Digital Picture Frame
Utilizing the Intel Xscale Architecture
Kelvin Bui

Background/Purpose:

The digital picture frame has been recently created as products to display the consumer’s massive amount of digital pictures taken. The speed at which the collection of digital images grows faster than even Moore’s Law[1]. I believe that this is a simple yet very much needed device. In previous years, the goal has been to allow the user to get data anytime and anywhere. The problem is however, that all this data will never be used because of the effort it takes to just manage the data. This is where the digital picture frame will help. It will display photos that the user can choose to delete anytime simply. Most of these are relatively simple embedded systems, taking pictures from a flash drive and displaying them in regular repeating intervals. However, we are in an age where PDAs verge on the edge of an mp3 player, mp3 players verge on the edge of video players, and cell phone are gaining more functionality than ever imagined.[2] So the existence of a stand alone picture viewer has a dim future indeed. Products now have to be aware of each other and be very configurable.

Currently, there are few manufacturers of these digital picture frames. There are some well known manufacturers as Sony, and Kensington. However, the more advanced features belong to unknown companies such as Ceiva and Storybox who allow picture downloading through the phone line.[3] The next generation Ceivas are reportedly going to be able to print them directly or order them online from the appliance. None of these units however, are multi-purpose. None include a simple camera, nor do they offer any other applications. The picture frame has the advantageous feature that people specifically look at it when they walk by. That makes the perfect device to leave notes, find the time, or even read the news.

Because it has the limitation of merely a couple buttons and a stylus, we do not want too much interaction with the user. It would be too cumbersome for them to do any task with high complexity or one that requires high precision. There will not be enough buttons to form a keyboard, nor powerful enough hardware to do heavy calculations. The digital picture frame is limited to more passive relationship. It’s primary goal is just to have the user “look” at it.
Project description:

The project is to actually make one of these devices and having it more integrated with its environment using the XScale’s I/O capabilities. We want to explore the steps and procedures that would be necessary to create a commercial product. The overall goal is to get a working prototype that will require no user commands. It will also have some features not available in markets yet. It will have: a touch-screen based navigation, news/traffic information, a clock function, notebook function, the ability to pull images off of internet image databases and simple image editing.

Platform:

The platform we are using for this project the Intel XScale platform. It is designed to optimize low power consumption with dynamic voltage scaling while providing high performance for the multitudes of I/O capabilities that it has.[4] It has a strong application inclination towards the cell phone market, as the cell phone is increasingly growing in its need for computational power, high I/O interaction, and low power requirements. Needless to say, this specific platform is probably much too expensive and powerful for our needs to create a digital picture frame. Besides image manipulation, the digital picture frame does not require any heavy computational power. It would probably be more cost efficient to make a stand alone picture frame with a lower cost CPU such as the Atmel chip. We must point out that it is very likely that the product will ship not by itself but as a package with many more utilities, of which may require more power.

API/GUI Platform:

We chose the Qtopia/qpe environment to create our application. Qtopia is one of the top leaders in their API development environment. They are the largest platform for linux-based PDAs and others.[5] They are also expanding into the cell phone market, which shows growing progress. One of the other main reasons we chose this platform is because an old version was included with Intel’s Mainstone setup. We got to see a taste of what the GUI environment was like. It was all touch screen based; something that is required if it is going to be made for commercial use.

Results:

Unfortunately, we were not able to actually create any applications for the Qtopia GUI. It was more difficult to install and setup everything needed to start programming for the environment than we had originally thought. Heavy package dependencies and compiler errors prevented us from installing a new version of the qpe on the OS. And
even if we had, it might have taken a very significant amount of time to learn and understand the new API. We spent a lot of time trying to debug the system specific errors we had, including manually editing the make files, substituting working files from an older version, and attempting to use precompiled versions. In the end, we decided that it took too much time, and we chose to create a picture from that would not run inside qpe’s GUI, but instead one that we created.

**Implementation:**

We began by capturing pictures from ci-capture and storing them into files. Then we would attempt to reload them at a different time. We discovered that the camera had a video mode and a still-picture mode. Of course, we would implement it in our code as a picture. For it to be a picture frame, it would have to cycle through the pictures like a slide show. We do this by using a system call to execute a “ls” and tell us what files to display using their extension. Like a book we would implement the ability to flip back and forth between a picture you just missed and a future photo you would like to see.

Next we implemented little features like displaying the name of the file with the date and time. The date and time must stay constant even while the program is outputting to screen. The time feature is a very important one for the user because it is something that they use a lot, even if it is simple. For example, the cell phone is used to look up the time very often nowadays, decreasing the market share of wrist watches.

Besides the obvious minimal features that we must implement to make it a digital picture frame, the next most important is the interface. We were able to get the touch screen to work even without the GUI. With the ability of the touch screen however, greatly increases the complexity of the program as it is no longer a serial program. It must now handle user input at any point in time. This required us to use multiple threads to execute and manage the system state, which looks something like below.

All the threads work concurrently with each other modifying the data tags if they need to send messages to other threads. We guarantee that we do not end up with any deadlocks by allowing only some threads to write to certain data tags. If the system state chart grew much larger, we would have to use locking mechanisms.
Conclusion:

Although we were not able to deliver a fully commercial production using a standardized platform (qtopia) we did manage to create a product that had the essentials that allowed the user to operate the system with ease. The problems that prevented us from fully utilizing a developed gui were mostly due to lack of experience. We understand now that in a commercial environment with the constraints of time to market, system costs, and labor costs, experience is the key to success.

Further exploration:

We came across a couple of qtopia IDEs that we did not have time to try out. Graphical IDE are becoming very popular now with a built in editor, compiler and debugger all in one package. This would greatly reduce the initial setup problems we had and would allow us to develop much more quickly. It is worth to explore these new tools to make efficient use of money and time, both which are heavily strained.

References:


http://www.shortcourses.com/how/digitalframe/digitaldisplays.htm

[4] Intel XScale Microarchitecture Product Brief 
http://www.intel.com/design/intelxscale/xscaleproductbriefweb.pdf

http://www.trolltech.com/products/qtopia/qtopia-overview


“Linux IP Masquerade HOW TO” Jerry Winegarden 
http://wwwherry.oit.duke.edu/linux/HOWTO/ipchains_with_adsl_v62.html

“how to get Started with Qtopia development” – A simple graphical program 