Department of Computer Science and Engineering
The University of Texas at Arlington

Detailed Design Specification

Team: AudioTEK

Project: dRadio System

Team Members:
José Fleites
Carter Ho
David Lopez
José Ortiz
Binkal Tandukar
Table of Contents

Table of Contents ..................................................................................................................................................1
List of Figures ......................................................................................................................................................3
List of Tables .......................................................................................................................................................4
1 Introduction ..........................................................................................................................................................1
  1.1 Document Overview .................................................................................................................................1
  1.2 Product Overview .......................................................................................................................................1
  1.3 Project Scope ..............................................................................................................................................2
  1.4 Definitions and Acronyms .........................................................................................................................2
2 Architecture Overview ......................................................................................................................................4
  2.1 The dRadio System Architecture ............................................................................................................4
  2.2 Module Decomposition Chart ................................................................................................................6
  2.3 Module Producer-Consumer Matrix ..........................................................................................................7
  2.4 Module Functional Description ................................................................................................................8
3 User Interface Layer ......................................................................................................................................10
  3.1 User Interface Controller .........................................................................................................................10
  3.2 Graphics ....................................................................................................................................................12
4 Processing Layer ..........................................................................................................................................13
  4.1 Radio Input Handler ..................................................................................................................................13
  4.2 System Logic Controller’s Subsystem .....................................................................................................14
5 Data Layer Component Design .....................................................................................................................19
  5.1 Data Controller ..........................................................................................................................................19
  5.2 Audio Player ............................................................................................................................................21
6 Hardware Components ...................................................................................................................................22
  6.1 Overview ..................................................................................................................................................22
  6.2 Intel NM10 Mini ITX ..................................................................................................................................22
  6.3 Pico-PSU-120WI-25v ................................................................................................................................23
  6.4 Hard Drive ...............................................................................................................................................24
  6.5 Touch screen ............................................................................................................................................24
7 Requirements Traceability Matrix ................................................................................................................26
7.1 Purpose ........................................................................................................................................... 26

8 Quality Assurance .......................................................................................................................... 27
  8.1 Test Plans and Procedures ........................................................................................................... 27
  8.2 Module/Unit Test ......................................................................................................................... 27
  8.3 Component Test .......................................................................................................................... 28
  8.4 Integration Test ............................................................................................................................ 29
  8.5 System Verification Test ............................................................................................................... 29

9 Acceptance Plan ............................................................................................................................. 30
  9.1 Overview ..................................................................................................................................... 30
  9.2 Packaging and Installation .......................................................................................................... 30
  9.3 Acceptance Testing ...................................................................................................................... 30
  9.4 Acceptance Criteria ..................................................................................................................... 30

10 Appendices ..................................................................................................................................... 32
  10.1 Sound Exchange ....................................................................................................................... 32
List of Figures

Figure 1-1 Product Overview........................................................................................................... 2
Figure 2-1 Architecture Overview................................................................................................. 4
Figure 2-2 Module Decomposition................................................................................................. 6
Figure 3-1 User Interface Layer..................................................................................................... 10
Figure 4-1 Processing Layer .......................................................................................................... 13
Figure 5-1 Data layer ....................................................................................................................... 19
Figure 10-1 Sound Exchange (SoX) Interaction............................................................................ 32
List of Tables

Table 2-1 Data Flow ............................................................................................................ 7
Table 2-2 Producer Consumer Matrix .................................................................................. 8
Table 5-1 Requirements Matrix ............................................................................................ 26
Table 9-1 Acceptance Criterion ............................................................................................ 31
1 Introduction

1.1 Document Overview
The Detailed Design Specification (DDS) will detail the design of the dRadio system in a lower level. The subsystems and layers defined in the Architecture Design Specification (ADS) will be further broken down into modules. Details of each modules relationship, the input/output, and pseudo codes will be documented in the DDS along with the hardware that Team AudioTEK expects to use.

1.2 Product Overview
The following is the “Product Overview” as defined in Section 1.1 “Purpose and Use” of the System Requirements Specification (SRS).

The dRadio is a recording device for the car audio system (car stereo and speakers) that enables the user to use the playback functionalities, such as rewind, play, pause, and fast-forward, live radio. This is what we call “Radio Mode.”

In addition, when using our system, the user will also have the ability to record live radio. If the user decides to record the content that the car radio is playing, then this recording will be stored in the dRadio’s memory system. The user will be able to interact with recordings and use the playback options previously mentioned to manipulate the recordings. This is what we call “Time Capsule Mode.” Furthermore, the user will also have the ability to stop listening to a recording and switch back to live radio at the touch of a button.

Figure 1-1 provides a graphical overview of how the dRadio system is integrated into the car audio system.
1.3 Project Scope

The dRadio project will start from the beginning of the Spring 2012 semester and conclude at the end of the Summer 2012 semester. This project is to be intended for educational purposes and application of previous and new knowledge learnt through the course of the college journey.

The project is split into two major phases, Project Planning and Product Implementation. During Project Planning, the system requirement specification for the dRadio project is to be drafted, decided on and finally defined. Along with the requirements, individual and team status reports are to be submitted biweekly on opposite weeks. This allows for accounting for the work done, and be recorded to give an overview of how the project is progressing. During Product Implementation, the Architectural Design Specification, Detailed Design Specification, Test Plan and Implementation of the project shall be completed. Team AudioTEK is made up of five members who are majoring in computer science, software engineering and computer engineering. Team AudioTEK will follow the hybrid staged delivery model for the duration of this project.

1.4 Definitions and Acronyms

ADS: Architectural Design Specification  
DDS: Detail Design Specification  
OS: Operating System  
SRS: System Requirements Specification  
UI: User Interface  
GUI: Graphical User Interface
UTA: The University of Texas at Arlington
SRS: System Requirements Specification
SoX: Sound Exchange, a sound processing program
2 Architecture Overview

2.1 The dRadio System Architecture

Figure 2-1 shows the high level architecture of the dRadio. The dRadio Architecture consists of three major layers:

1. UI Layer
2. Processing Layer
3. Data Layer
2.1.1 *User Interface Layer*

The UI Layer is the main means of interaction between the user and the system. The UI Layer will handle receiving user input and displaying relevant information to the user. This layer mainly interacts with the Processing Layer.

2.1.2 *Processing Layer*

The Processing Layer serves as the middleman between the UI Layer and the Data Layer. The Processing Layer handles the input from the car radio, and interacts with both the UI and the Data Layer.

2.1.3 *Data Layer*

The Data Layer is where all audio files will be stored and played.

2.1.4 *Operating System*

The operating system plays a significant role in the proper functioning of the dRadio system. dRadio will use Microsoft Windows XP Professional as the operating system. This operating system is compatible with the touch screen drivers that dRadio will use. In addition, by default, Windows XP provides the Microsoft Wave Mapper device, which means the direct use of the system’s default audio devices. The defaults can be controlled via the Windows Control Panel.

In the UI Layer, the operating system gets user inputs via the touch screen device drivers and passes the appropriate mouse event to the Event Handler module of the User Interface Controller subsystem. In addition, the graphics module uses the operating system graphics drivers to direct the touch screen to display the user interface components.

In the Processing Layer, the operating system gets sound information from the radio device and passes that information to the Record module. After the audio file has been encoded by the Record module, it is then stored and saved in the hard drive by the operating system for future retrieval. The operating system is also used to retrieve the names of the previously saved recordings from the hard drive as well as to increase or decrease the output volume of the dRadio system.

In the Data Layer, the Player module uses the operating system to retrieve audio files from the storage media. The operating system also uses the sound card drivers to play the respective audio files through the car speakers.

2.1.5 *API*

Various API’s are used in the dRadio system. The primary API is Sound Exchange (SoX). SoX is a cross-platform (Windows, Linux, MacOS X, etc.) command line utility that can convert various formats of computer audio files into other formats. It can also apply various effects to these sound files, and, as an added bonus, SoX can play and record audio files on most
platforms. A brief overview on SoX is provided on Section 10. Another API used is win32 API; this API allows SoX to directly access the native device drivers.

### 2.2 Module Decomposition Chart

Figure 2-2 further decomposes figure 2-1 into individual modules and defines the data flows between modules.
2.3 Module Producer-Consumer Matrix

The Module Producer – Consumer Matrix shows the data flows between modules and visually describe where these data flows are produced and where they are consumed.

<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Data Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User input to the Event Handler</td>
</tr>
<tr>
<td>2</td>
<td>Graphics is updated based on the user's input</td>
</tr>
<tr>
<td>3</td>
<td>Appropriate GUI components are displayed to the user</td>
</tr>
<tr>
<td>4</td>
<td>User input in a form of a String and an integer</td>
</tr>
<tr>
<td>5</td>
<td>OS - Sound Driver produces raw audio file to be saved and compressed by Record</td>
</tr>
<tr>
<td>6</td>
<td>Record produces compressed audio files and saves onto hard drive</td>
</tr>
<tr>
<td>7</td>
<td>Invokes Record and passes a string argument representing the folder where the audio files must be saved.</td>
</tr>
<tr>
<td>8</td>
<td>Integer value representing the amount of time a button was pressed</td>
</tr>
<tr>
<td>9</td>
<td>Logic calls play function</td>
</tr>
<tr>
<td>10</td>
<td>Logic calls Pause function</td>
</tr>
<tr>
<td>11</td>
<td>Logic calls fastforward function</td>
</tr>
<tr>
<td>12</td>
<td>Logic calls live function</td>
</tr>
<tr>
<td>13</td>
<td>Logic calls volume function</td>
</tr>
<tr>
<td>14</td>
<td>Logic call browse function</td>
</tr>
<tr>
<td>15</td>
<td>Logic calls makeDir</td>
</tr>
<tr>
<td>16</td>
<td>makeDir creates a folder and saves it in the hard drive</td>
</tr>
<tr>
<td>17</td>
<td>Rewind updates Player Controller with new currentFile location</td>
</tr>
<tr>
<td>18</td>
<td>Play call Player Controller</td>
</tr>
<tr>
<td>19</td>
<td>Pause interrupts player controller</td>
</tr>
<tr>
<td>20</td>
<td>Fast Forward updates player controller with new currentFile location</td>
</tr>
<tr>
<td>21</td>
<td>Live updates player controller with newest recording</td>
</tr>
<tr>
<td>22</td>
<td>increase volume in sound card</td>
</tr>
<tr>
<td>23</td>
<td>Browse retrieves a list of files from hard drive</td>
</tr>
<tr>
<td>24</td>
<td>player controller calls wait function</td>
</tr>
<tr>
<td>25</td>
<td>player controller passes file name and folder name to player</td>
</tr>
<tr>
<td>26</td>
<td>player retrieves audio file from folder to be played</td>
</tr>
<tr>
<td>27</td>
<td>player plays audio file through sound card and speakers</td>
</tr>
</tbody>
</table>
### 2.4 Module Functional Description

#### 2.4.1 Event Handler

The Event Handler will receive input from the user in the User Interface Layer. Event handler converts user input into strings to be used by the system.

#### 2.4.2 Graphics

Creates and displays graphics to the user using Java’s Swing and the OS’s graphic driver.

#### 2.4.3 Record

Record module uses the program SoX to capture audio from the sound driver and compresses and saves it. Files will be saved as a wav file.

---

**Table 2-2 Producer Consumer Matrix**

<table>
<thead>
<tr>
<th>Producer</th>
<th>User Input</th>
<th>Event Handler</th>
<th>Graphics</th>
<th>Display</th>
<th>OS - Sound Driver</th>
<th>Record</th>
<th>Hard Drive</th>
<th>Logic</th>
<th>Pause</th>
<th>Makedir</th>
<th>Fast Forward</th>
<th>Rewind</th>
<th>Live</th>
<th>Player Controller</th>
<th>Wait</th>
<th>Player</th>
<th>Speaker</th>
<th>play</th>
<th>volume</th>
<th>browse</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Input</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event Handler</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphics</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS - Sound Driver</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>15</td>
<td>11</td>
<td>8</td>
<td>12</td>
<td></td>
<td>19</td>
<td>13</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hard Drive</td>
<td>16</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pause</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Makedir</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast Forward</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rewind</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Live</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Player Controller</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Player</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speaker</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>play</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>volume</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>browse</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
2.4.4 **Logic**

The logic module receives and interprets user commands from the Event Handler. It is responsible for killing and restarting processes if needed and is in charge of saving the current state of the system.

2.4.5 **Pause**

Pause module will be responsible for stopping the Player module’s process and updating the logic module with the current state of the system.

2.4.6 **Makedir**

Makedir creates a folder where the audio files will be saved. Folders will be labeled with the date and time of creation.

2.4.7 **Fast Forward**

Calculates the amount of time the Fast Forward button is being pressed to determine how many files to go back and play.

2.4.8 **Rewind**

Calculates the amount of time the Rewind button is being pressed to determine how many files to go back and play.

2.4.9 **Live**

This module is responsible for restarting the recording process after resetting the current state of the system to its default value.

2.4.10 **Player Controller**

Player controller sends file to be played to the Player module.

2.4.11 **Player**

Given a path to an audio file, the Player module is responsible for playing the audio file with the help of the OS sound driver and the program SoX.

2.4.12 **Browse**

Browse is used to retrieve a list of folder containing previous recording and display it to the user.

2.4.13 **Play**

This module is used to resume the playing process after it has been paused. It restores all value that was saved including the file that was playing and the folder it was in.
3 User Interface Layer

The User Interface Layer allows communication and interaction between the user and the system. This layer contains two subsystems, the User Interface Controller and Graphics. Each subsystem contains one module and will be described below.

![User Interface Layer Diagram]

Figure 3-1 User Interface Layer

3.1 User Interface Controller

3.1.1 Event Handler

3.1.1.1 Module Description
The Event Handler module contains all possible buttons used by the dRadio. Because a touch screen will be used, on touch, each button reacts the same way as a mouse click.

3.1.1.2 Interfaces
Each button defined in this module will receive a MouseUp event.

3.1.1.3 Physical Data Structure
A total of 8 buttons
Play/Pause
Rewind
Fast-forward
Record
Live
Volume up
Volume down
Browse up
Browse down

3.1.1.4 Pseudo Code

```java
private void rewindActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void play_pauseActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void fast_forwardActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void recordActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void liveActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void volume_downActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void volume_upActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void browse_downActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
```

3.2 Graphics

3.2.1 Graphics

3.2.1.1 Module Description
The Graphics Module uses the OS graphics drivers to displays the graphics that the user will be able to see and interact with based on their input.

3.2.1.2 Interface
The graphics module interfaces with the Event Handler module in the User Interface Controller and the OS graphics driver. The graphics module connects to the touch screen via USB.

3.2.1.3 Physical Data Structure
This module will use java’s Swing, a part of Java Foundation Classes (JFC), an API for providing GUI in java.

3.2.1.4 Pseudo code

```java
private void initComponents() {
    Declare jPanels, jButtons, and jLabels for all buttons
    Specify jPanels, jButtons, and jLabels properties and position
}
```
4 Processing Layer

The Processing Layer serves as the middleman between the User Interface and the Data Layer. This layer contains two subsystems. Radio Input handler has one module, while the System Logic Controller has six.

![Figure 4-1 Processing Layer]

4.1 Radio Input Handler

4.1.1 Record

4.1.1.1 Module Description
The Record module calls the open source program, Sound Exchange (SoX) to compress and save incoming audio.

4.1.1.2 Interface
This module interacts with the logic module of the System Logic Controller and accepts a string argument from it. It will call SoX and pass a string as the folder name and another string as the wav name.

4.1.1.3 Physical Data Structure
String folderName
Unsigned int wavCount
String filename

4.1.1.4 Pseudo Code
record(String folderName){
    for(int wavName; wavName<MAX; wavName++){
        runs until max number of files is reached or user interrupts
        records audio by calling permRec.bat to execute sox which
        accepts two
        arguments, "rec -t alsa default
        ../Music/perm/${args[0]}/${args[1]}wav trim 0 5"
        first argument is the folder name, would be a date and time
        stamp
        second argument is a count, will be used as the audio file name
        calls file.wait(), wait for 5 seconds
        if mode is radio mode, exit
    }
}

4.2 System Logic Controller’s Subsystem

4.2.1 Logic

4.2.1.1 Module Description
This module will keep track of the current state of the system and the commands
given by the user. This module will be used to keep track of which threads are
active, which threads are sleeping, and which threads to kill and restart. Uses a
switch statement to determine which action to take after a button is pressed.

4.2.1.2 Interface
The Logic module interfaces with the UI Controller. UI Controller passes a String
argument to the Logic module. It calls the Radio Input Handler module, passing a
string to it. The Logic module also asks the Data Controller module for the
current playback status.

4.2.1.3 Physical Data Structure
int currentSoundFile
String folderName
int system_state //0 = capsuleMode, 1 = radioMode

4.2.1.4 Pseudo Code

logic(String fromUIController){
    case switch(string fromUIController)
        case 1: record
            if system_state(capsuleMode){
                kills recording process
                goes to live radio
            } if system_state(radioMode){
                system_state=capsuleMode;
                String folderName = makedir();
                permRecord(folderName);
            }
        case 2: pause
            save system state
}
if pause_state(false)
    call pause();

case 3: play
    update system state
    if pause_state(true)
        play();

case 4: fastForward
    get currentSoundFile
    get folderName
    fastforward(int howLongButtonIsPressed);

case 5: rewind
    get currentSoundFile
    get folderName
    fastforward(int howLongButtonIsPressed);

case 6: live
    system_state = radioMode;
    kill recording and playing
    clear temp directory, FileUtils.deleteDirectory(<File object of directory>);
    currentSoundFile = 0;
    folderName = temporary;
    start Record
    start playerController(int currentSoundFile, String folderName);

case 7: volUP
    call to SoX to increase volume, update volume value

case 8: volDOWN
    call to Sox to decrease volume, update volume value

case 9: browse
    browse() }

4.2.2 Pause

4.1.1.1 Module Description
This module will interrupt and stop the player process after the state of the system has been saved by the logic module.

4.1.1.2 Interface
This module interacts with the Player module to interrupt the player process and stop it from playing any more files.

4.1.1.3 Physical Data Structure
Thread.interrupt()
pause()
    get currentSoundFile
    get folderName
    Thread.interrupt()
}

4.2.2 Makedir

4.2.2.1 Module Description
Creates a new folder labeled with the date and time of creation. Audio files will
be saved in the newly created folder until a new recording is started.

4.2.2.2 Interface
Creates a folder, and then passes the folder name as a string to the Logic Module.

4.2.2.3 Physical Data Structure
Date date = new Date();
String folderName=date+"";
folderName = folderName.replaceAll(" ", ".");

4.2.2.4 Pseudo Code

makedir() {
    Date date = new Date();
    String folderName=date+"";
    folderName = folderName.replaceAll(" ", ".");
    String path = "/home/carter/Music/"+folderName;
    File dir = new File(path);
    dir.mkdir();
    return folderName;
}

4.2.3 Fast Forward

4.2.3.1 Module Description
The fast forward module will calculate the time the Fast Forward button is
pressed and increase the fast forward speed.

4.2.3.2 Interface
This module interacts with the Event Handler module to determine how long a
button was pressed.

4.2.3.3 Physical Data Structure
int howLongButtonIsPressed

4.2.3.4 Pseudo Code

fastforward(int howLongButtonIsPressed){
    if button is pressed, rewind 5 seconds
        check boundary, if time allows update currentfile location,
    else, set to MAX
        playercontroller(currentfile+1, folderName)
    if buttons is pressed and held for 3 seconds, rewind 10 seconds
        check boundary, if time allows update currentfile location,
    else, set to MAX
playercontroller(currentfile+2, folderName)
    if buttons is pressed and held for more than 4 seconds, rewind by 30 seconds per seconds
        check boundary, if time allows update currentfile location,
    else, set to MAX
        playercontroller(currentfile+(howLongButtonIsPressed*6), folderName)
}

4.2.4 Rewind

4.2.4.1 Module Description
The rewind module will calculate the time the rewind button is pressed and increase the rewind speed.

4.2.4.2 Interface
This module interacts with the Event Handler module to determine how long a button was pressed.

4.2.4.3 Physical Data Structure
int howLongButtonIsPressed

Pseudo Code

rewind()
{
    if button is pressed, rewind 5 seconds
        check boundary, if time allows update currentfile location,
    else, set to MIN
        playercontroller(currentfile-1, folderName)
    if buttons is pressed and held for 3 seconds, rewind 10 seconds
        check boundary, if time allows update currentfile location,
    else, set to MIN
        playercontroller(currentfile-2, folderName)
    if buttons is pressed and held for more than 4 seconds, rewind by 30 seconds per seconds
        check boundary, if time allows update currentfile location,
    else, set to MIN
        playercontroller(currentfile-(howLongButtonIsPressed*6), folderName)
}

4.2.5 Live

4.2.5.1 Module Description
This module will play the latest file saved while in radio mode. It will reset the system state to its default value and delete all files inside the temporary folder.

4.2.5.2 Interface
This module interfaces with the Logic module and starts the Record module. It passes an integer representing the current file to be played and a string value representing the folder that the file is saved.

4.2.5.3 Physical Data Structure
int currentSoundFile
4.2.5.4 Pseudo Code

live()
    system_state=radioMode;
    kill recording and playing
    clear temp directory, FileUtils.deleteDirectory(<File object of
directory>);
    currentSoundFile = 0;
    folderName = temporary;
    start Record
    start playerController(int currentSoundFile, String folderName)

4.2.6 **Browse**

4.2.6.1 Module Description
This module retrieves a list of files stored in an array and displayed to the user.

4.2.6.2 Interface
This module retrieves a list of file from the hard drive and returns it to the Logic Layer.

4.2.6.3 Physical Data Structure
String [] browseFolders;

4.2.6.4 Pseudo Code

browse()
    String [] browseFolders;
    retrieves a list of directories from the hard drive
    Store in array
    return browseFolders;

4.2.7 **Play**

4.2.7.1 Module Description
This module restarts the playing process after it has been paused.

4.2.7.2 Interface
This module is called by the Logic module and interacts with the Player module
to resume the player process and stop it from playing any more files.

4.2.7.3 Physical Data structure
int currentSoundFile
String folderName

4.2.7.4 Pseudo Code

play()
    get currentSoundFile
    get folderName
    start playerController(int currentSoundFile, String folderName);
5 Data Layer Component Design

The Data Layer manages and plays the audio files.

5.1 Data Controller

5.1.1 Player Controller

5.1.1.1 Module Description
This module feeds the Player module with the files that will need to be played.

5.1.1.2 Interface
Playback Controller is called by the Logic module of the System Logic Controller. It accepts three arguments two ints and one string. The first int is a number to determine which file to start playing first, the second is the volume value used by SoX to control the volume. The string is the directory in which the files are stored.

5.1.1.3 Physical Data Structure
unsigned int currentWavFile;
String folderName;

5.1.1.4 Pseudo Code

playerController(int wavFile, String folderName){
    for(wavFile;wavFile<totalNumberOfFiles; wavFile++){
        setsoundFile(wavFile);
        setFolderName(folderName);
        calls SoX, audioPlayer(wavFile,folderName);
        wait 5 seconds
    }
}
setFolderName(String folderName){
    String currentFolderName = folderName;
}

setSoundFile(int currentFile){
    int currentWavFile = currentFile;
}

getsoundFile(){
    return currentWavFile;
}

5.1.2 Wait

5.1.2.1 Module Description
This module pause the Player Controller for 5 seconds to allow enough time complete the playing of a file in the Player Module before the next file is played.

5.1.2.2 Interface
Accepts an integer value from the player controller, it represents the seconds that the loop should wait before resuming playing.

5.1.2.3 Physical Data Structure
int n;  //seconds to wait
long t0,t1  //start and end time

5.1.2.4 Pseudo Code

wait (int n){
    long t0, t1;
    t0 = System.currentTimeMillis();
    do{
        t1 = System.currentTimeMillis();
    } while ((t1 - t0) < (n * 1000));
}
5.2 Audio Player

5.2.1 Player

5.2.1.1 Module Description
This module is the .bat script containing the SoX commands.

5.2.1.2 Interface
The play module accepts 3 arguments from the playback controller to determine which file is to be played and at what volume. It accepts one integer and two strings. First integer is the volume at which SoX will play the file at. Second argument is a string to determine in which directory the file in is. The last argument is the name of the wav file to be played.

5.2.1.3 Physical Data Structure
\${args[1]} = directory
\${args[2]} = file name

5.2.1.4 Pseudo Code

play ../Music/\${args[1]}/\${args[2]}
6 Hardware Components

6.1 Overview

This section describes some of the main physical hardware components that will be used in the dRadio system. Along with that, this section will also cover the purpose, specification, and the interfaces of each component.

6.2 Intel NM10 Mini ITX

![Intel NM10 Motherboard](image)

**Figure 6-1 Intel NM10 motherboard**

6.2.1 Purpose

The purpose of the motherboard is to run our entire system. The motherboard will consist of different components that will be used for different purposes.

6.2.2 Specification

- Intel Atom D525 @ 1.8 GHz
- Integrated Audio
- Integrated Graphics

6.2.3 Interfaces
The Mini ITX will be powered by a Pico PSU power supply. It will receive radio input and output audio to speakers with its 3.5 mm input/output. A display will be connected to the board with a USB cable.

### 6.3 Pico-PSU-120W1-25v

6.3.1 Purpose

Provides filtered and stable power to the Mini ITX board

6.3.2 Specification

- 20 or 24 pin ATX connector
- 12V input
- 120 Watts output

6.3.3 Interfaces

Connects to the Mini ITX board and the power source.
6.4 Hard Drive

![Solid State Drive Image]

Figure 6-3 32 GB Solid State Drive

6.4.1 Purpose
The main purpose of the hard drive is to store the permanent recordings as wav files. Each audio file will be of a length of 5 seconds long.

6.4.2 Specifications
- Capacity: 32 GB
- Max read: 130 MB/s
- Max Write: 60 MB/s

6.4.3 Interfaces
The SSD will be connected to the Mini ITX with a SATA II cable.

6.5 Touch screen

![Touch Screen Image]

Figure 6-4 MIMO 7in Touch Screen

6.5.1 Purpose
The touch screen acts as the main component which helps user to interact with the dRadio system.

6.5.2 Specifications
   - USB powered
   - 7” display
   - 800x480 resolution
   - Resistive touch screen

6.5.3 Interfaces
   The touch screen will be connected to the Mini ITX via USB, it will be used to accept user’s input.
7 Requirements Traceability Matrix

7.1 Purpose

Requirements mapping shows that the architecture fulfills the requirements listed in the System Requirement Specification. The requirements listed are based on the acceptance criteria detailed in the SRS.

<table>
<thead>
<tr>
<th>Modules</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.1 Graphical User Interface</td>
</tr>
<tr>
<td></td>
<td>3.2 Speaker Output</td>
</tr>
<tr>
<td></td>
<td>3.3 Input from Radio</td>
</tr>
<tr>
<td></td>
<td>3.4 Saved Recording</td>
</tr>
<tr>
<td></td>
<td>3.5 Playing Audio</td>
</tr>
<tr>
<td></td>
<td>3.6 Live Radio Playback</td>
</tr>
<tr>
<td></td>
<td>3.7 Playback Controls</td>
</tr>
<tr>
<td></td>
<td>3.14 Delete Recording</td>
</tr>
<tr>
<td></td>
<td>5.3 Display</td>
</tr>
<tr>
<td></td>
<td>5.4 Storage</td>
</tr>
<tr>
<td>Event Handler</td>
<td>X</td>
</tr>
<tr>
<td>Graphics</td>
<td>X</td>
</tr>
<tr>
<td>Record</td>
<td>X X X X X</td>
</tr>
<tr>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td>Files</td>
<td>X X X X X</td>
</tr>
<tr>
<td>Player Controller</td>
<td></td>
</tr>
<tr>
<td>Player</td>
<td>X X X</td>
</tr>
</tbody>
</table>
8 Quality Assurance

8.1 Test Plans and Procedures

8.1.1 General
This section describes, briefly, how each architectural component will be tested to ensure the architecture meets the design requirements specification.

8.2 Module/Unit Test

8.2.1 User Interface layer

8.2.1.1 Event Handler

- Ensure that all buttons produces the correct result by checking the String value produced by each button.

8.2.1.2 Graphics

- Test to make sure the correct graphics is displayed.

8.2.2 Processing Layer

8.2.2.1 Logic

- Logic module must keep up with the current state of the system. It must know what mode the system is in, what file is currently being played and from which folder. This module must be tested to ensure that it correctly retrieves and stores those values.
- Test to ensure that the logic module is capable of killing and restarting a thread.
- Test to ensure that it correctly identifies and responds to the String value passed from the UI controller.

8.2.2.2 Record

- Test to ensure that the record module is capable of saving audio files in a given directory.
• Test to ensure that the files saved are playable and is 5 seconds in length.

8.2.3 **Data Controller**

8.2.3.1 **Player Controller**

• Player controller must be able to keep store and keep track of the current file being played and its location. It must be test to ensure that it is capable of accurately saving the state.
• Must be tested to ensure that it can continuously pass valid audio file to be played by the player module.

8.2.3.1 **Player**

• Tested to ensure that it can play wav files.

8.3 **Component Test**

8.3.1 **Power Supply**

The power supply will be tested by supplying power to it and verifying that it provides the appropriate voltage.

This will be done with a multimeter.

8.3.2 **Motherboard**

The motherboard will be tested by connecting a working power supply, a monitor, a keyboard, and a mouse and verifying that the motherboard takes inputs from the mouse and keyboard, and provides output via the monitor.

8.3.3 **Hard Drive**

The hard drive will be tested by connecting it to a working computer and verifying that we can read and write to the device.

8.3.4 **Touch screen**

The touch screen will be tested by connecting it to a working computer and verifying that we can provide input to the computer as well as be provided output from the computer through the touch screen.

8.3.5 **RAM**

The RAM will be tested by connecting it to a working motherboard and making sure the memory can be used by the computer for its tasks.
8.4 Integration Test

8.4.1 User Interface Layer

- When a button is pressed we will ensure that the correct method was executed.

8.4.2 Processing Layer

- When the Processing Layer is provided with a test string, we will ensure that the appropriate method is executed.
- When the Processing Layer is provided with audio input, we will make sure that the input is being recorded and save to the right location on the hard drive.

8.4.3 Data Control Layer

- When one of the Data Controller's methods is called, we will ensure that the appropriate file actions take place.
- We will ensure that audio is being outputted using the speaker port.
- When the pause command is issued, the data layer will stop outputting sound.
- When given the play command is issued, the data layer will resume outputting sound.

8.5 System Verification Test

The entire system will be tested as whole to ensure that it satisfies all high level requirements and all requirements listed in the Acceptance Criteria of the SRS. It will be tested to ensure that all interaction between layers is correct, and all playback functions are present and work correctly.
9 Acceptance Plan

9.1 Overview
This section describes the minimum criteria that the dRadio system must meet before it is considered complete.

9.2 Packaging and Installation
The dRadio will be packaged as one unit inside a storage medium, such as a cardboard box. The dRadio will also include, in addition to the product, an instruction manual that will detail the basic operation of the system. A safety pamphlet will also be included in order to make the user aware of the risks associated with the operation of such a device. All physical components will have preloaded software. The package will include a 1 foot long USB cable to connect the dRadio unit with its touch screen. The user will not need to perform any installation procedures other than the actual installation of the dRadio into the car, the connecting of the dRadio with its touch screen, the conversion of the car radio’s signal into the appropriate format, and the conversion of the dRadio’s audio output into a format that the car speakers will use.

9.3 Acceptance Testing
System testing will be conducted to ensure the product meets the acceptance criteria. The details of this testing will be provided in the System Test Plan document.

9.4 Acceptance Criteria
The following criteria must be met in order for the dRadio System to be considered acceptable:
<table>
<thead>
<tr>
<th>Acceptance Criterion</th>
<th>Requirements Addressed</th>
<th>Verification Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify that the dRadio allows for interaction via a GUI</td>
<td>3.1, 5.3</td>
<td>While the dRadio and car radio are both connected and turned on, the user should be able to see and interact with a GUI.</td>
</tr>
<tr>
<td>Verify that the dRadio plays live radio</td>
<td>3.2, 3.3, 3.6, 3.8</td>
<td>While the dRadio and car radio are both connected and turned on, the user should be able to push the jump to live button. The audio should be played through the existing car speakers with a small delay from true live radio.</td>
</tr>
<tr>
<td>Verify that the dRadio plays saved recordings</td>
<td>3.2, 3.3, 3.4, 3.5, 3.8, 5.4</td>
<td>While the dRadio and car radio are both connected and turned on, the user should be able to use the GUI to browse through a list of recordings and play a selected recording.</td>
</tr>
<tr>
<td>Verify that the dRadio allows the user to save and delete recordings</td>
<td>3.1, 3.3, 3.4, 3.14, 5.4</td>
<td>While the dRadio and car radio are both connected and turned on, the user should be able to push the record button while listening to live radio to save the current broadcasting into permanent storage. The recording will stop when the user pushes the record button again, or until there is no more memory. The user can then browse through the previous recordings and delete selected recordings.</td>
</tr>
<tr>
<td>Verify that the dRadio offers playback controls</td>
<td>3.7</td>
<td>While the dRadio and car radio are both connected and turned on, the user can use the GUI to issue basic playback controls like rewind, play, pause, fast-forward, and jump to live.</td>
</tr>
</tbody>
</table>
10 Appendices

10.1 Sound Exchange

10.1.1 Overview

Sound Exchange (SoX) is a command line utility that can be used to compress and save audio files. Known as the “Swiss Army Knife” of sound processing programs, SoX is published under the GNU General Public License which gives Team AudioTEK the permission to use and change the software.

10.1.2 Purpose

SoX will be used for all of our systems audio processing. It will be used to compress raw audio files from the sound card into wav files, and will also be used to play the files.

10.1.3 Interaction

Figure 10-2 depicts the interactions of SoX.

![Figure 10-1 Sound Exchange (SoX) Interaction](image)