THE INTEGRATED MEDIA SYSTEM

SENIOR DESIGN TEAM #10
STREAM COME TRUE

ANDREW OOSTERHOUSE
NICK GOOTE
BEN BUFFORD
JERMEY GAJADHAR

CALVIN COLLEGE
ENGR 340
MAY 12, 2006
Executive Summary

The purpose of the project is to design a streaming media system aimed at augmenting the existing Hekman Library Cayvan media area using modern digital archiving, storage, retrieval, and playback methods in conjunction with a wireless Ethernet infrastructure. For the 2005-2006 design year, the team proposed to design, construct, and demonstrate the concept of an Integrated Media System.
# Table of Contents

1. Introduction .................................................................................................................. 1
2. Terms.............................................................................................................................. 2
3. Project Scope.................................................................................................................. 3
   3.1 The Challenge............................................................................................................. 3
      3.1.1 Scope of Hardware ............................................................................................ 4
      3.1.1.1 Scope of the handheld device ........................................................................ 4
      3.1.1.2 Scope of Media Server .................................................................................. 4
      3.1.1.3 Scope of the Video Terminal ........................................................................ 4
      3.1.2 Scope of Software ............................................................................................. 4
      3.1.2.1 Software for the handheld device ............................................................... 4
      3.1.2.2 Software for the media server ..................................................................... 5
      3.1.2.3 Software for the video terminal ................................................................. 5
3.2 Design Norms.............................................................................................................. 5
3.3 Project Requirements................................................................................................. 7
4. The Solution .................................................................................................................. 9
   4.1 Hardware Solutions ................................................................................................. 9
      4.1.1 Overview .......................................................................................................... 9
      4.1.2 Design Strategy ................................................................................................ 9
      4.1.3 Display ............................................................................................................ 11
      4.1.4 Bluetooth ........................................................................................................ 13
      4.1.5 Audio solutions ............................................................................................... 14
      4.1.6 GPIO (General Purpose Input/Output) ............................................................ 16
      4.1.7 Processor Module ........................................................................................... 17
      4.1.8 Casing Solutions ............................................................................................. 17
      4.1.9 Power Supply .................................................................................................. 18
      4.1.10 PCB Design ................................................................................................... 19
   4.2 Software Solutions .................................................................................................. 20
      4.2.1 Handheld Software .......................................................................................... 20
      4.2.2 Server Software ............................................................................................. 22
      4.2.3 Software difficulties ....................................................................................... 23
   4.3 Budgeting performance ......................................................................................... 23
   4.4 Project Management .............................................................................................. 24
      4.4.1 Team Management .......................................................................................... 24
      4.4.1.1 Ben Bufford ............................................................................................... 24
      4.4.1.2 Jermey Gajadhar ....................................................................................... 25
      4.4.1.3 Nick Goote ................................................................................................. 25
      4.4.1.4 Andrew Oosterhouse ................................................................................ 25
      4.4.2 Schedule ......................................................................................................... 26
      4.4.3 Task specifications ......................................................................................... 27
5. Remaining Issues ......................................................................................................... 32
6. Conclusion ................................................................................................................... 32
7. Recommendations ....................................................................................................... 33
   7.1 Additions to the Integrated Media System ............................................................. 33
   7.2 New Applications for the IMS concept ................................................................. 33
8. Acknowledgements ...................................................................................................... 34
9. Appendices .................................................................................................................. 34
   9.1 UML diagram of central server software .............................................................. II
   9.2 UML for Client Software ....................................................................................... III
Table of Figures

Figure 1 - Cayvan Video Viewing Room and Listening Station .................................................... 1
Figure 2 - Overall System Hierarchy .......................................................................................... 3
Figure 3 - Boot options used .................................................................................................... 11
Figure 4 - Excerpt of design notes ........................................................................................... 12
Figure 5 - Final kernel output .................................................................................................. 12
Figure 6 - Original code ............................................................................................................ 15
Figure 7 - Altered code .............................................................................................................. 15
Figure 8 - Final flash memory layout ....................................................................................... 17
Figure 9 - Casing used for prototype ......................................................................................... 18
Figure 10 - Voltage regulators mounted on the final board ....................................................... 19
Figure 11 - Final PCB layout for handheld device ..................................................................... 20
Figure 12 Application screen capture ....................................................................................... 21
Figure 13 - List of available media ........................................................................................... 21
Figure 14 - Media options ......................................................................................................... 22
Figure 15 - Team organizational chart ....................................................................................... 24
Figure 16 - Summary of Project Schedule .................................................................................. 26
Figure 17 - Server UML ............................................................................................................. II
Figure 18 - UML diagram for handheld ..................................................................................... III
Figure 19 - Top layer of final PCB ............................................................................................ XXXIX
Figure 20 - Layer 2 of final PCB .............................................................................................. XXXIX
Figure 21 - Layer 3 of final PCB .............................................................................................. XL
Figure 22 - Bottom layer of final PCB ..................................................................................... XL
Table of Tables

Table 1 - Handheld device decision matrix ................................................................. 4
Table 2 - Decision matrix for media server ................................................................. 22
1. Introduction

The project is intended for use in multimedia areas of public institutions, such as university libraries. The multimedia area at Calvin College is a prime example of our target end-user. The multimedia area in Calvin’s Hekman Library is called the Cayvan media area. Currently the Cayvan collection is comprised of a large assortment of physical audio and video media that can be used by students at listening stations in pre-designated sections of the library. Figure 1 shows the current implementation of the Cayvan video viewing room (left) and the listening station (right). There are two primary drawbacks to this system. The first is the amount of effort that has to go into organizing and maintaining such a sizeable collection of physical media. The Cayvan collection is comprised of over 40,000 dollars worth of physical media, with over 2,000 choral pieces from Calvin’s choir alone\(^1\). The second primary drawback is the somewhat limited nature in which the various media types can be accessed. There are limited copies of all the media, so only a small number of students can view the media at any given time. Also, the media is physical, so it requires actual storage space. These disadvantages also apply to any library with a similar media collection. While the current system is adequate, using current generation digital media archival techniques could enhance this solution.

![Figure 1 - Cayvan Video Viewing Room and Listening Station](image)

Our project will use handheld computer systems to stream audio and control video streams from a centralized media storage network. The number of handheld computer is dependant on the desires of the institution purchasing the system. Library media will be accessible via streaming audio over the library's wireless network. The audio, a streaming mp3 format, will stream and playback through a handheld audio computer. Any media available on the library's media server will be accessible via the user interface on the handheld device. This server will be able to contain audio such as music selections, audio books, as well as any other audio media currently available. In terms of video, the handheld device will control a

---

\(^1\) Cayvan Services. 17 Aug. 05. The Hekman Digital Library. 3 Dec. 05.
<http://www.calvin.edu/library/cayvan/tutor/main.stm>
video stream that is displayed on a separate video terminal. The video will most likely be AVI format, and will include DVD’s, VHS tapes, and any other video media that is currently available.

The interface on the handheld computer will be context specific. The video aspect of the system will be implemented via thin client systems wired to the library LAN. These thin client systems will have no external human interface devices. Instead they will be controlled by the handheld computer systems via Bluetooth, which is a wireless communication standard. When a handheld computer detects a Bluetooth signal it will display a video selection interface. The user will then be able to select any available video media. The selected video will then stream to the user-specified thin client terminal.

2. Terms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS</td>
<td>Integrated Media System</td>
</tr>
<tr>
<td>ARM</td>
<td>Advanced RISC Machines</td>
</tr>
<tr>
<td>SCT</td>
<td>Stream Come True (Team 10)</td>
</tr>
<tr>
<td>BGA</td>
<td>Ball Grid Array</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronic Engineers</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PCMCIA</td>
<td>Personal Computer Memory Card International Association</td>
</tr>
<tr>
<td>SBC</td>
<td>Single Board Computer</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>WRL</td>
<td>Western Research Laboratory</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>VLC</td>
<td>Video LAN client</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>SDP</td>
<td>Service Discovery Protocol</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>HID</td>
<td>Human Interface Device</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input/Output</td>
</tr>
<tr>
<td>PLD</td>
<td>Programmable Logic Device</td>
</tr>
</tbody>
</table>
3. Project Scope

3.1 The Challenge

The challenge that the team presented to itself during this project was significant. The team decided to design and construct an Integrated Media System (IMS) which would utilize off the shelf components for the media server and video terminal, and concentrate most of the design work on the handheld device. The overall system hierarchy envisioned by the team can be seen in the figure below.

![Figure 2 - Overall System Hierarchy](image)

From the above diagram, it is clear that the handheld device is critical to the system’s success. However, the handheld device also presented Stream Come True (SCT) with the most challenges. First, a single board computer had to be selected. After this selection, SCT had to devise a way to incorporate Bluetooth and 802.11 communication hardware onto the device, as well as incorporate audio playback capabilities.
3.1.1 Scope of Hardware

3.1.1.1 Scope of the handheld device
Initially, SCT was considering building the handheld device from the ground up, using an open hardware device called the ITSY. The schematics and bill-of-materials for this device are in the public domain, and thus would have been easily accessible had SCT chose to pursue that route. However, SCT ultimately decided against using the ITSY and opted instead to use a pre-built single board computer. The decision matrix used by SCT to make this decision can be found in the table below. Ultimately, SCT did not have the resources available to fully design a version of the ITSY, while also designing the rest of the project. A single board computer was selected to narrow the scope of the project to a manageable level.

Table 1 - Handheld device decision matrix

<table>
<thead>
<tr>
<th>Handheld</th>
<th>Weighting</th>
<th>ITSY</th>
<th>Single Board Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td>8</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Availability</td>
<td>6</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Fabrication/Assembly</td>
<td>5</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Chance of Failure</td>
<td>7</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Expansion Capability</td>
<td>8</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>248</td>
<td>124</td>
</tr>
</tbody>
</table>

3.1.1.2 Scope of Media Server
SCT felt it would be prudent to limit the scope of the work involved with the media server given the large amount of design work associated with the handheld device. A standard Personal Computer (PC) was chosen to be the media server in the prototype implementation of the IMS. In the production model a much faster system would have to be implemented, especially if a large number of handheld clients were utilized.

3.1.1.3 Scope of the Video Terminal
SCT also felt it would be prudent to limit the scope of the work involved with the video terminal. A standard PC was chosen to serve in this role for prototyping purposes. In production, a thin client computer would most likely be used.

3.1.2 Scope of Software

3.1.2.1 Software for the handheld device
The scope of software design was also important to the project. Given SCT’s already formidable commitment to hardware design with the handheld device, it was important that software design be
minimized when possible. To this end, SCT utilized the pre-built Linux kernel that came with the Triton handheld module as a building block, and modified it to our own needs. The alternative would have been to start with the generic ARM Linux kernel and customize it to our Triton hardware.

Even with the use of the pre-built Triton kernel, the scope still contained software design work. However, this work was restricted to “tweaking” the existing software so that our custom hardware would operate properly.

3.1.2.2 Software for the media server
The media server will require custom software in order to interact with the rest of the system. Some sort of database structure would have to be implemented, along with an interface to both the handheld device and the video terminal.

3.1.2.3 Software for the video terminal
The video terminal will also require custom software in order to interact with the rest of the IMS. Software will be needed to interact with the handheld device via Bluetooth in order to ensure that video playback is synchronized with audio playback. Software will also be required to play streaming media from the central media server.

3.2 Design Norms
Design Norms are general principles that relate to how to design correctly. These norms, while they are moral rather than physical or legal guidelines, do have a strong influence over the scope of a project. By following these norms, the engineer is forced to take a step back and take a look at how the design affects the environment in which it will be used. SCT desires to take a look at how our design will affect the environment in which it is used, and will utilize design norms for this purpose.

According to Gayle Ermer, Associate Professor of Engineering at Calvin College, “Technology is a tool that needs to be shaped by Christian values to be effective in ministering to the needs of the world.” These ideas are epitomized by the design norms of cultural appropriateness, transparency, stewardship, and caring.

The first design norm is cultural appropriateness. This design norm focuses on how well new technology fits into the society that will make use of it. The handheld device is made with this design norm in mind. Given that our project will be designed for use in an academic library, it is critical that we design the Integrated Media System (IMS) to fit into the culture of an academic institution, and more generally, the culture of the USA. It is also important to consider how our project might be applied after

---

we finish our proof of concept. In light of that, the design needs to be easy to use and easily adaptable to all academic institutions. An excellent way in which this consideration can be implemented is to give the handheld device and its support equipment multi-language support, which will allow it to work on an international scale.

Next, it is important to consider the design norm of transparency when designing the IMS. Transparency is designing the system to be as open and understandable as possible to the end user and everyone involved in the design. This does not necessarily mean disclosing every design detail to the end user. For example, software companies rarely disclose source code because it may be proprietary. However, they do disclose information on exactly what the code is meant to do, and how reliable the code is. Our team is striving to fulfill this design norm by using Open Source software for the handheld device. Using Open Source allows for more detailed documentation of all hardware components, as well as all software programs. This allows our team to provide very thorough documentation about how the IMS will function to all end-users of the system.

Stewardship is also an important consideration for our design. Stewardship is the idea of designing a system that doesn’t use more resources than required to meet the goal. This can take many forms. These can include monetary stewardship, stewardship of natural resources, or stewardship of assets such as time, labor, buildings, etc. By utilizing our design, academic libraries stand to save both space and money, allowing for better stewardship of both physical and economic resources. Space will be saved by converting the large amount of physical media to digital files which can be archived on central media servers. The IMS will also allow for possible elimination of the separate viewing rooms since the system is completely portable and has headphones. Economic resources will be saved by reduced storage costs, as well as the expandability of the IMS. If, in the future, the end-users wish to increase their media capabilities, they could easily do so by simply obtaining more IMS equipment. The only limit to the scalability of the system would be the overall bandwidth of the available wireless network, the number of video terminals, and the size of the building. With the old system, expanding the media collection of the end-user could possibly require an addition to the library building, or at the very least additional storage space. The IMS serves as a single easy and economic solution to the expansion of the Cayvan media libraries. It is important to our team that we use resources efficiently. The cost that is saved in producing videos, DVDs, and also building materials could be funded to other areas.

Caring is the idea of designing a system that will have a positive effect on people. The main way in which the team applies this norm is by caring for people with physical disabilities. For a specific example, our project will enable greater access to Cayvan’s media library for all library patrons. This includes persons who may be physically handicapped and unable to use the current facilities. Under the current system, those in wheelchairs must make their way to the Cayvan desk, dodging students and
library furniture in the process. After reaching the desk, they must get themselves into a small viewing room where wheelchair mobility is restricted. The rooms are small, and given the amount of furniture in the room, making adjustments to the AV setup can be a hassle. For example, if a wheelchair bound person wanted to play a VHS tape, they first must obtain the tape from the desk, take the tape to the room, put the tape into the VCR to begin viewing. This is clearly feasible, but more difficult for someone in a wheelchair in a library crowded with students and furniture. If Cayvan was using the IMS, the media becomes much more accessible, allowing Cayvan to serve more people. Depending on the extent of the wireless media, Calvin could also bring their library archives to the student’s room through streaming media if needed, thus making Calvin a friendly environment for people with disabilities.

3.3 Project Requirements

In order to more clearly define the scope of the project, SCT implemented a formal set of design requirements. These requirements take the issues regarding project scope from the previous sections and combine them into a single cohesive set of requirements to guide the rest of the project. The requirements can be seen below:

A. Hardware Requirements

I. The Portable Client

a. General
   i. The portable client shall provide the housing and the interface for the integration board and modular components

   ii. The processor module shall contain the Intel PXA255 processor
   iii. The portable client shall feature a 3.5 inch LCD screen
   iv. The portable client shall include an audio jack connector

b. Integration Board
   i. The portable client integration board shall provide Bluetooth and 802.11b functionality and interconnect all of the component modules.

B. Interface requirements

I. Handheld interface requirements
   a. Primary Communication Interface Requirements
      i. The portable client shall access audio and server resources over Ethernet
      ii. The portable client shall send Bluetooth control signals to video terminals

II. Video Terminal Interface
   a. The video terminal shall play back video media over an Ethernet to the Central Media Server
   b. The video terminal shall signals to control the video playback via Bluetooth. These signals shall be sent from the portable client device.
III. Server Interface
   a. The server shall transmit the requested video media to the video terminal for playback.
   b. The server shall deliver all audio content over wireless Ethernet utilizing the IEEE 802.11b specification.

IV. Software Interface
   a. Inter-device communication shall be implemented using TCP/IP version 4 and the Bluetooth protocol stack

C. Software Requirements

I. The portable client with installed Daughter-card
   1. The portable client shall have preinstalled Linux 2.6 Kernel version
   2. The portable client device shall have an QT based Qtopia graphical interface system
   3. The unit will be able to recognize streaming media libraries
   4. The handheld device shall require some Ruby programming and use of its libraries

II. Media Server
   1. The media server shall have a Linux 2.6 based Kernel installed.
   2. The server shall also implement a PostgreSQL database engine
   3. Ruby programming shall be a requirement to get the media server functional with the handheld
   4. An Apache 2.0 version shall be installed in order to stream wirelessly with the media server and handheld.

D. Timetable and Deliverables
   a. The engineering team shall supply any ARM based board development system or board no later than the middle of March
   b. The engineering team shall finish board layout for the first portable prototype no later than the end of March.
   c. The computer science team shall include regularly scheduled software builds for hardware testing
   d. The computer science team shall provide a functional client/server streaming media software delivery by the end of the school year.
   e. The initial prototype shall be completed by May 6th
4. The Solution

4.1 Hardware Solutions

4.1.1 Overview
Since the entire first semester was spent acquiring funding, determining feasibility, and finalizing component selection very little non-administrative design work was done. After a comprehensive feasibility study the idea of designing a daughter card and constructing an Itsy pocket computer from scratch was abandoned due to budgetary and time constraints. Instead, modular components of every major subsystem would be combined to meet the project requirements. The broad range of device functions for the portable client lent its design well to this type of development. This decision was also made because modular components delivered the logical functionality of a particular subsystem but fully eliminated the design complexity of incorporating requisite support circuitry and/or engaging in rigorous and time consuming R/F design work for which the project was ill equipped. The subsystems identified and modules selected were: Bluetooth (A7 engineering eb100-HCI), display and input (Littlechips LP35), 802.11 Ethernet (DPAC WLN-B-ET-DP100), and processor module (Karo Triton-ETN). Of the modules, display and Ethernet were critical to the success of the project (with the processor module implied to be operational). The audio device (Philips UCB1400) was also determined to be a critical component but there was no module available so it required design around a chip rather than module. Collectively the project critical path included the three functional areas of audio, Ethernet, and display.

4.1.2 Design Strategy
Entering the second semester the project faced serious time constraints with virtually every design challenge still remaining. Consequently, the design strategy centered largely on identifying the most effective means of integrating the modular subsystems onto a carrier board of SCT’s own design as quickly as possible. Concisely described, the approach was to verify the functionality of each major subsystem by linking custom breakout boards to the Triton starter kit. Since speed of development was a primary focus COTS (commodity off the shelf) cabling and connectors were the exclusive means of interconnecting the subsystem breakout boards to the Triton development kit, and the breakout PCBs were fabricated entirely in house. These boards were designed with long, exposed, unconnected traces to allow for probing and quick rewiring. The ability to easily rewire these breakout boards allowed for multifaceted design testing and verification. This was indispensable throughout the course of the project. Specific hardware conditions could be wired to test software, and varying hardware configurations were easy to experiment with. However, the precision with which it was possible to fabricate breakout boards
was a limiting factor in testing some of the hardware. Knowing that these boards would be changed and re-fabricated repeatedly; being confident in the Ethernet portion of the schematic; and balancing the tradeoffs in price and turnaround time the design decision was made to not have the breakout boards professionally fabricated. It was determined that the DPAC 802.11 module had straightforward enough wiring to warrant exclusion from breakout board testing. For all other components the trace precision was sufficient or viable workarounds were determined.

As the correct connections to the processor module were determined for each subsystem a system wide schematic (in appendix) was updated. Leading into and during the early stages of break out board testing a global schematic was created and updated in parallel with hardware developments in the lab. This resulted in efficient schematic capture of all hardware development almost as quickly as they were made in the lab. The schematic was typically updated on the same day any substantial hardware changes were finalized on breakout boards. Once there was reasonable confidence in the hardware interconnects routing began. Re-routing did take place as hardware changes were made up to the day before the board layout was finalized for fabrication. The decision to route the PCB in parallel with ongoing hardware development was made entirely due to time constraints and ideally would have waited until all breakout board development was completed. This also directly influenced the design decision to use no more than four board layers. Additional layers would have simplified the process of routing at the expense of being unable to scratch out an errant trace. Due to the rapid nature in which the layout was finalized and the slightly less than finalized state of the breakout board development setup, this was a reasonable tradeoff. SCT was pleased to find out that all the component interconnects were indeed correct on the schematic level. The decision to use four layers also paid dividends when it was realized that the pads for the Bluetooth device had been incorrectly numbered, making all of the routed traces for the device incorrect. The final physical layout, along with the schematic, can be seen along in the appendix.

A software build environment was also an integral aspect of the design project. Nearly any UNIX platform that the GNU compiler collection and libraries are available for is suited for the task. Brief investigation was done into using Darwin as the primary build environment for ARM cross-compiling. Due to the ever-present time constraints, the ready availability of a high performance multi-processor x86 GNU/Linux platform, and pre-complied x86 cross compiler tools supplied by Karo (the Triton vendor), Acolyte (an x86 server hosted by the Calvin College computer science department) was selected as the build host. Since the pre-compiled toolset was used no decisions regarding compiler or library versions were possible. The versions used were gcc 3.4.3, glibc 2.3.3, and binutils 2.14. All compiled software was transferred to a Darwin work environment using GNU secure copy and subsequently transferred to the target using ftp or tftp.
4.1.3 Display

Hardware testing and development began with the LP35 display module. Work began on this device first since it was the most important of the three critical subsystems. The rationale behind this was simple. Assuming everything else works, the system is designed to interact with the user visually. As such, if the end-user is unable to see the interface the portable client is unusable. The LP35 was selected due to its high level of availability and affordable pricing. By approaching a company who was already using this module in their designs, SCT was able to leverage their buying power with the top level LCD vendor (TopSun) to get small part quantities SCT would have otherwise been unable to acquire. In addition, the modular advantage of this component prevented the team from having to confront the various voltage step up, power supply timing, and intermediate logic issues that would have been required with a component direct from an LCD vendor. During testing the LP35 module was confirmed to be functional once the backlight was wired correctly to a 5V source. The initial timing and resolution settings for the frame buffer were incorrect for the TopSun TS35ND5B01 LCD, but they did confirm that the module was indeed functioning correctly. The LCD datasheet provided detailed operating parameters for the device that were fine-tuned experimentally. The PXA255 LCD control registers determine the LCD parameters. These can be set externally by module options when the frame buffer module is loaded into the kernel, by command line arguments at boot time, or internally by modifying the pxafb kernel sources found in pxafb.c and the video section of karo.c (listed in appendix). After a few attempts at statically specifying the correct parameters for the TS35ND5B01 in the kernel source with mixed results the decision to use boot time command line arguments was made. Further reason for this decision is that the use of loadable modules is minimized in favor of device drivers compiled into the kernel in order to simplify hardware initialization when command line intervention is unavailable on the final carrier board implementation. Scripting capabilities built into the boot monitor made using boot time arguments simple. Kernel modification would have been ideal but was unsuitable given the time constraints of the project. A heavily patched LCD implementation for the Triton system meant going beyond understanding the generic kernel code. The command line argument, an excerpt from the technical notes detailing affected parameters, and the resulting kernel boot messages are shown:

```
video=pxafb:mode:320x240-16,active,color,hsync:30,left:20,right:38,upper:4,lower:15,vsync:3,hsync:0,vsync:0,pixclockpol:0,pixclock:6400000
```

**Figure 3 - Boot options used**
2.6 kernel pxafb boot param is:

pixel clock polarity is falling edge
pixclockpol: 0

outputen: POLARITY is currently unknown
dpc: DPC is unknown, probably false

vsync and hsync are both active low
hsync: 0
vsync: 0

active
color

See pg 11-12 of TS35ND5B01 data sheet
pixclock: 6400000 Pixel clock in picoseconds

front porch
left: LEFT == LCCR1_BLW + 1
should be THF from datasheet: 20 ticks -> 20 + 1 = 21

back porch
right: RIGHT == LCCR1_ELW + 1
should be THB from datasheet: 38 ticks -> 38 + 1 = 39

length of hsync pulse
hsynclen: HSYNC == LCCR1_HSW + 1
should be THP from datasheet: 30 ticks -> 30 + 1 = 31

number of line clocks for vsync front porch
upper: UPPER == LCCR2_BFW
should be TVF from datasheet: 4 cycles -> 4 + 1 = 5

number of line clocks for vsync back porch
lower: LOWER == LCCR2_EFR
should be TVB from datasheet: 15 cycles -> 15 + 1 = 16

length of hsync pulse
vsynclen: VSYNC == LCCR2_VSW + 1
should be TVP from datasheet: 3 cycles -> 3 + 1 = 4

Figure 4 - Excerpt of design notes

pxa2xx-fb pxa2xx-fb: overriding resolution: 320x240
pxa2xx-fb pxa2xx-fb: overriding bit depth: 16
pxa2xx-fb pxa2xx-fb: override hsynclen: 30
pxa2xx-fb pxa2xx-fb: override left: 20

Figure 5 - Final kernel output
Along with the timing and geometry issues encountered the display also exhibited unusual color inversion. This was traced back to an unsigned int member of the kernel struct pxafb_info called cmap_inverse. This element is used as a Boolean value to determine whether the color mapping should or should not be inverted for a particular display. The vendor specific patches to the kernel made altering this variable somewhat cumbersome. Changes to the controlling source module karo.c also seemed to impact whether the display was initialized as a static or TFT device. Since the display was active and functioning adequately for use a thorough resolution of this issue was postponed in order to address higher priority design problems. At this point the color inversion remains uncorrected.

4.1.4 Bluetooth

Though Bluetooth was non-critical, the other components of the system (audio and 802.11) were not yet ready to be tested. Thus, circumstance and necessity dictated work on this area next rather that a deliberated design decision. The sole criterion in this case was availability, and it required adapting to the situation at hand.

The Bluetooth device interfaces to the processor module by way of the dedicated Bluetooth UART built into the PXA255 microprocessor. Connections for hardware flow control between the CSR BlueCore device and the PXA255 are optional, but the decision to attach flow control the lines was made because it was preferable to have the entire spectrum of hardware options for the device physically available. During the testing phase for the Bluetooth hardware the kernel level BlueZ protocol stack and the UART HCI (Host Controller Interface) driver were compiled as modules to allow them to be unloaded and reloaded without restarting the kernel as hardware adjustments were made. The BlueZ protocol suite was selected because it is the official Bluetooth protocol stack fro the Linux kernel and it is integrated directly into the kernel source tree, making it the easiest of any available protocol stack to implement. Alternate options, though available, would have required far more time to implement and were not even superficially considered.

For this device numerous adjustments were required due to a large amount of uncertainty caused by the labeling conventions of the Triton and EB100 hardware vendors. For audio it was definitively confirmed that the pins labeled for input on the UCB1400 were to be connected to the PXA255 pins labeled for input and likewise with the output pins. The schematics seemed to indicate that this counter intuitive trend had also been followed for the BTUART as well. Furthermore there was no indication from any of the vendor documentation that the CTS and RTS hard flow control pins for the devices were supposed to be crossed over. This realization was made by further research into general UART interfaces (a prime example of something that would be assumed common knowledge for an experienced engineer). It was only after using the proc file system to monitor kernel transmit receive and transmit activity on the
BTUART that the correct interconnects could be determined. The node /proc/tty/drivers/PXA serial allowed the number of data and flow control bytes sent and received over the BTUART (/dev/ttyS1) to be displayed. This allowed the impact of hardware alterations to be viewed immediately after they were made and tested.

Since Bluetooth required user level software to fully initialize, in addition to drivers statically compiled into the kernel, an initialization script was also added to the default run level. This was done in order to integrate the Bluetooth hardware following existing Linux conventions as closely as possible. Using the init framework also made automating Bluetooth initialization trivially simple.

The entire collection of BlueZ libraries and utilities needed to be cross-compiled from source for ARM before they were available for use. A userspace daemon was required to run in the background to activate the Bluetooth device. The applications hciattach and hciconfig then had to be run to bring the device up. The hciattach application was responsible for setting the flow control options, speed and device protocol while hciconfig set the device status. These tools start the device at 38400 bps with hardware flow control enabled using the blue core serial protocol. Though the maximum UART speed is 921600 bps and the EB100 is stated by the vendor to have a UART speed of up to 1Mbps the ONLY speed it was able to successfully initialize was at 38400. The manufacturer is unaware of the cause for this slowdown and remains in correspondence to resolve this issue.

4.1.5 Audio solutions
Audio was the last subsystem to work correctly. Before work could even begin, additional parts that were originally unplanned had to be ordered. Naturally this introduced delays. In order to set its sampling frequency, internal, reset, and sync clock, the UCB1400 uses an internal phase locked loop connected to a 24.576MHz crystal external to the device which the team had not anticipated needing. The UCB1400 is a multifunction device that also serves as a touch screen controller. SCT was under the mistaken impression that the crystal and phase locked loop were for touch screen rather than audio use. Further delays were introduced when instead of a 24.576MHz crystal an oscillator was ordered. Through hole crystals were ordered for the breakout boards and surface mount crystals were ordered for the final board and work on audio did not progress until they arrived.

To get audio up and running kernel changes had to be made and multiple hardware bugs had to be addressed. The first of the mistakes made by the team to come to the team’s attention – after review under a microscope – was the fact that in spite of the team’s best efforts, the audio device was misaligned on the breakout board, shorting several of the pins. The chip was checked under a microscope after all attempts to load the drivers for the audio hardware met with failure. It was clear from the error messages that the AC’97 device on board the PXA255 was not receiving expected reset signals to synchronize it with the UCB1400 audio codec. Since the AC’97 is integrated into the same physical package as the
processor it could be readily concluded that any communication errors between the AC’97 device and
codec would almost certainly be caused by the audio codec. Confident that the first UCB1400 chip used
was damaged either from excessively high soldering temperatures or from being shorted on the breakout
board, and also confident that it would be far to difficult to correctly align a new chip accurately on the
breakout board, the bold approach of soldering bare wire to an isolated UCB1400 device was taken. The
motivation behind this decision was, while fragile, the bare wire setup would be the only way to test the
device with the available board resources. The team was quite divided over the effectiveness of this
decision, but it was ultimately concluded that time was slim enough and there were enough available
UCB1400 spares to warrant an attempt at the idea. It turned out to be the break through for audio.

After this change the reset errors were eliminated, but the read and write access errors that
followed them remained. Being at a loss for solutions and after many fruitless hours searching forums
and mailing lists, the vendor was contacted. Correspondence with Karo revealed that there were pre-
processor macros present in a custom source module that they had patched into the kernel that were
inhibiting the AC’97 device from being activated. The module karo_config.c needed to be modified to
activate the AC’97 feature set if ALSA (Advanced Linux Sound Architecture) drivers were selected as
kernel options rather than or along with OSS (Open Sound System) drivers. A simple code change,
shown in the figures below was all it took to activate AC’97 in the next kernel build.

```
#if defined(CONFIG_SOUND_PXA_AC97) ||
defined(CONFIG_SOUND_PXA_AC97_MODULE)
```

**Figure 6 - Original code**

```
#if defined(CONFIG_SND_PXA2xx_AC97) ||
defined(CONFIG_SND_PXA2xx_AC97_MODULE) ||
defined(CONFIG_SOUND_PXA_AC97) ||
defined(CONFIG_SOUND_PXA_AC97_MODULE)
```

**Figure 7 - Altered code**

After the code change, all reset, read, and write errors previously seen were eliminated leaving
only one esoteric error stating that AC’97 access was invalid and the critical mixer device was being
removed. Further review of the kernel source for the driver revealed that the location of the code where
the error was occurring was using data that could only be supplied by active communication with the
external audio codec. The error was being generated in the function snd_ac97_mixer and was occurring
only after the AC’97 device failed to receive valid capability and vendor ID data after calls to the
snd_ac97_read function. Debug messages were added to the area interest and the kernel modules for
audio were rebuilt. Loading the instrumented modules confirmed that invalid data was being stored by
the snd_ac97_read function. This lead us to revisit prior questions the team had raised but were unable to previously answer. The presence of two data input pints (SDATA_IN0 and SDATA_IN1) on the codec chip had been an early area of confusion with the AC’97 interface, but audio development up to this point had never progressed far enough for clarification to be of any consequence. Changing the connection from SDATA_IN1 to SDATA_IN0 was the final step. Reloading the instrumented driver showed valid data received from the UCB1400. Sound tests proceeded flawlessly, confirming both the audio hardware and an earlier untested cross-compile of the Vorbis decoding software. SCT knew from the data sheets for the UCB1400 and the PXA255 that the SDATA_IN pins were responsible for moving serial communication data from the UCB1400 to the AC’97, and SDATA_IN from the UCB1400 had been routed to SDATA_IN0 on our final board layout.

4.1.6 GPIO (General Purpose Input/Output)
The simplest way to attach the input buttons on the LP35 display module was to implement them as GPIO inputs. These buttons are active low and the GPIO pins for the PXA255 are internally pulled up, readily lending the GPIO pins to such use. Since a PLD (Programmable Logic Device) would have to be used to implement any other sort of peripheral HID (human interface device) protocol (e.g. serial or USB) using GPIO for input handling was the easiest and most obvious choice; especially given the time constraints. In order to implement input in this manner some currently occupied GPIO pins had to be freed. This was accomplished by removing PCMCIA and CF from the kernel since nothing of the sort was used in the project. Once removed in the kernel config options, the kar0_config.c source module also had to be modified to disable these hardware options and to prevent the GPIO pins from be allocated despite their absence in the kernel. The module uses an enumerated data type to store a list of all available features and the desired features are stored in an unbounded array. Preventing GPIO allocation by the custom Karo modules simply amounted to removing undesired hardware elements from the feature array (KARO_SYS_FEATURES_009[] in this case). Ideally a feature set array should be built for the custom carrier board designed in this project. Unfortunately, time did not permit this to take place. Apart from the kernel changes all of the PCMCIA and CF module auto-loading had to be disabled in addition to the PCMCIA/CF init scripts be linked as K10PCMCIA instead of S10PCMCIA in all run levels. Once this was done the breakout boards were wired to verify that the requisite GPIO pins had indeed been freed up. Forays into the various Xscale message boards had often mentioned a tool called pxaregs floating around the web. Its features were ideally suited for verifying GPIO functionality. This program delivered the capability to view a snapshot of register memory. By holding down a button wired to a specific GPIO pin and executing pxaregs or conversely, by executing pxaregs when a button was released the state of a GPIO bit could be determined. Experimentation quickly revealed the GPIOs 52 and 53 were reserved by
the hardware, so the I²C feature set was removed from the kernel and disabled in karō_config.c. The GPIOs this freed were used to account for the reserved 52 and 53.

The GPIO inputs are mapped to registers that can be accessed through the memory region beginning at 0x40E00000. Using the mmap system call this memory region can be mapped and manipulated by a user space application. This is what pxaregs uses, and though currently unwritten the input “driver” application will use the same system call to read memory from the GPIO register memory region and deliver information to a named pipe based on what it detects. The interface application will have an open file descriptor to the pipe that will allow it to instantly receive data from the GPIO input application once it has determined an input by checking GPIO memory. This is a requisite design approach since there is no other way to access the status of GPIO pins from user space apart from writing an input decoding program in assembly. A kernel implementation would be unwieldy and not worth the effort for the needs of this design.

4.1.7 Processor Module

The processor module required very little work before it was running exactly as needed. As hardware development progressed new kernels had to be built, and the addition of debug code to the kernel as well as the Bluetooth additions made the kernel large enough that more flash space had to be allocated for it. The final flash layout is as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>FLASH addr</th>
<th>Mem addr</th>
<th>Length</th>
<th>Entry point</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedBoot</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00040000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>RedBoot config</td>
<td>0x01F80000</td>
<td>0x01F80000</td>
<td>0x00001000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>FIS directory</td>
<td>0x01FC0000</td>
<td>0x01FC0000</td>
<td>0x00040000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>rootfs</td>
<td>0x00080000</td>
<td>0xA0108000</td>
<td>0x00C00000</td>
<td>0xA0108000</td>
</tr>
<tr>
<td>userfs</td>
<td>0x00C40000</td>
<td>0xA0108000</td>
<td>0x01140000</td>
<td>0xA0108000</td>
</tr>
<tr>
<td>linux</td>
<td>0x01E40000</td>
<td>0xA0108000</td>
<td>0x00140000</td>
<td>0xA0108000</td>
</tr>
</tbody>
</table>

Figure 8 - Final flash memory layout

Flash memory had to be allocated in 256KB blocks so any kernel greater than 1MB in size (as all of SCT’s were) required 1310720 bytes of flash space. No power management or clock adjustment changes were attempted due to lack of time to effective address these issues.

4.1.8 Casing Solutions

The casing for the prototype was designed using Autodesk Inventor. It was roughly 3.5 inches tall, 4.25 inches wide and 2 inches deep. It has an opening in the front to accommodate the LCD screen, as well as holes for the buttons and the joystick. The walls of the case are 3mm thick, to ensure that the case will be sturdy and will be able to withstand regular use. The case was donated to us from a company called Select Manufacturing Services Inc, and can be seen in the figure below.
4.1.9 Power Supply

The power supply was designed using Texas Instruments PTH08080 switching regulators, which can be seen in the following figure. The voltages required for the design were 3.3 and 5 volts. The battery that was chosen was a 12 volt battery with a rating of 2300 mA hours. Both of the regulators were hooked up to the twelve volt battery, one regulated the voltage to five volts, and one regulated the voltage to 3.3 volts. Each regulator is rated to a maximum of 2.5 Amps, and the project will take roughly .7 Amps running at full capacity. Because switching regulators are being used, the voltages are efficiently reduced, making the battery provide roughly 8000 mA hours. This will allow the unit to be powered for 11.5 hours before the battery needs to be recharged. This system delivers power to the unit, and has been proven to work properly.
4.1.10 PCB Design

Designing the PCB (Printed Circuit board), which would serve as the final version of the handheld device, was one of the most challenging aspects of this project. While many of the handheld’s components were modular, which limited the scope of our PCB design, it was still vital that the PCB be laid out correctly. SCT decided to use a 4 layer PCB with the outer two layers being used for signal traces, one of the inner layers being used for a ground plane, and the other inner layer being used for a power plane. This arrangement allowed for quick accessibility to signal traces and components, which made assembly of the final design much easier.

While designing the PCB layout, SCT had to consider many factors. Trace length was important, since fast signals (such as the LCD data bus, and signals from the 802.11 transceiver) can degrade quickly if the traces connecting them are too long. Trace impedance was also an important consideration. The components on the board had to have a similar impedance, or signal integrity would be adversely affected. Thankfully the default board impedance used by PCB express was very close to the impedance of most of the parts purchased for the handheld device. The final design that SCT used can be seen in the figure below, while a picture of each individual layer can be found in Appendix 9.7.
4.2 Software Solutions

4.2.1 Handheld Software

SCT chose to use the very popular and very powerful Qt3 windowing library for the GUI after discovering the embedded version of the library known as Qtopia Core. Qt in itself is a very large library, but with Qtopia SCT was able to build a client that was small enough for the space constraints on the device, while keeping the configurability and ease of development of the full library. Using this library allowed SCT to spend more time in the design and development of the application by making the implementation of the GUI much easier. For playing the audio streams, SCT chose the ogg123 program as it is small, easily compiled for any platform, and included support in reading from streaming sources as well as local files. For communication with the central server SCT found a small library called XmlRpc++. XmlRpc is the communication protocol SCT chose to use for this project as it is easy to set up, easy to use, and very lightweight. This library, like the ogg123 program, is also easy to compile for any architecture, a common theme when deciding what tools to use.

The finished handheld application allows the end-user to select from a variety of media available on the central server. This media is sorted by department and course. When the program is first started up, the user is presented with the screen seen below.
After selecting the desired department and course, the application displays a list of available media, as seen in the figure below.

Finally, the user is able to select the media they want. Upon selection, they are given the options of playing, pausing, or stopping the media, as seen below.
4.2.2 Server Software

There really are two different servers in this project, though both will be running on the same computer. The first server is the one which uses XmlRpc to communicate with the client. This server controls a database of available departments, classes, and files and the relationships required for proper ordering. This server is written using the Ruby programming language and uses a PostgreSQL database.

The second server is the official audio streaming server. SCT investigated a good number of different streaming servers including Apple’s Darwin Streaming Server³ and a popular streaming library called Icecast⁴. In the end, SCT found and decided to use a small server named GNUmp3d⁵. SCT chose this program because it was the only one with both a simple web interface, special support for OggVorbis files (an open source file type much like the popular MP3), and an ability to stream either a file or a play-list from the server. The other servers investigated by SCT only allowed for specifying play-lists, which would require the database server to create these files, adding complexity to the server, along with not being compatible with the ogg123 program for playing. A decision matrix reflecting these decisions can be found in the table below.

Table 2 - Decision matrix for media server

<table>
<thead>
<tr>
<th></th>
<th>Weighting</th>
<th>Darwin Streaming Server</th>
<th>IceCast</th>
<th>GNUmp3d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplicity of interface</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Support for OggVorbis</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Ability to stream files and playlists</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

### 4.2.3 Software difficulties

The most difficult part of this project in terms of software was setting up the development machine for ARM compiling. This required setting up what is known as a cross-compiling tool chain, a system in which a compiler on a certain architecture will produce files meant to be run on a different architecture. Creating such a tool chain required the recompiling of many different system libraries for this new architecture, thus opening up many possibilities for file incompatibilities. This process also takes a long time so any mistake made may only show up after thirty minutes to an hour of recompiling, requiring the entire process to be restarted. Thankfully SCT was able to find a few tools that simplified the creation of the tool chain.

### 4.3 Budgeting performance

The first thing that should be said is that the team stayed within their allocated budget. The original budget from Calvin was three hundred dollars, and Stream Come True received four thousand six hundred dollars from Smiths Aerospace to be used on the project. The first thing that was purchased was a Triton development board with two Triton ETN processors, one to be used as backup. The next major purchase was two LCD modules. These were purchased from a company called LittleChips, and cost one hundred sixty dollars each. The team also purchased a wireless Ethernet evaluation kit, and an extra module to be used as backup. This purchase came to a total of four hundred sixty nine dollars. The last major component to be used in the design was a Bluetooth development kit and extra modules. The team split the cost of this with Team 6, so the total for each team was two hundred nine dollars. Stream Come True also purchased two printed circuit boards from PCB Express for 496 dollars. These are the major purchases that Stream Come True had to make, and where most of the prototype budget went. Along with these major purchases there are many smaller purchases, such as resistors and capacitors and other small components that were required to make the project work. A document containing every purchase the team made, and the prices of those items, can be found in Appendix 9.4.

Appendix 9.5, the bill of materials, shows a listing of all of the parts that are necessary to construct a handheld device. Everything listed is a direct price that was found, except for the casing which is based on information from Ross Gates, who made the prototype case for the team. When purchasing items for 100 handhelds, there is generally a discount on the price. A 25% discount is assumed for the DPAC modules, wireless Bluetooth modules, the battery, the Triton modules, and the LCD, because they did not have listed discounts for buying in bulk. Twenty five percent was chosen based on the discounts that were listed for other items. Stream Come True’s system, based on the listing of all the parts that goes into a board, cost $1126.69 because the case was donated, and the assembly was done by the team. Including the development boards and everything else that the team needed to buy, Stream
Come True spent a total of $4609.69 on the project. While this may seem like a lot of money, Stream Come True considers the budged acceptable, because $4900 was allotted to finish the project, and the team came in under budget. Starting from scratch, if only 1 board was to be made, the entire handheld system would cost $1286.69. If the handhelds were made 100 at a time, then a large discount could be applied to some of the items, reducing the cost of each handheld device to $681.72.

4.4 Project Management

4.4.1 Team Management

SCT teams roles were not defined however, each member had assumed roles where their strengths were applicable as well as where they are needed most. A graphical organization chart can be seen below in figure below. SCT had defined since the beginning of the project that Ben Bufford would be the team leader. The team lead is ultimately responsible of the progress of the project, deliverables being delivered on time, and communication resolving team dynamic problems. A summary of the work done by each team member is shown below.

![Team organizational chart](image)

4.4.1.1 Ben Bufford

Ben had been responsible for setting up the team FTP server, where the team stores materials related to the project. Ben also worked on parts procurement, for both the initial Itsy project as well as for the SBC contingency plan. Ben also submitted a budget proposal to Smiths Aerospace. Ben also performed research on the Itsy and several SBC alternatives. Ben was also part of the project scheduling and PPFS documentation. Ben continued through second semester with the development of software
requirements for our computer science team member. Ben worked diligently on debugging and understanding the system platform and integration of all components to the SBC. Ben also aided in production of printed circuit boards at Calvin College. Ben also reviewed the final schematics for the prototype board as well as aided in routing the prototype board. Ben has also prepared and given presentations on project night along with writing the final report.

4.4.1.2  Jermey Gajadhar

In the first semester, Jermey Gajadhar did some initial research on PCMCIA and how to integrate the communication protocol into the project. Jermey also did a separate budget from Ben Bufford and submitted it to Smiths Aerospace. Jermey worked diligently on parts procurement for both the Itsy as well as the SBC contingency plan. Jermey also did separate research on SBC and worked on developing a project schedule. Jermey also contribute to the PPFS documentation. In the second semester, Jermey did research on each component and integrated them both on component boards as well as a final schematic in Eagle Cad. Jermey also began all board integration through in house PCBs at Calvin College. Jermey was the initial funnel for all orders produced by the team. Jermey fabricated all libraries for parts necessary in Eagle Cad, researched issues on trace impedances, width and lengths as well routed the final prototype board. Jermey also worked on soldering components on PCBs as well as the final prototype board. The team member also assisted on software issues and integration. Jermey has also prepared and given presentations along with writing the final report.

4.4.1.3  Nick Goote

In the first semester, Nick Goote did some research on Bluetooth technology and how to integrate the wireless protocol into the project. Nick also spent some time procuring parts. Nick contributed to the PPFS documentation and project scheduling. In the second semester Nick developed a power supply design for the SBC. Nick also worked diligently in developing a case for housing the handheld device. Nick also spent time soldering components for breadboard integration purposes as well as developing PCBs for integration. Nick worked on software debugging and integration issues. Nick was also reviewed the final schematic for the prototype. Nick has also prepared and given presentations on project night along with writing the final report.

4.4.1.4  Andrew Oosterhouse

In the first semester Andrew Oosterhouse worked on meeting minutes for important meeting with industrial consultants and mentors. Andrew has done research on PCMCIA for integration into the project as we as research on SBCs. Andrew has also been involved in part procurement and project
scheduling. Andrew worked on writing and editing the PPFS document. In the second semester Andrew has worked on meeting all academic deadlines concerning the project. Andrew has also programmed the development kit for the 802.11 integration of the project. Andrew continually delivered status reports to advisors, updated the team website, kept track on team hours and performed backups of the server. Andrew was also responsible for poster making and preparing the final report. Andrew has also prepared and given presentations.

### 4.4.2 Schedule

The project schedule was developed at the beginning of the project using Microsoft Project 2003. The schedule reflects all the tasks necessary to complete the project as well as the estimated time required for each project task. SCT had assumed an initial working day of 1 hour per team member however, the time commitment kept increasing as the deadline came closer.

Team SCT performed poorly with its predicted schedule. SCT fell behind schedule early on due to funding issues. This lack of funding, and the subsequent waiting, forced SCT to abandon plans for the Itsy and switch to our SBC contingency plan. The team received funding during the month of January and was aware of the significant time loss. In light of that, the team planned a revision of the project schedule that included a SBC project line. All revisions to the schedule were managed through Subversion. Even with the revised schedule SCT still did not allot enough time for project tasks. Based on the task specifications SCT saw that the estimates used to create the project timeline were incorrect. The final numbers in comparison to the projects’ are shown below. A full detailed schedule and hour management can be found in the appendix.

![Figure 16 - Summary of Project Schedule](image-url)
One may notice that SCT had hoped to deliver a functional system on April 2, 2006, while in actuality SCT delivered a functional system on Friday May 5 2006. While SCT had originally planned to finish the entire project by the end of April, the month of May was very important in wrapping up the project. SCT also realized based on the task specification and the cumulative hours reported that SCT significantly underestimated the required hours. SCT had allotted 1,146 hours for the completion of the project, yet the actual hours came to be over 1,400 hours. The team noticed that on average the best way to estimate the time a task would take is to double the specified hours.

### 4.4.3 Task specifications

The team’s task specifications can be found below. As the semester progressed, the team discovered that certain tasks were redundant or completely unnecessary. These tasks, while the team budgeted time to complete them, were actually not needed, and thus no time was spent on them. For example, under the “subsystem” section, time was budgeted to test and debug specific components. However, since most of the components were modular or purchased from outside vendors, extensive testing was not necessary.

<table>
<thead>
<tr>
<th>Task</th>
<th>Estimated Time Needed</th>
<th>Actual Time Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Reports</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Keeping</td>
<td>8 hours</td>
<td>8 hours</td>
</tr>
<tr>
<td>Writing Report</td>
<td>8 hours</td>
<td>8 hours</td>
</tr>
<tr>
<td>Schedule Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Making Schedule</td>
<td>8 hours</td>
<td>6 hours</td>
</tr>
<tr>
<td>Updating Schedule</td>
<td>8 hours</td>
<td>8 hours</td>
</tr>
<tr>
<td>Budget Reporting</td>
<td>8 hours</td>
<td>5 hours</td>
</tr>
<tr>
<td>Team Meetings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meeting with Industrial Consultant</td>
<td>4 hours</td>
<td>4 hours</td>
</tr>
<tr>
<td>Formal Team Meetings</td>
<td>8 hours</td>
<td>20 hours</td>
</tr>
<tr>
<td>Informal Team Meetings</td>
<td>20 hours</td>
<td>25 hours</td>
</tr>
<tr>
<td>Overall System Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall project goals</td>
<td>8 hours</td>
<td>8 hours</td>
</tr>
<tr>
<td>Viable project alternatives</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial ITSY Feasibility</td>
<td>4 hours</td>
<td>4 hours</td>
</tr>
<tr>
<td>Initial Single Board Computer Feasibility</td>
<td>4 hours</td>
<td>4 hours</td>
</tr>
<tr>
<td>Initial PDA Feasibility</td>
<td>4 hours</td>
<td>4 hours</td>
</tr>
<tr>
<td>ITSY Device</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feasibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine overall cost of fabrication</td>
<td>8 hours</td>
<td>3 hours</td>
</tr>
<tr>
<td>Activity</td>
<td>Time (hours)</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>Determine overall cost of assembly</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Determine overall cost of Digikey parts</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Determine overall cost of obsolete parts</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Determine interface possibilities</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Determine overall Time for assembly</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Determine tools required</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Single Board Computer Device</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Feasibility</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine interface possibilities</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Determine tools required</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Assembly</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procure Digikey parts</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Procure obsolete parts</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Assemble Parts</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Software (EE tasks)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile and install Operating System</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Find necessary device drivers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examine Linux Kernel for native device support</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Search for pre-made open source drivers</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Research implementation of pre-made drivers</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Design necessary device drivers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research programming of device drivers</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Search for device driver examples</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Search for similar device drivers</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Program Device Driver</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Debug any OS or driver related problems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug OS related problems</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Debug driver installation problems</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Debug driver compile problems</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Debug driver functionality problems</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Testing and Debugging</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify requirements to be tested</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Identify additional tests</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Write test checklist</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Identify/Procure test equipment needed</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Run driver tests</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Run handheld tests</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Document test results</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Debug problems discovered during test</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Re-run tests</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Sub-systems</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research necessary components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research Bluetooth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research IC's</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Research Development Kits</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Hours</td>
<td>Hours</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Research Bluetooth Software 8 hours 5 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyze Options 8 hours 5 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Solution 8 hours 7 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research 802.11 hardware and software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research IC’s 8 hours 8 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research Development Kits 8 hours 8 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research Software 8 hours 5 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyze Options 8 hours 4 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Solution 8 hours 2 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research power requirements and implementation of daughter-card</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research Power Requirements 8 hours 8 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research Possible Interfaces 8 hours 8 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choose Best Interface 8 hours 5 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research board fabrication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research trace length/width requirements 4 hours 6 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research number of required board layers 8 hours 8 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research local board fabricators 8 hours 8 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schematic Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAD software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyze different CAD options 8 hours 5 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test different options 8 hours 4 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choose best option 8 hours 6 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Tutorials 8 hours 20 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Practice PCB design 8 hours 20 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Practice Schematic design 8 hours 8 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design Daughter card 8 hours 150 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layout daughter card on PCB 8 hours 100 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify relevant requirements 8 hours 8 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obtain Simulation Software 8 hours 8 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run necessary simulations 8 hours 2 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have schematic reviewed by Corrin 8 hours 4 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send board out for fabrication 2 hours 2 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test and debug sub-systems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify requirements to be tested 8 hours 1 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify additional tests 4 hours 1 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write test checklist 4 hours 0 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify/Procure test equipment needed 8 hour 0 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run tests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run IC tests 8 hours 1 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perform Continuity tests 8 hours 0 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document test results 6 hour 0 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug problems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify problems 7 hours 0 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design tests to isolate problems 7 hours 0 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Description</td>
<td>Hours</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Test isolated problems</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Repair problems</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Document repair</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Update Schematic</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Document test results</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Re-run tests</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Design daughter-card to handheld Interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify interface options</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Analyze advantages of each option</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Choose best option</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Design interface based on best option</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Test interface</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Implement interface</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Test and debug combined system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define tests</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Run tests</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Document results</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Isolate problems</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Test isolated problems</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Repair problems</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Document results</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Re-run tests</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Update Schematic</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td><strong>Media cataloguing system</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EE tasks</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Catalog sample audio and video</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Procure Thin Client Terminals</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Procure Server Computer</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Obtain Bluetooth cards</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Vendor</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Decide on card</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Procure Card</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td><strong>Budget</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obtain external funding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submit proposal/budget to Smiths</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Contact PCB Express for donations</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Procure funding from Smiths</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Produce detailed budget for both handheld and daughter-card</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Produce detailed budget for ITSY</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>Produce detailed budget for SBC</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Produce detailed budget for daughter-card</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>Produce production budget</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>Periodically revise budgets</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td><strong>CS tasks</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Media Cataloguing System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Hours 1</td>
<td>Hours 2</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Design custom interface software for thin client/server interface</td>
<td>60 hours</td>
<td>60 hours</td>
</tr>
<tr>
<td>Debug and test custom software</td>
<td>40 hours</td>
<td>40 hours</td>
</tr>
<tr>
<td>Implement custom software</td>
<td>10 hours</td>
<td>10 hours</td>
</tr>
<tr>
<td><strong>Handheld Software</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design and compile custom software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBC media selection software</td>
<td>30 hours</td>
<td>30 hours</td>
</tr>
<tr>
<td>Thin Client Software</td>
<td>10 hours</td>
<td>10 hours</td>
</tr>
<tr>
<td>Upload custom software to memory</td>
<td>1 hour</td>
<td>1 hours</td>
</tr>
<tr>
<td>Debug custom software problems</td>
<td>20 hours</td>
<td>20 hours</td>
</tr>
<tr>
<td>Prototype casing</td>
<td>0 hours</td>
<td>30 hours</td>
</tr>
<tr>
<td><strong>Prepare PPFS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write detailed table of contents</td>
<td>5 hours</td>
<td>5 hours</td>
</tr>
<tr>
<td>Write PPFS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Introduction/Conclusion</td>
<td>3 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write Educational and Design Objectives</td>
<td>6 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write Design Norms</td>
<td>4 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write Project task list and time allocation</td>
<td>8 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write Research section</td>
<td>8 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write Project Management</td>
<td>5 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write Team Organization</td>
<td>5 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write Preliminary Feasibility Analysis</td>
<td>8 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write Software Design</td>
<td>5 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write Project Requirements</td>
<td>8 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td>Write System Design</td>
<td>8 hours</td>
<td>9 hours</td>
</tr>
<tr>
<td><strong>Prepare final presentation for project night</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write presentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Project Night Presentation</td>
<td>8 hours</td>
<td>8 hours</td>
</tr>
<tr>
<td>Write Final Class Presentation</td>
<td>7 hours</td>
<td>2.5 hours</td>
</tr>
<tr>
<td>Rehearse presentation</td>
<td>20 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td><strong>Prepare final report</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write final report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Chapter 1</td>
<td>8 hours</td>
<td>3 hours</td>
</tr>
<tr>
<td>Write Chapter 2</td>
<td>8 hours</td>
<td>3 hours</td>
</tr>
<tr>
<td>Write Chapter 3</td>
<td>8 hours</td>
<td>3 hours</td>
</tr>
<tr>
<td>Write Chapter 4</td>
<td>8 hours</td>
<td>3 hours</td>
</tr>
<tr>
<td>Write Chapter 5</td>
<td>8 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td>Write Chapter 6</td>
<td>8 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td>Write Chapter 7</td>
<td>8 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td>Write Chapter 8</td>
<td>8 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td>Write Chapter 9</td>
<td>8 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td>Write Chapter 10</td>
<td>8 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td>Write Chapter 11</td>
<td>8 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td>Write Chapter 12</td>
<td>8 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td>Proofread final report</td>
<td>15 hours</td>
<td>15 hours</td>
</tr>
</tbody>
</table>
5. Remaining Issues

While SCT was able to make significant progress during the course of this project, there were a few issues which SCT was not able to resolve. The first of these issues is the incorporation of the video terminal into the prototype IMS system. Due to time constraints, SCT was unable to bring the video portion of the IMS to fruition. The hardware needed for that portion of the system, namely a standard PC running Linux and a Bluetooth capable handheld, were completed on schedule. The last remaining hurdle is the software. Custom software would be needed on the handheld device, as well as the video terminal PC to establish our custom Bluetooth connection. SCT currently does not have the code available to make the video terminal functional. However, given more time, SCT could easily complete this facet of the system.

Another remaining issue with the project is the availability of user I/O. While the GUI was successfully implemented, the buttons remain unavailable for user input. The hardware is available, but the software to read signals from the General Purpose Input and Output (GPIO) pins remains unfinished. This makes it difficult for the user to interact properly with the handheld device. SCT is confident that this issue could also have been resolved, if more time was available.

Finally, SCT was not able to produce a fully functional, handheld prototype. Our handheld prototype was partially functional, and would have attained full functionality if more time was available to configure all of the components properly. SCT is confident that a fully functional handheld prototype can be produced, since the team was able to produce a fully functional prototype using break-out boards to connect the system components together. With the knowledge gained from our first, non-mobile prototype, it would only be a matter of time before SCT was able to make the handheld version fully functional as well.

6. Conclusion

Overall, the project was a success. SCT was able to construct a working prototype of the handheld device, and thus was able to prove the concept behind the IMS. While not all of the major goals were completed by the established deadlines, SCT was able to complete the majority of the tasks specified at the beginning of the project (the exception being the video aspect of the IMS). SCT believes the work that has been put into the project has not only satisfied the immediate project objective, but opened the door to more project possibilities in the future.
7. Recommendations

This attempt by SCT to generate a functional prototype of the Integrated Media System has brought many other exciting opportunities to light. These new opportunities can be divided into two groups: additions to the current project, and new applications for the concept behind the IMS.

7.1 Additions to the Integrated Media System

While SCT was able to make excellent progress on this project, some issues remain to be resolved. The video terminal remains non-functional, and a fully functional, handheld prototype was never fully implemented. Both of these would make excellent project ideas for someone with the software expertise to make the hardware generated by SCT operational.

Additionally, there are multiple improvements that can be made to the prototype which SCT generated. First, it should be possible to incorporate the use of a touch screen interface into a future revision of the IMS. While the LCD that SCT purchased has hardware support for a touch screen interface, SCT believed it was beyond the scope of feasibility to utilize such an interface in the current design. All that is needed for this to happen is a method by which the touch screen interface would interact with the Triton processor. As the version of the Triton obtained by SCT does not have built-in touch screen support, this would take significant hardware and software design.

Finally, the IMS would benefit from a project to make the components much more integrated. This integration would save greatly on costs. For example, the DPAC module which allows for wireless 802.11 communication is rather expensive, with a price of $129 per module\(^6\). It would make an excellent project to design a custom integrated 802.11 module for the IMS, and also investigate if this would lower the overall cost of the IMS in the process.

7.2 New Applications for the IMS concept

In addition to the possible additions to the IMS system, the IMS system itself can be applied in new ways. One of these new applications could be as a self-guided tour around a college campus. In this scenario, the handheld device would not require any hardware modification, and software modification would be minimal. The setup could consist of a wireless network infrastructure in the areas where the tour will take place. A Bluetooth base station could be placed at specific points of significance along the tour. At these points, the base station would trigger the playback of a certain audio file stored on a central media server. This file could be a pre-recorded explanation of the area currently being viewed by the tour. The LCD would not be utilized in the same manner as in the IMS envisioned by SCT, since the recordings would be selected based on where the user was located, not by what they wanted to listen to.

\(^6\) http://www.quatech.com/catalog/airborne_wirelessdeviceserver_modules_emb.php
However, it would be possible to have the LCD display a map of the campus (or other area being toured) which would update every time the user came to a new base station. In this way, the user would know their exact location throughout the course of the tour.

8. Acknowledgements

SCT received assistance from many sources in order to complete this project. Without this external assistance, the project would not have been possible. Specifically, SCT would like to thank:

- Smiths Aerospace, for funding the project
- DornerWorks, for allowing SCT to use their facility to solder PCBs
- Corrin Meyer, for his PCB design expertise
- Brian Krug, for his assistance in designing the power supply for the handheld device
- Tim Theriault, for serving as SCT’s industrial consultant
- PCB Express, for donating their time to fabricate our PCBs
- Professor Steve VanderLeest, for being an excellent team advisor, and helping to keep the team focused on the project objectives
- Senior design team #6 (Silence is Golden) for their assistance with Bluetooth
- Ross Gates, for his willingness to help SCT fabricate the prototype casing for the handheld device
- Andy Wallner, for helping various members of SCT solder components
9. Appendices

Contents

9.1 UML diagram of central server software ................................................................. II
9.2 UML for Client Software .......................................................................................... III
9.3 Source Code .............................................................................................................. IV
  9.3.1 Handheld application source code ......................................................................... IV
      9.3.1.1 CayvanView.cpp ......................................................................................... IV
      9.3.1.2 CayvanView.h .............................................................................................. VI
      9.3.1.3 CayvanApplication.cpp ............................................................................... VII
      9.3.1.4 CayvanApplication.h ................................................................................... VIII
      9.3.1.5 PlayerDialog.cpp ......................................................................................... IX
      9.3.1.6 PlayerDialog.h ............................................................................................. XI
      9.3.1.7 CayvanConfig.cpp ....................................................................................... XII
      9.3.1.8 CayvanConfig.h .......................................................................................... XIII
      9.3.1.9 ServerConnection.cpp .................................................................................. XIV
      9.3.1.10 ServerConnection.h ................................................................................... XVI
      9.3.1.11 OggPlayer.cpp ............................................................................................. XVII
      9.3.1.12 OggPlayer.h ................................................................................................ XVIII
  9.3.2 Server Code .......................................................................................................... XXI
      9.3.2.1 CayvanServer.rb .......................................................................................... XXI
      9.3.2.2 Server.rb ..................................................................................................... XXII
  9.3.3 Triton Module Source Code .................................................................................... XXIV
      9.3.3.1 Excerpts from pxafb.h .................................................................................. XXIV
      9.3.3.2 Excerpts from karo-config.c ......................................................................... XXV
      9.3.3.3 Excerpts from karo.c .................................................................................... XXVI
  9.4 Budget Summary ....................................................................................................... XXXI
  9.5 Bill of Materials ........................................................................................................ XXXII
  9.6 Complete Project Schedule ........................................................................................ XXXIII
  9.7 Pictures of final handheld PCB layout ....................................................................... XXXIX
  9.8 Final circuit layout of handheld device ....................................................................... XLI
9.1 UML diagram of central server software

```
CayvanServer

+get_departments(): Array
+get_classes(dept: String): Array
+get_audio_files(dept: String, class_num: String): Array
+get_video_files(dept: String, class_num: String): Array
```

Figure 17 - Server UML
9.2 UML for Client Software

Figure 18 - UML diagram for handheld
9.3 Source Code

9.3.1 Handheld application source code

9.3.1.1 CayvanView.cpp

```cpp
#include "CayvanView.h"
#include <QGridLayout>
#include <QVBoxLayout>
#include <Qt>
#include <QSpacerItem>

CayvanView::CayvanView(QWidget *parent)
    : QWidget(parent)
{
    m_List = new QListWidget();
    m_QuitButton = new QPushButton(tr("Quit"));
    m_QuitButton->setFont(QFont("Times", 18, QFont::Bold));

    QGridLayout *mainLayout = new QGridLayout();
    QVBoxLayout *infoLayout = new QVBoxLayout();

    // Set up information screen
    m_DeptLabel = new QLabel("Department: ");
    m_CourseLabel = new QLabel("Course: ");
    m_FileLabel = new QLabel("File: ");

    infoLayout->addWidget(new QLabel("Current Information:"));
    infoLayout->addWidget(m_DeptLabel);
    infoLayout->addWidget(m_CourseLabel);
    infoLayout->addWidget(m_FileLabel);

    mainLayout->setColumnMinimumWidth(0, 20);
    mainLayout->setColumnMinimumWidth(2, 40);
    mainLayout->setColumnMinimumWidth(3, 100);
    mainLayout->addWidget(m_List, 2, 1);
    mainLayout->addLayout(infoLayout, 2, 3);

    // Get rid of title bar
    setWindowFlags(Qt::FramelessWindowHint);

    // Make the UI take up the whole screen
    // TODO better way to do this?
    setMaximumHeight(240);
    setMinimumHeight(240);
    setMaximumWidth(320);
    setMinimumWidth(320);
    setLayout(mainLayout);

    connect(m_QuitButton, SIGNAL(clicked()), this, SIGNAL(quit()));
    connect(m_List, SIGNAL(itemActivated(QListWidgetItem*)), this, SIGNAL(itemClicked(QListWidgetItem*)));
}

CayvanView::~CayvanView()
{
    delete m_QuitButton;
}
```
```cpp
delete m_List;
}

void CayvanView::setListContents(QStringList items)
{
    m_List->clear();
    m_List->addItem(items);

    // Make sure first item is selected to prevent confusion
    //m_List->setItemSelected(m_List->item(0), true);
    m_List->setCurrentRow(0);
}

void CayvanView::keyPressEvent(QKeyEvent *event)
{
    emit keyPressed(event);
}

void CayvanView::showPlayer(IMediaPlayer* player, QString filename, CayvanConfig* config)
{
    // TODO Put this somewhere else?
    // Weird. It seg faults. Definite memory leak here I'm afraid
    // if (m_Player)
    // {
    //     cout << "Player exists, kill it" << endl;
    //     delete m_Player;
    // }
    m_Player = new PlayerDialog(player, filename, config);
    m_Player->show();
}

void CayvanView::setState(AppState newState, QString dept, QString course, QString file)
{
    QString deptText = "Department: " + dept;
    QString courseText = "Course: " + course;
    QString fileText = "File: " + file;

    switch(newState)
    {
    case DEPT_LIST:
        deptText = "<b>" + deptText + "</b>";
        break;
    case COURSE_LIST:
        courseText = "<b>" + courseText + "</b>";
        break;
    case FILE_LIST:
    case PLAYING_FILE:
        fileText = "<b>" + fileText + "</b>";
        break;
    }

    m_DeptLabel->setText(deptText);
    m_CourseLabel->setText(courseText);
    m_FileLabel->setText(fileText);
}
```
#ifndef __CAYVAN_VIEW_H__
#define __CAYVAN_VIEW_H__

#include <QWidget>
#include <QPushButton>
#include <QListWidget>
#include <QListWidgetItem>
#include <QKeyEvent>
#include <QLabel>
#include <QString>
#include "ServerConnection.h"
#include "PlayerDialog.h"
#include "common.h"
#include "CayvanConfig.h"
#include "IMediaPlayer.h"

/**
 * This class contains all of the view logic of the application. It creates
 * the needed widgets and controls the PlayerDialog, communicating through
 * the Application class when needed.
 */

class CayvanView : public QWidget
{
    Q_OBJECT

public:
    CayvanView(QWidget *parent = 0);
    ~CayvanView();

    /** Updates the list widget with new entries */
    void setListContents(const QStringList list);

    /** Called when a key is pressed */
    void keyPressEvent(QKeyEvent *event);

    /** Sets the state labels to show what the user has selected */
    void setState(AppState newState, QString dept = "", QString course = "",
                  QString file = "");

    /** This function shows the PlayerDialog with a given media player */
    void showPlayer(IMediaPlayer *player, QString filename, CayvanConfig* config);

    signals:
    void quit();
    void itemClicked(QListWidgetItem *item);
    void keyPressed(QKeyEvent *event);

private:
    QPushButton *m_QuitButton;
    QListWidget *m_List;
    QLabel *m_DeptLabel;
    QLabel *m_CourseLabel;
    QLabel *m_FileLabel;
    PlayerDialog *m_Player;
};

#endif // __CAYVAN_VIEW_H__
CayvanApplication.cpp

```cpp
#include "CayvanConfig.h"
#include <QFile>

CayvanConfig::CayvanConfig(QString configFile)
{
    QFile file(configFile);
    QString line;
    QString key, value;

    // Don't care if we can't open the file
    if (file.open(QIODevice::ReadOnly | QIODevice::Text))
    {
        while (!file.atEnd())
        {
            line = QString(file.readLine());
            line = line.trimmed();

            if (line.length() == 0)
            {
                continue;
            }

            key = line.section('=', 0, 0);
            value = line.section ('=', 1);

            m_Opts.insert(key, value);
        }
    }
    file.close();
}

CayvanConfig::~CayvanConfig()
{
}

QString CayvanConfig::getConfigOption(QString key)
{
    QString out = m_Opts.value(key);
    return out;
}
```
9.3.1.4 CayvanApplication.h

```c++
#ifndef __CAYVAN_APPLICATION_H__
#define __CAYVAN_APPLICATION_H__

#include <QApplication>
#include <QListWidgetItem>
#include <QStringList>
#include <QKeyEvent>
#include <QString>
#include "CayvanView.h"
#include "ServerConnection.h"
#include "CayvanConfig.h"
#include "common.h"

/**
 * This is the main runner class for the application. It creates the view and
 * server connection and controls the communication between the two classes.
 */

class CayvanApplication : public QApplication
{
    Q_OBJECT

public:
    CayvanApplication(int argc, char** argv);
    ~CayvanApplication();

    /** Tells the view to update the current state of the application */
    void updateViewState();

    /** Tells the view to show the player dialog */
    void showPlayer();

    /** Wrapper for choosing file lists from the server depending on the
     * current state of the application, whether it's showing audio
     * or video clips
     */
    QStringList getFileList();

public slots:
    void itemSelected(QListWidgetItem *item);
    void keyPressed(QKeyEvent *event);

private:
    CayvanView *m_View;
    ServerConnection *m_Model;
    CayvanConfig *m_Config;

    QString m_SelectedDept;
    QString m_SelectedCourse;
    QString m_SelectedFile;
    AppState m_CurrentState;
    AppMode m_PlayerMode;
};

#endif // __CAYVAN_APPLICATION_H__
```
9.3.1.5 PlayerDialog.cpp

#include <QGridLayout>
#include "PlayerDialog.h"
#include "OggPlayer.h"
#include "VideoPlayer.h"

PlayerDialog::PlayerDialog(IMediaPlayer* player, QString filename, CayvanConfig* config)
: m_Paused(false)
{
    // TODO? move to CayvanApplication, could be taking a long time.
    // compare time of this with time of opening up a new dialog
    m_Player = player;
    m_Player->initialize();

    m_CurrentFile = filename;

    m_Label = new QLabel("Now Playing");
    m_List = new QListWidget();

    QStringList list;
    list.push_back("Play");
    list.push_back("Pause");
    list.push_back("Stop");
    list.push_back("" );
    list.push_back("Close");

    m_List->addItems(list);
    m_List->setCurrentRow(0);

    QGridLayout *layout = new QGridLayout();
    layout->addWidget(m_Label, 0, 0);
    layout->addWidget(m_List, 2, 0);

    connect(m_List, SIGNAL(itemActivated(QListWidgetItem*)),
            this, SLOT(itemSelected(QListWidgetItem*)));

    setLayout(layout);
}

PlayerDialog::~PlayerDialog()
{
    if (m_Player)
    {
        delete m_Player;
    }
}

void PlayerDialog::itemSelected(QListWidgetItem* item)
{
    QString selection = item->text();

    if (selection != "")
    {
        if (selection == "Play")
        {
            play();
        }
        else if (selection == "Pause")
        {
            // do nothing
        }
    }
}
void PlayerDialog::accept()
{
    m_Player->stop();
    if (m_Player)
    {
        delete m_Player;
    }
    QDialog::accept();
}

void PlayerDialog::show()
{
    play();
    QDialog::show();
}

void PlayerDialog::play()
{
    m_Player->play(m_CurrentFile);
    updateLabel("Now Playing");
    m_Paused = false;
}

void PlayerDialog::pause()
{
    m_Paused = !m_Paused;
    m_Player->pause();
    if (m_Paused)
    {
        updateLabel("Paused");
    }
    else
    {
        updateLabel("Now Playing");
    }
}

void PlayerDialog::stop()
{
    m_Player->stop();
    updateLabel("Stopped");
}

void PlayerDialog::updateLabel(QString newState)
{
    m_Label->setText(newState + ": " + m_CurrentFile);
}

else if (selection == "Stop")
{
    stop();
}
else if (selection == "Close")
{
    accept();
}
}
9.3.1.6 PlayerDialog.h

#ifndef __PLAYER_DIALOG_H__
#define __PLAYER_DIALOG_H__

#include <QDialog>
#include <QListWidget>
#include <QString>
#include <QLabel>
#include "IMediaPlayer.h"
#include "CayvanConfig.h"

/**
 * This is the Dialog that the user uses to control playing of audio
 * and video. Because of the limited input mechanics of Up, Down, and Enter,
 * this dialog is simply a list widget with a few commands that are
 * selectable by the user.
 */

class PlayerDialog : public QDialog
{
    Q_OBJECT

public:
    PlayerDialog(IMediaPlayer* player, QString filename, CayvanConfig* config);
    ~PlayerDialog();

    /** From QDialog, shows the widget */
    void show();

    /** Updates the label of the widget to show what is currently happening */
    void updateLabel(QString newState);

    /** From QDialog, closes the dialog */
    void accept();

    /** Manipulation commands that are passed down to the appropriate player */
    void play();
    void pause();
    void stop();

public slots:
    void itemSelected(QListWidgetItem* item);

private:
    QString m_CurrentFile;
    QLabel *m_Label;
    QListWidget *m_List;
    bool m_Paused;
    IMediaPlayer* m_Player;
};

#endif // __PLAYER_DIALOG_H__
#include "CayvanConfig.h"
#include <QFile>

CayvanConfig::CayvanConfig(QString configFile)
{
    QFile file(configFile);
    QString line;
    QString key, value;

    // Don't care if we can't open the file
    if (file.open(QIODevice::ReadOnly | QIODevice::Text))
    {
        while (!file.atEnd())
        {
            line = QString(file.readLine());
            line = line.trimmed();

            if (line.length() == 0)
            {
                continue;
            }

            key = line.section('=', 0, 0);
            value = line.section ('=', 1);

            m_Opts.insert(key, value);
        }
    }
    file.close();
}

CayvanConfig::~CayvanConfig()
{
}

QString CayvanConfig::getConfigOption(QString key)
{
    QString out = m_Opts.value(key);
    return out;
}
```cpp
#ifndef __CAYVAN_CONFIG_H__
#define __CAYVAN_CONFIG_H__

#include <QHash>
#include <QString>

/**
 * This is a simple class that reads in a config file and saves <key, value>
 * pairs for reading by the application
 *
 * Config file is simply formatted as:
 * key=value
 * key2=value2
 */

class CayvanConfig
{
public:
    CayvanConfig(QString configFile = "config.ini");
    ~CayvanConfig();
    QString getConfigOption(QString key);

private:
    QHash<QString, QString> m_Opts;
};
#endif // __CAYVAN_CONFIG_H__
```
#include <string>
#include "ServerConnection.h"
#include <iostream>
using namespace std;

ServerConnection::ServerConnection(CayvanConfig *config)
{
    m_Config = config;
}

ServerConnection::~ServerConnection()
{
}

QStringList ServerConnection::getDepartmentList()
{
    return sendRequest("get_departments");
}

QStringList ServerConnection::getCourseList(QString department)
{
    XmlRpcValue param(department.toStdString());
    return sendRequest("get_courses", param);
}

QStringList ServerConnection::getAudioFileList(QString department, QString course)
{
    XmlRpcValue params;
    params[0] = department.toStdString();
    params[1] = course.toStdString();
    return sendRequest("get_audio_files", params);
}

QStringList ServerConnection::getVideoFileList(QString department, QString course)
{
    XmlRpcValue params;
    params[0] = department.toStdString();
    params[1] = course.toStdString();
    return sendRequest("get_video_files", params);
}

QStringList ServerConnection::sendRequest(QString methodName, XmlRpcValue params)
{
    QStringList list;
    XmlRpcValue retVal;
    QString serverURL = m_Config->getConfigOption("serverurl");
    QString serverPort = m_Config->getConfigOption("serverport");
    XmlRpcClient server(serverURL.toStdString().c_str(), serverPort.toInt());

    if (server.execute(methodName.toStdString().c_str(), params, retVal))
    {
        for (int i=0; i < retVal.size(); i++) {

XIV
list.push_back(QString{ ((std::string) retVal[i]).c_str() });
}

else
{
    list.push_back("ERROR: ");
    list.push_back("There was a problem");
    list.push_back("communicating with the");
    list.push_back("server.");
}

server.close();

return list;
9.3.1.10 ServerConnection.h

#ifndef __SERVER_CONNECTION_H__
#define __SERVER_CONNECTION_H__

#include <QString>
#include <QStringList>
#include "include/XmlRpc.h"
#include "include/XmlRpcValue.h"
#include "CayvanConfig.h"

using namespace XmlRpc;

/**
 * This class holds all of the xmlrpc communication code to interact with
 * the server.
 */

class ServerConnection
{
    public:
        ServerConnection(CayvanConfig *config);
        ~ServerConnection();

        /**
         * Get the complete list of available departments
         *
         */
        QStringList getDepartmentList();

        /**
         * Get all course numbers for a given department
         *
         */
        QStringList getCourseList(QString department);

        /**
         * Get all files available for the given department
         * and course
         */
        QStringList getAudioFileList(QString department, QString course);
        QStringList getVideoFileList(QString department, QString course);

    protected:
        /**
         * Wrapper function to prevent duplication of code in sending an XMLRPC
         * request
         */
        QStringList sendRequest(QString methodName, XmlRpcValue params = XmlRpcValue());

    private:
        CayvanConfig *m_Config;
};

#endif //__SERVER_CONNECTION_H__
9.3.1.11  OggPlayer.cpp

```cpp
#include "OggPlayer.h"

#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <string>
using namespace std;

#define MAX_PATH_LEN 512
#define OGG123_PATH "./ogg123"

OggPlayer::OggPlayer(CayvanConfig* config)
  : IMediaPlayer()
  {
    m_Config = config;
  }

OggPlayer::~OggPlayer()
  {
    stop();
    close(m_SendFD);
    close(m_RecvFD);
    kill(m_Ogg123PID, SIGTERM);
    // execute a waitpid, so we do not end up with those nasty zombies...
    // waitpid(m_Ogg123PID, NULL, 0);
  }

bool OggPlayer::initialize()
  {
    int fd_send[2];
    int fd_recv[2];
    int oldstat;

    // make socketpair to communicate with player
    //
    if (socketpair(AF_UNIX, SOCK_STREAM, 0, fd_send) < 0) {
      //printf("mod_ogg123_start(): unable to setup send pipe for ogg123 (%s).\n", strerror(errno));
      return false;
    }
    if (socketpair(AF_UNIX, SOCK_STREAM, 0, fd_recv) < 0) {
```
// printf("mod_ogg123_start(): unable to setup recv pipe for ogg123 (%s).\n", strerror(errno));
return false;
}

// fork child process
m_Ogg123PID = fork();
if (m_Ogg123PID == 0)
{
    // pipe in/output through socket to parent
dup2(fd_send[0], STDIN_FILENO);
close(fd_send[0]);
close(fd_send[1]);
dup2(fd_recv[0], STDOUT_FILENO);
dup2(fd_recv[0], STDERR_FILENO);
close(fd_recv[0]);
close(fd_recv[1]);

    // spawn player
    execvp(OGG123_PATH, args);

    // never reached if exec was ok
    exit(-1);
}

// parent continues here

close(fd_send[0]);
close(fd_recv[0]);
m_SendFD = fd_send[1];
m_RecvFD = fd_recv[1];

oldstat = fcntl(m_RecvFD, F_GETFL, 0);
fcntl(m_RecvFD, F_SETFL, oldstat | O_NONBLOCK);

// check if player is running
sleep(1);
if (!m_Ogg123PID)
{
    //printf("mod_ogg123_start(): player process didn't start!!\n");
    return false;
}

return true;
}

void OggPlayer::play(QString filename)
{
    char buf[MAX_PATH_LEN];
    string serverURL = m_Config->getConfigOption("streamserverurl").toString();
    sprintf(buf, "LOAD %s/%s\n", serverURL.c_str(), filename.toString().c_str());
    sendCommand(buf);
}
void OggPlayer::pause()
{
    sendCommand("PAUSE\n");
}

void OggPlayer::stop()
{
    sendCommand("STOP\n");
}

void OggPlayer::sendCommand(const char* cmd)
{
    ssize_t len;
    //cout << "Sending: " << cmd << " to " << m_SendFD << endl;
    len = write(m_SendFD, cmd, strlen(cmd));
}
#ifndef __OGG_PLAYER_H__
#define __OGG_PLAYER_H__

#include "IMediaPlayer.h"
#include "CayvanConfig.h"

/**
 * This class communicates through pipes with the ogg123
 * process for playing audio files.
 * A BIG thanks from AudioBox, from whose source I used
 * heavily when creating this class (http://audiobox.sourceforge.net)
 */

class OggPlayer : public IMediaPlayer
{
    public:
        OggPlayer(CayvanConfig* config);
        ~OggPlayer();

        /**
         * Initialize the player, including opening up sockets and pipes
         * to the ogg123 remote application.
         * This is done here because you can't return values from a
         * constructor
         */
        bool initialize();

        /**
         * File control functions
         */
        void play(QString filename);
        void stop();
        void pause();

    protected:
        /**
         * Wrapper for sending a command to the ogg123 process.
         * Be sure to end the command string with a new line.
         */
        void sendCommand(const char* cmd);

    private:
        int m_Ogg123PID;
        int m_SendFD;
        int m_RecvFD;
        CayvanConfig* m_Config;
};
#endif //__OGG_PLAYER_H__
9.3.2 Server Code

9.3.2.1 CayvanServer.rb

#!/usr/bin/env ruby

## CayvanServer
##
## This is the Cayvan web server. It connects to the database and takes care
## of
## all client requests.
require "postgres"
require 'xmlrpc/server'
class CayvanServer
  # Initialize our connection to the database
  def initialize
    end

  ## This function allows for function local connects and closes of the
  ## database connection
  def get_connection
    conn = nil
    begin
      conn = PGconn.connect("localhost", nil, nil, nil, "cayvan",
"cayvan")
      yield(conn)
    rescue PGError => e
      puts "Database Error: #{e.message}"
    rescue => ex
      puts "Error: #{ex.message}"
    ensure
      conn.close unless conn.nil?
    end
  end

  def get_departments
    ret = []
    get_connection { |conn|
      result = conn.exec("SELECT abbr FROM departments")
      result.each do |row|
        ret << row
      end
    }
    ret.flatten
  end

  def get_classes(dept)
    ret = []
    get_connection { |conn|
      result = conn.exec("SELECT num FROM classes " +
        "LEFT JOIN classes_departments ON class_id =

XXI
"LEFT JOIN departments ON departments.id = department_id " +
    "WHERE departments.abbr = '#{dept}' ")
result.each do |row|
    ret << row
end
}
ret.flatten
end

def get_audio_files(dept, class_num)
    get_files dept, class_num, 'a'
end

def get_video_files(dept, class_num)
    get_files dept, class_num, 'v'
end

def get_files(dept, class_num, type)
    ret = []
    get_connection { |conn|
        result = conn.exec("SELECT filename FROM files " +
        "LEFT JOIN departments ON departments.id =
        files.department_id " +
        "LEFT JOIN classes ON classes.id = files.class_id " +
        "WHERE departments.abbr = '#{dept}' AND classes.num =
        '#{class_num}' " +
        " AND file_type = '#{type}'")
        result.each do |row|
            ret << row
        end
    }
    ret.flatten
end

class UnknownUserException < RuntimeError
end

class UnauthorizedException < RuntimeError
end

end
9.3.2.2  Server.rb

#!/usr/bin/env ruby

## server.rb
##
## This file is the wrapper / redirector ruby file that defines the functions
## the server makes available to the outside world.

require 'webrick'
require 'xmlrpc/server'
require 'lib/CayvanServer'

server = CayvanServer.new

s = XMLRPC::WEBrickServlet.new(9000)
s.set_parser(XMLRPC::XMLParser::REXMLStreamParser.new)

s.add_handler("get_departments") do
  server.get_departments
end

s.add_handler("get_courses") do |dept|
  server.get_classes dept
end

s.add_handler("get_audio_files") do |dept, class_num|
  server.get_audio_files dept, class_num
end

s.add_handler("get_video_files") do |dept, class_num|
  server.get_video_files dept, class_num
end

s.set_default_handler do |name, *args|
  raise XMLRPC::FaultException.new(-99,
    "Method #{name} missing or wrong number of parameters!")
end

httpserver = WEBrick::HTTPServer.new(:Port => 9000)
httpserver.mount("/RPC2", s)
trap("INT") { httpserver.shutdown }
httpserver.start
9.3.3 Triton Module Source Code

9.3.3.1 Excerpts from pxafb.h

```c
struct pxafb_info {
    struct fb_info            fb;
    struct device    *dev;
    u_int        max_bpp;
    u_int        max_xres;
    u_int        max_yres;
    /*
     * These are the addresses we mapped
     * the framebuffer memory region to.
     */
    /* raw memory addresses */
    dma_addr_t    map_dma;    /* physical */
    u_char        *map_cpu;    /* virtual */
    u_int         map_size;
    /* addresses of pieces placed in raw buffer */
    u_char        *screen_cpu;    /* virtual address of frame buffer */
    dma_addr_t    screen_dma;    /* physical address of frame buffer */
    u16           *palette_cpu;    /* virtual address of palette memory */
    dma_addr_t    palette_dma;    /* physical address of palette memory */
    u_int         palette_size;
    /* DMA descriptors */
    struct pxafb_dma_descriptor * dmadesc_fblow_cpu;
    dma_addr_t    dmadesc_fblow_dma;
    struct pxafb_dma_descriptor * dmadesc_fbhigh_cpu;
    dma_addr_t    dmadesc_fbhigh_dma;
    struct pxafb_dma_descriptor * dmadesc_palette_cpu;
    dma_addr_t    dmadesc_palette_dma;
    dma_addr_t    fdadr0;
    dma_addr_t    fdadr1;
    u_int        lccr0;
    u_int        lccr3;
    u_int        cmap_inverse:1,
                  cmap_static:1,
                  unused:30;
    u_int        reg_lccr0;
    u_int        reg_lccr1;
    u_int        reg_lccr2;
    u_int        reg_lccr3;
    volatile u_char        state;
    volatile u_char        task_state;
    struct semaphore ctrlr_sem;
    wait_queue_head_t ctrlr_wait;
    struct work_struct task;

#if CONFIG_CPU_FREQ
    struct notifier_block freq_transition;
    struct notifier_block freq_policy;
#endif
};
```
9.3.3.2 Excerpts from karo-config.c

    // Starterkit II
    static int KARO_SYS_FEATURES_009[] __initdata = {
        KARO_HW_DETECT_BOARD,
        KARO_HW_I2C_EEPROM,
        KARO_HW_USB_HOST,
        KARO_HW_USB_SL811,
        KARO_HW_USB_DEVICE,
        KARO_HW_USER10,
        KARO_HW_CFCARD_IDE,
        KARO_HW_CFCARD_MEM,
        KARO_HW_I2C_PCF8574,
        KARO_HW_I2C_PXA,
        KARO_HW_PCIEA,
    #if defined(CONFIG_UCB1400) || defined(CONFIG_UCB1400_MODULE)
            KARO_HW_UCB1400,
    #endif
    #ifdef CONFIG_KARO_LCD
            KARO_HW_LCD,
    #endif
    };
9.3.3.3 Excerpts from karo.c

static struct pxafb_mach_info __initdata karo_lcd_data = {
#if defined(CONFIG_KARO_LCD_480_320)
    .pixclock = 221038,  // clock period in ps
    .bpp = 8,
    .xres = 480,
    .yres = 320,
    .left_margin = 10,  // LCD_BEGIN_OF_LINE_WAIT_COUNT,
    .right_margin = 10,  // LCD_END_OF_LINE_WAIT_COUNT,
    .hsync_len = 1,  // LCD_HORIZONTAL_SYNC_PULSE_WIDTH,
    .upper_margin = 0,  // LCD_BEGIN_FRAME_WAIT_COUNT,
    .lower_margin = 0,  // LCD_END_OF_FRAME_WAIT_COUNT,
    .vsync_len = 1,  // LCD_VERTICAL_SYNC_PULSE_WIDTH,
    .sync = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT,
    .lccr0 = LCCR0_CMS,
    .lccr3 = LCCR3_PCP | LCCR3_Acb(13),
    .cmap_greyscale = 1,
    .cmap_inverse = 1,
    //.cmap_static = 0,
    .pxafb_backlight_power = karo_lcd_backlight,
    .pxafb_lcd_power = karo_lcd_power,
#elif defined(CONFIG_KARO_LCD_320_240)
    .pixclock = 6400000,  // clock period in ps
    .bpp = 16,
    .xres = 320,
    .yres = 240,
    .hsync_len = 30,  // LCD_HORIZONTAL_SYNC_PULSE_WIDTH,
    .vsync_len = 3,  // LCD_VERTICAL_SYNC_PULSE_WIDTH,
    .left_margin = 20,  // LCD_BEGIN_OF_LINE_WAIT_COUNT,
    .upper_margin = 4,  // LCD_BEGIN_FRAME_WAIT_COUNT,
    .right_margin = 38,  // LCD_END_OF_LINE_WAIT_COUNT,
    .lower_margin = 15,  // LCD_END_OF_FRAME_WAIT_COUNT,
    .sync = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT,
    .lccr0 = LCCR0_CMS,
    .lccr3 = LCCR3_PCP | LCCR3_Acb(255),
    .cmap_greyscale = 0,
    .cmap_inverse = 1,
    //.cmap_static = 0,
    .pxafb_backlight_power = karo_lcd_backlight,
    .pxafb_lcd_power = karo_lcd_power,
#elif defined(CONFIG_KARO_LCD_1024_768)
    .pixclock = 60283,  // clock period in ps => 16.588 MHz
    .xres = 1024,
    .yres = 768,
    .hsync_len = 1,  // LCD_HORIZONTAL_SYNC_PULSE_WIDTH,
    .vsync_len = 1,  // LCD_VERTICAL_SYNC_PULSE_WIDTH,
    .left_margin = 10,  // LCD_BEGIN_OF_LINE_WAIT_COUNT,
    .upper_margin = 0,  // LCD_BEGIN_FRAME_WAIT_COUNT,
    .right_margin = 10,  // LCD_END_OF_LINE_WAIT_COUNT,
    .lower_margin = 0,  // LCD_END_OF_FRAME_WAIT_COUNT,
    .sync = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT,
    .lccr0 = LCCR0_CMS,
    .lccr3 = LCCR3_PCP | LCCR3_Acb(255),
    .cmap_greyscale = 1,
    .cmap_inverse = 1,
    .cmap_static = 0,
    .pxafb_backlight_power = karo_lcd_backlight,
    .pxafb_lcd_power = karo_lcd_power,*/
#endif

*/
#if defined(CONFIG_KARO_LCD_320_240)
    .pixclock = 500000,  // clock period in ps
    .bpp = 8,
    .xres = 320,
    .yres = 240,
    .hsync_len = 1,  // LCD_HORIZONTAL_SYNC_PULSE_WIDTH,
    .vsync_len = 1,  // LCD_VERTICAL_SYNC_PULSE_WIDTH,
    .left_margin = 10,  // LCD_BEGIN_OF_LINE_WAIT_COUNT,
    .upper_margin = 0,  // LCD_BEGIN_FRAME_WAIT_COUNT,
    .right_margin = 10,  // LCD_END_OF_LINE_WAIT_COUNT,
    .lower_margin = 0,  // LCD_END_OF_FRAME_WAIT_COUNT,
    .sync = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT,
    .lccr0 = LCCR0_CMS,
    .lccr3 = LCCR3_PCP | LCCR3_Acb(255),
    .cmap_greyscale = 1,
    .cmap_inverse = 1,
    .cmap_static = 0,
    .pxafb_backlight_power = karo_lcd_backlight,
    .pxafb_lcd_power = karo_lcd_power,*/
#endif

*/
#endif

*/
#if defined(CONFIG_KARO_LCD_1024_768)
    .pixclock = 60283,  // clock period in ps => 16.588 MHz */

XXVI
.bpp = 8,
.xres = 1024,
.yres = 768,
.hsync_len = 1, // LCD_HORIZONTAL_SYNC_PULSE_WIDTH,
.vsync_len = 1, // LCD_VERTICAL_SYNC_PULSE_WIDTH,
.left_margin = 128, // LCD_BEGIN_OF_LINE_WAIT_COUNT,
.upper_margin = 8, // LCD_BEGIN_FRAME_WAIT_COUNT,
.right_margin = 80, // LCD_END_OF_LINE_WAIT_COUNT,
.lower_margin = 18, // LCD_END_OF_FRAME_WAIT_COUNT,
.sync = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT,
.lccr0 = LCCR0_PAS | LCCR0_DMADel(0),
.lccr3 = LCCR3_Acb(0),
.cmap_inverse = 1,
// cmap_greyscale = 0,
.pxafb_backlight_power = karo_lcd_backlight,
.pxafb_lcd_power = karo_lcd_power,
#endif
#define XRES 800
#define YRES 600
#define nSEC_TO_PIX(ns,clk) (((ns) * ((clk) / 1000U) + 500000) / 1000000U)
#define uSEC_TO_LINE(us,clk) (((us) * ((clk) / 1000U) + 500) / 1000U)
#define LEFT_MARGIN nSEC_TO_PIX(3770, PCLK)
#define RIGHT_MARGIN nSEC_TO_PIX( 940, PCLK)
#define HSYNC_LEN nSEC_TO_PIX(1890, PCLK)
#define LINE_CLK (PCLK / (XRES + LEFT_MARGIN + RIGHT_MARGIN + HSYNC_LEN))
#define UPPER_MARGIN uSEC_TO_LINE(1020, LINE_CLK)
#define LOWER_MARGIN uSEC_TO_LINE( 350, LINE_CLK)
#define VSYNC_LEN uSEC_TO_LINE( 60, LINE_CLK)
#define BASE_CLK 13000000UL
#define LCLK (8 * BASE_CLK)
#define CLK_DIV 1
#define PIXCLOCK (34666666UL) // designated pixel clock in Hz
// #define PIXCLOCK (52000000UL) // designated pixel clock in Hz
#define PCLK_PERIOD (1000000000UL / (PIXCLOCK / 1000))
#define PCD (LCLK / CLK_DIV / PIXCLOCK)
#define PCLK (LCLK / (CLK_DIV * ((PCD) + 1)))
// use 640x480 as standard
// line width: 31.77µs == 807 pixel
// active width: 25.17µs == 640 pixel
// hsync width: 3.77µs == 95 pixel => 55
// left margin: 1.89µs == 48 pixel => 88
// right margin: 0.94µs == 23 pixel
// frame width: 16.68ms == 525 lines
// active width: 15.25ms == 480 lines
.pixclock = PCLK_PERIOD, /* clock period in ps => 25.4 MHz */
.bpp = 16,
.xres = XRES,
.yres = YRES,
.left_margin = LEFT_MARGIN, /* LCD_BEGIN_OF_LINE_WAIT_COUNT,
.right_margin = RIGHT_MARGIN, /* LCD_END_OF_LINE_WAIT_COUNT,
.hsync_len = HSYNC_LEN, /* LCD_HORIZONTAL_SYNC_PULSE_WIDTH,
.upper_margin = UPPER_MARGIN, /* LCD_BEGIN_FRAME_WAIT_COUNT,
.lower_margin = LOWER_MARGIN, /* LCD_END_OF_FRAME_WAIT_COUNT,
.vsync_len = VSYNC_LEN, /* LCD_VERTICAL_SYNC_PULSE_WIDTH,
.sync = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT, /* HSYNC,
VSYNC polarity
.lccr0 = LCCR0_PAS | LCCR0_DMAdel(0),
.lccr3 = LCCR3_Acb(0) | LCCR3_PdFor(0),
#ifdef CONFIG_PXA27x
.lccr4 = LCCR4_PALFOR_18 | ((CLK_DIV == 1) ? LCCR4_PCDIV : 0),
#endif
.cmap_inverse = 1,
.cmap_greyscale = 0,
.pxafb_backlight_power = karo_lcd_backlight,
.pxafb_lcd_power = karo_lcd_power,
#elif defined(CONFIG_KARO_LCD_640_480)
#define XRES 640
#define YRES 480
#define nSEC_TO_PIX(ns, clk) (((ns) * ((clk) / 1000U) + 500000) / 1000000U)
#define uSEC_TO_LINE(us, clk) (((us) * ((clk) / 1000U) + 500) / 1000U)
#define LEFT_MARGIN nSEC_TO_PIX(3770, PCLK)
#define RIGHT_MARGIN nSEC_TO_PIX(940, PCLK)
#define HSYNC_LEN nSEC_TO_PIX(1890, PCLK)
#define LINE_CLK (PCLK / (XRES + LEFT_MARGIN + RIGHT_MARGIN + HSYNC_LEN))
#define UPPER_MARGIN uSEC_TO_LINE(1020, LINE_CLK)
#define LOWER_MARGIN uSEC_TO_LINE(350, LINE_CLK)
#define VSYNC_LEN uSEC_TO_LINE(60, LINE_CLK)
#define BASE_CLK 13000000UL
#define LCLK (8 * BASE_CLK)
#define CLK_DIV 1
#define PIXCLOCK (27781967U) /* designated pixel clock in Hz
#define PCLK_PERIOD (1000000000UL / (PIXCLOCK / 1000))
#define PCD (LCLK / CLK_DIV / PIXCLOCK)
#define PCLK (LCLK / (CLK_DIV * ((PCD) + 1)))
// use 640x480 as standard
// line width: 31.77µs == 807 pixel
// active width: 25.17µs == 640 pixel
// hsync width: 3.77µs == 95 pixel => 55
// left margin: 1.89µs == 48 pixel => 88
// right margin: 0.94µs == 23 pixel
// frame width: 16.68ms == 525 lines
// active width: 15.25ms == 480 lines
// upper margin: 1.02ms == 32 lines
// lower margin: 0.35ms == 11 lines
// vsync len: 0.06ms == 2 lines
.pixclock = PCLK_PERIOD, /* clock period in ps => 25.4 MHz */
.bpp = 8,
.xres = XRES,
.yres = YRES,
.left_margin = LEFT_MARGIN, /* LCD_BEGIN_OF_LINE_WAIT_COUNT,
.right_margin = RIGHT_MARGIN, /* LCD_END_OF_LINE_WAIT_COUNT,
.hsync_len = HSYNC_LEN, /* LCD_HORIZONTAL_SYNC_PULSE_WIDTH,
.upper_margin = UPPER_MARGIN, /* LCD_BEGIN_FRAME_WAIT_COUNT,
.lower_margin = LOWER_MARGIN, // LCD_END_OF_FRAME_WAIT_COUNT,
.vsync_len = VSYNC_LEN, // LCD_VERTICAL_SYNC_PULSE_WIDTH,
.sync = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT, // HSYNC,
VSYNC polarity
\.lccr0 = LCCR0_PAS | LCCR0_DMADe1(0),
\.lccr3 = LCCR3_Acb(0) | LCCR3_PdFor(3),
#ifdef CONFIG_PXA27x
\.lccr4 = LCCR4_PALFOR_18 | ((CLK_DIV == 1) ? LCCR4_PCMDIV : 0),
#endif
\.cmap_inverse = 1,
\.cmap_greyscale = 0,
.pxafb_backlight_power = karo_lcd_backlight,
.pxafb_lcd_power = karo_lcd_power,
#else defined(CONFIG_KARO_LCD_800_480)
#define XRES 800
#define YRES 480
#define nSEC_TO_PIX(ns, clk) (((ns) * ((clk) / 1000U) + 500000) / 1000000U)
#define uSEC_TO_LINE(us, clk) (((us) * ((clk) / 1000U) + 50) / 1000U)
#define LEFT_MARGIN nSEC_TO_PIX(2595, PCLK) // s/b: 1905
#define RIGHT_MARGIN nSEC_TO_PIX(1905, PCLK)
#define HSYNC_LEN nSEC_TO_PIX(1850, PCLK) // s/b: 2540
#define LINE_CLK (PCLK / (XRES + LEFT_MARGIN + RIGHT_MARGIN + HSYNC_LEN))
#define UPER_MARGIN uSEC_TO_LINE( 889, LINE_CLK)
#define LOWER_MARGIN uSEC_TO_LINE( 444, LINE_CLK)
#define VSYNC_LEN uSEC_TO_LINE( 95, LINE_CLK)
#define BASE_CLK 13000000UL
#define LCLK (8 * BASE_CLK)
#define CLK_DIV 1
//#define PIXCLOCK (30464500U) // designated pixel clock in Hz
#define PIXCLOCK (31500000U) // designated pixel clock in Hz
#define PCLK_PERIOD (1000000000UL / (PIXCLOCK / 1000))
#define PCD (LCLK / CLK_DIV / PIXCLOCK)
#define PCLK (LCLK / (CLK_DIV * ((PCD) + 1)))
\.pixclock = PCLK_PERIOD, /* clock period in ps => 25.4 MHz */
\.bpp = 8,
\.xres = XRES,
\.yres = YRES,
\.left_margin = LEFT_MARGIN, // LCD_BEGIN_OF_LINE_WAIT_COUNT[1..256],
\.right_margin = RIGHT_MARGIN, // LCD_END_OF_LINE_WAIT_COUNT [1..256],
\.hsync_len = HSYNC_LEN, // LCD_HORIZONTAL_SYNC_PULSE_WIDTH [1..64],
\.upper_margin = UPER_MARGIN, // LCD_BEGIN_FRAME_WAIT_COUNT [0..255],
\.lower_margin = LOWER_MARGIN, // LCD_END_OF_FRAME_WAIT_COUNT [0..255],
\.vsync_len = VSYNC_LEN, // LCD_VERTICAL_SYNC_PULSE_WIDTH [1..64],
\.sync = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT, // HSYNC,
VSYNC polarity
\.lccr0 = LCCR0_PAS | LCCR0_DMADe1(0),
\.lccr3 = LCCR3_Acb(0) | LCCR3_PdFor(3),
#endif CONFIG_PXA27x
\.lccr4 = LCCR4_PALFOR_18 | ((CLK_DIV == 1) ? LCCR4_PCMDIV : 0),
#else
// use 640x480 as standard
\.pixclock = 20000, //80193, /* clock period in ps => 24.88 MHz */
\.bpp = 8,
\.xres = 640,
\.yres = 480,
\.left_margin = 64, // LCD_BEGIN_OF_LINE_WAIT_COUNT,
\.right_margin = 72, // LCD_END_OF_LINE_WAIT_COUNT,
.hsync_len = 16, // LCD_HORIZONTAL_SYNC_PULSE_WIDTH,
.upper_margin = 31, // LCD_BEGIN_FRAME_WAIT_COUNT,
.lower_margin = 31, // LCD_END_OF_FRAME_WAIT_COUNT,
.vsync_len = 6, // LCD_VERTICAL_SYNC_PULSE_WIDTH,
.sync = FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT, // HSYNC,

VSYNC polarity
.lccr0 = LCCR0_PAS | LCCR0_DMAEn(0),
.lccr3 = LCCR3_Acb(0) | LCCR3_PdFor(3),
#ifdef CONFIG_PXA27x
.lccr4 = LCCR4_PALFOR_18,
#endif
.cmap_inverse = 1,
.cmap_greyscale = 0,
.pxafb_backlight_power = karo_lcd_backlight,
.pxafb_lcd_power = karo_lcd_power,
#endif
);
### 9.4 Budget Summary

<table>
<thead>
<tr>
<th>Date</th>
<th>Item Description</th>
<th>Vendor</th>
<th>Cost</th>
<th>Out of Pocket?</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-Oct-05</td>
<td>No purchases made</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-Nov-05</td>
<td>No purchases made</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18-Jan</td>
<td>Gift from Smiths Industries of $4,600.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31-Jan</td>
<td>Triton Starter Kit 2 with Triton-ETN module</td>
<td>Direct Insight</td>
<td>2278</td>
<td>N</td>
</tr>
<tr>
<td>31-Jan</td>
<td>Triton-ETN Fast Ethernet PXA255 miniature SBC</td>
<td>Direct Insight</td>
<td>378.1</td>
<td>N</td>
</tr>
<tr>
<td>31-Jan</td>
<td>LP-35 3.5inch LCDs (2)</td>
<td>Littlechips</td>
<td>339.98</td>
<td>N</td>
</tr>
<tr>
<td>23-Feb</td>
<td>DPAC Wireless Ethernet Eval kit</td>
<td>DPAC</td>
<td>249</td>
<td>N</td>
</tr>
<tr>
<td>23-Feb</td>
<td>DPAC Wireless Ethernet Bridge Module</td>
<td>DPAC</td>
<td>220</td>
<td>N</td>
</tr>
<tr>
<td>23-Feb</td>
<td>Connectors (IDE, 44 pin LCD connector, 144 pin DIMM connector)</td>
<td>Digikey</td>
<td>53.05</td>
<td>N</td>
</tr>
<tr>
<td>23-Feb</td>
<td>UCB1400BE-T Audio codec chip</td>
<td>Digikey</td>
<td>21.6</td>
<td>N</td>
</tr>
<tr>
<td>28-Feb</td>
<td>Bluetooth Development Kit (includes 10 modules)</td>
<td>A7engineering</td>
<td>209.1</td>
<td>N</td>
</tr>
<tr>
<td>29-Mar</td>
<td>Digikey order (DPAC connectors and antennas)</td>
<td>Digikey</td>
<td>108.41</td>
<td>N</td>
</tr>
<tr>
<td>4-Apr</td>
<td>Digikey order (Oscillator, audio jack, more codecs)</td>
<td>Digikey</td>
<td>59</td>
<td>N</td>
</tr>
<tr>
<td>20-Apr</td>
<td>12V battery</td>
<td><a href="http://www.DefenseDevices.com">www.DefenseDevices.com</a></td>
<td>55.96</td>
<td>N</td>
</tr>
<tr>
<td>1-May</td>
<td>24,576 Crystal</td>
<td>Digikey</td>
<td>44.7</td>
<td>N</td>
</tr>
<tr>
<td>1-May</td>
<td>Components for board Population</td>
<td>Digikey</td>
<td>71.8</td>
<td>N</td>
</tr>
<tr>
<td>1-May</td>
<td>Shipping PCB from</td>
<td>PCB Express</td>
<td>25</td>
<td>N</td>
</tr>
<tr>
<td>3-May</td>
<td>Additional board from PCB express</td>
<td>PCB Express</td>
<td>496</td>
<td>N</td>
</tr>
</tbody>
</table>

**Budget:** $4,900.00  
**Grand Total:** $4,609.69  
**Remaining:** $290.31
### 9.5 Bill of Materials

<table>
<thead>
<tr>
<th>Part</th>
<th>Value</th>
<th>Quantity</th>
<th>Package</th>
<th>Digi-Key Part Number</th>
<th>Need for 1 board</th>
<th>Price</th>
<th>Need for 100 boards</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resistor</td>
<td>0</td>
<td>10</td>
<td>R1206W</td>
<td>311-0.0ERCT-ND</td>
<td>4</td>
<td>0.76</td>
<td>400</td>
<td>9.7</td>
</tr>
<tr>
<td>Resistor</td>
<td>10</td>
<td>10</td>
<td>R1206W</td>
<td>311-10.0FRCT-ND</td>
<td>4</td>
<td>0.98</td>
<td>400</td>
<td>10.88</td>
</tr>
<tr>
<td>Resistor</td>
<td>22</td>
<td>10</td>
<td>R1206W</td>
<td>311-22.0FRCT-ND</td>
<td>4</td>
<td>0.98</td>
<td>400</td>
<td>10.88</td>
</tr>
<tr>
<td>Resistor</td>
<td>100</td>
<td>10</td>
<td>R1206W</td>
<td>311-100CT-ND</td>
<td>4</td>
<td>0.76</td>
<td>400</td>
<td>9.7</td>
</tr>
<tr>
<td>Resistor</td>
<td>348</td>
<td>10</td>
<td>R1206W</td>
<td>311-348FCT-ND</td>
<td>4</td>
<td>0.38</td>
<td>400</td>
<td>10.88</td>
</tr>
<tr>
<td>Resistor</td>
<td>1.87K</td>
<td>10</td>
<td>R1206W</td>
<td>311-1.87KFCT-ND</td>
<td>4</td>
<td>0.38</td>
<td>400</td>
<td>10.88</td>
</tr>
<tr>
<td>Capacitor</td>
<td>1uF</td>
<td>100</td>
<td>C1206K</td>
<td>PCC1870CT-ND</td>
<td>50</td>
<td>8.4</td>
<td>5000</td>
<td>360.75</td>
</tr>
<tr>
<td>Capacitor</td>
<td>1uF</td>
<td>10</td>
<td>CD050</td>
<td>311-1142-1-ND</td>
<td>4</td>
<td>0.82</td>
<td>400</td>
<td>19.88</td>
</tr>
<tr>
<td>Capacitor</td>
<td>100uF</td>
<td>5</td>
<td>CT3216</td>
<td>478-3345-1-ND</td>
<td>4</td>
<td>12.4</td>
<td>400</td>
<td>942.4</td>
</tr>
<tr>
<td>Capacitor</td>
<td>150pF</td>
<td>10</td>
<td>C1206K</td>
<td>478-1420-1-ND</td>
<td>4</td>
<td>5.17</td>
<td>400</td>
<td>122.2</td>
</tr>
<tr>
<td>Capacitor</td>
<td>22uF</td>
<td>40</td>
<td>CT3216</td>
<td>495-2183-1-ND</td>
<td>34</td>
<td>8.16</td>
<td>3400</td>
<td>423.94</td>
</tr>
<tr>
<td>Capacitor</td>
<td>33pF</td>
<td>10</td>
<td>C1206K</td>
<td>311-1156-1-ND</td>
<td>4</td>
<td>1.35</td>
<td>400</td>
<td>32.88</td>
</tr>
<tr>
<td>Capacitor</td>
<td>47uF</td>
<td>4</td>
<td>CT3216</td>
<td>478-3079-1-ND</td>
<td>2</td>
<td>1.6</td>
<td>200</td>
<td>136.8</td>
</tr>
<tr>
<td>50 Pin Connector (2.54mm)</td>
<td>50 Pins</td>
<td>3</td>
<td>Through Hole</td>
<td>929975-01-25-Nd</td>
<td>2</td>
<td>7.84</td>
<td>200</td>
<td>562.94</td>
</tr>
<tr>
<td>12 Volt DC Jack</td>
<td>12 Volts</td>
<td>3</td>
<td>PJ-102AH</td>
<td></td>
<td>2</td>
<td>1.88</td>
<td>200</td>
<td>56</td>
</tr>
<tr>
<td>Reset Monitor (Active Low)</td>
<td>3.3V</td>
<td>3</td>
<td>SOT-143-4</td>
<td>TC1270SERRCT-ND</td>
<td>2</td>
<td>1.28</td>
<td>200</td>
<td>103.5</td>
</tr>
<tr>
<td>Reset Monitor (Active High)</td>
<td>3.3V</td>
<td>3</td>
<td>SOT-143-4</td>
<td>TCMB12TERC OR TC1270TERCTR??</td>
<td>2</td>
<td>1.28</td>
<td>200</td>
<td>192</td>
</tr>
<tr>
<td>Power Supply Chips</td>
<td>12V</td>
<td>2</td>
<td>PTH08080W2AH-ND??</td>
<td></td>
<td>4</td>
<td>4.0</td>
<td>400</td>
<td>2786.4</td>
</tr>
<tr>
<td>Ferrite Beads</td>
<td>3.3V</td>
<td>10</td>
<td>005</td>
<td>490-1030-1-ND</td>
<td>6</td>
<td>0.74</td>
<td>600</td>
<td>35.64</td>
</tr>
<tr>
<td>LED</td>
<td>2.1V</td>
<td>5</td>
<td>CHIPLED_1206</td>
<td>P111141CT-ND</td>
<td>4</td>
<td>1.32</td>
<td>400</td>
<td>111.24</td>
</tr>
<tr>
<td>DPAC Screws</td>
<td>100</td>
<td>1</td>
<td>H781-ND</td>
<td></td>
<td>6</td>
<td>0.12</td>
<td>600</td>
<td>12</td>
</tr>
<tr>
<td>LCD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.80</td>
<td>100</td>
<td>12000</td>
</tr>
<tr>
<td>Triton ETN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>371.36</td>
<td>100</td>
<td>27862</td>
</tr>
<tr>
<td>DPAC Module</td>
<td>1.590-1012-ND</td>
<td>1</td>
<td></td>
<td>129.42</td>
<td>100</td>
<td>9424.97</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EB100 Module</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>40</td>
<td>100</td>
<td>2000</td>
</tr>
<tr>
<td>Rechargeable Battery</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>55.95</td>
<td>100</td>
<td>1398.75</td>
</tr>
</tbody>
</table>

**Parts Total**          **853.97**  **586.3721**

- Board Fabrication (PCB Express): 274  1625
- Board Assembly: 43.04
- Case: 10000

**Total**                  **1287.97**  **655.6621**
9.6 Complete Project Schedule

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Milestone</th>
<th>External Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Budget</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obtain external funding.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submit Proposal/Budget to Smiths</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Budget approval (From Smiths includes acquisition of funds)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contact PCB Express for donations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Produce detailed budget for handheld and daughterboard</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Produce detailed tny Budget</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Produce detailed SBC Budget</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Produce detailed daughterboard Budget</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Produce production Budget</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Periodically revise Budget</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single Board Computer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feasibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine interface possibilities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine tools required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assembly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procure digkey parts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procure obsolete parts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assemble parts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software (EE tasks)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile and install Operating System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Find necessary device drivers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examine Linux Kernel for native device support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search for premade open source drivers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research implementation of premade drivers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design necessary device drivers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research programming of device drivers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search for device driver examples</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search for similar device drivers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Project schedule (prelim)
Date: Thu 5/11/05

XXXIII
### Task List

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>Program device drivers</td>
</tr>
<tr>
<td>31</td>
<td>Debug any OS or driver related problems</td>
</tr>
<tr>
<td>32</td>
<td>Debug OS related problems</td>
</tr>
<tr>
<td>33</td>
<td>Debug driver installation problems</td>
</tr>
<tr>
<td>34</td>
<td>Debug driver compiled problems</td>
</tr>
<tr>
<td>35</td>
<td>Debug driver functionality problems</td>
</tr>
</tbody>
</table>

#### Testing and Debugging
- Identify requirements to be tested
- Identify additional tests
- Write test checklist
- Identify/procure test equipment needed
- Run driver tests
- Run handheld tests
- Document test results
- Debug problems
- Re-run tests

#### Daughter Card

#### Research necessary components
- Research Bluetooth hardware and software
  - Research ICs
- Research development kits
- Research Bluetooth software
- Analyze options
- Select solution

#### Research 802.11 hardware and software
- Research ICs
- Research development kits
- Research Bluetooth software
- Analyze options
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>Select solution</td>
</tr>
<tr>
<td>60</td>
<td>Research power requirements and implementation of daughter</td>
</tr>
<tr>
<td>61</td>
<td>Research power requirements</td>
</tr>
<tr>
<td>62</td>
<td>Research possible interfaces</td>
</tr>
<tr>
<td>63</td>
<td>Choose best interface</td>
</tr>
<tr>
<td>64</td>
<td><strong>Research board fabrication</strong></td>
</tr>
<tr>
<td>65</td>
<td>Research trace length/width requirements</td>
</tr>
<tr>
<td>66</td>
<td>Research number of required board layers</td>
</tr>
<tr>
<td>67</td>
<td>Research local board fabricators</td>
</tr>
<tr>
<td>68</td>
<td><strong>Schematic Design</strong></td>
</tr>
<tr>
<td>69</td>
<td><strong>CAD software</strong></td>
</tr>
<tr>
<td>70</td>
<td>Analyze different CAD options</td>
</tr>
<tr>
<td>71</td>
<td>Test different options</td>
</tr>
<tr>
<td>72</td>
<td>Choose best option</td>
</tr>
<tr>
<td>73</td>
<td>Software tutorials</td>
</tr>
<tr>
<td>74</td>
<td>Practice PCB design</td>
</tr>
<tr>
<td>75</td>
<td>Practice schematic design</td>
</tr>
<tr>
<td>76</td>
<td>Design daughter card</td>
</tr>
<tr>
<td>77</td>
<td>Layout daughter card on PCB</td>
</tr>
<tr>
<td>78</td>
<td>Identify relevant requirements</td>
</tr>
<tr>
<td>79</td>
<td>Obtain simulation software</td>
</tr>
<tr>
<td>80</td>
<td>Run necessary simulations</td>
</tr>
<tr>
<td>81</td>
<td>Have schematic reviewed by Corin</td>
</tr>
<tr>
<td>82</td>
<td>Send board out for fabrication</td>
</tr>
<tr>
<td>83</td>
<td><strong>Test and debug board</strong></td>
</tr>
<tr>
<td>84</td>
<td>Identify requirements to be tested</td>
</tr>
<tr>
<td>85</td>
<td>Identify additional tests</td>
</tr>
<tr>
<td>86</td>
<td>Write test checklist</td>
</tr>
<tr>
<td>87</td>
<td>Identify/procure test equipment needed</td>
</tr>
</tbody>
</table>

Project schedule (prelim)
Date: Thu 5/11/06

XXXV
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>Run tests</td>
</tr>
<tr>
<td>89</td>
<td>Run IC test</td>
</tr>
<tr>
<td>90</td>
<td>Perform continuity test</td>
</tr>
<tr>
<td>91</td>
<td>Document test results</td>
</tr>
<tr>
<td>92</td>
<td><strong>Debug problems</strong></td>
</tr>
<tr>
<td>93</td>
<td>Identify problems</td>
</tr>
<tr>
<td>94</td>
<td>Design test to isolate problems</td>
</tr>
<tr>
<td>95</td>
<td>Test isolated problems</td>
</tr>
<tr>
<td>96</td>
<td>Repair problems</td>
</tr>
<tr>
<td>97</td>
<td>Document repair</td>
</tr>
<tr>
<td>98</td>
<td>Update schematic</td>
</tr>
<tr>
<td>99</td>
<td>Document test results</td>
</tr>
<tr>
<td>100</td>
<td>Rerun tests</td>
</tr>
<tr>
<td>101</td>
<td><strong>Design daugthercard to handheld interface</strong></td>
</tr>
<tr>
<td>102</td>
<td>Identify interface options</td>
</tr>
<tr>
<td>103</td>
<td>Analyze advantages of each option</td>
</tr>
<tr>
<td>104</td>
<td>Choose best option</td>
</tr>
<tr>
<td>105</td>
<td>Design interface based on best option</td>
</tr>
<tr>
<td>106</td>
<td>Test interface</td>
</tr>
<tr>
<td>107</td>
<td>Implement interface</td>
</tr>
<tr>
<td>108</td>
<td><strong>Test and debug combined system</strong></td>
</tr>
<tr>
<td>109</td>
<td>Define tests</td>
</tr>
<tr>
<td>110</td>
<td>Run tests</td>
</tr>
<tr>
<td>111</td>
<td>Document results</td>
</tr>
<tr>
<td>112</td>
<td>isolate problems</td>
</tr>
<tr>
<td>113</td>
<td>Test isolated problems</td>
</tr>
<tr>
<td>114</td>
<td>Repair problems</td>
</tr>
<tr>
<td>115</td>
<td>Document results</td>
</tr>
<tr>
<td>116</td>
<td>Rerun tests</td>
</tr>
</tbody>
</table>

**Project: schedule (prelim)**
Date: Thu 5/11/06

XXXVI
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>117</td>
<td>Update schematic</td>
</tr>
<tr>
<td>118</td>
<td>SYSTEM FULLY FUNCTIONAL</td>
</tr>
<tr>
<td>119</td>
<td>Software (CS tasks)</td>
</tr>
<tr>
<td>120</td>
<td>Media Cataloging system</td>
</tr>
<tr>
<td>121</td>
<td>Design custom interface software for thin client/server interface</td>
</tr>
<tr>
<td>122</td>
<td>Debug and test custom software</td>
</tr>
<tr>
<td>123</td>
<td>Implement custom software</td>
</tr>
<tr>
<td>124</td>
<td>Handheld Software</td>
</tr>
<tr>
<td>125</td>
<td>Design and compile custom software</td>
</tr>
<tr>
<td>126</td>
<td>SBC media selection software</td>
</tr>
<tr>
<td>127</td>
<td>Thin client software</td>
</tr>
<tr>
<td>128</td>
<td>Upload custom software</td>
</tr>
<tr>
<td>129</td>
<td>Debug custom software problems</td>
</tr>
<tr>
<td>130</td>
<td>Media cataloguing system</td>
</tr>
<tr>
<td>131</td>
<td>EE Task</td>
</tr>
<tr>
<td>132</td>
<td>Catalogue audio and video</td>
</tr>
<tr>
<td>133</td>
<td>Procure thin client terminal</td>
</tr>
<tr>
<td>134</td>
<td>Procure server computer</td>
</tr>
<tr>
<td>135</td>
<td>Obtain bluetooth cards</td>
</tr>
<tr>
<td>136</td>
<td>Select vendor</td>
</tr>
<tr>
<td>137</td>
<td>Decide on card</td>
</tr>
<tr>
<td>138</td>
<td>Procure cards</td>
</tr>
<tr>
<td>139</td>
<td>Prepare PPFS</td>
</tr>
<tr>
<td>140</td>
<td>Write detailed table of contents</td>
</tr>
<tr>
<td>141</td>
<td>Write PPFS</td>
</tr>
<tr>
<td>142</td>
<td>Write Introduction/conclusion</td>
</tr>
<tr>
<td>143</td>
<td>Write Educational and Design Objective</td>
</tr>
<tr>
<td>144</td>
<td>Write Design Norms</td>
</tr>
<tr>
<td>145</td>
<td>Write Project task list and time allocation</td>
</tr>
</tbody>
</table>

Project: schedule(prelim)
Date: Thu 5/11/06
XXXVIII
9.7 Pictures of final handheld PCB layout

Figure 19 - Top layer of final PCB

Figure 20 - Layer 2 of final PCB
Figure 21 - Layer 3 of final PCB

Figure 22 - Bottom layer of final PCB
9.8 Final circuit layout of handheld device