A Low Power Touch Screen Document Viewer

By

Chris Perfetto

California Polytechnic University San Luis Obispo

CPE 461 & 462 - Senior Project I & II

June 2010

Advisor

Dr. John Y. Oliver

Table of Contents

List of Tables & Figures

Abstract

This document will covers the requirements, design, implementation and testing of a low power touch screen document viewer to serve as a replacement bulletin board. It was Implemented using a BeagleBoard development board, a 4.3” touchscreen LCD and a SD card with a Ubuntu Linux OS installed. The testing of this device is primarily testing the power consumption. The device is not yet complete and will require further development for it to completely satisfy the requirements.

I. Introduction

The goal of this project is to create a device that could be used as an electronic replacement for a small bulletin board, replacing the need to print a pamphlet or document in order to post it. The device will need to display documents clearly and have a simple yet intuitive interface as to make as easy if not easier to navigate different postings as would be with a normal bulletin board. It is also a goal of this project to design the device such that it consumes little enough power so that at a later date it would be possible to set up a small battery / solar panel system to power it, removing its reliance on both paper / ink and power resource; this, however, is not within the scope of this project.

For the scope of this senior project I initially projected that I would have a graphical display, the ability to view documents, and maybe a touchscreen interface working by the end of the project. The planned implementation schedule to achieve this can be seen in Figure 8 the appendix.

II. Background

The applications for this device are quite broad; it is of course useful for displaying general information normally found on a bulletin board but a microcontroller allows for many more possibilities. Simply adding wireless capability would allow the device to be updated remotely (and hopefully securely) without having to go out and make the changes to the physical device. Several of these devices could be updated simultaneously given wireless capability. It could be used to display teacher's schedules, notices, and other information you might find on the bulletin boards outside most professor's offices. It could be used in central locations where most people go to find just general information. But it shouldn't be thought of as just another computer terminal where you can go to access this information. It is different from this in that the information is already there at your fingertips without having to try and find it online; the important information has found and put there for all to see. In essence it combines the simplicity of a bulletin board with the ease of posting, updating, and maintaining online material. Not to mention, if used on a large scale, it could help to reduce our reliance on paper and ink resources.

III. Requirements

Power

The device should be able to run for a full day or more without recharge. This is assuming the display turns itself off after a specified period of time to conserver power. In addition, any low power modes in both hardware and software should be taken advantage of in order to conserver power.

Size

The device needs a display large enough to display documents typically found on a bulletin board without causing unnecessary eye strain to the viewer. It should, however, not be so large that wall mounting would be made difficult. A letter sized (8.5 x 11) screen, or something close to this should meet both requirements.

Interface

A viewer needs to be able to interact with it to cycle through the documents being displayed, change options on how the documents are displayed, and update the displayed information.

Storage

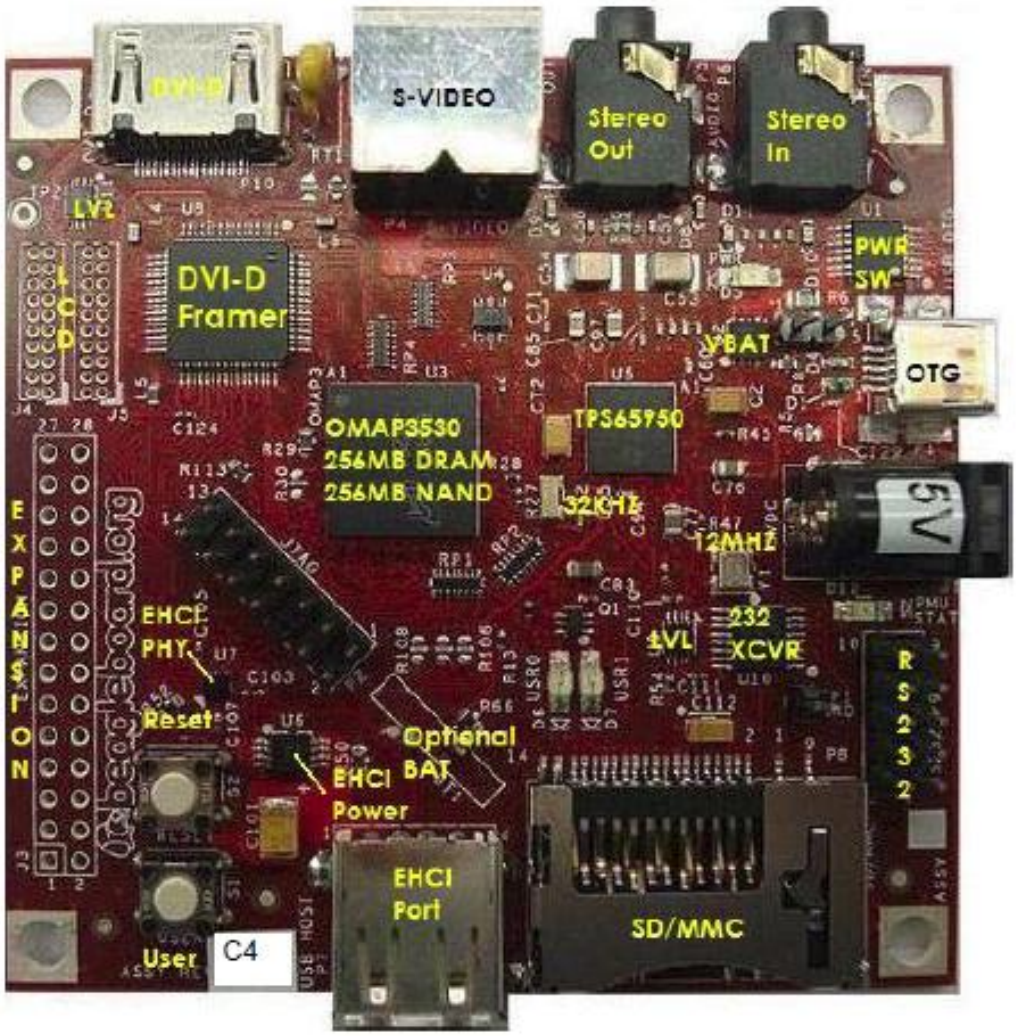
The storage unit should be large enough to contain an operating system as well as any documents to be displayed by the device. An SD card storage solution would best satisfy this and could allow for further expansion.

Cost

The cost of this project should be less than $500, which is the budget for this project.

IV. Design

BeagleBoard



[1]Figure 1: BeagleBoard Rev. C4

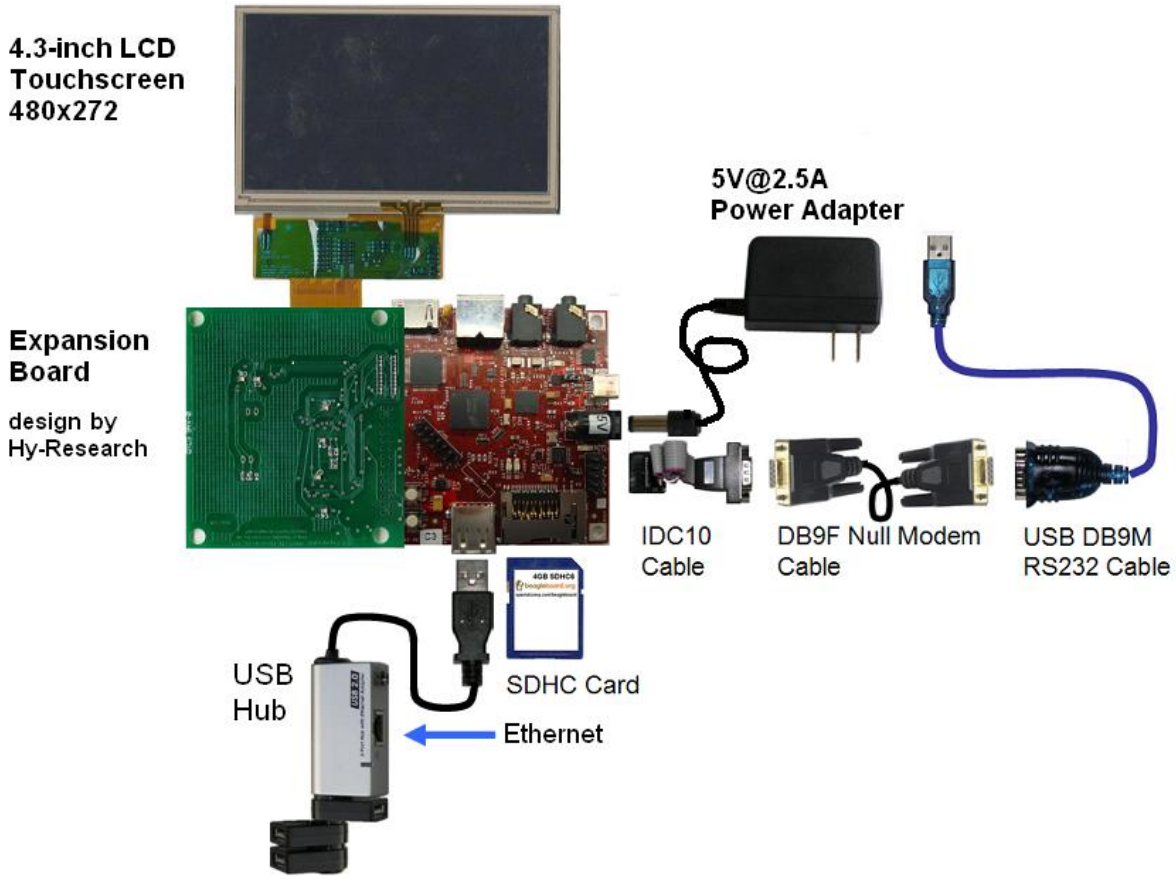
The development board used for this project was the BeagleBaord which is powered by a Texas Instruments OMAP3530 MCU containing an ARM Cortex-A8 CPU. The Cortex-A8 processor is a powerful, low power applications processor already in use in several common embeds systems including the Nokia

[1] Figure taken from BeagleBoard Reference Manual (Pg. 42)

N900 phone and the 3rd gen IPod Touch [1]. It was chosen for its low power consumption as well as the ability to run an operating system. The devices of interest labeled in Figure 1 above are (1) the SD/MMC slot (and SD card) allow booting of an operating system much larger than could be contained on the 256MB NAND flash. (2) The “LCD” and “Expansion” slots (required headers to be soldered) allowed interfacing with the LCD expansion board seen in Figure 2 below; this includes the 27 pins (8 pins per color, pixel clock pin, horizontal sync pin, vertical sync pin, and a pixel data enable pin) to control the LCD display and 4 pins to interface the touchscreen using SPI. (3) The OMAP3530 MCU as well as the NAND flash and DRAM can be seen in the center of the board, the NAND flash and DRAM actually being mounted on top of the OMAP3530 MCU. It is worth mentioning since it is the device controlling everything else on board. (4) The RS232 header seen in the bottom left was used to connect to the board over a serial port, much work was done through this interface up until the LCD screen was working, and even after, since the LCD screen was a bit small. (5) The EHCI port, or USB port, seen below was used to connect a USB Hub / Ethernet device allowing the installation of additional software for the operating system in use. This information was taken from the BeagleBoard Reference Manual Rev C4 [6].

LCD Displays

In order to get the LCD display working some research into LCD interfaces and timings was conducted. I will give a brief overview of the functionality of a LCD display including what most signals do and the timings associated with these signals. Much of this information was taken from a presentation on NXP’s website [2].



[2]Figure 2: Additional Hardware Used in the Project

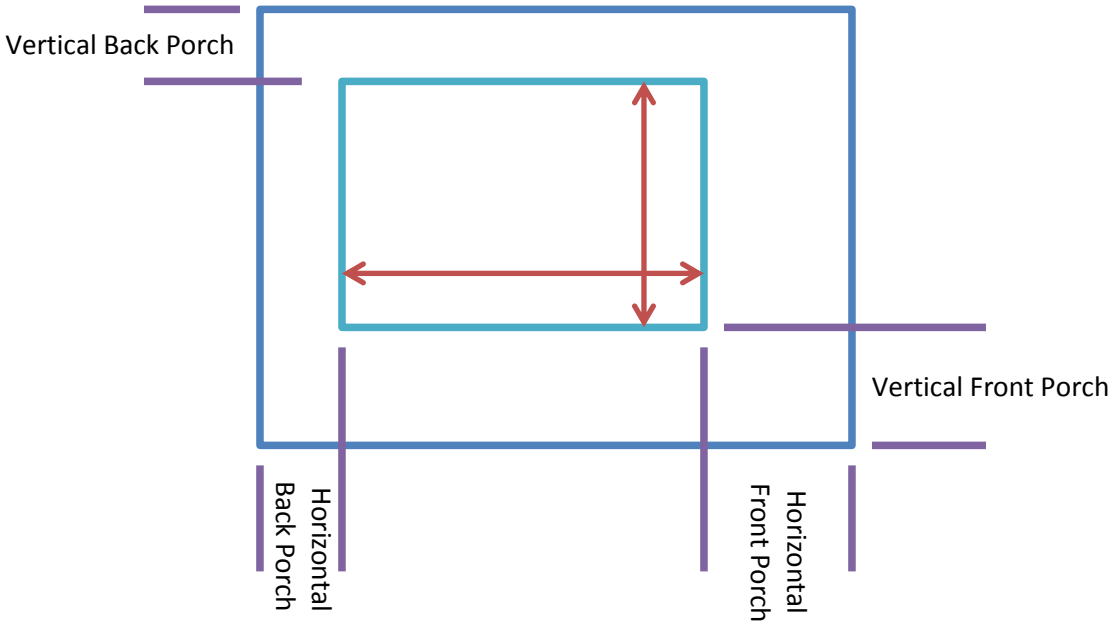


Figure 3: LCD Timing Representation

[2] Figure taken and modified from https://specialcomp.com/beagleboard/BeagleLCD2.htm

First a few signals and explanations of their purpose:

CLOCK – synchronizes timings

VSYNC – resets the circuitry so that the next pixel to be updated will be the pixel in the upper left corner

HSYNC – sets the next pixel to be the first pixel in the next row down

ENABLE – indicates valid pixel data (optional)

DATA – pixel data lines

Figure 3 above describes some of the basic properties of LCD timings. First of all the light blue square in the center represents the time where pixel data is actually being output to the screen. The Vertical Back Porch (VBP) is the time immediately after a VSYNC pulse where no pixel data is displayed. The Vertical Front Porch (VFP) similarly is the time immediately before a VSYNC pulse where no pixel data is displayed. Mirroring the VFP in functionality the Horizontal Back Porch (HBP) and Horizontal Front Porch (HFP) are the times immediately after and before a HSYNC pulse respectively, where no pixel data is displayed. These parameters will differ between different LCD screens and need not be equal, as is shown in Figure 3.

My interest in this information came about while attempting to get the LCD screen working by changing the frame buffer data under Linux running on the BeagleBoard. Using the `fbset` utility one can change these display parameters for the device. For Example one such call to fbset can be seen in Figure 4 below.

```
ubuntu@beagleboard:~$ fbset -i
mode "1280x720-60"
    # D: 64.000 MHz, H: 44.444 kHz, V: 59.979 Hz
    geometry 1280 720 1280 720 16
    timings 15625 80 48 3 13 32 5
    rgba 5/11,6/5,5/0,0/0
endmode
```

Figure 4: Example Call to fbset

The numbers next to timings indicate pixel clock (in picoseconds), HBP (in pixels), HFP (in pixels), VBP (in pixel lines), VFP (in pixel lines), Horizontal Sync Length (in pixels), and Vertical Sync Length (in pixels). Pixel lines are typically less than pixels since they incorporate a whole lines worth of pixels in them. I found that, even though I could change the resolution and timings, I was unable to change the pixel clock for some reason (running the command to change it did absolutely nothing). I'm not sure what the cause of this is but it is a noteworthy observation.

E-Ink

E-Ink is a type of display that, although it has a low refresh rate, has extremely low power consumption. This technology is used in Amazon's Kindle and several other E-Readers out there for just this property. Ideally this device would use an E-Ink screen in order to minimize the amount of power consumed, this ended up not being a viable plan due to the extremely large cost just for a prototyping kit, (on the order of a few thousand dollars, much more than the budget of this project). Perhaps as this technology matures it will become a more viable option for the device.

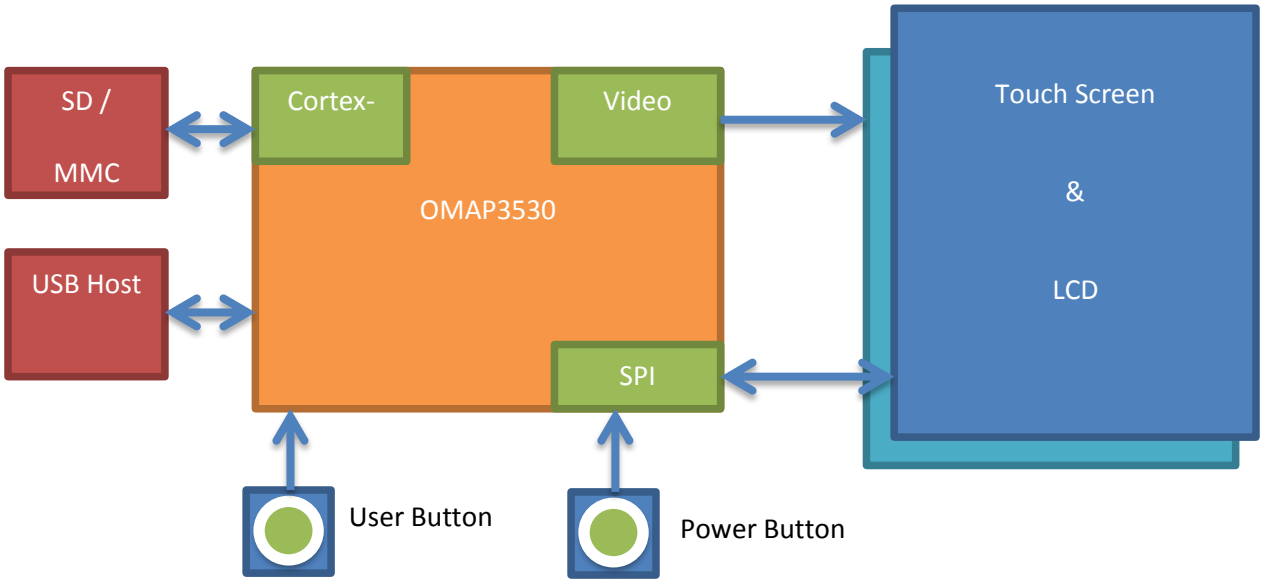


Figure 5: Simple System Level Block Diagram

The design chosen uses the BeagleBoard to drive an LCD display with a touch screen interface and will run Linux (Ubuntu 10.04) off of a 4GB SD card. The Power requirements are satisfied by the choice of a processor that is power efficient and has options for low power and standby modes. The size requirement cannot currently because the screen size in use currently will not be the screen used in the final product but given an adequate screen size it should be able to meet the size requirements. The interface to the device is not yet complete but it is also not incapable of meeting these requirements. A 4GB SD card is currently being used to store the operating system and any additional files needed. This is plenty of space for an OS, document viewer, and documents. This also allows for plenty of room for expansion and further development. Figure 5 above provides a system level block diagram of the components of this project and how they are connected.

Development Board Decision

		Cost	Power	Performance	Features			Totals
					MMU	Video	Other	
	Weight	9	9	7	3	2	3	330
BeagleBoard ***(ARM Cortex-A8 / OMAP 3530)***	***Value***	$150	300mW	2.0 DMIPS / MHz, 600Mhz-1GHz	Yes	DVI out, Headers for LCD, S-Video	USB, 256MB NAND flash, 128 MB RAM, SD/MMC, Audio in/out, UART over RS232 port, community Support	
	Score	8	8	7	10	7	8	261
Atmel SAM3U-EK ***(ARM Cortex-M3 / SAM3U)***	***Value***	$235	no conclusive power consumption data found for Cortex-M3, will assume power consumption is similar to that of the Cortex-A8	1.25 DMIPS/MHz, 96Mhz	None	Board includes LCD screen for development	USB, 3D accelerometer, UART & USART RS232 port, SD/MMC, 256MB NAND flash, 128 PSRAM, Audio in/out, 10bit and 12 bit ADC connectors	
	Score	7	8	5	0	8	8	210
HawkBoard ***(ARM 926EJ-S / OMAP L138)*** - *looked promising but doesn't seem to be readily available at the moment*	***Value***	Around $100 ?	420mW	220MIPS @ 200Mhz	Yes	VGA out	128MB DDR RAM, 128MB NAND flash, UART over RS232, Audio in/out, SATA, Ethernet, USB, SD/MMC, Expansion Headers for SPI, UART, GPIO, I2C, PWM	
	Score	9	7	4	10	7	9	243

Figure 6: Development Board Decision Matrix

Deciding on a development board was a challenging task in itself. Of the things that had to be taken into condideration, cost and power consumption were the two most important factors in deciding between boards. This can be seen in Figure 6 Below where both cost and power have a weight of 9. Next most important feature is performance, if the board is going to be running an operating system and providing a responsive interface to the user it needs to have enough computing power to do this. Next are the

additional features needed, these are not necessarily required but added benefits. Video is under this category because all three boards have some sort of video out , they just differ in the output formats that are avaliable to each board.

Software

For software it was decided that an OS would make being able to manage files, view multiple document, adding enhancemnents in the future easier. For this the Linux distro Ubuntu was chosen, specificly Ubuntu 10.04 (Lucid). Much work has already been done by the online community on getting linux running on the BeagleBoard and there were esentially a step by step instructions on getting Ubuntu up and running so this is what won out in terms of both the community support and how quickly it could be installed.

Originaly the plan was to Use Angstrom, a verry light weight Linux distro, and have it run from the NAND flash and at one point Angstom was actually booting from the NAND flash. The only problem with this is that first, I was not very familiar with the Angstrom distro, and second, although it all fit on the nand flash it didn't really leave room for much else. This limited how much development that could be done on the system itself,as opposed to developing on annother machine and cross compiling which would add annother layer of complexity that was not really necessary, and placed limitations on further features to be added beyond the scope of this project. Thus I decided to run a mainstreem, well supported, OS from a SD card with plenty of space for development and additional features.

For the cost of the project, it can be seen in Figure 7 in the parts list section of the Appendix that the total cost for the parts was $436, less than the requirement of being less then $500. The majority of these parts were purchased from a company called special computing [5].

V. Development & Construction

Step 1: A serial Connection to the BeagleBoard

A simple enough procedure, connect the IDC10 cable to the Null Modem Cable to the RS232 to USB cable (as shown in Figure 2) and plug it into the computer that will connect to the BeagleBoard (assuming the computer has no RS232 serial port). If Windows is being used then you can use PuTTY to connect to the device over serial, just choose the serial radio button and enter the com port that was assigned to the USB to serial device, using 115200 for the baud rate. I did run into some connection problems using PuTTY however, (it would occasionally just stop sending characters from the terminal to the BeagleBoard) so I switched over to Linux and "screen" which can be used to connect to a serial device. To use screen enter screen <device name> <baud rate> and this will get you connected. Also to exit use <ctrl> + A followed by K, and to scroll up use <ctrl> + A followed by <esc>.

Step 2: Understanding Uboot

Uboot is the boot loader that comes pre-installed on the BeagleBoard. A few things to note about Uboot are the environment variable commands `setenv, printenv, saveenv`. These are used to manage many of the environment variable that are available, including the baud rate to use, the boot arguments to use for Linux, and any boot scripts you may want to run. Many of the other commands are self-explanatory and there is a help menu as well. One thing to note however is that the command `md` (memory dump) should not be used with addresses lower than 0x80000000, any address lower than this is not referencing memory and will cause the board to freeze [3]. Also noteworthy, different versions of Uboot may use different commands, most notably `mmcinit` vs. `mmc init` (which initialize the memory card) and `source` vs. `autoscr` which (run the contents of the named environment variable).

A few common error messages I ran into include “bad NAND or CRC using default environment,” this usually occurs after flashing a new version of Uboot to the NAND flash using one of the boot scripts out there that does this such as the one found here http://beagleboard.googlecode.com/files/boot_v3.scr and is the result of the environment variables on the NAND flash having been cleared. To fix this simply do a `saveenv` to write the default environment variables to NAND. Another common error I ran into was one saying “can’t fatload from mmc 0:1,” this is likely because mmc init /mmcinit has not been run or the wrong one was used for this particular version of Uboot.

If the BeagleBoard becomes unbootable for some reason, say the section of the NAND flash containing Uboot gets overwritten or corrupted somehow, the board can be recovered by following the instructions found here:

http://elinux.org/BeagleBoardRecovery

Step 3: Installing Linux

As mentioned in the design section I initially had tried using Angstrom but switched over to Ubuntu for a more familiar environment. From here I managed to get networking up and running using the BeagleBoard Ubuntu page here http://elinux.org/BeagleBoardUbuntu. From that point It wasn’t too much more effort to get x-server and gnome installed.

Step4: Getting Something to Display on the LCD

Getting something to display on the LCD was both simple and frustrating. This is because of one small difficulty that impeded my progress in this area for the latter part of winter quarter and the beginning of spring quarter. The difficulty was that when plugging the LCD expansion into the headers on the BeagleBoard pushing it all the way down, for some reason, causes it to not connect correctly and thus not work.

IMORTANT If the LCD backlight does not display anything after a few seconds try adjusting the connection between the BeagleBoard and the LCD Expansion board.

Also, in the boot args, set the frame buffer mode to dvi:480x272@60, for some reason this defaults to 480x300 but it at least gets something on the screen. It would Appear the resolution 480x272 isn't natively supported but I wasn't able to figure out how to add support for it.

Step5: Getting the Touch Screen Working

I unfortunately was unable to get this working before running out of time. I was in the process of compiling and installing a newer version of the kernel version 2.6.34 from this repository http://www.rcn-ee.net/deb/lucid/. One thing to note on this, compiling the kernel on the BeagleBoard itself, although simpler than trying to cross compile, takes a very long time.

Useful resources

BeagleBoard Wiki - http://elinux.org/BeagleBoard

BeagleBoard Google Group - http://groups.google.com/group/beagleboard

BeagleBoard Ubuntu - http://elinux.org/BeagleBoardUbuntu

BeagleBoard Recovery http://elinux.org/BeagleBoardRecovery

Kernels for BeagleBoard - http://www.rcn-ee.net/deb/lucid/

VI. Conclusion

Thus far, the SD card has a working version of Linux with a GUI interface installed, and the LCD screen is working. The resolution on the LCD screen however is not quite right, it is displaying at a resolution of 480x300 when it should be 480x272. This is due to the resolution not being supported by the current Linux kernel. The touch screen Interface is also still not functional. Because of the difficulties encountered in just getting the LCD screen working not much progress was made beyond this. Figure 8 in the appendix shows what the planned schedule was, due to the above stated difficulties the interface with the touchscreen, front end software development and power analysis were not able to be completed.

Future Development for the device should aim to get these two hardware components functioning correctly and begin development of custom software for the device. Looking into the power saving options of the MCU and how to control them should also be a future goal.

Through this project I gained quite a bit of experience in working with an application based microcontroller, an LCD screen, a Touch Screen, and setting up Linux to work with these devices. In setting up Linux to run on the BeagleBoard I became familiar with some aspects of Linux that I had not explored until this point, this includes setting the network connection, Installing and setting up x-server and a window manager, obtaining and compiling a new kernel from scratch, configuring boot parameters and setting the frame buffer resolution. Most of this I had known of but never had the need to explore it until now. I am also coming out of this with a much better understanding of what control signals are sent to the LCD and much of the terminology for the timing requirements for LCDs.

VII. Bibliography

[1] “Cortex-A8 Processor.” ARM. June 3, 2010. June 14 2010.

<http://www.arm.com/products/processors/cortex-a/cortex-a8.php>

[2] “Introduction to Graphics and LCD Technologies.”NXP. February 2009. May 29, 2010.

<http://ics.nxp.com/literature/presentations/microcontrollers/pdf/graphics.lcd.technologies.pdf>

[3] Lilja, Magnus. “Uboot md crash on PDK.” Mail-Archive. April 9, 2009. March 12, 2010.

<http://www.mail-archive.com/u-boot@lists.denx.de/msg12129.html>

[4] “E Ink Corporation | Low Power, Electronic Paper Displays.” E Ink Corporation. May 24, 2010. June 12, 2010. <http://www.eink.com/kits/amepd.html>

[5] “SpecialComp Products.” SpecialComp. June 14, 2010. June 14, 2010.

<https://specialcomp.com/beagleboard/order.htm>

[6] “BeagleBoard System Reference Manual Rev C4.” BeagleBoard. December 15, 2009. February 15. 2010. <http://beagleboard.org/static/BBSRM_latest.pdf>

VIII. Appendices

A. Parts List

Part Name	Cost
BeagleBoard RevC4	$149.00
LCD Expansion Board and 4.3-in 480x272 LCD Screen with 4-wire Resistive Touchscreen	$149.00
BeagleBoard Clear Acrylic Case with Nylon Fasteners	$49.00
IDC10 to DB9M Bulkhead (RS-232) Cable	$5.00
DB9F Null Modem (RS-232) Cable (6-ft)	$4.00
USB to DB9M RS-232 Cable (6-ft)	$10.00
USB 2.0 Extension Cable	$2.00
USB Mini-A to USB A Female OTG Cable (6-in)	$9.00
USB Mini-B Male to USB A Male Device Cable (6-ft)	$5.00
USB 2.0 High-Speed 3-Port Hub with Ethernet	$30.00
AC Power Adaptor 5VDC@2.5A for BeagleBoard	$10.00
USB A Male to 5.5mm Power Plug Cable	$8.00
HDMI Male to DVI-D Male Cable (6-ft)	$6.00
Total	**$436.00**

Figure 7: Parts List with Costs

B. Schedule

Winter 2010 Quarter		
1/20/2010	2/5/2010	research and decide on hardware
2/5/2010	2/17/2010	project specification document
2/5/2010	2/12/2010	order and receive hardware
2/12/2010	2/26/2010	get Linux up and running
2/26/2010	EoQ*	interface with LCD
Spring 2010 Quarter		
3/29/2010	4/5/2010	Interface with LCD
3/29/2010	4/12/2010	Interface with touchscreen
4/14/2010	5/19/2010	software front end
5/19/2010	EoQ*	Power Analysis

**End of Quarter*

Figure 8: Project Schedule