Abstract

This report will cover the progress of the media center that was created for the COE718 final project. The report is split into 8 parts as follows: introduction, past work/review, methodology, design, experimental results, conclusion, references and appendix. The introduction will introduce the project and specify the purpose and the project's intentions. The past work/review section will outline my experience with hardware and software and their effect on the development of the media center on the board. The methodology section will describe the method that I used to design the project, from a top-level and general perspective, followed by the design section which will give the details based on my methodology for the various components, modules and functions that I implemented in my project. The experimental results will show the results of the media center running on the board itself. The conclusion will reflect on the overall project and discuss the strengths and weaknesses of my implementation, followed by the references and the appendix, which will include all the C files that I modified.

Introduction

This final project's purpose was to make use of all of the programming concepts learned throughout the semester in order to implement a media center. The media center's features include a photo gallery for viewing images, an MP3 player for streaming audio from the PC through USB, and a game that users can play with the on-board joystick. All of these features are connected through a main menu, controlled by the joystick. This report will outline the progress that has been made with implementing the media center, the hardware that was used on the MCB1700, the software specifications and a description of the overall system.

Past Work/Review

Many previous labs and extracurricular experiences played a large role in aiding my implementation of this media center. Outside of school, I actively develop games, and am the lead developer for a MMO made with C# in Unity. The experience I've gained through game development has helped me greatly with regards to the logic implementation. For example, the joystick controls for both menus and the game were very easy to implement due to my experience in programming game controls. For the game specifically, I created a remake of Flappy Bird to run on the MCB1700, which I had already implemented for Android a few years ago. The logic was very familiar to me and easy to reproduce in C for use on the board, only having to make a few small changes.

In addition to past game development experience, the labs that were conducted in this course and other courses in the Computer Engineering curriculum also helped make the implementation of the media center simpler. Other courses such as COE328, COE538, and COE608 all make use of software development with implementation on hardware in environments similar to this course, COE718, where we are using the MCB1700 board with Keil uVision IDE for programming. Having experience in developing for different hardware and getting used to VHDL environments has made it easier to implement the media center.

Methodology

For the media center, the required hardware includes:

- A 320x240 pixel LCD (landscape) that connects using a 16-bit interface.
- A joystick with 10 states: neutral, up, down, left, right and clicks in each direction.
- A potentiometer with a voltage range of 0.0V to 3.3V.
- A USB interface for connecting to a PC for audio streaming.

The LCD screen is used for showing the output of the media center to the user so that they can interact with the program. The joystick is what is used to interact with the media center, specifically for navigating the menu and controlling the game. The potentiometer is used during the mp3 audio streaming to control the volume of the playback. Finally, the USB interface is what connects the MCB1700 board to the computer in order to facilitate the audio streaming.

The software designed for the media center is meant to interact with the hardware specified above. The main components of the media center are separated into different screens. Essentially, there are four screens: the main menu screen, the photo gallery screen, an MP3 player screen, and a game screen.

The game, photo gallery, and MP3 player screens are all connected through the main menu, which has support for the joystick on the MCB1700 board. The MCB1700's joystick was programmed using the knowledge acquired through completing lab1, such that the code efficiently polls the joystick and only updates the LCD when a valid input has been made. This way, when a user is holding down on the joystick in the menu, the cursor isn't jumping past menu options constantly but instead will only move one position at a time until the joystick is back in the neutral position. Since the joystick will be used for different purposes on different screens, the valid inputs will vary between them depending on their functionalities. For the main menu, the joystick can only be in the up position, down position, neutral position, and neutral-click position. All other joystick configurations are ignored, and only the valid positions affect the user experience. Moving the joystick up and down will position the cursor next to text representing one of the three other screens. When the joystick is in the neutral-click position, whatever text it is beside is the screen that will be entered next. The LCD clears itself and then draws the necessary images for the selected screen and the code enters the correct block representing the logic of the screen that is displayed.

The project makes use of images uploaded to the MCB1700's memory and displayed on the LCD. For example, the cursor on the main menu is actually a 34x34 pixel image of the character that will be controllable in the game. The image moves up and down depending on the

position of the cursor. Other examples of images used in this project are the logo and foreground of the game on the game screen as a starting screen, as well as a background for the MP3 player screen. The images are still a work-in-progress, and will be altered or upgraded as the project progresses. What won't change, however, is how the loading of images is implemented in the code behind the scenes. As shown in the example, the image that is planned to go on the MCB1700 board must be exported as a C source file in 16-bit RGB (RGB565). The image must also fit on the LCD, or in other words it must be at max 300x240 pixels. When following the default implementation from the example project for displaying images on the LCD, it was found that the images were always flipped vertically. To fix this, a change was made to how the images were displayed in the GLCD_Bitmap function in the GLCD_SPI_LPC1700.c file by adding a small bit of code. This same approach of loading images and manipulating their positions used for the cursor image will be used to control the character sprite for the game.

In addition to joystick control and image loading, audio streaming via USB from the PC to the MCB1700 board is also working as expected. By following the USBAudio example project provided, the audio streaming was implemented into the media center through USB. When audio is played on the PC, the sound will come out through the MCB1700 board's internal speakers located underneath the LCD screen. The potentiometer is used as a volume controller, where a clockwise rotation decreases the volume and a counter-clockwise rotation increases the volume of the audio being played. Although the functionality is nearly complete, the screen itself will undergo visual changes to make it match that of the other screens and the theme of the game. One thing that needs to be completed in future versions of this project is the disconnection of the MP3 player when signaled to exit from the player by some means, such as using the joystick or the keyboard when it is implemented for the game in the future.

Design

The media center is split into 4 parts, the main menu, game, photo gallery, and MP3 player. The main menu connects to each of the other screens and is navigated using a joystick, and each screen will link back to the main menu once a user finishes whatever they needed to do on the screens by pressing left on the joystick.

In the main loop, each screen is represented by its own block, distinguished by an integer which serves as a Boolean variable. The default screen when the media center starts up is the main menu, so the menuSelected integer is set to 1. When one of the screens is active, all of the integer "Booleans" associated with the other screens are set to zero, so that only the block of code that is associated with the active screen is running and they don't interfere with each other. For example, since joystick functionality will be different for each of the screens, the media center must have a way to enable and disable the different control schemes depending on which screen is active. This method proves to be reliable and easy to implement in comparison to other methods such as separating the screens into different C files. ***Figure 1*** below shows an example of these code blocks for each screen and their associated variables.

Figure 1: Code block for different screens.

```
if(photoGallerySelected == 1)
{
  menuSelected = 0;
  photoGallerySelected = 0;
  mp3PlayerSelected = 0;
  if(justOpened == 1)
  {
    justOpened = 0;
    GLCD_SetBackColor(Black);
    GLCD_SetTextColor(Yellow);
    GLCD_DisplayString(0, 0, __FI, "                    ");
    GLCD_DisplayString(1, 0, __FI, "                    ");
    GLCD_DisplayString(2, 0, __FI, "                    ");
    GLCD_DisplayString(3, 0, __FI, "                    ");
    GLCD_DisplayString(4, 0, __FI, "                    ");
    GLCD_DisplayString(5, 0, __FI, "                    ");
    GLCD_DisplayString(6, 0, __FI, "                    ");
    GLCD_DisplayString(7, 0, __FI, "                    ");
    GLCD_DisplayString(8, 0, __FI, "                    ");
    GLCD_DisplayString(9, 0, __FI, "                    ");
    GLCD_DisplayString(10, 0, __FI, "                    ");
    GLCD_Bitmap(0, 150, 320, 80, foreground);
    GLCD_Bitmap(0, 0, 300, 97, logo);
  }
}
else if(mp3PlayerSelected == 1)
{
  menuSelected = 0;
  photoGallerySelected = 0;
  gameSelected = 0;
    • • •
```

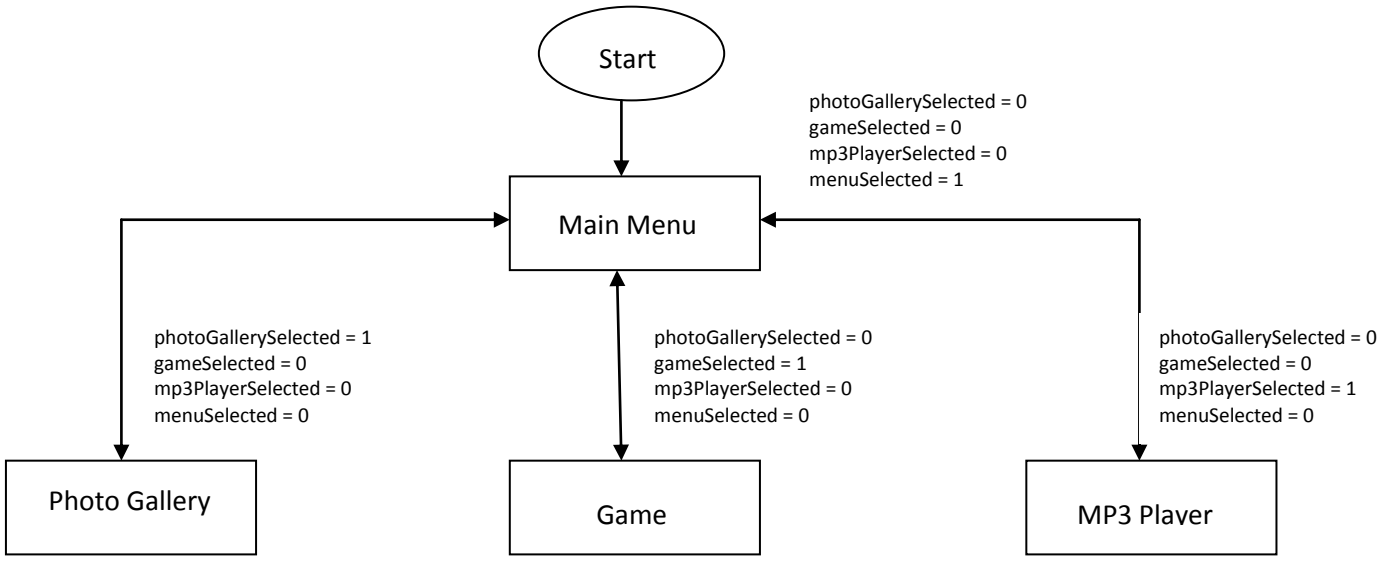
Navigating the main menu is done using the joystick, and updates to the LCD are only pushed when a valid joystick input is made, preventing the LCD from constantly updating for no reason. The joystick is polled using the get_button() function every loop of the main method and the resulting value is checked for whether it is a valid input or not. If it is, then one of two things happen: if the joystick is moving up or down then the pointer's location is updated accordingly, and if the joystick is in the neutral position and clicked then the screen change process is started by setting the associated Boolean integer to 1, as described above, in order to switch from the main menu to that code block. ***Figure 2*** below illustrates the code for this functionality.

Figure 2: Joystick configuration for main menu navigation.

```
/********* The main Function is an endless loop ***********/
while (1) {
    joystickCode = get_button();

    if(lastCode != joystickCode && (joystickCode == 0x00 || joystickCode == KBD_UP || joystickCode == KBD_DOWN
      || joystickCode == 0x01))
    {
      lastCode = joystickCode;
      switch(joystickCode){
        case 0x00:
          if(menuSelected == 1)
          {
            sprintf(direction, "N");
            if(pointerLocation != 85 && pointerLocation != 135 && pointerLocation != 185) pointerLocation = 85;
          }
          break;
        case KBD_UP:
          if(menuSelected == 1)
          {
            sprintf(direction, "U");
            if(pointerLocation == 135) pointerLocation = 85;
            else if(pointerLocation == 185) pointerLocation = 135;
          }
          break;
        case KBD_DOWN:
          if(menuSelected == 1)
          {
            sprintf(direction, "D");
            if(pointerLocation == 85) pointerLocation = 135;
            else if(pointerLocation == 135) pointerLocation = 185;
          }
          break;
        case 0x01:
          sprintf(direction, "N Click");
          if(pointerLocation == 85){ justOpened = 1; photoGallerySelected = 1; }
          else if(pointerLocation == 135) { justOpened = 1; mp3PlayerSelected = 1; }
          else if(pointerLocation == 185) { justOpened = 1; gameSelected = 1; }
          break;
```

Figure 3: Flow chart of overall system.



The photo gallery was designed to show different images by scrolling through them using the joystick. When the joystick is moved to the right position, the photo gallery replaces the image on the screen with the next image, looping back to the first image when the last image is passed. When the joystick is moved to the left position at any time, the photo gallery exits and the media center returns control to the main menu, no matter what picture is currently active. In order to keep track of which image is currently active, the media center stores the current photo number as an integer. Depending on the current photo number, the next photo number can be determined by incrementing it, or returning back to 1 if the current photo is the last photo in the queue. When a photo is changed, the screen is cleared so that the last photo is erased and only the current photo is visible on the screen. The photo gallery pictures can be seen in the experimental results section as ***Figure 6, 7,*** and ***8***.

When the mp3 player screen is selected, the USB initialization begins automatically and the audio streaming service begins, displaying a splash screen to indicate to the user that streaming has begun. The implementation of this is shown in the experimental results section as ***Figure 9.*** Similar to the photo gallery screen, pressing the joystick to the left position will allow the user to exit the streaming at any time and return control to the main menu. Apart from this, the user can also control the volume of the audio playback using the potentiometer on the board itself.

Upon selecting the game option from the main menu screen, the media center goes to the game's start screen, as shown in ***Figure 10***. Here, the player sprite is shown and the user is given two options: either press the joystick in the neutral position to begin the game, or move the joystick left to return control to the main menu screen in the same process as the photo gallery and mp3 player screen. The game itself is a clone of the mobile game Flappy Bird, popularized for its simple one-touch control scheme and its difficulty. Similarly, the game that was implemented for the media center has only one input, which is the neutral click of the joystick. When the game is active, the bird is constantly moving downwards by the effect of gravity, and clicking the joystick causes it to flap its wings and go upwards for a short period of time. Clicking multiple times will make the bird flap more and go higher. The bird has four different sprites depending on its state. There is a bird_up sprite when the bird is moving up, a bird_down sprite when the bird is moving down, a bird_mid sprite when the bird is idle and a bird_dead sprite when the bird has died and the game is over. All of these are shown below in ***Figure 4***.

Figure 4: Bird states for the media center game (idle, up, down, dead).

The main purpose of the game is to avoid the pipes that come in from the right side of the screen and sweep towards the left. There is a small opening between the pipes where the user can fly through, but touching the pipes will put the bird in the dead state and the game will be over. Navigating through each sprite adds to the score of the user, which is displayed in the top left when passing through the first pipe. There are two pipes active on the screen at all times, each 120 pixels apart from each other (half the size of the screen's width). When each pipe goes past the left edge of the screen, it reappears on the right side to simulate an infinite game length, only ending when the user loses. The pipes have three different positions that they cycle through: one with a gap in the center, one with a gap towards the top of the screen, and one with a gap towards the bottom of the screen. Each of the two active pipes will cycle through one of these three configurations, and they are offset at the beginning so that it gives a pseudo-random position during the game. The collision check between the bird and the pipe is done on the top and bottom portions of the left side of the pipe, as well as the center part which is above and below the gap so that a player cannot fly up or down through the pipe when they are halfway through it.

When the bird touches the ground or the ceiling however, the game is over and the bird is in the dead position. The ground and the ceiling are represented by the bottom edge and the top edge of the LCD screen respectively. In code, these would be represented as y = -5 and y = 205. -5 and 205 were chosen instead of 0 and 240, which are the actual pixel boundaries of the screen, because they account for the offset of the bird's pixels since the coordinates of the bird begin at the top left rather than the center. When in the dead state, if the bird is in the air then it falls straight down until it hits the ground, and the pipes stop moving. Once the bird is on the ground, clicking again returns the user to the start screen to either retry the game or leave. The player can also leave at any time by pressing the joystick left during the game.

Experimental Results

Figure 5: Main menu with picture cursor.

Figure 6: Photo gallery picture 1.

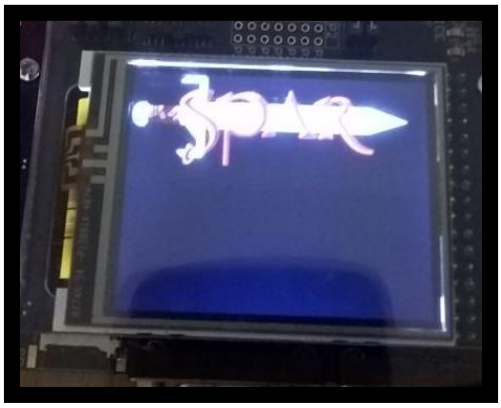


Figure 7: Photo gallery picture 2.

Figure 8: Photo gallery picture 3.

Figure 9: Mp3 audio streaming with light active.

Figure 10: Game start/restart screen.

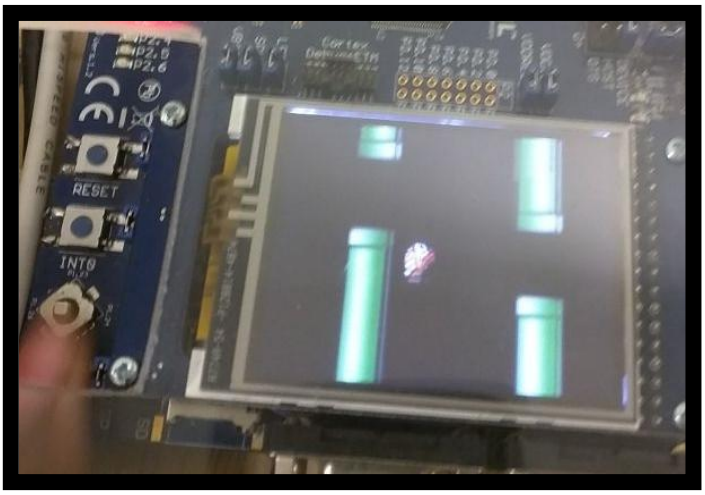


Figure 11: Gameplay screen.

Conclusion

The purpose of this project was successfully met, having created a working media center which covers all the initial specifications. The media center has a main menu, a game, a photo gallery, and an mp3 streamer, all using the hardware on the board and the PC for the streaming. Knowledge from previous labs and personal experience in programming and game development were taken advantage of in order to implement the media center and all its required components successfully. I ran into issues after overestimating the specifications of the board with regards to the game and was having trouble with the refresh rate of the game and game play lag, resulting in the decision of choosing to deal with graphical issues rather than a slow-paced game. Looking forward, I could choose to either make a simpler game or take advantage of more advanced programming techniques in order to manage memory better and implement the SD card reader. Overall, however, the media center was a good learning project for development on the MCB1700 board and the Kiel uVision environment.

References

Media Center. (n.d.). Retrieved from

https://www.ee.ryerson.ca/~courses/coe718/labs/Media-Center.pdf

ARM Ltd and ARM Germany GmbH. (n.d.). RL-ARM User's Guide. Retrieved

November 28, 2017, from
http://www.keil.com/support/man/docs/rlarm/rlarm_ar_artxarm.htm