and solving issues with a “let’s-not-reinvent-the-wheel” approach. Instead, they build together a solid common core which sustains their respective business models.

In our opinion, the qualifying term “open source” is not in itself very significant. The important factor is to apply the proper rights to the code, once it is written. This is where software licensing comes into play. Projects are said to be free/libre Open Source Software (FLOSS) when their licensing terms offer flexibility in the usage rights while remaining accessible to the largest possible community. The Free Software Foundation classifies software licences and promotes those that are, in accordance with their criteria, genuinely open. The Open Source Initiative (OSI) is also a recognized reference body that leads the reviewing and approving of licences that conform to the Open Source Definition [3].

There is an obvious trend in motion now. Industry has adopted many open source software solutions. Several OSS products are widely deployed: GNU/Linux, Apache, OpenOffice, Firefox, Asterisk, Eclipse and MySQL. Companies using these tools see several benefits in the form of flexibility, independence, ability to fix, to enhance and to tweak the software that they need. For all of those involved, OSS is a key to success.

The popularity of OSS has even reached one of the technology-sector bastions. We can now purchase consumer electronics (CE) products that do “run” on OSS. Here are some important examples of such devices:

- The Linksys WRT54G Wi-Fi wireless router [4] is a product for which the GNU/Linux firmware was released. Since this release, users have enhanced its functionality to enterprise-grade routers.
- Neuros [5], an Internet set-top box manufacturer, will shortly release its next product called OSD2. Neuros relies partly on users and external developers to create or integrate new applications for their platforms which also use OSS.
- The DASH Express [6] is a new type of GPS car navigation system with a two-way telecommunication data channel. It can receive real-time traffic data information and can also offer Internet search for points of interest. The DASH device came on the market a year ago and seems to be very successful in the USA. Interestingly, DASH is based on the FIC Inc. Openmoko first hardware release. The DASH is a good example of a vertically-integrated device. It is a great case study of how OSS can be used with the right licensing scheme on closed business models.
- Some developers have also enabled OSS frameworks on top of closed CE devices. The Rockbox project [7] creates open source replacement firmware for several brands of portable digital audio players like Apple, Archos and iRiver. Audio codecs are amongst the numerous features added: FLAC, WavPack, AC3 (A/52), AAC/MP4 and WMA. These were not available on the iPod models sold by Apple.

Handhelds go open source

Today, there is an important push towards OSS on handhelds promoted by many industry players. Their interest to participate in the mobility value chain is motivated by the current trends described above. Some key projects with the potential to impact BTH are described in this section.

Openmoko

Openmoko [8] is an OSS project that was initiated by First International Computer, Inc. (FIC Inc.), an important manufacturer of motherboards for personal computers. Early on, the company made the bet that their best option to become competitive with their new smartphone products was to open up a complete software stack that would enable and leverage user innovation. In July 2007, FIC Inc. released its first “developer preview” prototype called the Neo 1973 with the Openmoko software stack.

To many developers, the Neo represents the first truly open mobile phone platform. The Neo incorporates several interesting connectivity options: GSM, GPRS, GPS and Bluetooth. Interestingly, the Neo 1973 initially shipped with a primitive software stack that did not even allow the completion of phone calls. We had to wait a few more months before the “phoning feature” became available through software updates.

The Neo FreeRunner (GTA02), was the next version of the device. It was released in June 2008 with enhanced usability, hardware improvements and Wi-Fi connectivity.

Openmoko was conceived to enable various business models. Openmoko Inc., the company, does sell devices for profit. Independent developers can sell proprietary software applications thanks to the LGPL licence which covers the Openmoko software stack. With the DASH Express, FIC Inc. has also hinted that vertically integrated devices could be constructed on Openmoko. With such a framework, both closed and open applications are on a level playing field for competition. In this environment, users are just “one click away” from one or the other type of applications. The end-user will decide which best fits his/her needs.

Another interesting fact is that shortly after the release of the FreeRunner, another developer community was able to successfully port the Qtopia OSS distribution onto the device. Qtopia had been developed some years before by Trolltech, a company acquired by Nokia in 2008. At the current time, Koolu, a Canadian company, announced that it will port Android (another OSS distribution introduced in the next section) to the FreeRunner by the end of November 2009. This shows how organic and efficient an OSS ecosystem can be. Just a few months after its introduction, the FreeRunner device can already host several new software platforms.

Android

Android [9] was originally conceived to be the fundamental building block of new mobile devices. It is a software distribution that includes an operating system, a middleware layer and some key applications. It is being developed by the Open Handset Alliance (OHA), a consortium of 34 members including Google, HTC, T-Mobile and other important players in the field.

The Android Java-based software development kit (SDK) was released in November 2007 to allow application development long before any Android device was even produced. Since then, an application development contest sponsored by Google was initiated to stimulate the development of new Android applications. A total of US\$5 million was awarded to 50 submissions featuring the most innovative applications [10]. The first Android-based mobile phone (T-Mobile G1) was released in October 2008 in selected markets.

Initially, the degree of openness of Android was limited despite the fact that the SDK was available for free. The OHA recently decided to release the Android open source project [11] which will encompass most components of the Android platform. With the Android OS and virtual machine becoming open, new exciting development projects are now possible. This could even lead to the creation of new hardware components. Details to come about the licensing and the governance of the project will ultimately reveal to what extent Android will be open. But in its current configuration, Android still presents a compelling option for the design of open source handhelds.

Other open mobile platforms

Other open platforms which do not include mobile phone network interfaces are available to create new handhelds as well. These devices represent potential candidates for broadcast reception.

The Nokia Internet tablet computers based on Maemo [12] are such devices. The Maemo platform provides lots of functionalities and shows great potential for many useful usage scenarios. Maemo is a software platform that is based on OSS projects such as Debian GNU/Linux and GNOME.

The Ubuntu Mobile Edition is another example of a GNU/Linux based software stack that could fulfil the requirements for new handhelds. This effort was launched by Canonical Inc. to support the development of an OSS distribution for mobile internet devices.

Another potential major advance for OSS on mobile phones could come from Nokia. The company announced the creation of the Symbian Foundation and the release of its Symbian OS as an open source software [13]. This OS was designed for mobile devices and comes with libraries, user interface frameworks and reference implementations. Symbian, with a market share currently surpassing 50%, is still the platform deployed on most smart phones in the world.

The Openmokast project

None of the open software frameworks and devices encountered in our study did support digital broadcasting hardware. Since BTH is a main field of research for our group at the CRC, we were motivated to explore the possibility of integrating broadcast functionality into such an open device. When FIC Inc. announced in February 2007 the launch of their open mobile phone prototype using the Openmoko framework, we decided to initiate the **Openmokast** (“OPEN MOBILE broadKASTing”) project in our lab.

The aim of the project was to integrate a DAB receiver in a fully-functional mobile phone. We would design, build and test, with a live DAB signal, a prototype capable of decoding typical DAB audio services as well as some of the *missing apps*. Based on our previous experience in the lab, we chose to work with GNU/Linux and other OSS packages. We have learned that using OSS accelerates the integration of a prototype by reusing common SBBs found in those packages. In order to build a final product, we had to find a DAB reception platform and integrate it into the prototype. Other major software components such as the receiver control unit, the bitstream demultiplexer and the decoder had to be developed from scratch. We even had to manufacture a physical extension to the original handset to be able to embed the small USB DAB receiver and its required antenna into our prototype.

The Openmokast software platform

CRC-DABRMS is a stable software platform developed previously at the CRC to control commercial computer-based DAB receivers. This original effort provided access to raw DAB bitstreams on typical personal computers. CRC-DABRMS can decode signalling information contained in the fast information channel and dispatch desired sub-channels to various types of outputs. This in turn permitted the demonstration and testing of new applications geared toward DAB but not yet standardized. This system had been implemented for Windows and GNU/Linux platforms.

The GNU/Linux version of CRC-DABRMS was ported to the Openmoko platform and renamed *Openmokast*. Porting it involved recompiling the application for the new target AP. It also meant adapting the code while verifying that all required libraries would be available on the new platform at runtime. The architecture of the Openmokast middleware is shown in *Fig. 1*.

The original interface of Openmokast was the command line. Later we developed a GUI using Openmoko’s GTK libraries. Screenshots of a running Openmokast are shown in *Fig. 2*. At start-up, Openmokast presents a menu where the input device must be selected (*Fig. 2a*). The system can

also accept, as input, a locally-stored DAB multiplex file. This feature enables off-line testing and development of new applications without the need for a physical receiver and a live signal.

Different applications were either developed or were direct integrations of existing OSS projects. The standard DAB radio application was done with an HTTP wrapper which forwards MP2 audio to the Mplayer media player. A package for Mplayer was readily available. The DAB+ application was constructed in the same manner except that the transport protocol had to be removed prior to forwarding the AAC+ stream to Mplayer.

Two data applications were integrated by reusing source code made available under the OSS project called Dream [14]. Dream is an SDR receiver for DRM which includes an MOT transport protocol decoder, as well as two of the *missing apps*: Journaline and Slideshow (Fig. 2c). The packet mode decoder had to be developed in-house. Since we were able to re-use code from other OSS projects, these applications could be developed quickly and efficiently. This experience reinforced our views that OSS carries significant benefits for developers.

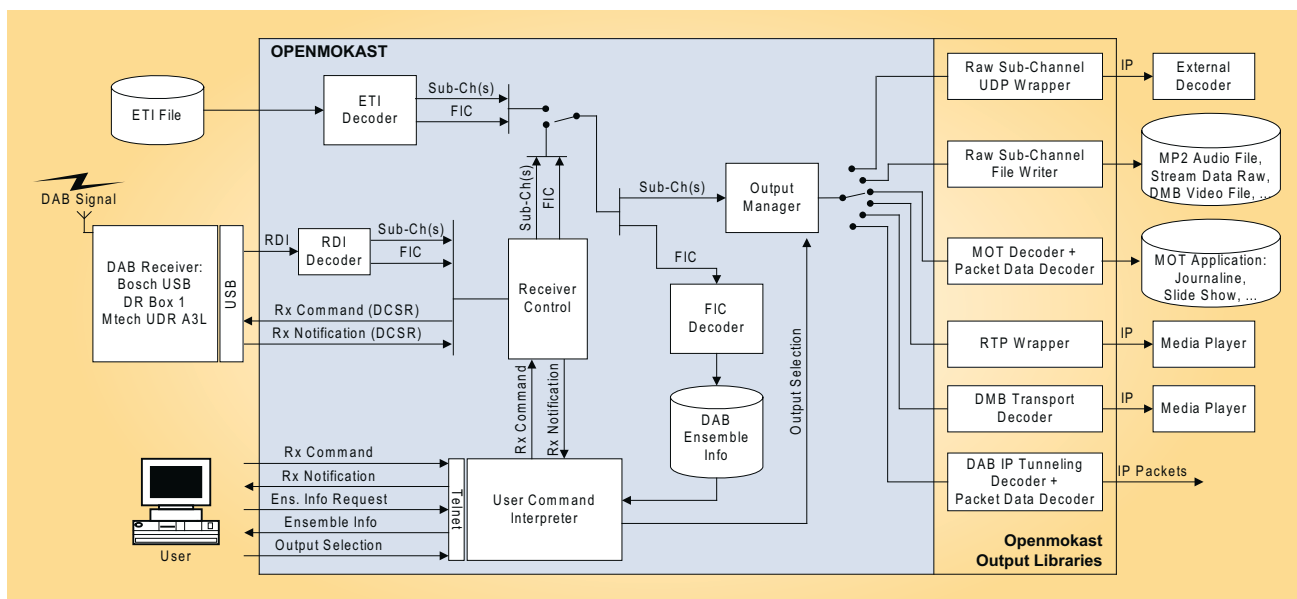


Figure 1 Architecture of the Openmokast middleware

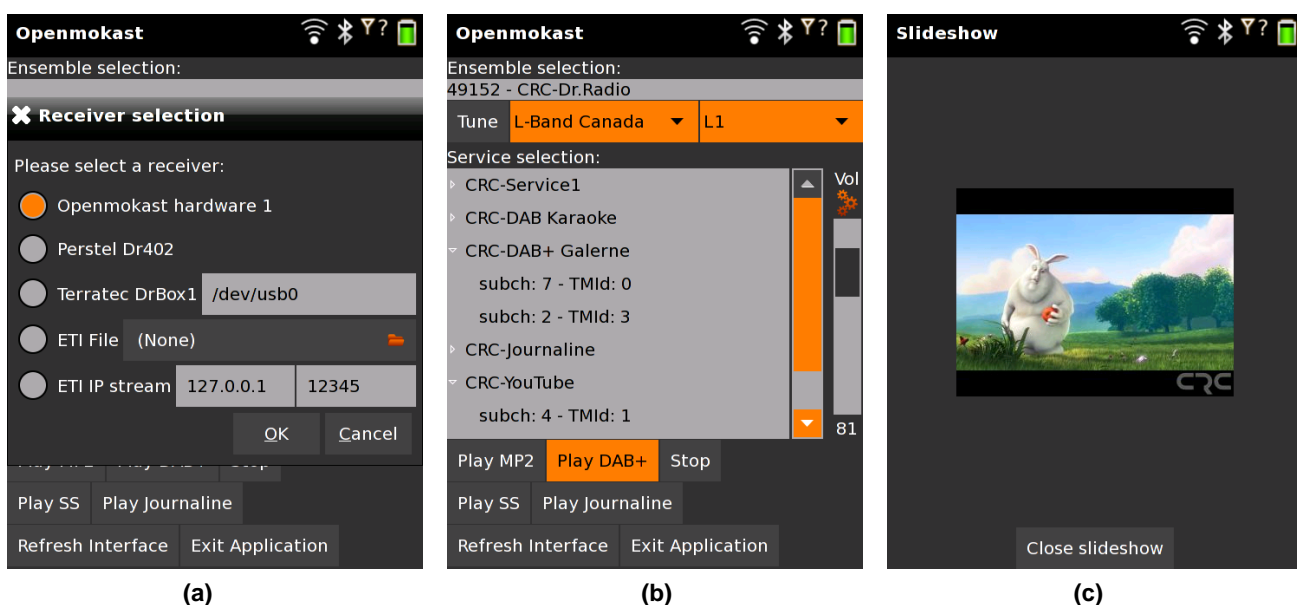


Figure 2 Screenshots of a running Openmokast device

The DAB receiver

A fundamental component required in the design of the prototype was a DAB receiver. Our early analysis suggested that a USB receiver could be a good fit for the unit. First, the FreeRunner has one USB port available. We also knew that the FreeRunner's AP was not powerful enough to perform DAB signal demodulation in real-time. We therefore searched for a USB component where this task could be performed efficiently on silicon.

We first tested the USB Terratec DrBox. The device drivers (DABUSB) for this device were already included in the Openmoko kernel. This should have made the integration task easier but unfortunately, the firmware for the unit could not be uploaded correctly. We had similar experiences with other USB receivers that did not succeed our initial test runs.

An alternative in our quest for the appropriate receiver was to obtain development kits offered by chipset makers. We contacted many companies but none could provide the components required. In general, their offerings did not meet the needs of a research development project like ours. Those companies were unwilling to support our initiative which would likely not lead to significant device sales. Furthermore, the development kits offered are expensive and the USB device drivers provided are usually built for Windows instead of GNU/Linux, and the usage of those kits requires the signature of non-disclosure agreements. We did not want to follow such a path of development and were stalled for a while.

We continued our research and found an off-the-shelf product which could fulfil our requirements. The MTECH USB key model UDR-A3L had all the features we needed. The libusb library was chosen to communicate with it. Libusb is a Unix suite of user-mode routines that control data transfer between USB devices and the host systems. We were able to establish basic communications between the MTECH key and the FreeRunner USB port. A reference to this key was added as a new input source in Openmokast (hardware 1 in *Fig. 2a*).

Prototype construction



Figure 3
ABS extension for
FreeRunner

The openness practices promoted by the Openmoko project went beyond our initial expectations. We found that even the CAD drawings for the FreeRunner mechanical casing were released to the public and available freely for download. Using those drawings for reference, we then modified the original mechanical design and produced a clip-on plastic extension which provides the extra space inside the device to embed the stripped down USB key. *Fig. 3* shows this extension.

To manufacture the physical extension part, we contracted the services of a 3D printing shop. We sent them the modified Pro/Engineer formatted 3D file and received the finished ABS part within 48 hours for the price of US\$90. We were happy to find that just like with OSS, even mechanical hardware components of a prototype benefit from the “democratization” of production means. The CRC will release

the schematics of this extension under a non-restrictive *creative commons* licence.

Fig. 4 shows the thickness of the final Openmokast prototype.



Figure 4
Comparison of thickness between the original
device and the Openmokast prototype

The MTECH receiver was connected directly onto the internal USB test points on the printed circuit board. In this configuration, it could draw its power from the device's main battery. This setup also had the advantage of freeing the external USB connector on the handset. This port remains the most convenient way to recharge the device. Once disassembled, the Free-Runner and extension exposed enough internal free space to install the dedicated L-Band antenna.



Figure 5
Inside the Openmokast prototype

Fig. 5 shows the prototype's internals while Fig. 6 shows the final product. Notice the colourful "skin" with our organization's brand as well as the Openmokast logo.



Figure 6
The final Openmokast prototype

Some results

We are satisfied with the overall test results of the Openmokast prototype so far. This device was put together by a small team in a short period of time and it has performed well since its release. The form factor of the device is appreciated by current testers. Some DAB *missing apps* which are usually not available on commercial receivers could be demonstrated.

The Openmokast prototype was introduced at the IBC 2008 exhibition and was hailed as the first open mobile broadcasting handset .

Abbreviations

ABS	Acrylonitrile Butadiene Styrene (thermoplastic)	GPS	Global Positioning System
AP	Application Processor	GSM	Global System for Mobile communications
ATSC	Advanced Television Systems Committee http://www.atsc.org/	GTK	Graphical user interface Tool Kit
BTH	Broadcasting To Handhelds	GUI	Graphical User Interface
BWS	(DAB) Broadcast Web Site	HTTP	HyperText Transfer Protocol
CAD	Computer-Aided Design	IP	Intellectual Property
CPU	Central Processing Unit	LGPL	(GNU) Lesser General Public Licence http://www.gnu.org/licenses/lgpl.html
DAB	Digital Audio Broadcasting (Eureka-147) http://www.worlddab.org/	MNO	Mobile Network Operator
DLS	(DAB) Dynamic Label Segment	MOT	(DAB) Multimedia Object Transfer
DMB	Digital Multimedia Broadcasting http://www.t-dmb.org/	OHA	Open Handset Alliance http://www.openhandsetalliance.com/
DRM	Digital Radio Mondiale http://www.drm.org/	OS	Operating System
DVB	Digital Video Broadcasting http://www.dvb.org/	OSI	Open Source Initiative http://www.opensource.org/
EPG	Electronic Programme Guide	OSS	Open Source Software
FLOSS	Free/Libre Open Source Software	SDK	Software Development Kit
FTA	Free-To-Air	SDR	Software Defined Radio
GPRS	General Packet Radio Service	TPEG	Transport Protocol Experts Group http://www.tisa.org/

The overall performance of the receiver component is good. The MTECH provided good signal reception in the L-Band and Band III frequency ranges, under various conditions. The device could receive signals coming from either the CRC-mmbTools LiveCD transmitter [15] or from standard commercial equipment.

The total CPU computational load measured for real-time DAB and DAB+ audio decoding on the FreeRunner was low. The GNU/Linux *tops* utility was used to estimate the processing cycles for two audio decoding scenarios (*Table 1*).

Table 2 shows the power autonomy estimates based on measurements made during three different usage scenarios.

Table 1
CPU usage for two audio decoding scenarios (%)

	Codec type	Bitrate (kbit/s)	Mplayer	Open-mokast	DAB+	Total
Scenario 1	Musicam	192	12.70%	1.00%		13.70%
Scenario 2	HE-AACv2	64	14.30%	0.60%	0.60%	15.50%

Table 2
Openmokast power consumption and autonomy with 1200 mAh battery

	Source	Measured consumption	Estimated autonomy
Scenario 1	Receiver in Band III	650-670 mA	1h49
Scenario 2	ETI file	220-230 mA	5h20
Scenario 3	None	190 mA	6h19

The Openmokast software was installed and tested on typical GNU/Linux PCs. The code could execute successfully on an emulator of the Neo 1973 available through the Openmoko project. In this setup, both Openmoko as well as the Openmokast GUI could be tested. We could not repeat a similar test for the FreeRunner configuration since no emulator for this device exists to date.

Openmokast's software could also be tested directly on two different GNU/Linux distributions, without the need for an Openmoko device emulator. We can conclude from these experiments that we could deploy Openmokast on other platforms and operate DAB devices in those environments.

Opening Openmokast

During this project, we had to face the issue of Intellectual Property (IP) in relation to OSS implementations. In fact, it is important to realize that implementing most of today's international open broadcast standards implies using proprietary IP. As a consequence, an implementation like Openmokast has to include proprietary algorithms or techniques to fully implement such a standard. Most of the time, licence fees must be paid to the rightful holder for each implementation of technology "sold".

Consequently, standards including proprietary IP appear to be a barrier to OSS implementations for two reasons. OSS projects are given away for free and do not generate revenues. It is also impossible to control the distribution of such software. Therefore, it would be very hard to collect any licence fees. Some countries have recognized the need for free standards and are promoting the adoption of new standards that are freely available and that can be implemented at no charge. The issue of open source and standardization was discussed in a report commissioned by ETSI in 2005 [16].

One possible solution for implementations of non-IP-free standards in OSS is to design frameworks with licensing schemes that allow the integration of IP-free as well as non-IP-free modules. With this approach, the non-IP-free components must be extracted from the overall distribution. A carved-up framework can be widely distributed along with the IP-free modules. The non-IP-Free components have to be distributed separately with provision for the correct attribution of the licensing fees. We plan to use this approach for the Openmokast project. The framework could be distributed openly without the module that requires special licensing (MP2 and HE-AACv2 decoding, etc..).

Conclusions

In this article, we have identified trends shaping the current environment that can impact the development of new BTH services. For a long time now, Moore's law has been the driver for advances in hardware and functionality that previously was provided by specialized circuits – but is now done on generic chips. The shift to software in the implementation of powerful handhelds is also a well-established trend. With software and its flexibility, developers have been able to create in a few months, thousands of new innovative applications for an Android or an iPhone.

We believe that a transformational force may be laying beyond the possibilities created by Moore's law and flexible software. The most important trend that we have identified during our study is that Open Source Software, once integrated on handhelds, carries an enormous potential for innovation. This is a revolution in the making that could reach its true potential once the right mix of collaboration and openness is found. Broadcasters, if they choose to follow this trend, could find new opportunities to promote their technologies. This could catapult broadcaster-led applications in the foreground and push further the deployment of new BTH networks.



François Lefebvre joined the Broadcast Technologies research branch at the Communications Research Centre, Canada, in 1999 to lead its Mobile Multimedia Broadcasting team. Since then, he has contributed to numerous national and international standardization efforts and R&D projects. His recent work has focused on creating and developing open software building blocks for next-generation mobile broadcasting networks, devices and applications.

Mr Lefebvre graduated from Laval University in Electrical Engineering where he also completed his M.A.Sc. in 1989. He then moved to Europe where he worked for ten years as an engineer in R&D laboratories and as a freelancer on several multimedia and Internet projects in Germany.

Jean-Michel Bouffard graduated in Computer Engineering (B.A.Sc.) in 2003 from Sherbrooke University, Canada. He then joined the Broadcast Technologies research branch at the Communications Research Centre, in Ottawa, where he is involved in projects related to mobile multimedia broadcasting systems.

Mr. Bouffard's expertise in multimedia communications and his interest in convergence led to studies about the applications of the participatory Web concepts to broadcasting and mobility. In the same vein, his current occupation focuses on enabling and promoting broadcasting on open mobile devices. He is also completing his M.A.Sc in Systems and Computer Engineering at Carleton University.



Pascal Charest graduated in Computer Engineering (B.A.Sc.) in December 2000 from Laval University, Quebec City, Canada. He then joined the Broadcast Technologies research branch at the Communications Research Centre, in Ottawa, as a Research Engineer. He has been involved in several research projects in the area of mobile multimedia broadcasting.

Mr Charest's recent work has focused on system building blocks and open source software, with emphasis on encoding, decoding, modulation, device control and system integration. In addition, he is a member of the Club Linux Gatineau where he has served both as the Vice-President and President in the past.



In this article, we presented several open handset projects, similar consumer electronics devices and the Openmokast prototype developed at the CRC. We believe, following our study, that broadcasters have an opportunity now to sponsor the development of broadcaster-led handhelds. If they did, chipset manufacturers could be encouraged to participate in the process. In fact, the Openmokast framework could easily be adapted to support other technologies such as DVB-T or ATSC-M/H by exploiting current building blocks redundancy.

In an open device with both BTH and mobile telecommunications network interfaces, we could hope that some synergies would happen. It is a simple matter of providing software on the handheld to bridge those two networks. With Openmokast, we can claim at last that we have reached convergence.

In an attempt to support our vision of convergence and the new opportunities that arise from it, the CRC plans to release, as open source software, several tools which were developed for the Openmokast project (<http://www.openmokast.org>).

Acknowledgements

The authors would like to thank their colleague Martin Quenneville for his work on the Openmokast prototype. Particular thanks should go to Karl Boutin for his thorough review and to André Carr, René Voyer, Bernard Caron, Silvia Barkany and Bruce Livesey for their comments and support.

References

- [1] C. Weck and E. Wilson: [Broadcasting to Handhelds — an overview of systems and services](#)
EBU Technical Review No. 305, January 2006
- [2] [2] E. von Hippel: **Democratizing Innovation**
MIT Press, 2006
- [3] <http://www.opensource.org/docs/osd>
- [4] <http://en.wikipedia.org/wiki/Wrt54g>
- [5] http://wiki.neurostechnology.com/index.php/OSD2.0_Development
- [6] <http://www.dash.net/>
- [7] <http://www.rockbox.org/>
- [8] <http://www.openmoko.com> — <http://www.openmoko.org>
- [9] <http://code.google.com/android/>
- [10] http://code.google.com/android/adc_gallery/
- [11] <http://source.android.com/>
- [12] <http://maemo.org/>
- [13] <http://www.symbianfoundation.org/>
- [14] <http://drm.sourceforge.net/>
- [15] P. Charest, F. Lefebvre: **Software DAB Transmitter on Live CD**
Published in the IASTED WOC 2008 conference proceedings, Québec, QC, May 2007
- [16] Rannou & Soufron: **Open Source Impacts on ICT Standardization**
Report – Analysis Part – VA6, ETSI

Appendix A: Anatomy of a handheld

Fig. 7 depicts typical hardware and software components on today's handhelds. The hardware building blocks (HBBs) operate around an application processor (AP). Given the portable nature of handhelds, special features support mobility: wireless connectivity, GPS and accelerometers are such examples.

Although APs can run most processing tasks, some of the "heavy lifting" has to be performed by specialized processors such as Digital Signal Processors (DSPs). The AP just does not have the processing power needed. Besides, even if it could manage the computational tasks, the related energy consumption requirement would be prohibitive. Instead, DSPs are used and these tasks can be efficiently accomplished while consuming less energy.

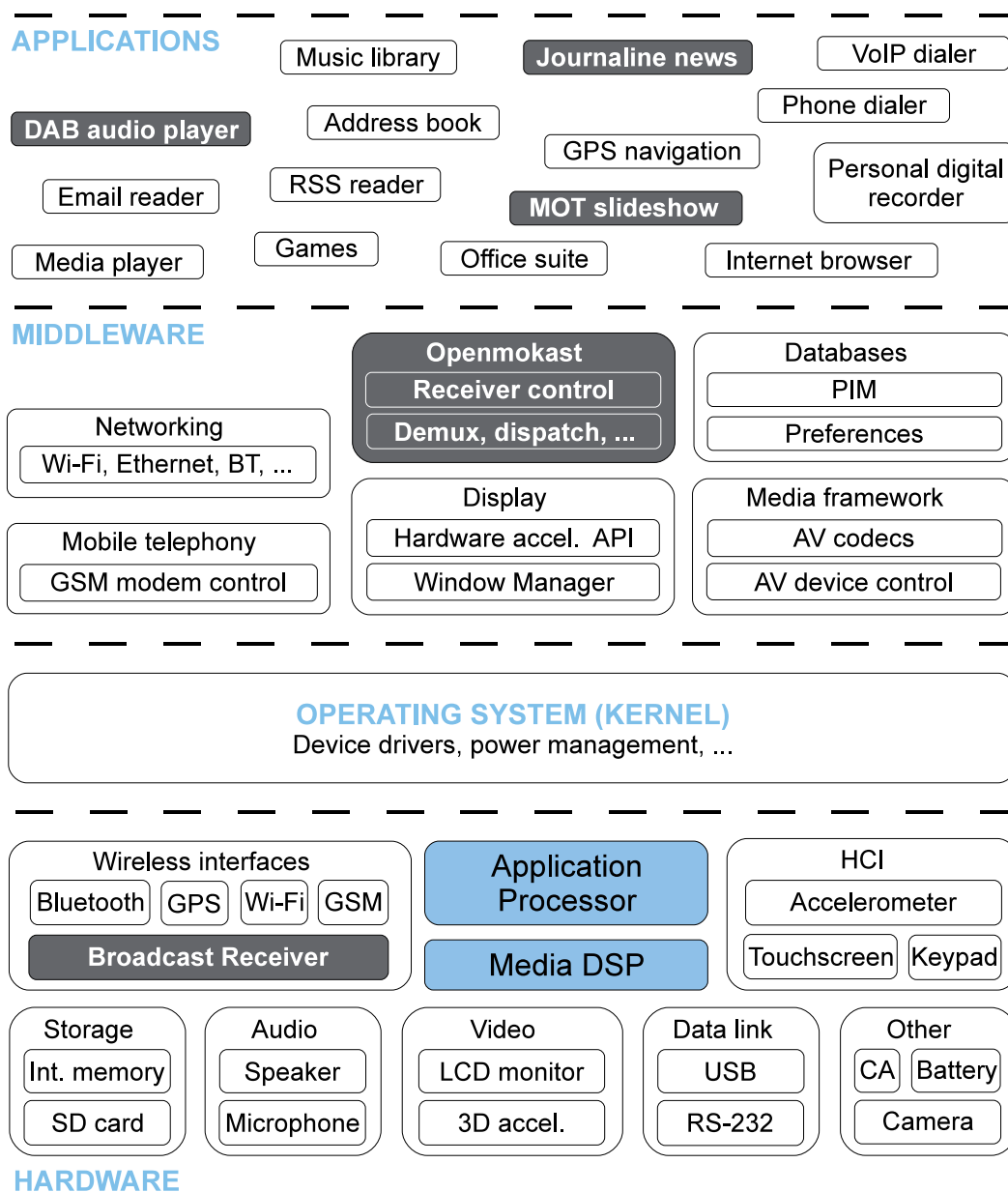


Figure 7
Typical hardware and software components of a contemporary handheld

One such set of DSPs are the wireless Hardware Building Blocks (HBBs) found on the receiver presented in *Fig. 7*. As an example, the broadcast receiver front-end filters the signal then digitizes and translates it. The signal then emerges at an intermediate frequency or directly at baseband. At this point, a DSP performs the demodulation to produce a bitstream that can be processed by the AP.

Media processing normally requires some degree of powerful yet efficient processing and is performed using DSPs. Most current systems include specialized media processing units to decode media streams such as H.264 video and AAC audio.

Another typical hardware component that needs consideration in the design of a handheld is the conditional access HBB. This function usually relies on hardware to perform access management for pay services. Fortunately, since this is not a requirement for FTA services, the design of broadcaster-led handhelds is simplified.

The rapid evolution of APs makes us believe that in the foreseeable future, software defined radios (SDRs) will perform the demodulation of broadcast and other signals in real-time, with versatile wideband front-ends and A/D converters on the AP itself.

Three main levels of software are also depicted in *Fig. 7*: the operating system (OS), the middleware and the application layer. There is no clear separation between the layers and some software building blocks (SBBs) could actually overlap two or three of those layers. A complete implementation, including all SBBs required to operate a device, is often referred to as a software stack or a distribution. It provides the device with all of the basic software needed to operate.

In the software stack described above, the lower layer components provide their “Application Programming Interfaces” (APIs) to the upper layer components. Device drivers present the APIs of HBBs to upper software components.

