Ryerson University

Department of Electrical, Computer & Biomedical Engineering
Faculty of Engineering & Architectural Science

COE718 – Embedded Systems Design

LAB REPORT

Semester/Year:	Fall 2024
Lab Number:	Interim Report

Instructor:	Dr. Gul N. Khan
Section No:	012
Submission Date:	2024-11-14
Due Date:	2024-11-14

Student Name	Student ID	Signature
Carlo Dumlao	501018239	C.D

By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a 0 on the work, an F in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: www.ryerson.ca/senate/current/pol60.pdf.

Table of Contents

Progress

During the initial weeks, the provided examples were firstly tested including the LCD_Blinky and USBAudio to understand the generalities on how images and audio can be generated in the MCB1700 board. In particular, from the LCD_Blinky, the methods that are crucial for initializing and loading an image in the LCD were noted; and, from the USBAudio, the importance of the interrupt handler was recognized, as well as the methods for connecting and disconnecting to the USB. With this understanding, the overall structure of the application was realized thereafter such as its state diagram and the multiple threads or tasks that would execute during runtime.

Introduction

The Figure1 below showcases the overall structure of the media-center. Notice that the input of the joystick controller is firstly observed, which is then fed into the controller. This controller acts as the *brain* of the application: it dictates the media center when it should be generating an image, audio, or game, and it also manages the varying screens to be displayed in the LCD (ie. homepage and menu screen). Put simply, the application will execute three different threads concurrently :

1) First thread is the joystick input. This will continuously read the input of the user via the joystick controller.
2) Second thread is for the controller. This will run indefinitely, and it will manage the 3rd thread to be executed.
3) Third thread is reserved for tasks that are responsible for displaying the screen and processing the features (image, audio, and game) of the application.

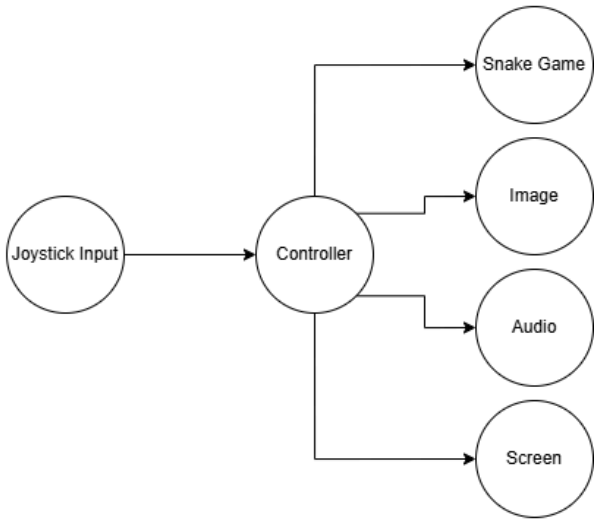

Figure 1: Structure of the media-center.

State Diagram and Threads

The flow diagram of the controller can be seen in Figure 2. Note that the states will not be implemented sequentially, but rather it will be coded in a multithreaded manner. In this case, if the controller thread wishes to execute a particular thread (or a state) after a transition, then it will simply unblock it through an inter-thread communication. The collection of threads are shown in Table 1.

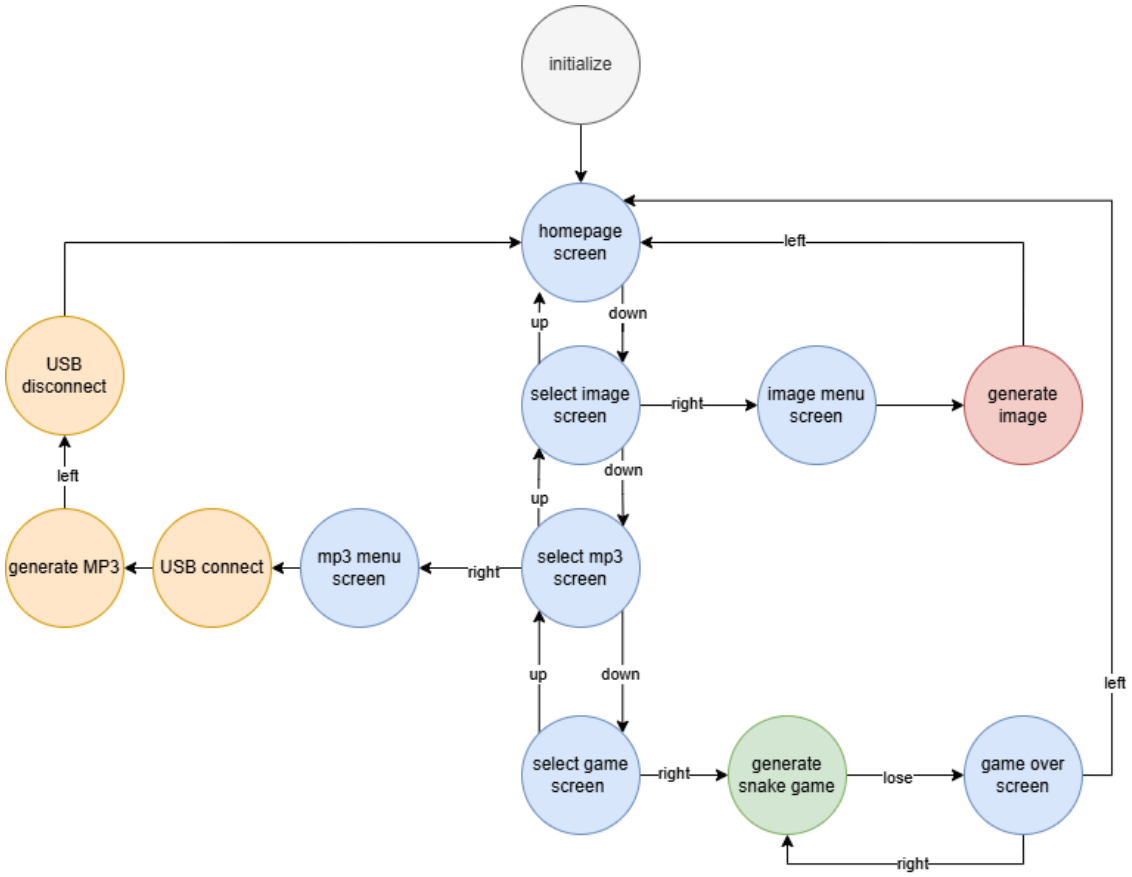

Figure 1: State diagram of the controller.

Table 1: Threads of the application and its functionality.

Thread	Functionality
Initialize	• Setups the registers, priorities, interrupts, and other methods that are crucial for initializing the LCD, Audio, and multithreading.
Joystick Input	• Determines whether the joystick has been tilted to the left, right, down, or up.
Controller	• Determines the next thread to be executed depending on

	its current state and the readings from the Joystick Input thread.
Homepage Screen	● Displays the three features of the application (ie. image, mp3, and game)
Select Image Screen	● Displays a *select image screen* to indicate that the user has selected the gallery option
Select MP3 Screen	● Displays a *select mp3 screen* to indicate that the user has selected the mp3 option
Select Game Screen	● Displays a *select game screen* to indicate that the user has selected the game option
Image Menu Screen	● Displays a control menu for the gallery
Generate Image	● Generates a random image from the collection of images ● If the user tilts the joystick to the left, it will exit and go back to the homepage. ● If it is on the right, the gallery displays the next image
MP3 Menu Screen	● Displays a control menu for the MP3
USB Connect	● Establishes a connection to the PC
Generate MP3	● Streams the incoming audio signal from the PC ● This will be implemented by: ○ Assigning it to a higher priority to ensure that the audio is generated smoothly ○ Using a virtual timer to achieve the 32kHz audio sampling rate
USB Disconnect	● Terminates the connection to the PC
Generate Snake Game	● Displays a snake game to be played by the user ● The variables associated with the game will be updated periodically through the implementation of virtual timer
Game Over Screen	● Displays a *game over screen* indicating that the game has ended ● If the user tilts the joystick to the left, it will exit and go back to the homepage. ● If it is on the right, the game will restart.

Snake Game

The application will offer a single game – that is, a snake game. A simple pseudo code is provided below:

```
//Note: the game will be scaled by a WxH grid

//Randomly generate food. It appears outside of the snake's body
generate _food(W, H);

//Run the game until the the head of the snake hits its body, or goes
beyond the boundary
while(!gameover()){

      //Move the joystick based on the user input
      move_snake(joystick_input);

      //If the head of the snake hits the food, increase the length
of the snake
      if(ate_food()){
            add_snake_length();
            generate_food(W,H)
      }

      //display the snake every 500ms
      displaySnake();
      wait(500ms);
}

//Signal the game over screen thread
signalGameOverScreen()
```

Future Plans

1) Construct the joystick input thread. Furthermore, make a read-only global resource – a resource reflecting the input of the user – that will be shared among the threads, especially the controller thread.
2) Implement the screen threads that are responsible for displaying the homepage and menu page for each of the features.
3) Implement the threads correlating to the image, audio, and game generation. Reuse and modify the provided LCD_Blinky and USBAudio such that it agrees with the requirement of the media center:
    a) Modify the LCD_Blinky such that it is also capable of displaying multiple images one at a time
    b) Modify the USB audio so that the audio stream is executed through a virtual timer, rather than the Timer 0 interrupt handler
4) Finally, implement the controller thread, and integrate all the threads in accordance to the state diagram depicted in Figure 2.