Kyle Stewart Greg Bray

Wi-Fi Clock Radio

CS 3992 Final Proposal

Website: http://www.theg2.net/kingme/ 4/24/2007

Abstract

The purpose of this project is to alleviate the stress associated with an alarm clock by giving it features that make the experience of waking up in the morning more enjoyable. The alarm clock will be able to connect to an existing wireless network on which the user runs a personal computer that stores a personal music collection. The user can configure the alarm clock to connect to this collection of music so that they may fall asleep and wake up to whatever music they wish. The user will be able to configure the device through a touch screen interface in order to keep the amount of technical knowledge and setup to a minimum.

Introduction & Motivation

Waking up early every morning in order to get to work on time is an almost daily occurrence for nearly all working class citizens in industrialized countries. The use of alarm clocks came along with the need to arise earlier than the normal circadian rhythms of the human body. Due to the fact that this process is inherently unnatural, the daily task of waking up early for work and school is often an unpleasant experience.

Unfortunately, the technology of alarm clocks has not progressed in the same fashion as its electronic counterparts such as televisions, personal computers, and video game systems. The desire to hear something besides the harsh “beep-beep-beep” that an alarm clock normally emits was first satisfied by clock radios that incorporate Amplitude Modulation / Frequency Modulation (AM/FM) tuners, which allowed alarm clock patrons to wake up to their choice of radio stations. The alarm clock passed through another innovation with the popularization of the compact disc (CD), which gave users the ability to select their own music to wake up to in the morning and provided a crystal clear signal free of the interference associated with AM/FM radio. The principle limitation of CDs is the number of songs they are able to store, usually no more than 72 minutes of music and oftentimes much less. Though more an extension rather than an innovation, the use of compression formats such as MP3 and AAC allow hundreds of songs to be stored on one CD; however alarm clocks capable of using this format are rare and often lack features such as playlist selection.

While alarm clock technology has stagnated for many years, several other technologies have made significant advancements. Two technologies that relate to the future of alarm clocks are music and wireless networking. The market interest for users desiring quick and easy access to large volumes of music is clearly demonstrated by the success of the iPod and other music players. Wireless networking has also become commonplace now that many personal computers ship with wireless capabilities and offer wireless routers for the ever popular price of “Free with rebate.”

Despite the obvious market potential for alarm clocks that can wireless connect to music collections stored on a personal computer; no one has presented a budget alarm clock to do exactly that. The one product to come close to filling this market gap is the Roku SoundBridge Radio; however, the $400 price tag not only places it beyond the reach of the vast majority of consumers, it also seems an unreasonable amount considering the dedicated nature of the alarm clock functionality.

Proposed Work

For our senior project we have chosen to create a wireless clock radio that will allow a user to wakeup to music from their personal music collection. The system will consist of a small touch-screen device that can be used to program the alarm clock and connect to a server that hosts the music collection and play lists. The system will utilize existing wireless networking and home media server solutions that interact with a custom built alarm clock client.

The alarm clock will be built using a small form factor computer with an embedded liquid crystal display (LCD) touch screen. A touch screen will allow the most flexible user interface options since it does not depend on a predetermined button set. In order to minimize the time spent on the hardware component of the project, we will use off the shelf components as much as possible. The TC-10 product from EarthLCD.com [1] provides a low-cost solution for these requirements. It consists of a 10” touch sensitive LCD screen that is directly attached to an embedded system. This embedded system consists of a National Geode x86 based processor running at 300 MHz. Since the processor is x86 based, this allows for a greater degree of flexibility in operating systems since there are many free versions of the Linux operating system that work on this architecture. The system also provides a low profile peripheral component interconnect (PCI) bus. This PCI bus allows us to use a low profile wireless card such as the Novatech NV-926W [2] to connect to a 802.11 wireless network. This card is particularly useful since drivers already exist for a variety of operating systems, including Linux. We will use a secure digital (SD) flash-based storage device to store the operating system and all the necessary software components that will interface with the hardware. The final hardware component will be a set of passive speakers connected to the audio line out output of the TC-10 through an amplifier. If time permits, a suitable housing for the unit will be constructed using a transparent material such as Plexiglas; however, this should be considered a stretch goal based on esthetics rather than functionality.

The hardware will run an embedded version of the Linux operation system. We will use the Damn Small Linux (DSL) [3] distribution of Linux due to its extremely small footprint and popular use in embedded systems. We will utilize the wireless networking card to connect to a user’s home network. The drivers to do this are already available, but must be loaded into the operating system and tested for compatibility. The software to handle Transport Control Protocol/Internet Protocol (TCP/IP) communication is also built into the operating system, but additional software must be created to implement the features of a clock radio. Also a graphical user interface (GUI) must be created that will allow users to configure and use the different features available for the radio. This GUI will use the MiniGUI [4] software library in order to reduce the time required to build the GUI. MiniGUI is a set of open-source, cross-platform utilities for creating GUIs used in real-time embedded systems. There will also be a software component that is installed on the users home PC to enable it to act as a media server for streaming audio files. The media server will allow the user to select playlists to be used by the alarm clock and will stream the selected playlist to the clock radio when it is time to get up.

Demonstration

Our project has three primary features that we plan to demonstrate. The first and most basic feature is the integration of the hardware components and operating system to create a functioning embedded system. This is demonstrated by successfully booting the TC-10 into the Linux operating system that should automatically load the wireless clock radio application. This will all be clearly observable through the LCD screen. This demonstrates the interfaces used between the hardware and the operating system. Once the operating system is up and going, it also demonstrates the graphical interfaces required for the GUI. A successful demonstration of the GUI will include demonstrating the ability of the user to navigate menus configure options through the touch screen interface. The goal is to set up the user interface in an intuitive manner so that users do not need to reference a set of instructions in order to configure the device.

The second feature we plan to demonstrate is the ability to configure the wireless clock radio. This differs from the general GUI navigation described previously in that here we describe the specific configuration options that will be available. The first configuration requirement will be configuring access to a network. The primary means of accessing a network will be through the wireless 802.11 networking protocol. The user should be able to intuitively set the wireless network settings including any security settings and passwords that are required to access the wireless network. After configuring the network settings, the radio should automatically authenticate to the wireless network and obtain a network address. This will allow the radio to see the music available on a PC running the server-side software. In addition to wireless network settings, the user should also be able to set the current time, alarm time, and music location. The user will also be able to select music to be played when the alarm goes off.

The final and most important feature of the clock radio will be to successfully play the user's selection of music when the current time matches the preset alarm time. This requires that all other features work properly including the wireless connection, user interface controls, media server, media client as well as the physical connections to the speaker system.

Task List

- Client (Kyle)
  - Obtain touch screen hardware
  - Install OS (Linux)
  - Load drivers (Wi-Fi, touch screen, Audio)
  - Configure for Wi-Fi network
  - Develop Client Software for Clock Radio
- Server (Greg)
  - Setup standard media server (FireFly) [5]
  - Customize code base to work with client software
  - Extend media server API to include alarm clock features
- Integration Testing (Kyle and Greg)
  - Test Firefly Media server using default client

    - Test Alarm Clock client using default media server
    - Integrate customized media server with Alarm Clock client
- User Documentation (Kyle and Greg)
    - Installation guide
    - Users guide
    - Advanced users guide for customizing Alarm Clock client

Interface Issues

There are three separate interfaces that must be dealt with for the Alarm clock to be successful. The first is interfacing the Alarm clock software and underlying operating system with the embedded client hardware, which will include a touch screen for user input, 200-500Mhz x86 CPU for processing power, WiFi card for wireless networking, and an audio output jack for connecting to external speakers. The second is interfacing the alarm clock client software with the media server software running on the network.

Careful selection has to be put in place to make sure that the components purchased will be compatible with the Linux OS used to run the system. This interface is simplified somewhat by basing the system on the x86 architecture. This architecture has been along for a long time and many operating systems work very well with it, especially in the embedded systems category. This also allows for mature tool chains such as those found in the GNU toolkits to be used in the development of our software. It also means that the software can be developed and tested on workstation machines that are also based on the x86, which will shorten the development cycle. The first step for interfacing the operating system to the board will be to load a bootable version of the Damn Small Linux (DSL) operating system [3] onto a compact flash card. The TC-10 uses compact flash as a primary means of persistent storage. DSL is a Linux distribution with an extremely small memory footprint (around 50MB) that is designed to be used in embedded systems with small amounts of storage available. Loading the operating system will be done on a workstation machine that can mount compact flash devices as storage. The installation of DSL will then take place by selecting the compact flash device as the destination for the operating system. The flash device will then plug into the TC-10 using the built-in interface for compact flash devices. Since the TC-10 is designed to look for this flash device on startup, a proper installation of a bootable operating system will then load the DSL operating system on the TC-10.

After the operating system is loaded onto the TC-10, the next interfacing task is to interface the operating system with the other components on the board. The touch screen sends a generic PS/2 signal used by virtually all mouse and keyboard devices. DSL already has the drivers to interact with these devices. One potential problem with the touch screen is the calibration so that the touch screen knows how to map the touch input to the pixels being displayed on the screen. This may require additional development to create a calibration program that will calibrate the touch screen when necessary. Another hardware component that the operating system needs to interact with is the wireless card. The particular card we chose, the Novatech NV-926W, was specifically chosen because of known compatibility with Linux operating systems. The card is built on the RaLink 2560F chip which is reported to work with the Linux rt2500 driver [7]. The drivers for the PCI bus itself are also included in the DSL package.

The next interface is the API between the alarm clock software running on the client and the media server software running on the server. The current plan is to customize an existing Open Source media server to fit the needs of our alarm clock application. For streaming audio we will program the client software to connect using the existing API of the media server software, but a new API will have to be created for the alarm clock configuration settings.

The final interface will be that of the media server software, which will provide access to the user's music collection and playlists. Our research has found an open source media server called Firefly [5], which offers most of the features required for our alarm clock application. This software will be customized to include additional features such as play list selection and alarm date/time configuration.

The Firefly media server utilizes an XML based API called mt-daapd [6] that is very similar to the iTunes API and allows you to query for information using simple HTTP requests. The mt-daapd will allow you to remotely browse through the media library and access information such as the song title, album, genre, artist, bit rate, and length. It will also let you search for a specific song, create a new play list, or even add and remove songs from an existing play list. These features will allow our alarm clock client to have all the features of a standard media player like iTunes even though it does not store any of the media files locally.

Testing Policy

Our chosen project is designed around the KISS principle: It's just an Alarm Clock. This should help as we look at creating our testing policy, as our desired feature set is quite small. We plan on creating a set of use-case scenarios that list what a customer would try to do with the product (i.e. set alarm clock for 7AM, select the play list they would like to wake up to, etc...) and using these as the basis when testing the system. We also plan on getting the system into an operational condition as early as possible, so that we can identify bugs and have a short, iterative approach to fixing bugs in the system. Also we would like to have at least one beta user from a non-technical background test the system to help identify any usability issues before we make our final presentation at the end of the semester.

Schedule & Milestones

- August
  - Collect all hardware components (3 days)
  - Finish planning all interface requirements (2 weeks)
- September
  - First Client/Server interoperability check (1 week)
  - Lower level API implemented (3 weeks)
- October
  - Connect GUI to lower level API (2 weeks)
  - Begin integration testing (2 weeks)
- November
  - Finish integration testing (2 weeks)
  - Write user documentation (1 week)

    - Start beta user testing (1 week)
- December
    - Incorporate beta user feedback (2 weeks)
    - Final demonstration date (1 week)

Bill of Materials

- Hardware
    a. Touch Screen/Embedded System
        i. TC-10 from EarthLCD.com ($259) [1]
    b. Low profile wireless PCI card
        i. Novatech NV-926W from LogicSupply.com ($28) [2]
    c. 1GB CompactFlash Card
        i. Generic from NewEgg.com ($15.99) [7]
    d. Generic passive speaker and amplifier
        i. Provided by team members
- Software
    a. Operating System and Applications
        i. Damn Small Linux (Open Source, customized by team) [3]
        ii. Clock Radio Application GUI (Developed by team)
        iii. Clock Radio Application Backend (Developed by team)
        iv. Cross-Platform Music Server (Custom version of Firefly developed by team)
    b. Drivers
        i. Novatech NV-926W (Open Source and available for Linux) [2]
        ii. TC-10 (Uses generic drivers for touch screen and audio) [1]
    c. Required Libraries Not Provided by Operating System
        i. MiniGUI (Open Source, cross-platform) [4]

Risk Assessment

Our biggest concern right now is that the documentation for the TC-10 is somewhat lacking, which may become an issue as we look at adding additional devices to it. Luckily the device includes all of the essential components, but if in the future we choose to add an external hard drive it may require a bit of trial-and-error to get everything connected correctly. Linux support for our selected devices is our next major concern, but most of the components were specifically selected because of their support for Linux. Streaming audio is a low concern, since we are relying on existing software components that have proven to be reliable.

- Client
    – EarthLCD Supply (Low)
    – TC-10 has incomplete documentation (High)
    – Install Linux OS (Low)
    – Load drivers (Medium)
    – Configure for Wi-Fi network (Medium)
    – Develop Client Software for Clock Radio (Medium)
- Server

- Setup FireFly media server (Low)
- Customize code base to work with client software (Medium)
- Extend media server API to include alarm clock features (Medium)

Stretch Goals

After the initial product release, we would look at expanding the platform to work with other Personal Information Management (PIM) systems. These could include personal email (POP or Web-based), calendar/scheduling, or a To-Do list. Another area that we would want to look into is user customization for both the software (UI skins, sounds, scripts) and hardware (cases/covers, memory upgrade, USB expantion). We also might look into integrating a Flash [9] player to allow the alarm clock to access flash based widgets available on the internet.

Conclusion

An alarm clock that will let you choose what you want to hear when you first wake up in the morning will not only make sure that you get to school or work on time, but it will also make sure that you are in a better mood when you get there. Currently, the cost of such a luxury is outside the price range of most average consumers, but we feel that by using existing software and hardware components the price of waking up happy can be significantly reduced.

Bibliography

[1] Earth LCD TC-10 board, http://store.earthlcd.com/TC-10-TC10?sc=7&category=13
[2] Novatech NV-926W, http://www.logicsupply.com/product_info.php/products_id/461
[3] Damn Small Linux distribution, http://www.damnsmalllinux.org/
[4] MiniGui UI system, http://www.minigui.org/
[5] Firefly media server, http://www.fireflymediaserver.org/
[6] mt-daapd xml client API, http://wiki.mt-daapd.org/wiki/XML_Client_API
[7] 1GB CF, http://www.newegg.com/Product/Product.aspx?Item=N82E16820134020
[8] http://tuxinside.us/?Matts_Corner:Reviews:NovaTech_NV-926W
[9] http://www.adobe.com/products/flash/