EMBEDDED LINUX BASED ALBUM BROWSER SYSTEM AT MUSIC STORES

Suryadiputra Liawatimena [1]
[1]Computer Engineering Department, Computer Science Faculty
Bina Nusantara University, Jl. KH. Syahdan 9, Kemanggisan/Palmerah. Jakarta. 11480.
Email: suryadi@binus.edu

The goal of this research is the creation of an album browser system at a music store based on embedded Linux. It is expected with this system; it will help the promotion of said music store and make the customers activity at the store simpler and easier. This system uses NFS for networking, database system, ripping software, and GUI development. The research method used are and laboratory experiments to test the system's hardware using TPC-57 (Touch Panel Computer 5.7" SA2410 ARM-9 Medallion CPU Module) and software using QtopiaCore. The result of the research are; 1. The database query process is working properly; 2. The audio data buffering process is working properly. With those experiment results, it can be concluded that the summary of this research is that the system is ready to be implemented and used in the music stores.

Keywords: *Embedded Linux, Database, Cross compile, Bitbake, QtopiaCore*

Nearly every aspect of life in this world has been enhanced by the development of technology. In addition, the development of information technology has been accompanied by the development of the recent music industry of today. This can be observed from the development of the music producing media all the way to music storage media. The evolution of music storage media has always been very fast indeed; from records to MP3 data. Every corner of malls in Jakarta has been filled with music stores that sell albums in cassettes or CDs. From this perspective (the fast growing music industry), the development of technology can be made to ease the activities that customers will perform at a music store.

The fast growing music industry has an impact of more and more music stores opening up in Indonesia. The collection of these music stores are enormous, ranging from hundreds to thousands of CDs. The arrangements of these CDs are usually grouped by genres, like pop, rock, jazz, etc. It also can be arranged based on the alphabet of the artist, from A to Z.

Generally, the motivation of the consumers of music stores to buy cassettes or CDs are they are fans of the artist, they hear information of the album from various sources; they saw the songs' video clip on television, or references from other sources. Definitely, the consumers dislike to be disappointed by the album that they will buy. When the consumers are at the music store, it will be difficult for them to find information on the album that they want because of the huge amount of CD collection at that store. They want to hear a preview of the album so they wouldn't be disappointed with the album.

With this reason, consumers would need a way to get more information of the album they would like to buy, which is able to preview the songs of the album along with other information of the album. Because of this, we are interested in solving this problem by creating a system of album browser at the music store that utilizes the development of information technology. This system is created using the application of Embedded Linux using a media TPC-57 [1] that will be used in the music stores.

MODEL, ANALYIS, DESIGN, AND IMPLEMENTATION

Qt and QtopiaCore

Qt is a toolkit for graphic application development which have character across platform. Qt is known as configuration foundation of KDE, a famous graphic environment in Linux. Qt has been configured with C++ and can used in Unix Platform, windows and Mac OS X. Qt also support endorsement toward internationalization, data base access, XML and sheaf handling [2] .

Qt for Embedded Linux (formerly known as Qtopia Core) is the leading application framework for single-purpose devices powered by embedded Linux. It provides a robust and proven development environment enabling manufacturers to efficiently create devices with applications that are tailored to market needs [3] .

Qt for Embedded Linux inherits the power and advantages of Qt 4, Trolltech's leading C++ cross-platform application framework. Trolltech has always demonstrated both commitment and ability to remain ahead of the technology curve, freeing customers to focus on front-end value-adding innovation rather than maintaining the software infrastructure [3, 4, 5].

In Qt, signals and slots are used. A signal is emitted when a particular event occurs. Qt's widgets have many predefined signals, but user can always subclass widgets to add his/ her own signals to them. A slot is a function that is called in response to a particular signal. Qt's widgets have many pre-defined slots, but it is common practice to subclass widgets and add his/ her own slots so that he/she can handle the signals that he/she are interested in.

The signals and slots mechanism is type safe: The signature of a signal must match the signature of the receiving slot (see Figure 1). (In fact, a slot may have a shorter signature than the signal it receives because it can ignore extra arguments.) Since the signatures are compatible, the compiler can help us detect mismatch type. Signals and slots are loosely coupled: A class which emits a signal neither knows nor cares which slots receive the signal. Qt's signals and slots mechanism ensures that if you connect a signal

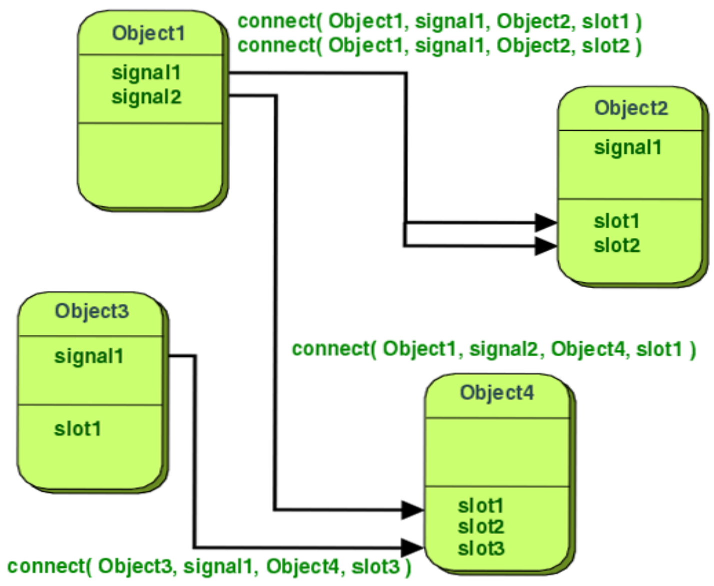

connect(Object1, signal1, Object2, slot1)
connect(Object1, signal1, Object2, slot2)
Object1
signal1
signal2
Object2
signal1
slot1
slot2
Object3
signal1
slot1
connect(Object1, signal2, Object4, slot1)
Object4
slot1
slot2
slot3
connect(Object3, signal1, Object4, slot3)

Figure 1: Signal and slot mechanism

to a slot, the slot will be called with the signal's parameters at the right time. Signals and slots can take any number of arguments of any type. They are completely type safe.

All classes that inherit from QObject or one of its subclasses (e.g., QWidget) can contain signals and slots. Signals are emitted by objects when they change their state in a way that may be interesting to other objects. This is all the object does to communicate. It does not know or care whether anything is receiving the signals it emits. This is true information encapsulation, and ensures that the object can be used as a software component.

Slots can be used for receiving signals, but they are also normal member functions. Just as an object does not know if anything receives its signals, a slot does not know if it has any signals connected to it. This ensures that truly independent components can be created with Qt.

User can connect as many signals as he/she want to a single slot, and a signal can be connected to as many slots as he/she need. It is even possible to connect a signal directly to another signal. (This will emit the second signal immediately whenever the first is emitted). Together, signals and slots make up a powerful component programming mechanism.

With full source code and documentation provided, Qt for Embedded Linux offers the flexibility to create and innovate. Device and application developers using Qt for Embedded Linux can efficiently differentiate their products by taking control of the user experience.

Qt for Embedded Linux not only has minimal hardware dependencies and will run unchanged on most standard embedded Linux set-ups, but can also be easily customized to take advantage of hardware specific accelerations.

Cross Compiling

Cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is run. Cross compiler tools are generally found in use to generate compiles for embedded system or multiple platforms [6].

It is a tool that one must use for a platform where it is inconvenient or impossible to compile on that platform, like microcontrollers that run with a minimal amount of memory for their own purpose. It has become more common to use this tool for paravirtualization where a system may have one or more platforms in use. The fundamental use of a cross compiler is to separate the build environment from the target environment.

Bitbake

Bitbake is a simple tool for the execution of tasks. It is derived from Portage, which is the package management system used by the Gentoo Linux distribution. It is most commonly used to build packages, and is used as the basis of the OpenEmbedded project. Bitbake is used as a tool to automate the compilation process. For example, a MySQL program compiled using Bitbake to be used on an embedded device such as the TPC-57 [7].

NFS

Network File System (NFS) is a network file system protocol originally jointly developed by Sun Microsystems and IBM in 1984, allowing a user on a client computer to access files over a network as easily as if the network devices were attached to its local disks. NFS, like many other protocols, is built on the Open Network Computing Remote Procedure Call (ONC RPC) system. The NFS Client will then import the Remote Archive System from the NFS server, while the NFS server exports the Local Archive System to the client.

Hardware Development

The functions of each part of the module in Figure 2 are as following. **Personal computer(PC)** is used as a

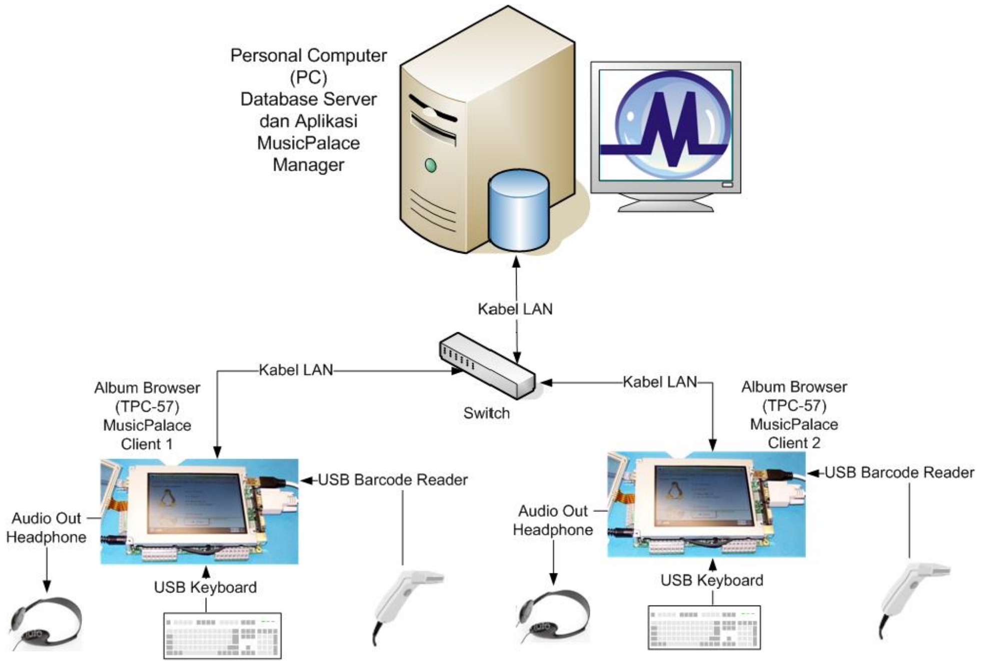

Personal Computer (PC) Database Server dan Aplikasi MusicPalace Manager
Kabel LAN
Switch
Kabel LAN
Kabel LAN
Album Browser (TPC-57) MusicPalace Client 1
Album Browser (TPC-57) MusicPalace Client 2
USB Barcode Reader
USB Barcode Reader
Audio Out Headphone
Audio Out Headphone
USB Keyboard
USB Keyboard

Figure 2: System modeling

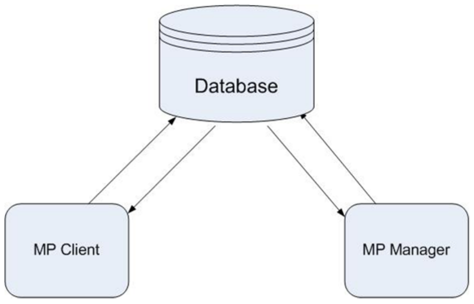


Figure 3: Running system diagram

data storage media and the bridge between MusicPalace Client and MusicPalace Manager Applications. **The MusicPalace Manager application** is used to control the activities of the whole system. **Album Browser (TPC-57)** is used as an interface media for the MusicPalace Client application. **Keyboard** is used as an input device for the MusicPalace Client application. **Barcode Reader** is used as an input device for the MusicPalace Client application as an alternative to the keyboard. **Headphone** is used as an output device for the music. **Network Switch and LAN cables** are used as connecting devices between the PC and the TPC-57.

Software Development

The diagram of the running system is shown in Figure 3. This display is the interface that displays all the activities of all the clients in real time. There are 6 clients, each with its own status for client standby, client ID, the song that is being played, and the shopping cart list for that particular client.

For example on client 1, the client is being used by the user "radip", on which the user is not hearing any song for the moment nor have any list in the shopping cart. Client 2 is on standby state, while clients 3, 4, 5, and 6 are not active.

Software Processing Using Bitbake for Use on the TPC-57 Module

The MP Client application that is used on the TPC-57 (see Figure 4) need a MySQL program, thus the MP Client will get the database information from the MP Manager. Whereas the default programs inside the TPC-57 that came from Techsol did not include the MySQL program. Therefore, MySQL can be used on the TPC-57, a source file ipkg (Debian installer) is needed.

It is done by compiling the MySQL source to be used on embedded ARM using the Bitbake software. For the compiling process using Bitbake, prepare the files that are needed:
- build-chroot-dallas_branch-svn-734.tar.bz2
containing chroot setup directory, where the Bitbake compiling process will take place.
- bitbake-1.6.0-svn-570.tar.bz2
the Bitbake program itself.
- chroot-i386-sarge-tsioe-devel-20060216.tar.gz
contains a Linux OS for the chroot.
- org.openembedded.dev-20060819.tar.bz2
containing specific recipes for open embedded programs.
- org.openembedded.oz354x-20060819.tar.bz2
containing specific recipes for openzaurus - open embedded.
Those files can be obtained via downloading.

Extract the file tarball build-chroot-dallas_branch-svn-734.tar.bz2 to the following work directory: *user@suse:* >

tar xvf build-chroot-dallas_branch-svn-734.tar.bz2 then enter said directory;
Move the files b to f listed above to the *build-chroot-dallas _branch-svn-734* directory, so that the files that are inside the build-chroot-dallas_branch-svn-734 directory are as follows:
1. bitbake-1.6.0-svn-570.tar.bz2
2. chroot-i386-sarge-tsioe-devel-20060216.tar.gz
3. dallas_branch-svn-734.tar.bz2
4. org.openembedded.dev-20060819.tar.bz2
5. org.openembedded.oz354x-20060819.tar.bz2
6. readme
7. README.setup.conf
8. setup.conf
9. suid-exec
10. tsioe-utils.sh

Read the readme file that contains instructions on how to install tarball files. After reading the readme file, do the following first step:
user@suse: /build-chroot-dallas_branch-svn-734> ./suid-exec "./tsioe-utils.sh setup"
This would produce a rootfs folder chroot.

Put the source files that have already been downloaded into the directory */build-chroot-dallas_branch-svn-0734 /rootfs/home/devel/tsidist-dallas_branch-svn-734/build/* and extract the tarball file with the following command:
user@suse: /build-chroot-dallas_branch-svn-734> ./suid-exec "./tsioe-utils.sh build"

Wait until the extraction process is finished as the process takes quite a long time to finish (at least 4 hours on some systems). If the computer is connected to the internet then the process will automatically download the newest sources so that the process will take more time and require more hard disk space.

After the extracting process is finished, the result will be a burnable image of the OS that can be installed on the TPC-57 and ipkg files that are located at the directory *build-chroot-dallas_branch-svn-734/rootfs/home/devel /tsidist-dallas_branch-svn-734/build/tmp-glibc-stable/ deploy.*

The result of extracting the source tarball does not have the MySQL ipkg files yet, to get those files do a Bitbake process for MySQL as follows:

Set the proc mount for MySQL with the following command:
mount -t proc none $(pwd)/proc/
pwd is the directory inside rootfs.

Set the working directory to chroot with the following command:
user@suse: /build-chroot-dallas_branch-svn-734> ./suid-exec "./tsioe-utils.sh chroot_login"
The operation is now down inside the chroot:
devel@suse: /tsidist-dallas_branch- svn-734/build$

MySQL Bitbake command:
devel@suse: /tsidist-dallas_branch-svn-734 /build$ Bitbake MySQL

The result of the MySQL Bitbake process is an ipkg file located at the following directory: */build-chroot-dallas_branch-svn-734/rootfs/home/devel/tsidist-dallas_branch-svn-734/build/tmp-glibc-stable/ deploy/ipk*

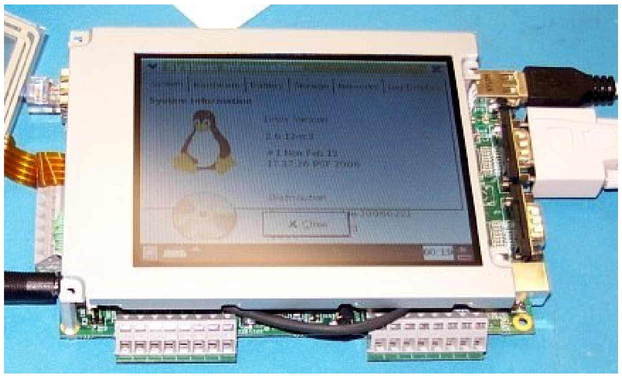


Figure 4: TPC-57

Cross Platforming using Qtopia Core

Cross platforming is used so that the MenuClient application that was developed using a Linux OS with a x86 processor can be used on a TPC-57 with a ARM processor. The following are the steps to cross compile using Qtopia Core:

Prepare the latest version Qtopia Core tarball file, for example:
qtopia-core-opensource-src-4.3.3.tar.gz

Extract the tarball file from above to the following /opt/ directory:
suse:/opt/ # tar xvzf qtopia-core-opensource-src-4.3.3.tar.gz

Rename the folder resulting from the extract to the desired folder name, for example:
QtopiaCore-4.3.3-arm

Make a folder inside /opt/cross/ with the desired fol-der name, for example: *tpc57*

Copy all the contents in the folder */home/radip/build-chroot-dallas_branch-svn-734/ rootfs/home/devel/tsidist-dallas_branch-svn-734/ build/ tmp-glibc-stable/cross/* into the directory */opt/cross/tpc57*. The contents should be the following folders: arm-Linux, bin, lib, libexec, share.

Go into the folder QtopiaCore-4.3.3-arm with the following command:
# cd /opt/QtopiaCore-4.3.3-arm
Then proceed to install Qtopia Core:
# ./configure -prefix /opt/cross/tpc57/QtopiaCore-4.3.3-arm/

The prefix functions here are used to determine the location of the Qtopia Core installation, where the default location is at /usr/local/Trolltech.

The reconfigure for a embedded ARM target:
# ./configure -help
# ./configure -prefix /opt/cross/tpc57/QtopiaCore-4.3.3-arm/ -qt-libjpeg -qt-libpng -qt-mouse-tslib -qt-kbd-usb -embedded arm

The options above are used according to the needs. In this case the options for JPEG library, PNG library, tslib driver for the mouse and touch screen, USB driver for the keyboard are needed. For further instructions on how to configure the Qtopia Core with the desired/needed options can be seen at *./configure -help*

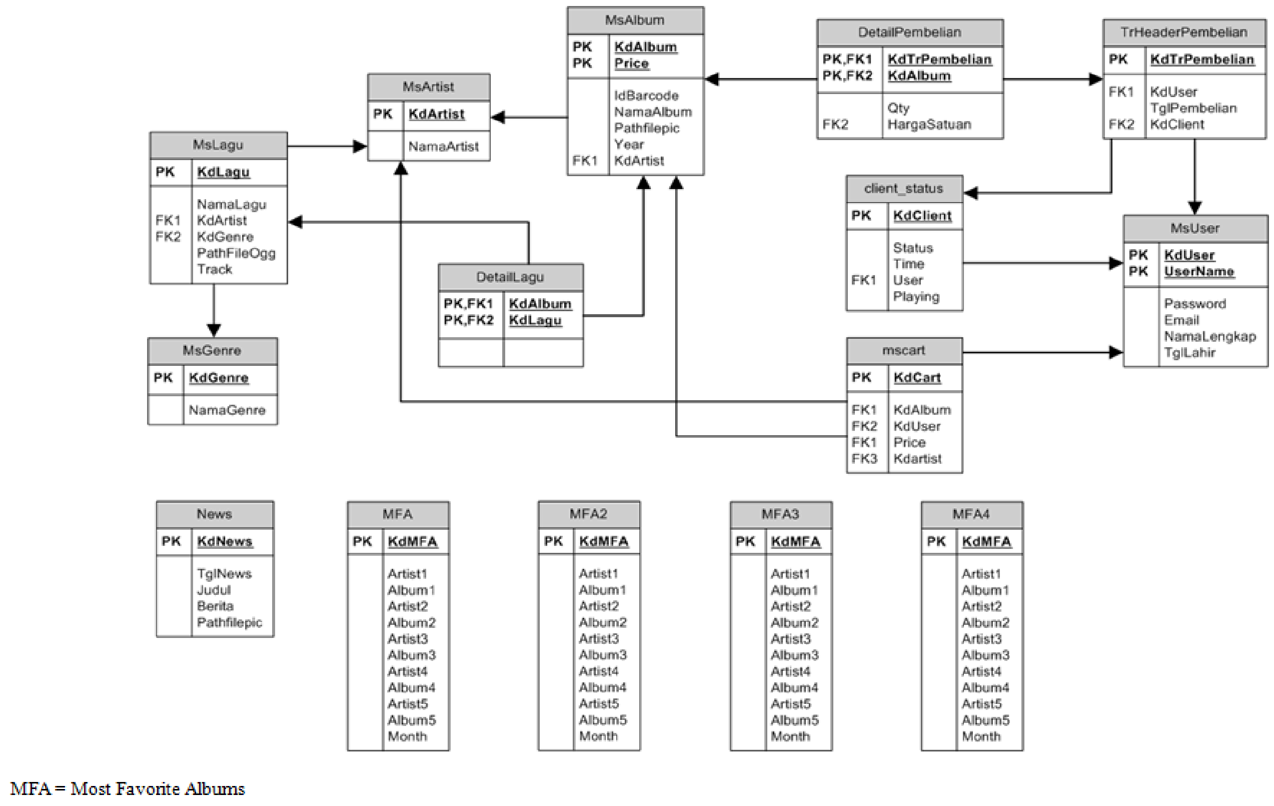


Figure 5: ERD of music palace system

Before beginning the make process, set the PATH with the following command: *# ARM_BOARD=tpc57*
# PATH=/opt/cross/tpc57/bin:/usr/lib/qt4/bin:$PATH
# export ARM_BOARD PATH
# QTDIR=/opt/cross/$ARM_BOARD/QtopiaCore-4.3.3-arm
# PATH=$QTDIR/bin:$PATH
# QMAKESPEC=$QTDIR/mkspecs/qws/Linux–arm-g++
# export QTDIR PATH QMAKESPEC

After the above process has been completed, the Qtopia Core program is successfully installed and the application code that was developed using Qt can be cross compiled. So the executable program that has been created can be made into binary that can be processed on a ARM processor. The commands for cross compiling are as follows:
(Qt application code folder)#qmake -project
(Qt application code folder)#qmake
(Qt application code folder)#make

After the above process has been completed, copy the library folder and the plug-ins that are located at the folder QtopiaCore-4.3.3-arm into the TPC-57 module with the same folder directories. In this case, */opt/cross/tpc57/QtopiaCore-4.3.3-arm*. Then create a .so format link file in the folder */opt/cross/tpc57/QtopiaCore-4.3.3-arm/lib* according to the needed library files. For example:
# ln -s libQtGui.so.4.3.3 libQtGui.so.4
Copy the resulting executable file into the TPC-57, and run the program with the optional -qws.

Creating A MySQL Plug-In Library In Qtopia Core

This system uses a MySQL program to get data from the client program on the TPC-57. So Qtopia Core can access the MySQL database. It would be needed to install a Qt MySQL plug-in library for the Qtopia Core.

This system uses a MySQL program to get data from the client application on the TPC-57. So Qtopia Core can access that MySQL database. The Qtopia Core program needs to have a Qt plug-in library for MySQL. The method accomplish this task is the same as the cross compile from the above explanation. With the work folder that is being used is */QtopiaCore-4.3.3-arm/src/plugins/sqldrivers/MySQL*. For example:
suse:/opt/QtopiaCore-4.3.3-arm/src/plugins/sqldrivers/MySQL#make

With that command, Qt will ask for the MySQL include file, with the MySQL include file already been cross-compiled using Bitbake beforehand. For example the location of the MySQL include file that has been cross-compiled using Bitbake can be found at the following folder:
/home/radip/build-chroot-dallas_branch-svn-734/rootfs/home/devel/tsidist-dallas_branch-svn-734/build/tmp-glibc-stable/work/arm-linux/MySQL-4.1.20-r0/MySQL-4.1.20/include/

The next step is copy all the MySQL include files that has a *.h format from the above folder into the folder */opt/QtopiaCore-4.3.3-arm/src/sql/drivers/MySQL*. Inside this

folder there is a file named *qsql_MySQL.h* that needs a header file named MySQL.h with the code
#include <MySQL.h>,
edit this code into
#include </opt/QtopiaCore-4.3.3-arm/src/sql/drivers/My SQL/MySQL.h>
to get the correct path.

Then, copy all the MySQL library into the Qtopia Core library, for example a library with a folder */home/radip /build-chroot-dallas_branch-svn-734/rootfs/home/ devel/tsidist-dallas_branch-svn-734/build/tmp-glib c-stable/work/arm-linux/MySQL-4.1.20-r0/install/ libMySQLclient/usr/lib copied into /opt/QtopiaCore-4.3.3-arm/lib*.

Finally, after the above steps, then do another make in the MySQL plug-in, with the following command:
suse:/opt/QtopiaCore-4.3.3-arm/src/plugins/sqldri-vers/ MySQL#make

ARM Processor Based TPC-57 Module As MP Client

TPC-57 is the album browser and the interface media for the MP Client application of the system. The TPC-57 has a touch-screen feature that will make it easy for users to operate and maximize the features that are available. This will make the album browser more attractive. A Linux based Operating System will be used on the TPC-57, will have the MP Client application for user interface.

This album browser will be connected to the database so it will expect for the MP Client to run smoothly even when there is interaction between the MP Manager and the MP Client. The process that happen here are the MP Client will request and send data to the database server. The data here is the data containing the information and location of a particular song. The requested *.ogg song file and file image will be placed on the TPC-57 on a folder created for NFS transfers.

MP Manager Development

The processes that happens at the MP Manager are processes that involves the database data; adding, changing, deleting a portion or the whole database; and to call an external program for CD ripping process. The database processes of this application will be connected to the database server. After the operator connects the application through the connection menu.

The databases that will be used are those for showing the album and song information, the registered user information, the transaction information, and the client status information.

For the processes of editing a song, an album, or a user data, the MP Manager operator will have to search the desired song, album, or user information. After finding the data that the operator wants to edit, the MP Manager will display an editing form that provides the selected song/album/user information to be edited or deleted. The process of deleting a song/album/user information on the MP Manager will empty the appropriate entry on the database table that is contained within the data-base server. The transactions information menu on the MP Manager is divided into two parts. The first part is the transactions history that is made by all the clients. The second part is the transactions history that was made by each of the clients. These transactions histories are recorded whenever a client user presses the icon "Buy" at the shopping cart menu on the MP Client.

The ripping process here is that the MP Manager application has a feature to call an external program with *.exe extensions, which the MP Manager operator need to locate to folder in which the external program is placed, for example *"/home/user_name/grip"*, then the MP Manager will call the selected Grip program that is located inside the folder *"/home/user_name/"*. The MP Manager is also capable of closing or terminating said called program.

Database Development

In this system, a database server is used. So the MP Client and MP Manager are able to interact with each other. In this way, the MP Client will have access to the database; to add users when they register, to request or to send the data for the News menu, Most Favorite Album menu, Album Searching menu, Shopping Cart menu, changing the status of the clients on the Client Information menu on the MP Manager, and to get the file path information of the files that will be played or displayed.

The database server is MySQL program query. The ERD of the database can be seen in the Figure 5 (At the end of this paper) [8].

Implementation

Hardware minimum specification

a) MusicPalace Manager and Database Server: Pentium III 1000 MHz, RAM 128 Mb, Hard disk 80 GB minimum, DVD-CD ROM, LAN CARD
b) Techsol TPC-57 Arm Processor for MusicPalace Client (`www.medallionsystem.com/products/News/ TechsolProductNews\_TPC57.html`)
c) Networking: LAN Cable and Switch
d) Input device for Client: USB Keyboard and USB Barcode Reader
e) Output device for Client: Headphone

Software specification

The specification of the Software that is used on this system is as follows:
a) PC Operating System using Linux SUSE 10.3
b) Qt 4 for developing the MP Client and MP Manager applications
c) MySQL for the database application
d) Qtopia Core 4.3.3
e) Bitbake 1.6.0
f) Ogg123 for playing song Ogg
g) Grip for CD Ripping

RESULT

MusicPalace Manager Experiment

MusicPalace manager main menu display

The display in Figure 6 is the interface that displays all the activities of all the clients in real time. There are 6 clients, each with its own status.
Each cliet has status for client standby, client ID, the song

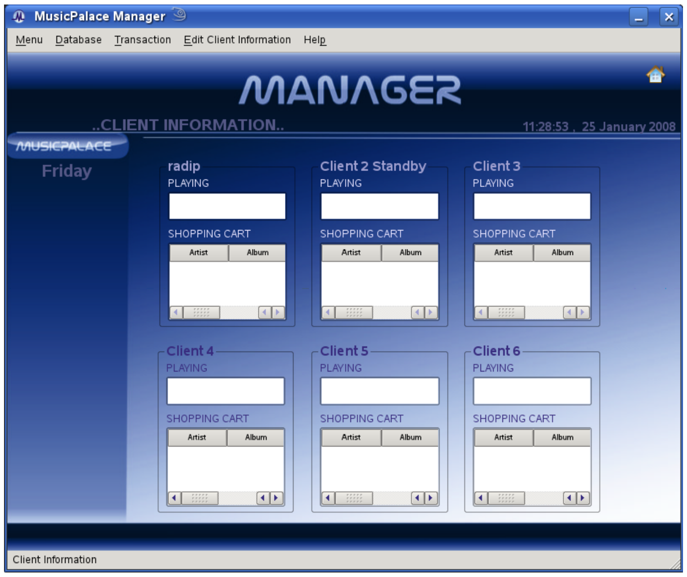


Figure 6: Client information

Table 1: Search query result table

NamaAlbum	NamaArtist	NamaLagu
Timeless Rawk!	BitterBallen	Pesta
Timeless Rawk!	BitterBallen	Telenovela
Cosmic Harmony	Captain Lord	Demi Cinta
Perjalanan	Ebiet G. Ade	Lagu Untuk Sebuah
Perjalanan	Ebiet G. Ade	Ada Sisa Sisa Suara
Perjalanan	Ebiet G. Ade	Asmara Suatu Ketika

Table 2: User information query result table

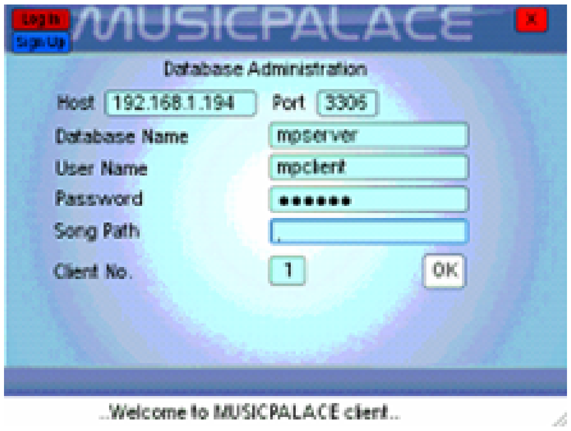
UserName	Password	NamaLengkap	TglLahir	Email
admin	admin			
dewo	dewo	Raden Dewantoro	22/Dec/1985	de_ballen@yahoo.com
kidal	kidal	Fadly Faizal	6/Mar/1986	kidalyoi@yahoo.com

Table 3: News query result table

judul	berita	Pathfilepic
Robert Plant on The Le	We could not fit every quote	/File/Audio/Image News/ledzeppe

Table 4: Most favourite album query result table

KdMFA	Artist1	Album1	Artist2	Album2	Artist3
1	Nidji	Breakthru	Krisdayanti	Mencintaimu	Samsons
2	Nidji	Breakthru	Krisdayanti	Mencintaimu	Samsons



Host 192.168.1.194 Port 3306

Database Name mpserver

User Name mpclient

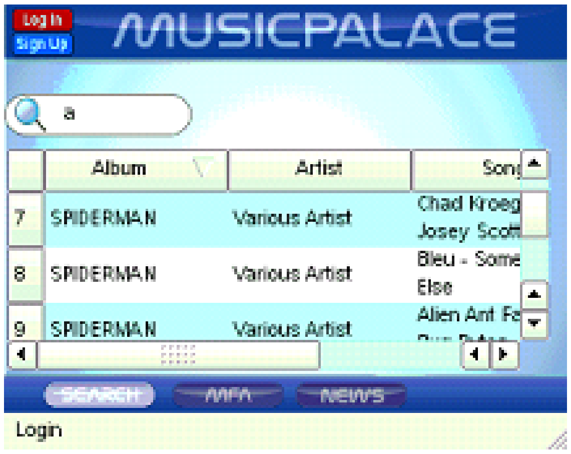
Password ••••••

Song Path

Client No. 1

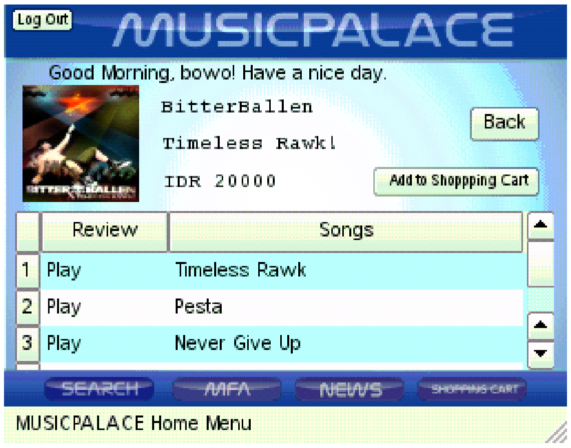
..Welcome to MUSICPALACE client..

Figure 7: MP client GUI display



	Album	Artist	Song
7	SPIDERMAN	Various Artist	Chad Kroeg Josey Scott
8	SPIDERMAN	Various Artist	Bleu - Some Else
9	SPIDERMAN	Various Artist	Alien Ant Fa

Figure 8: Searching process at MP client



Good Morning, bowo! Have a nice day.

BitterBallen

Timeless Rawk!

IDR 20000

	Review	Songs
1	Play	Timeless Rawk
2	Play	Pesta
3	Play	Never Give Up

Figure 9: Album information

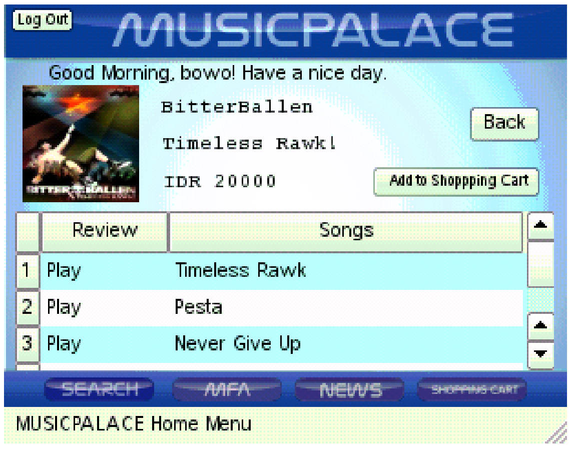


Figure 10: Playing telenovela

that is being played, and the shopping cart list for that particular client.

For example on client 1, the client is being used by the user "radip", on which the user is not hearing any song for the moment nor have any list in the shopping cart. Client 2 is on standby state, while clients 3, 4, 5, and 6 are not active.

Song information

Query to get song information from the database is (see Table 1):

```
SELECT IdBarcode, namalagu, namaartist,
       Namaalbum, year, track, namagenre,
       price, pathfilepic, pathfileogg
FROM msalbum m, mslagu u, detaillagu d,
     msartist a, msgenre r
WHERE  m.kdalbum = d.kdalbum
     AND u.kdlagu = d.kdlagu
     AND a.kdartist = u.kdartist
     AND u.kdgenre = r.kdgenre;
```

User information

Query to get user information from the database is (see Table 2):

```
SELECT username, password, namalengkap,
       tgllahir, email FROM msuser;
```

Transaction history

Query to get Transaction History from the database is :

```
SELECT username, namaartist, namaalbum,
       hargasatuan, qty, hargasatuan*qty
       as 'total harga', tglpembelian
FROM msuser U, msartist T, msalbum A,
     detailpembelian D,
     trheaderpembelian H
WHERE H.kduser = U.kdUser and A.kdartist
      = T.kdartist and H.kdtrpembelian
      = D.kdtrpembelian and D.kdalbum
      = A.kdAlbum;
```

Client information transaction

Query to get Client Information Transaction from the database is :

```
SELECT username, namaartist, namaalbum,
       Qty, price
FROM detailpembelian d, msartist a,
     msalbum m, trheaderpembelian p,
     msuser u, client_status c
WHERE d.kdalbum = m.kdalbum AND
      a.kdartist = m.kdartist AND
      d.kdtrpembelian = p.kdtrpembelian
      AND p.kduser = u.kduser
      AND c.kdclient = p.kdclient
      AND p.kdclient ='1';
```

Edit news

Query to insert news to the database is (see Table 3):

```
INSERT INTO news
   (judul, berita, Pathfilepic)
   VALUES ('%1','%2','%3');
```

Edit most favourite album

Query to insert Most Favourite Album to database is (see Table 4):

```
INSERT INTO mfa (Artist1, Artist2,
      Artist3, Artist4, Artist5,
      Album1, Album2, Album3,
      Album4, Album5 ,Month)
VALUES ('%1', '%2', '%3', '%4',
        '%5', '%6', '%7', '%8',
        '%9', '%10', '%11');
```

MusicPalace Client Experiment

MusicPalace client main menu display is shown in Figure 7.

Signup menu

Query to send to database when a user signs up is :

```
INSERT INTO msuser (UserName,
       Password, Email, NamaLengkap,
       TglLahir)
VALUES ('%1','%2','%3','%4','%5');
```

Log in menu

Query to send to database when a user log in is :

```
SELECT UserName,Password
FROM msuser WHERE UserName
     = '%1'").arg(usernamelogin))
```

Main menu

Searching Process:
Type in the desired keyword on the menu, then on the available table it will begin searching the database based on album name, artist name, and song name, according to the given keyword. The result can be seen on the Album Information screen that contains a picture of the cover of the chosen album, the name of the album, the artist, and the price of the album. For example if the given keyword is "a", then the result table will show all the albums, artists, and songs that contains the keyword "a" (see Figure 8).

Query on search process is :

```
SELECT NamaAlbum, NamaArtist,
       NamaLagu
FROM msartist t, msalbum m,
     mslagu u
WHERE NamaAlbum LIKE '%%1%'
   AND m.kdartist = t.kdartist
   AND u.kdartist = m.kdartist
   OR NamaArtist LIKE '%%1%'
   AND t.kdartist = m.kdartist
   AND t.kdartist = u.kdartist
   OR NamaLagu LIKE '%%1%'
   AND u.kdartist = t.kdartist
   AND m.kdartist =
   u.kdartist").arg(searchquery))
```

Album Information is shown in Figure 9. On the Album Information screen, the user can listen to the songs on the selected album by pressing the "Play" button besides the desired song, and to stop playing the song, the user can press the "Stop" button. For example a user presses "Play" to listen to a song titled "Telenovela".

Playing a song is shown in Figure 10. If the user have logged in beforehand, then he/she can use the "Add To Shopping Cart" facility on the Album Information screen, to order the selected album. The selected album will then be added to the client's Shopping Cart list.

Most Favorite's Album (MFA) menu

Query to get Most Favourite Album from database:

```
SELECT Artist1, Artist2, Artist3,
       Artist4, Artist5, Album1,
       Album2, Album3, Album4,
       Album5
FROM mfa
```

News menu

Query to get News from the database:

```
SELECT judul, berita,
       Pathfilepic
FROM news
```

CONCLUSION

Based on the results, we can conclude four conclusions. Firstly, database query processes by the MusicPalace Manager and MusicPalace Client application are working properly. Secondly, the display on the MusicPalace Client cannot be the desired quality of color, because the TPC-57 can only support 8-bits of color. Thirdly, the initial settings of the TPC-57 still needs to be done manually. Fourthly, playing and stopping of songs on preview is working properly.

REFERENCES

[1] Anonym: *Medallion Touch Panel Computer*. `http://www.medallionsystem.com/products/News/TechsolProductNews_TPC57.html` (2007)

[2] Anonym: *Qt 4.0 Tutorial*. `http://trolltech.de/products/qtopia/learnmore/whitepapers/qtopia_core_whitepaper_us_letter.pdf` (2005)

[3] Anonym: *Qtopia Core 4.1. 2006*. `http:trolltech.de/products/qtopia/learnmore/whitepapers/qtopia_core_whitepaper_us_letter.pdf` (2006)

[4] Blanchette, J., Jasmin: *C++ GUI Programming with Qt 4*. Prentice Hall Inc., Massachusetts (An Introduction To Design Patterns In C++ with Qt42006)

[5] Ezust, A.: *An Introduction To Design Patterns In C++ with Qt4*. Prentice Hall Inc., Massachusetts (2007)

[6] Liawatimena, S., Wiedjaja, S., Linggajari, S.: *Soft Embedded Development*. BiNus Computer Engineering (2006)

[7] Larson, C., Blundell, P.: *BitBake User Manual*. `http:bitbake.berlios.de/manual` (2006)

[8] Nugroho, B.: *Administrasi Database MySQL pada Server Linux dan Windows*. Graha Ilmu, Yogyakarta (2005)