Gumstix-based Touchscreen Piano and MP3 Player

Team PIN
Liangxiao Xin xlx@bu.edu
Rana Alrabeh ralrabeh@bu.edu
Riya Thakkar riyathak@bu.edu

Abstract— *For our EC535 final project, we Implemented a gumstix based, touchscreen piano and MP3 player housed in a 3D printed plastic cover designed to look like a miniature grand piano. The final product connects to any bluetooth enabled-device to play the piano keys or MP3 music loaded on the gumstix's microSD card. The piano sound sequence can be saved and played back. The MP3 player includes standard functionality: play, stop, skip/previous, playlist, artist info and volume controls.*

1. Introduction

The project is to simulate a piano keyboard on an LCD touchscreen that connects to any Bluetooth sound enabled device to help everyone from kids to adults enhance their skills in piano and have fun. It helps users learn the piano in an easy, portable and cost effective way. The song that user played gets stored in micro SD card. Thus, it can be played again and again to correct erroneous musical notes. Users can choose a song from their personal playlist to play via bluetooth. This helps users emulate actual songs and helps user get benefits of both a Piano and an MP3 Player. In our design, the gumstix is the central module, it runs the LCD touch screen interface. The touchscreen communicates back to the gumstix to control the bluetooth and audio functionality. In our final demo, we were able to achieve the desired features. One setback, however, is that the piano keys can only be played once at a time, which makes our piano slower than a real piano.

2. Design Flow

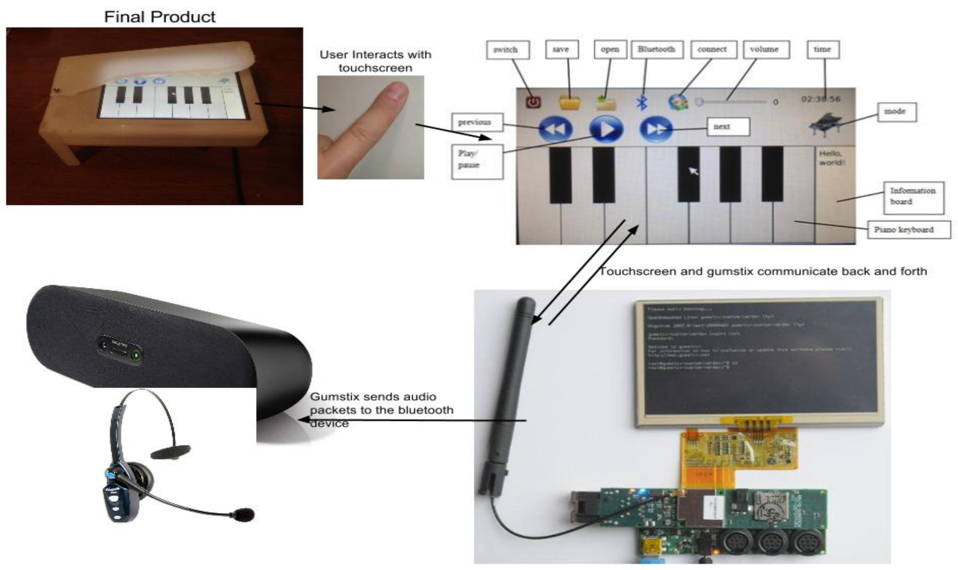
Final Product
User Interacts with touchscreen
switch
save
open
Bluetooth
connect
volume
time
previous
Play/pause
next
mode
Hello, world!
Information board
Piano keyboard
Touchscreen and gumstix communicate back and forth
Gumstix sends audio packets to the bluetooth device

Figure 1: Flow and design of the product

Module	Task Name	Description	Lead Member
Touchscreen	QT Module	The software for QT to control the touchscreen. Designing and implementing the interface, implementing event detection for button presses.	Liangxiao
	Touchscreen Interface	All hardware related to the touch screen, installing and running QT on the gumstix.	Liangxiao
Bluetooth	Bluetooth interface	All hardware related to the bluetooth including installing, running and debugging bluetooth.	Riya
	Finding suitable gumstix	Testing ~8 gumstix for all required modules (bluetooth audio, SD card and ethernet to install all the software using ipkg).	Riya
Audio	Audio Module	Installing all audio packages needed, testing for audio functionality and dealing with speakers.	Rana
	Writing functions for QT modules to call	The code for audio functionality, functions for Bluetooth setup, scanning, connection, audio playlist generation, play, skip, artist info ..etc.	Rana
	More features	Stop button and volume generation.	Liangxiao
	Multiprocessing	Making MP3 playing a multiprocess as to not block the touchscreen while playing a song.	Rana
	Piano tune Speed optimization	A big problem we faced was the the piano notes were slow in playing. We timed and tested various functionality in Aplay to achieve the optimal performance. (see top of communicate.cpp for list of flags used).	Riya
	Finding suitable Piano sounds	that can be played by madplay.	Riya
Integration & Debugging	Interfacing	Final testing, debugging and interfacing	All
Cover	Piano Cover	3D printed a cover to house the gumstix, touchscreen and connections and make it look like a miniature grand piano	Liangxiao

3. Project Details

There are three components in our project, QT module, Audio module and Bluetooth module. Each module was worked separately then integrated them together.

a. **QT Module**

A comfortable and convenient interface will not only improve user experience, but attract more customers to those products as well. Thus, according to recent popularity, a touchscreen which can be used both as display and operation equipment was used as the only human-computer interface in the project.

Before using touchscreen, several environmental parameters in Gumstix has to be set. These steps can be followed by the project slides and a script file was recommended so that these parameters will be set in a easier way. If it was the first time to run a touchscreen on Gumstix, the command “ts_calibrate” may be of help if the touchscreen cannot be pressed since Gumstix need to know there the edges of the display are.

The software we used for the graphical interface design is QT. QT is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI). In a normal operation system environment, QT is quite convenient to learn and use. However, in terms of the requirement of the project, QT need to be used in embedded system which is much more complicated than that in other systems such as Windows.

The first step to use QT in the project is to choose the property version of QT. There are two versions of QT available in the computers of lab, QT3 and QT4. After logging into the engineering network, the command
“source /ad/eng/courses/ec/ec535/mixed/bashrc_ec535”
needs be typed so that the library of QT4 will be linked. Otherwise, QT3 is the default version. The version can be also checked in the Makefile after the QT project is created. We used is QT4 and the following introduction is based on QT4.

Unlike in the ordinary system environment, some manual modification is required before compiling successfully. Following the tutorial given in the project slides, the command “qmake” cannot be executed successfully after “qmake -project”. It is because a line “MAKEFILE_GENERATOR=UNIX” was missing in the end of .pro file. Then Makefile will be created automatically by QT. In order to make the QT program run in Gumstix, we still have to modify the Makefile followed by the project slides. Then the code can be compiled successfully if the macro variable “Q_OBJECT” was commented. Nonetheless, the execute file theretofore is only able to achieve part of the functions in QT because of the missing “Q_OBJECT”. To make this important macro work, a “moc” component need to be added into Makefile. It seems the easiest method to obtain a Makefile with “moc” is using QTcreator though we added “moc” manually.

To run QT on Gumstix, we also need to add the relative QT library into Gumstix. An SD card is a recommendable route to store QT library and the reason will be discussed later. Then link the library to /lib or /usr/lib of Gumstix with the command “ln -s”. These links are also recommended to be written in the script file since the library may need be linked again.

After finishing all the preparation above, we can start to add the button and functions we

need. In our project, Push Button is the most widely used. A Push Button with icon can be defined as,
QPushButton *button_save = new QPushButton(this); //define a new button and its type
button_save->setGeometry(60, 10, 30, 30); // set the size and location of the button
button_save->setIcon(QIcon(QPixmap("/media/mmcblk0/Save.png")));
// load the icon into button
button_save->setIconSize(QSize(30, 30)); //set the size of icon
button_save->setStyleSheet("QPushButton{border: none;outline: none;}");

Since the piano keyboard icon is a whole picture that cannot be separated, a feasible method is to design several transparent Push Buttons above the piano keyboard icon. The function switch such as mode switch between Piano and MP3 can be presented by changing the icon of Push Button as well as the parameters inside the program. Or the other way is to define two buttons with “show()” and “hide()” control which can be used in all kind of QT buttons.

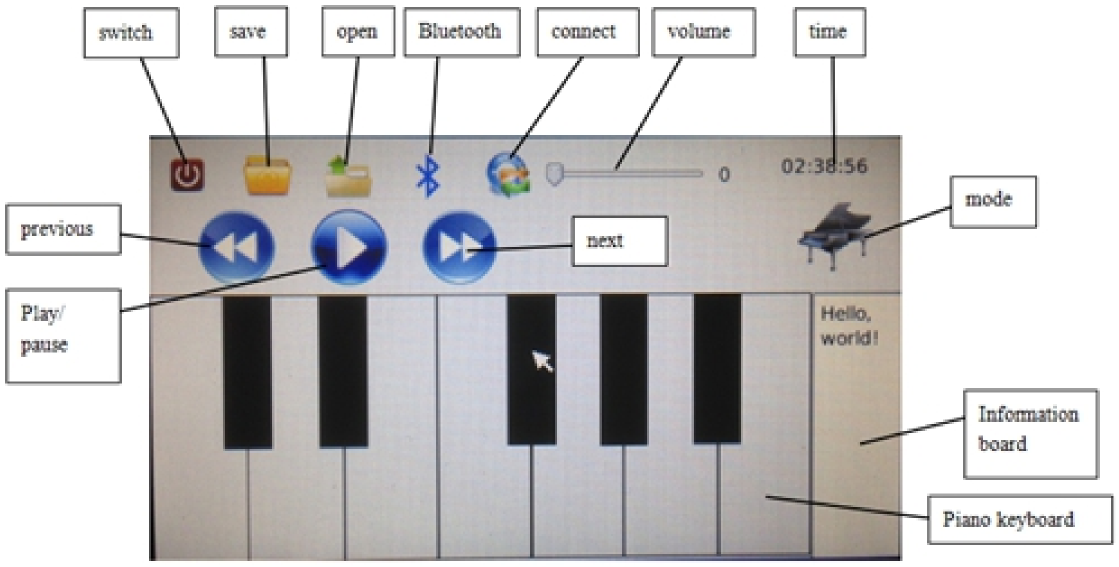

Figure 2. Main Interface

Additionally, WidgetList was used for listing the BlueTooth devices and MP3 songs. Timer was used for time display as well as screen refresh. Qslider can be used for volume control. WidgetText can print the information on the screen.

When finishing all the buttons we need in our project, the following work is to connect the buttons to the functions need to be called. A slot, the function in QT, was able to be called by the signal produced by the button. For example,
connect(button_quit, SIGNAL(clicked()), qApp, SLOT(quit()));
By calling a slot, it became a easy job to call the external functions of other modules in the slot.

Finally, all the software codes were integrated into one executable document. However, until now, the Gumstix had to be connected to the computer to run the program. The key to make the program boot up automatically when we turn on the Gumstix is the script file we mentioned above. If the script file was renamed as “S98XX .sh” and put in the path of “/etc/ ”, it will execute before we entered the login information. But before doing such function, we need to ensure that there some methods can be used to quit the process. One way, as mentioned before, is to store the QT library in SD card. In that situation, if SD card was not inserted in Gumstix, the program cannot find the shared library and quit. Another way is to set a QT quit button so that we can quit the program via touchscreen. If there is no method to quit the process, then Gumstix can never be logged in.

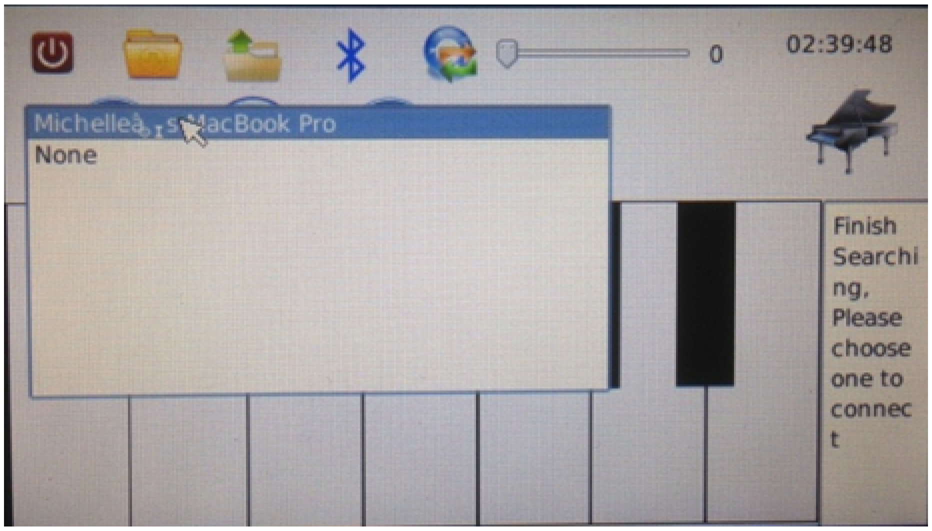

Figure 3. Bluetooth Search Example

b. **Bluetooth Module**

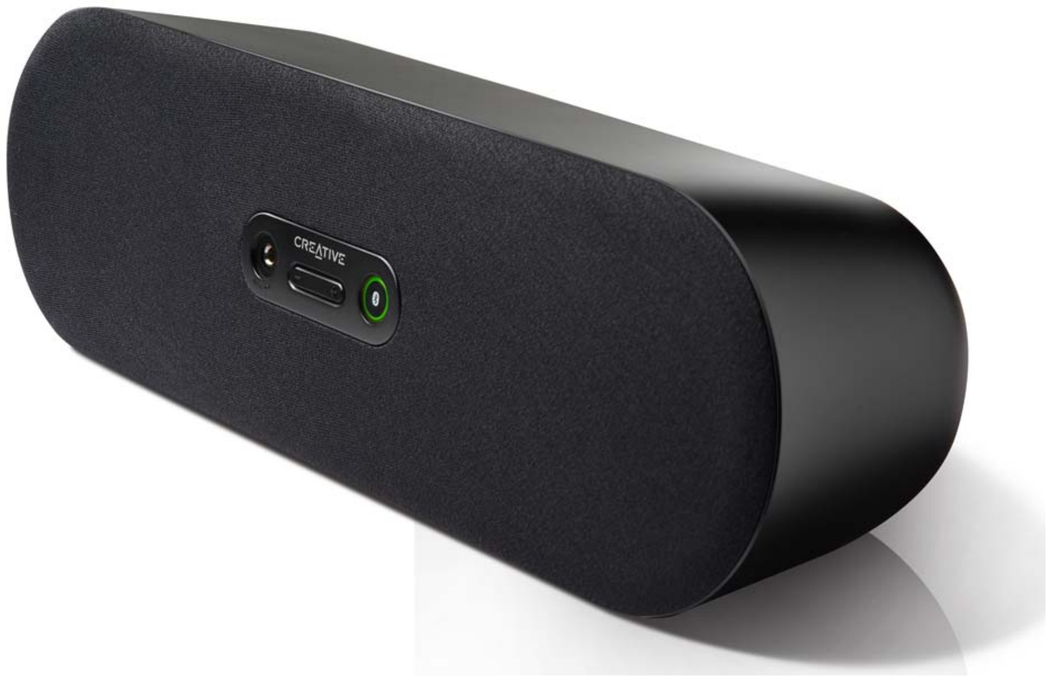

Figure 4: Bluetooth speakers used in this project

Bluetooth is a wireless technology standard used for exchanging data over short distances. Bluetooth uses short wavelength microwave transmissions in the ISM band from 2400-2480 MHz. Here, the data to be transferred are the audio signals from the gumstix board to the bluetooth speaker.

For bluetooth connection, Bluez-utils was installed using ipkg command on gumstix board. The setting up of bluetooth connection required start and stop of bluetooth.
First task of bluetooth module was to search for the devices present in the bluetooth range. The command used was hcitool scan. The result of it was then passed to the text file, from which the address of the device to be connected was parsed. The device address was given to ALSA (Advanced Linux Sound Architecture). ALSA did connection to the bluetooth speaker.

As shown in the figure above, Creative D80 speaker was used for playing piano sounds and mp3 songs.
After the speaker is powered ON LED status indicator will list steady. To allow pairing to other Bluetooth devices, press and hold the Connect button for 3 seconds. Release when 2 beep tones are heard with the LED status indicator blinking rapidly.

Additionally we enabled the bluetooth audio service by editing configuration file /etc/bluetooth/audio.service. Sample file is shown as follows

```
[Bluetooth Service]
Identifier=audio
Name=Audio service
Description=Bluetooth Audio service
Autostart=true
```

For all the tasks mentioned above two buttons on LCD are provided for SCAN and CONNECT :
1) SCAN: button is used for scanning all bluetooth enabled devices and listing those.
2) CONNECT: To connect to a particular device from the list.

c. Audio Module

For the audio backend, separate functions were written for each button or task. When a button is pressed on the touchscreen the QT module calls the corresponding function from the bluetooth, piano or mp3 player backend. One of the first problems encountered was that the libraries we elected to use: bluez, madplay, and ALSA were written in C while QT is written in C++. This made a problem in interfacing the the touch screen with the music player backend. To solve this issue, there were several options, from which we elected to simply write the backend in C++ and use system calls when appropriate to control the modules from the command line. Each button on the touchscreen interface calls a backend button that handles the work and provides the results needed as appropriate. For example, the 'scan' button makes a system call for the gumstix to scan for devices via the commandline and saves the list to a file. The function then opens the file, parses the strings and updates the appropriate device files with the device information to enable a bluetooth connection to that device. Most of the functions work in a similar manner; the playlist function makes a system call to find and list all MP3 files on the gumstix. It then displays the song list and saves the addresses of them internally to be played back later. The MP3 player, on the other hand, works a little differently. Since we discovered that it takes control of the process, thereby blocking the touchscreen from running, we made a multiprocess application that launches a new thread from the main thread each time an MP3 song is played. A nice feature of madplay is that it provides all the song info: title, artist name, album, year genre...etc. We therefore made a function that reads this information and displays it on the screen while the song is playing. For a full list of functions, please refer back to the source code.

d. **3D Printer Piano Cover**
Our topic of the project is digital piano. Thus we designed a piano cover specially for our project. The cover was designed by Protel and printed by 3D printer.

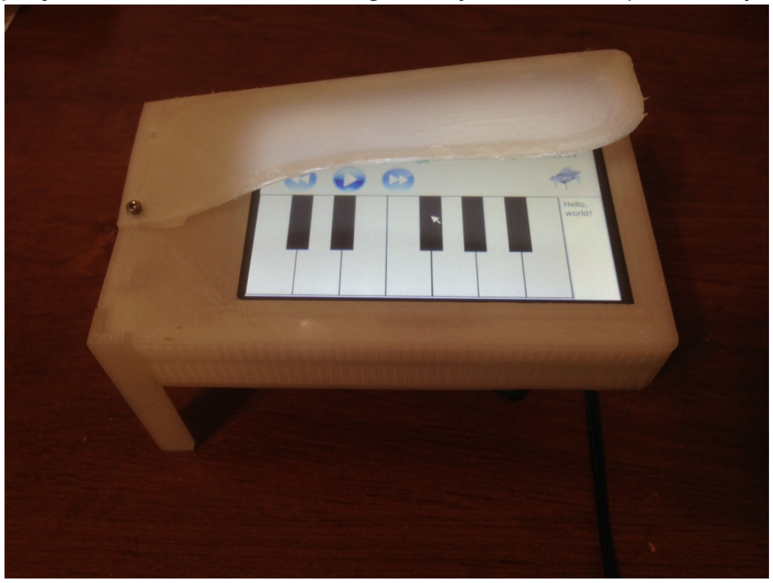

Figure 3. Final Project

In order to make the product easy to debug, several important ports was considered during the design. Eventually, power, Bluetooth, serial and Ethernet were allowed to be external connection.

4. Summary

Most of the function in our plan were achieved successfully. The project can be switched between piano mode and MP3 mode. In piano mode, we simulate the real piano tone and play the piano music by pressing the keyboard on touchscreen. Saving and Playback function makes it possible to listen the music played before. We also achieved the basic functions of a MP3 player. By switching to MP3 mode, songs can be played as well as stopped once choose a song on the list. We can skip the songs to the previous or next one. Volume adjustment is also added into the project. The information dialog window helped to make the manipulation easier. Besides, Bluetooth device can be scanned, chosen and connected optionally. Eventually, we reached the goal of our project and obtain a good performance. The future job will be to improve the performance of the project. We still need to find the way to decline the delay of the reaction of piano keyboard.

References

[1]BluetoothUser, https://wiki.debian.org/BluetoothUser
[2]aplay, http://linux.die.net/man/1/aplay
[3]How to bluetooth, http://wiki.gumstix.org/index.php?title=Category:How_to_-_bluetooth
[4]madplay, http://www.underbit.com/products/mad
[5]QT4, http://www.zetcode.com/gui/qt4/