Working Paper Series
ISSN 1177-777X

# Digital Libraries on an iPod: beyond the client-server model

**David Bainbridge & Steve Jones & Sam McIntosh & Matt Jones**

Working Paper: 06/2007
October 24, 2007


Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

# Digital Libraries on an iPod: beyond the client-server model

**David Bainbridge, Steve Jones, Sam McIntosh**
Department of Computer Science
University of Waikato
Hamilton, NZ
{davidb,stevej,sjm64}@cs.waikato.ac.nz

**Matt Jones**
Department of Computer Science
University of Swansea
Swansea, UK
m.jones@swan.ac.uk

## ABSTRACT

This paper describes an experimental system that enhanced an iPod with digital library capabilities. Using the open source digital library software Greenstone as a base, this paper more specifically maps out the technical steps necessary to achieve this, along with an account of our subsequent experimentation. This included command-line usage of Greenstone's basic runtime system on the device, augmenting the iPod's main interactive menu-driven application to include searching and hierarchical browsing of digital library collections stored locally, and a selection of "launcher" applications for target documents such as text files, images and audio. Media rich applications for digital stories and collaging were also developed. We also configured the iPod to run as a web server to provide digital library content to others over a network, effectively turning the traditional mobile client-server upsidedown.

**Keywords:** Mobile Digital Libraries, Open Source, Multimedia.

## 1 Introduction

Digital Libraries are increasingly commonplace resources we turn to for authoritative information. Starting first in the scholarly world, their public profile has increased significantly through initiatives at national libraries—such as the US Library of Congress" American Memory—and other egalitarian organisations, such as the Internet Archive. Access is predominantly (nearly exclusively) through the web, with the assumption that the user is working from a desktop machine.

Some research has been conducted with mobile devices in a digital libraries context, e.g. [1, 4, 5]. Typical usage sees a device such as a PDA with wireless networking, connecting to a central DL server through a web-based interface tailored for a small screen. We characterise this work as additive: fitting into the existing web-based client-server model. This is largely due to most categories of mobile device being currently "lightweight" on storage capacity, necessitating such a client-server model.

As with all things digital, continued technological development of mobile devices means the equipment goes faster and has more capacity as each year passes. In the case of mobile digital libraries development, however, there is a danger that the dominance of the client-server model encapsulated in existing DL software for mainstream use means that alternative modes of access might be overlooked. In this paper we report on our experimentation with going beyond this form of interaction; in particular we look at what happens when mobile devices have large-scale storage capabilities.

Our work is based on iPod-Linux[1] and Greenstone [6]. The former allows us to subvert a conventional iPod so we can compile and run software (written in C and C++) on it, namely the latter: Greenstone. Greenstone is a software toolkit that encapsulates over a decade of digital library research at the University of Waikato, home to three of the four authors of this paper. Downloaded hundreds of times a day, one of its key adopters is the UN, who use it to deliver full searchable content on humanitarian aid to developed countries on CD-ROM that can be run on "legacy" computers going back to Windows 3.1. Consequently Greenstone has a small execution footprint which lends itself well to the project in hand, in addition to the authors' familiarity with its code base. For this project its client-server model is picked apart and reshaped as needed.

The structure of the paper is as follows. First we provide details of the software foundation upon which the work was built, and a basic command-line test that turned out to be trickier to get working than one would have liked. The groundwork established, we discuss various types of digital library development on an iPod: an interactive menu applications with rudimentary "launcher" applications for text, image and audio; richer media presentations through the collaging of images and digital stories; and—turning the mobile client-server movdel on its head—running the iPod as a web server. We conclude with a summary of our findings.



[1] See *www.ipod-linux.org* for more details.

## 2 Software foundation

Most mobile devices—PDAs, phones and the like—are "lightweight" on storage capabilities. A device that counters this trend in the iPod (with, in 2007, 80 GBytes the norm), which allows for the intriguing possibility of running a self-contained mobile digital library.

Imagine being able to carry around a digital library in your pocket! Full, finger tip access to millions of items: text, image, audio, video, wherever you are. Really, *wherever* you are: no need to be within a wireless hotspot, mess around with ISP registration, or waiting for rich-media content to be transferred to your device, as necessitated by the mobile client-server model and the accompanying rapid depletion of your battery power that goes with all that communication.

Take the UN CD-ROMs on humanitarian aid that have been developed so far: there are over 40 of them. Given the full-text indexing with compression technology Greenstone utilises, all could be fitted on to one iPod. Alternatively, if literacy rates of end-users was of concern, the digital material could be audio or video based—over 1,000 hours of audio or 100 hours of video comfortably fit on to one 80 GByte iPod. If an village's information centre has a local area network, then why not allow the iPod to be connected to that and run a web server? This would allow those on the network shared access to the resource [2]. You wouldn't even have to do this with the latest, top of the range but pricey iPod model as there is a buoyant market in second-hand iPods. Even Apple sell reconditioned older models through their website for considerably less.

To explore such possibilities, we have combined iPod-Linux and the Greenstone Digital Library software. We provide general details about these two components, before explaining the experimental work we undertook.

### 2.1 iPod-Linux

iPod-Linux (*www.ipodlinux.org*) is an open source project, launched in 2004, with the aim of porting Linux to the various generations of iPod. Based on a default installation—which is achieved through a simple "Wizard" style graphical interface—a user's iPod-Linux experience is interacting with a top-level application called Podzilla, which serves as a substitute for the Apple iPod interactive menu application. Traversing a hierarchy of menus, many of the same features are available: for example, launching MP3 files (playlists artist lists, *etc.* accessed from the iTunes Database on the iPod) and having them play in the background while the user accesses further functions is clearly the most desirable of these.

Of course, in addition to this, one of the points of the project is that you can run other things as well. Podzilla is designed around a module architecture and thus highly configurable. Some of the example modules include games such as Mastermind and Space Invaders, utilities such as a calculator and colour picker; even the first-person shoot-em-up game Doom has been ported! Applications don't even need to be dressed up as a module, as one of Podzilla's applications is a file browser, which can be used to launch any such addition to the file system.


Figure 1: A second generation iPod booting up in Linux

The system of modules installed is highly configurable: the graphical installer also servers as an updater. The device can also be booted up in "Disk mode" a files copied across or updated that way.

The work is based around uClinux—a minimalistic distribution of Linux tailored for micro-controllers without Memory Management Units (MMUs)—and successful ports for 1st, 2nd, and 3rd generation iPods have already been achieved. Figure 1 shows a picture of a 2nd generation iPod booting up in Linux. For more recent iPods (4th, 5th, photo, video, nano) the core operating system functionality is provided, however kernel modules for interacting with devices are less developed; two omissions of note are audio in (recording) and network communication (Firewire/USB). As a result, these models are not (currently) officially supported by the project, but in practical terms following the well-documented installation procedures is as straightforward for these models as for the supported ones. At the time of writing, 2nd generation nano does not run Linux at all.

Fundamentally, to set up an iPod to run Linux one reformats its harddrive. This step is not as drastic as it first might appear, as the installation instructions take you through a procedure that partitions the drive such that the device becomes dual-boot if desired. It is therefore possible to retain your existing iPod data (music, photos and so forth), although it is strongly recommended by the developers that a copy of such personal data be taken prior to installation. It is even possible for the Linux side of things to be able to access this data file area, such as the iTunes Database.

As can be appreciated in a project where the full technical details of the device have not been disclosed, mess-

ing around with components at this level can have its hitches, although in our experience this is extremely rare. Apple provides software (initially a separate program, but now a feature of iTunes) capable of returning an iPod to its factory settings. So if things do get too badly scrambled, then there is a straightforward procedure to reformat the device as an Apple-only software "pure" iPod, upon which the installation procedure for Linux can be redone.

In the early days of the iPod-Linux project, reconfiguring an iPod to run Linux required the use of a series of command-line utilities. This has, for the main part, been replaced with a GUI-based installer that provides a layer of abstraction from this (as mentioned above). Done in the form of a wizard, clicking through the sequence of questions leaving values at their default will quickly yield an iPod running Linux for you to explore.

Considered in more detail, the information garnered by the installer controls whether the device is dual-bootable or not, which operating system is the default (if dual bootable), whether you want a copy of apple's operating system partition to be taken first (useful for restoration purposes, although not essential) and which software modules for the main graphical application, Podzilla, are to be installed. The command-line approach affords better control over setup, however unless you have especially unusual needs, the graphical installer is normally sufficient. Even if your needs include changing the version of the kernel used or having a new module (that was not previously an option in the installer) included, modest editing of a configuration file for the installer is all that is needed.

### 2.2 Development environment

Developing software to run on iPod-Linux requires a cross-compiler that is run on a desktop machine (the host), and then the generated executables are copied across to the iPod (the target). When development of the iPod application (or Podzilla module) is completed, the executables would ultimately be bundled together and included as part of the installer.

Development is most straightforward on a Unix-based host machine: precompiled binaries for the cross-compiler are available for Mac and Linux, for instance, and these operating systems can access natively the Linux partition on the iPod making it easy to adjust and update the installation with freshly compiled code.

Things are a little more convoluted when using Windows as the host but still workable. First, for the cross-compiler to work Cygwin[2] (a Unix-like environment) needs to be installed—precompiled binaries are available, however this port of the compiler is not as considered as reliable as the others. In our experience, the Windows cross-compiler worked fine compiling Greenstone and other application code, but failed to successfully compile the kernel. On Mac and Linux platforms all worked without a problem. Second, updating the iPod is more tedious under Windows because the host cannot natively access the iPod's Linux partition. We found LTools (Linux Tools) helped mitigate the latter somewhat by providing Windows command-line utilities and a graphical interface for manipulating a Unix (ext2 and ext3) formatted partition [7].

### 2.3 Greenstone

Greenstone is an open source digital library toolkit [6]. Used out of the box it provides the ability to create collections of digital content, to display the content in a web browser and to access and search the collections that have been built. Through UNESCO sponsorship the software is fully documented in English, French, Spanish, and Russian; in addition, its web interface has been translated into over 40 languages through volunteer efforts.

Countless digital libraries have been built with Greenstone since its public release on SourceForge in 2000: from historic newspapers to books on humanitarian aid; from eclectic multimedia content on pop-artists to curated first editions of works by Chopin; from scientific institutional repositories to personal collections of photos and other document formats. All manner of topics are covered: the black abolitionist movement, bridge construction, flora and fauna, the history of the Indian working class, medical artwork, and shipping statistics are just a random selection.

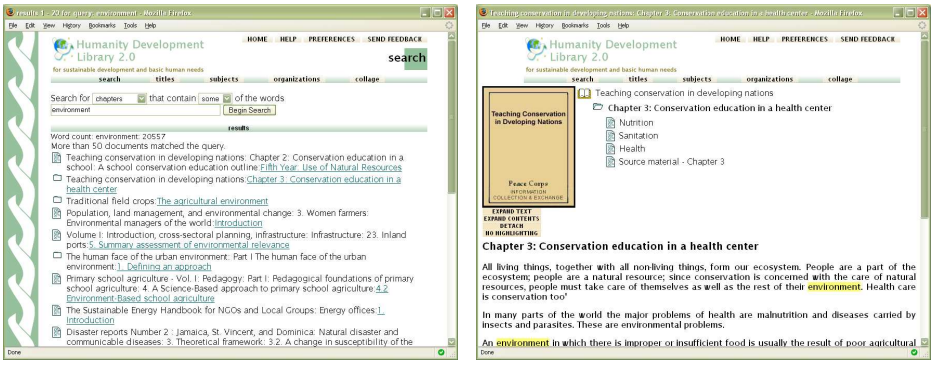
A wide variety of formats are accommodated, including HTML, PDF, OpenOffice, Word, PowerPoint, and Excel document formats; MARC, Refer, Dublin Core, LOM (Learning Object Metadata) and BibTeX metadata formats; as well as a variety of image, audio, and video formats. Greenstone also supports numerous standards including OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting), Z39.50 and METS (Metadata Encoding and Transmission Standard) to assist interoperability. Export options include Fedora, DSpace and MARC. See the web-site *www.greenstone.org* for more details.

An end-user's experience of Greenstone is through a web interface, such as the one shown in Figure 2, taken from the Human Info NGO's *Humanity Development Library*.[3] Documents in this collection can be searched by chapter title, in addition to full text searching by chapter or entire document. Alternatively, users might choose to browse alphabetically by title, or hierarchically by subject or organisation. In Figure 2(a) the user has searched within chapters for the word "environment" with a ranked listed of matches displayed; in Figure 2(b) the user is viewing the document that results from selecting the second matching item: Chapter 3 of *Teaching conservation in developing nations*.

Figure 3 shows the Greenstone Librarian Interface (GLI), a graphical application for creating and maintaining collections such as the *Humanity Development Library*. Through a system of tabbed panels accessed along the top of the interface, the digital librarian decides what files to include in the collection, what metadata is manually assigned (in addition to that automatically extracted by Greenstone from the source files), the collection's searching and browsing capabilities, and the customisation of presentation details.

[2] See *www.cygwin.com* for more details.

[3] *www.nzdl.org/hdl*




(a) Greenstone Collection interface (b) Greenstone Librarian Interface

Figure 2: Screenshots of Greenstone readers' interface.

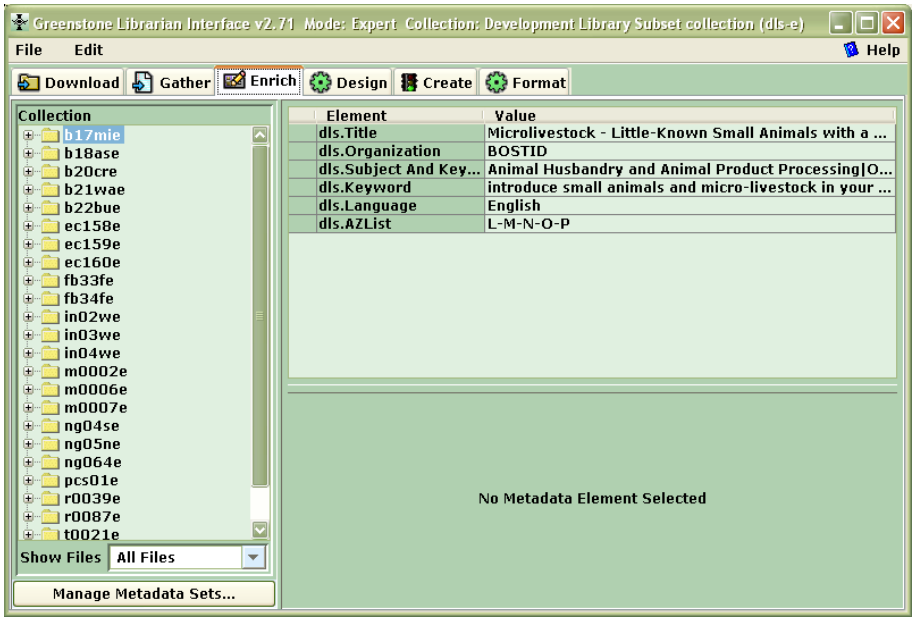



Figure 3: Screenshot of the Greenstone Librarian Interface

## 3 Software development and experimentation

Our development and exploration of digital library capabilities on an iPod can be divided into five areas: command-line test, interactive applications, collaging, digital stories and web server. We review each of these in turn.

### 3.1 Command Line Test

To immerse ourselves in the world of software development for iPod-Linux, the first goal we set ourselves was to compile runtime Greenstone (C++ code) such that is could be run from the command-line. Executing the compiled program—either through the file browser provided by Podzilla, or by adding it into the boot-up script—would not be particularly useful for a user of the iPod *per se*; Greenstone runtime is a CGI program and the result of running it is to generate raw HTML as its output.

There was little problem compiling the software. The cross-compiler available for the iPod is based around gcc/g++, a compiler that has regularly been used to compile the digital library project for over a decade. We were, however, surprised by the results of running the compiled program which instantly resulted in a segmentation fault the moment any printing or file input/output was attempted. Some of the issues we found documented on the *ipod-linux.org* site, the others we had to resolve from first principles.

First the well documented issues. There are in fact two cross-compilers in use for iPod-Linux: gcc/g++ 2.95 and 3.4.4. The former, which is a fairly old incarnation of the compiler by today's standards, is required to compile the version of the Linux kernel used; the latter is used mostly for applications for the iPod. It is well documented that the 2.95 version of the cross-compiler does not implement C++ IO terribly well, hence the moment Greenstone—which exclusively uses C++ IO mechanisms over C—does any file IO, it ends with a segmentation fault. We wrote several small test programs to confirm this.

Moving to using 3.4.4 solved the *file* IO issue, however a new complication was that now it was any printing

to the screen (standard out and error using the C++ objects `cout` and `cerr`) that led to immediate segmentation fault.[4] Perversely, file IO was now working perfectly and printing to the screen using C syntax (`printf`) was fine. Again a series of small test programs were devised to shed light on the problem, and led to the following conclusion.

Not many developers for iPod-Linux have been using C++ and its possible the problem of using `cout/cerr` syntax has not come to light if those that are coding C++ sure using C-level `printf/fprintf` syntax. The situation is quite likely, as `cout/cerr` is generally considered more verbose that the C approach. The C++ standard explicitly allows for freely mixing the two approaches with the same body of code, and while the C method is more arcane, once mastered is a more concise and faster way to achieve results making it an attractive choice. In the Greenstone codebase, full and consistent exploitation of the object oriented paradigm are embodies in its design, which is why `cout/cerr` syntax is used exclusively throughout the code.

Initialisation of global *objects* in C++ (as opposed to primitive types such as `int` and `float`) is a tricky component for a compiler to implement, due to the role constructors play in this, and the authors have had some experience with this failing to trigger correctly in previous project work. A sample program was written (defining a class, and instantiating an instance as a global objects) that did indeed confirm there was a problem in the cross-compiler in this regard. Extrapolating to `cout/cerr`, these are, of course, global objects and if they have not been properly initialised then it is not surprising that the program has a segmentation fault when they are used for printing. The hypothesis is consistent with what happens with C++ file IO since this is done in terms of local variables declared and used within the program; it also explained why C-based IO syntax both to file and to the screen embedded in C++ code worked unimpeded as they do not require global object instantiation.

In terms of looking for a solution that would allow us to run the Greenstone code on an iPod, we did not view replacing all `cout/cerr` lines with C equivalents as practical, or desirable as it would conflict with existing object oriented programming aims. The solution that was struck upon, after reading around the subject matter for some time, was to use a part of the C++ standard that governs the mixing of C and C++ IO. By default, these two approaches are defined to be synchronised: a `cout` in the code, followed by a `printf`, followed by a `cout` again strictly preserves the order of printing. The standard allows for this strict synchronisation to be relaxed with the call:

```
ios::sync\_with\_stdio(false);
```

If put as the first line of the `main()` program, the outcome of this—in terms of the problem we had—is that `cout` and `cerr` are re-initialised as a result of calling this—or rather, given the problem we had, these objects were finally initialised correctly for the first time. With this modification made, the command-line test of Greenstone ran without fault.

[4] Presumably `cin` is affected too, however our tests did not encompass this as it was not being used in the code being trialed.

### 3.2 Interactive Menu

The next step in developing Greenstone capabilities on an iPod was to shift to an interactive mode of access. There is a clear and direct mapping of the hierarchical browsing capability in Greenstone (in a web browser) to the iPod style of hierarchical menus.

To achieve this, we acquainted ourselves with the developer's guide for Podzilla and TTK (a GUI toolkit loosely specially written for iPod-Linux) and set about developing a new module for the application that included Greenstone browsing. The details of this are largely mundane, however a few comments are worthy of remark.

The iPod Greenstone module works directly from the files generated for a collection built on a host machine running a typical install of Greenstone (see Sectionr̃efsec:greenstone). The iPod module needs to be instructed, through a configuration file in the Podzilla modules folder, where the home file area to Greenstone is on the iPod, but once that is done the rest of the process in automatic. The browsing implementation centres largely around access to metadata information in a GDBM (Gnu's DataBase Manager) database. For each Greenstone collection the module finds the collection's name which is then appended as a child menu item to the main Greenstone menu item in Podzilla. After than, menu items are created on the fly as the user traverses the hierarchy. This contrasts with the first version of the software we wrote that build the full hierarchy of menus when the Greenstone module was initialised. Apart from this meaning Podzilla took a lot longer to load up at the start, we also started to experience out of memory errors depending on how many other modules were part of the Podzilla configuration.

Document "launcher" applications were written for for text, image and audio media types, so when a user browsed to to a leaf node in a Greenstone hierarhcy, an appropriate action could be triggered. Text and image launcher applications made use of the TTK toolkit; for audio, TTK was used in combination with accessing the Linux digital signal processing file-mapped device `/dev/dsp`, enabling the user to play, pause, fast-forward and rewind through an audio file.

In a subsequent phase of development, a search capability to the digital library on the iPod was added to round out the features set provided. Here we came up against issues relating to the minimal user interface of the iPod—perfectly designed for selecting and playing audio: click-wheel and five "push" buttons (play/pause, fast-forward to end, rewind to beginning, menu and enter). Our task was to decide how best to utilise this for text entry?

The iPod-Linux comes with various text entry widgets: from a full range of characters displayed linearly on the screen (a-z, 0-9, punctuation, *etc.*) and the click-wheel used to scroll forward and backwards, through to tapping out charters using Morse code. We felt we were unlikely to come up with any significant innovation in this area, and so allow the user to choose the form of text-entry widget they prefer from the existing list (this is actually already a feature of Podzilla) and acquire the text for query terms this way. Once acquired, initiating a query and displaying the search results is straightforward.

At the time of this work, there were rumours of a fu-

ture iPod generation soon to be released with a touch sensitive screen. This would solve many of the user interface issues, however, such an iPod is yet to materialise and in all likelihood it has now been eclipsed by the introduction of the Mac iPhone.

### 3.3 Collaging

Collaging is a technique devised by Kerne [3] for use in the more general environment of the web. In one example of its use, a user keys in several web domains—say *mtv.com*, *archive.org/audio*, and *ceolas.org*—presses *go* and then observes a visualisation of images drawn from crawling these web site concurrently. Images already in the visualisation fade over time and as new images are retrieved from the various sites, they are added in to the visualisation on top of what is already there. If at any stage a displayed image sparks some interest, the user can click on it and have a new web browser open up displaying the web page the image comes from.

Greenstone has utilised this idea based on collection source documents. In addition to browse by title or author, a more serendipitous form of browsing—collaging—is available. Based on images that are found in the source documents, be they the source document themselves, such as photos, or components of a larger work, such as artwork or figures included in Word and PDF documents, these are randomly shown over time, fading out as in Kerne's work. When a user click upon one, the relevant document the image is from is shown, locating the view of the document to be where that image appears.

In implementing a collage browser on the iPod, a view of graphics capabilities under iPod-Linux was conducted. A strong candidate that emerged from this was SDL. Written in C, SDL stands for Simple DirectMedia Layer and covers basic 2D graphics and sounds, through to fancier things such as 3D graphics, and video. It is a popular choice for many games. While the port to the iPod is patchy, it is still a popular choice for Ipod-Linux developing in many of the games that had been ported to the device, such as the afore mentioned Space Invaders. This was then combined with TTK's timer events, to allow images to be displayed at regular intervals.

The iPod collage version doesn't implement alpha values, as this capability is not available in the SDL port. It could be done in software, but this line of work was not pursued due to concerns over efficiency. Instead of alpha values, we decided to frame the top most image, and found this to be a satisfactory substitute based on overall anecdotal feedback from those that tried the system.

The affordance of the iPod's physical layout of buttons, designed for navigating and playing music, led to a novel variation to the standard collaging technique. Activating "rewind" through the click-wheel in the iPod collage was mapped to removing photos currently in the collage sequence, in a first in first out manner (akin to a stack operation). Activation "fast-forward" replaces the images in order. Such a feature does not exist in Greenstone's main implementation, nor the original by Kerne. To some extent, the ability in the iPod was introduced to help compensate for the lose of direct point and click with the mouse in the desktop versions, however in these versions it is still the case that sometimes a user clicks a spot where an image of interest is just as a new image is added that overlaps this region. The results is the new image being selected rather than the one intended, and worse it is sometimes difficult to get to that image as more images can overlap in the time it takes to recover from the initial mistake. An interesting line of investigation would be to evaluate the benefit of adding a comparable rewind and fast-forward feature to these main versions.

### 3.4 Digital Stories

Digital Storytelling is a technique pioneered by the BBC. In a format that is usually around 2 minutes long, a story such as a personal narrative is presented through digital media. The media component is typically encapsulated as video, and may involve a combination of music, still images—often incorporating a slideshow/screensaver feel with movement within the image through zoom and panning—augmented with text, and source video clips.

As an example of an enriched media object, it was decided to experiment with this form of artifact in a digital library context, and a proof of concept digital story player undertaken. While video makes an excellent carrier for delivery, from the point of view of ingest into a digital library, the item is a rather impoverished form of information. Having a more structured item that still retains what the logical components in the story are—such as this image is shown at this point in time, and then gradually panned in an north-east direction when this audio is played—is far more worthwhile. Playback is also rather trivial if being rendered as a single video option is pursued in terms of development work on the device since iPod-Linux already comes equipped with a video player (iPod photo and video versions).

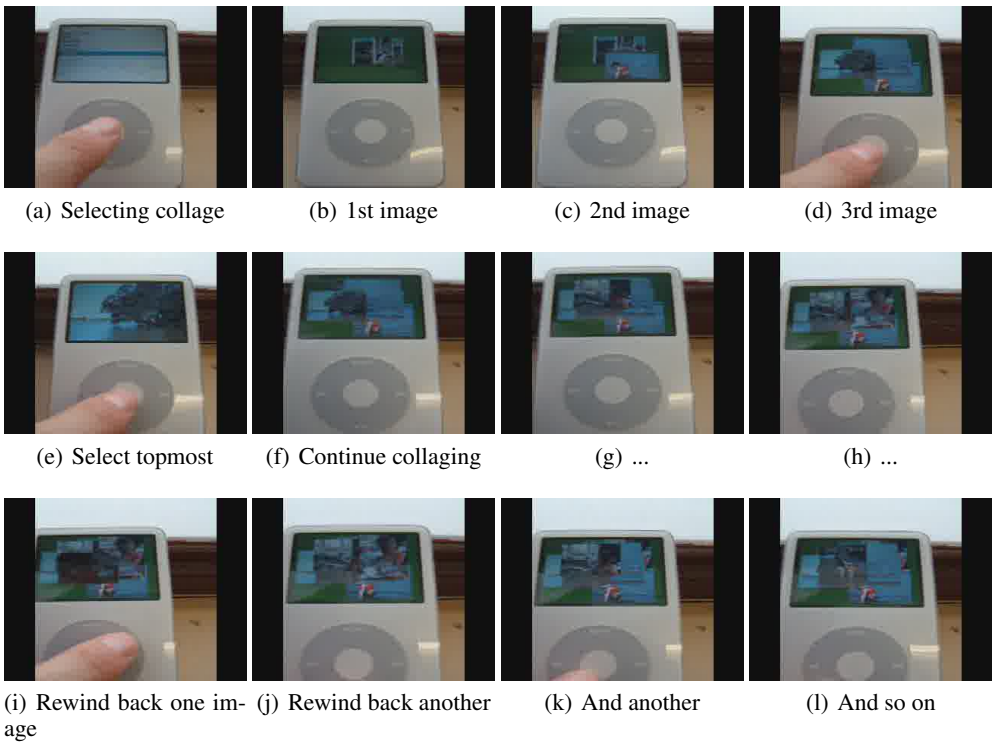
To develop a structured digital story capability, the following capabilities were used:

- SDL was again used to provide the basic image manipulation work.
- Linux file-map device support was used for audio (`/dev/dsp`).
- Timer events were used to control the synchronisation of the timeline between image display and audio buffering.
- The TTK render text ability to augment frames with textual content.

Stories at this stage are encoded as a data-structure (an array of digital story events) compiled into an executable. It would be a fairly straightforward matter to externalise the events to a file that is read in at runtime.

### 3.5 Web Server

Many of the key elements to running an iPod as a web server are around and (seemingly) tantalisingly close to realisation, however there are no reports of anyone successfully achieving this. There has been some discussion



(a) Selecting collage (b) 1st image (c) 2nd image (d) 3rd image

(e) Select topmost (f) Continue collaging (g) ... (h) ...

(i) Rewind back one image (j) Rewind back another (k) And another (l) And so on

Figure 4: Snapshot sequence of collaging on the iPod.

in the iPod-Linux forum on and off for a couple of years, but ominously in one case the "ticket" opened on the site's TRAC[5] system, requesting such a capability, was later closed (shortly afterwards) by one of the developers with the comment that they did not see any use for such a thing!

We can report there that we have indeed been successful in running the iPod as a web-server, generating both static content and running the Greenstone library CGI executable. There are quite a few hoops to jump through to set this up, which limits its applicability at the moment. We provide the details below. The open source nature of iPod-Linux means that this limiting situation can be rectified given time should this line of investigation be pursued further.

Existing work on the iPod-Linux site explains how to set up an iPod so you can Telnet into it. It relies on running IP over Firewire, comes with several caveats, and is a line of work carried out a couple of years ago and not actively pursued. This was our starting point for the development of a web server.

In theory IP over Firewire from a Linux host machine should be straightforward, given the general design of modules to the operating system. However, one of the iPod caveats is that the Ethernet module written for the iPod has a bug in it that requires a special Ethernet module compiled up (`eth1394`) and installed on the host machine. A following caveat—and a fairly large one at that—is that this new module does not work with Symmetric Multi-Processing (SMP) and preemptible kernels, necessitating a fairly conservative operating system configuration, which requires system administrator rights. This is still better than the situation for Windows and MacOS, where the iPod-Linux developer site provides a description of how this should work using IP over Firewire but then goes on to say it doesn't and nobody knows why!

The necessary Linux host machine (a decommissioned lab machine) was configured, but disappointing the specialised Ethernet module `eth1394` did not function correctly. Indeed, the act of loading it caused the machine to lock up and the machine could only be restarted by physically powering down and back up. As mentioned above, the iPod work for IP over Firewire had not been actively developed for at least a couple of years. Armed with an iteratively developed smattering of print statements, many power down/up cycles and a decent portion of patience (!) the fault was eventually traced, and the one line fix necessary added.[6] At this point we were able to demonstrate 'ping' running successfully and Telnet into the iPod.

Having reviewed the options for a web server, we selected Boa—an open source web server that is a popular choice for embedded systems—as a basis for our implementation. The version distributed uses `fork()` in various places, such as launching a CGI executable. This was an issue as iPod-Linux does not implement this function due to the uClinux base that assumes there is no memory management unit to the device. Fortunately, uClinux does implement `vfork()` a "lightweight" (or virtual) way of forking a process. What it does is create a child process and blocks the parent. What it relies upon is that the child process does not do anything too sophisticated—either the

[5] An open source system used in conjunction with the software version control system SVN commonly used to keep track of bugs and features requests.

[6] The updated version of the device drive is available through the web site associated with this EPSRC fellowship, *www.nzdl.org/epsrc-fellow-2007/*.

child exits shortly after the fork or runs an `exec` command that effectively replaces the child process with a new binary executable. The latter is perfect for what happens when a CGI program is initiated through the web server.

Suitable modifications were made to the sections of code involving `fork()`, along with changes relating the setting of user and group id, given the impoverished state of this information on the iPod. The modified version of the Boa server is available through *www.nzdl.org/epsrc-fellow-2007/ipod*.

## 4 Conclusions

In conclusion, a variety of digital library techniques have been trialled and tested on an iPod installed to run Linux. The open source Greenstone digital library software forms the basis of the work, due to the authors' familiarity with the software, but moreover due to its suitability to the task. Greenstone's runtime is written in C++, unlike other main contenders for open source DL projects which are all in Java. To date, no work has been done on running Java applications on iPod-Linux, which would be a significant hurdle to overcome.

The work undertaken was overall successful, achieving the goals set. A steep learning curve is inevitable when moving to a new hardware platform for development, especially when the operating system being used has been developed for a proprietary device that has not disclosed its design. This was the case for us, and the issue of printing to the screen using standard C++ classes was a particularly thorny issue that took considerable time to resolve. That said, after this was solved rapid progress was made. The other key impediment encountered was getting the Ethernet driver for the host Linux machine working, and here too—due to the open source orientation of the implementation—we were able to analyse the problem and to correct the defect.

The command-line test allowed us to confirm that the bulk of the Greenstone runtime code ran successfully on the mobile device, although its output (raw HTML) was not particularly useful from a user's perspective. The lion's share of the implementation time was spent developing an interactive hierarchical menu system that provided access to existing, pre-built Greenstone collections: both searching and browsing were supported. Making a particular Greenstone collection available on the iPod is a simple matter of copying (in the iPod's disk mode) the collection's index and configuration files (`index` and `etc`) from the host machine (where typically the collection was built) to the iPod.

Two media rich forms of delivery were also experimented with, as proof of concepts: collaging and digital stories. Running the iPod as a web server was also accomplished, however the list of requirements necessary to make this work means its usefulness is emasculated. This notwithstanding, all the other aspect of the project achieved a high level of success.

## References


[1] Thorsten Büring and Harald Reiterer. Zuiscat: querying and visualizing information spaces on personal digital assistants. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 129–136, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-089-2. doi: http://doi.acm.org/10.1145/1085777.1085799.

[2] Matt Jones, Will Harwood, George Buchanan, and Mounia Lalmas. Storybank: an indian village community digital library. In *JCDL '07: Proceedings of the 2007 conference on Digital libraries*, pages 257–258, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-644-8. doi: http://doi.acm.org/10.1145/1255175.1255225.

[3] A. Kerne. CollageMachine: an interactive agent of web recombination. *Leonardo*, 33(5):347–350, 2000.

[4] Gary Marsden, Robert Cherry, and Alan Haefele. Small screen access to digital libraries. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 786–787, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-454-1. doi: http://doi.acm.org/10.1145/506443.506597.

[5] A. F. Smeaton, N. Murphy, N. E. O'Connor, S. Marlow, H. Lee, K. McDonald, P. Browne, and J. Ye. The Fischlár digital video system: a digital library of broadcast TV programmes. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 312–313, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-345-6. doi: http://doi.acm.org/10.1145/379437.379696.

[6] Ian H. Witten and David Bainbridge. *How to Build a Digital Library*. Elsevier Science Inc., 2002. ISBN 1558607900.

[7] Werner Zimmermann. LTOOLS - accessing your Linux files from Windows 9x and Windows NT. *Linux Journal*, November 2000. URL `http://www.linuxjournal.com/article/4138`.