Media Center

COE718: Embedded System Design

Daniel Osawaru
Computer Engineering (BEng)
Toronto Metropolitan University
Toronto, Canada
dosawaru@ryerson.ca

I. INTRODUCTION

This project demonstrates a real-time application, creating a media center using the MCB1700 board, uVision, and incorporating all the concepts learned throughout the semester. The media center consists of 3 main components: photo gallery, audio player, and media game. All of these components can be accessed through the main menu of the media center.

When the Photo Gallery is selected, a new gallery menu is displayed which shows the list of all the images. The user can use the joystick to navigate to which image they would like to view and then select it by moving the joystick to the right. From here, the image is displayed on the LCD. The user must move the joystick to the left to return to the gallery menu.

The Audio Player can stream audio from the PC to the MCB1700 board. The potentiometer on the board can control the volume of the audio by rotating it counterclockwise to increase the volume and clockwise to decrease the volume. To return to the menu, the user must move the joystick to the left which will stop the sound from transmitting from the board.

The media game is created based on the classic game Tic Tac Toe. Two players take turns placing down game pieces, X and O, in a 3 by 3 square grid. The winner is the first player to get 3 pieces in a row either up, down, across, or diagonally. If all 9 squares are full and no player connects 3 of their pieces in a row, the game ends in a tie. Once a game is over, it returns to the game menu and the game can be played again.

II. PAST WORK/REVIEW

This program was written in C programming language using the Keil uVision Integrated Development environment.

The sample USB demo project is incorporated to help stream audio via USB from the PC to the board. GLCD_Bitmap() function was also implemented which is used to display an image on a graphic LCD.

Knowledge gained from previous labs in this course was also utilized like sample codes, documentation, and function analysis. The files LED.c, GLCD_SPI_LPC1700.c, and KBD.c were used to help with the functionality of the program. The KBD.c file was needed to read the joystick input from the user, navigate through the menu, and play the media game. GLCD_SPI_LPC1700.c is used to draw images onto the LCD and display text. To manipulate the LED on the board, the built-in function in LED.c were incorporated.

A. *Hardware Components*

The hardware component used for this project is the MCB1700 board. The board enables the user to perform functions by taking inputs from the controllers and outputting the result through the display, speakers, or LEDs.

The components used within the project are listed below:

- Controllers
    - Potentiometer
    - KBD directional joystick
    - KBD select button
    - Reset button
- LCD Screen
- Speakers
- Port LEDs

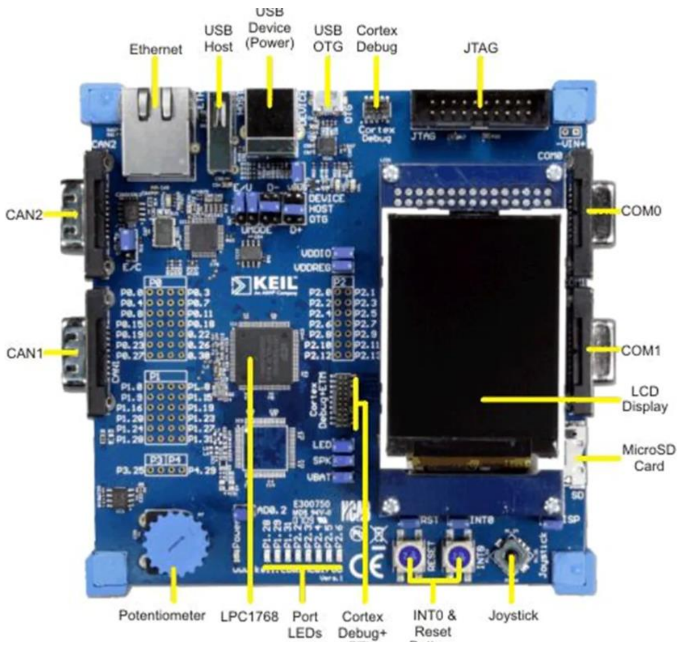

Figure 1: MCB1700 board

B. *Software Components*

The software component is needed to execute a set of instructions that are processed by the user. The software used in this project was Keil uVision which is coded in the C programming language.

Both hardware and software components are interlinked and connected to create a functional system. The software constantly waits for the user input to perform the necessary actions by sending signals to the hardware. The hardware performs the required action by retrieving the information from the software and outputting the results.

III. Design

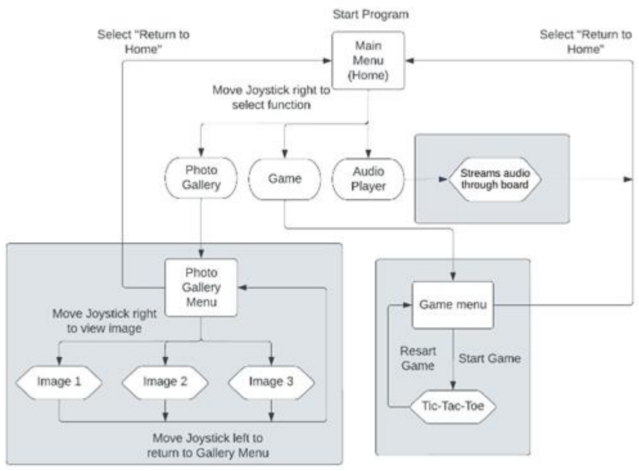

Figure 2: Block Diagram of Multimedia Center

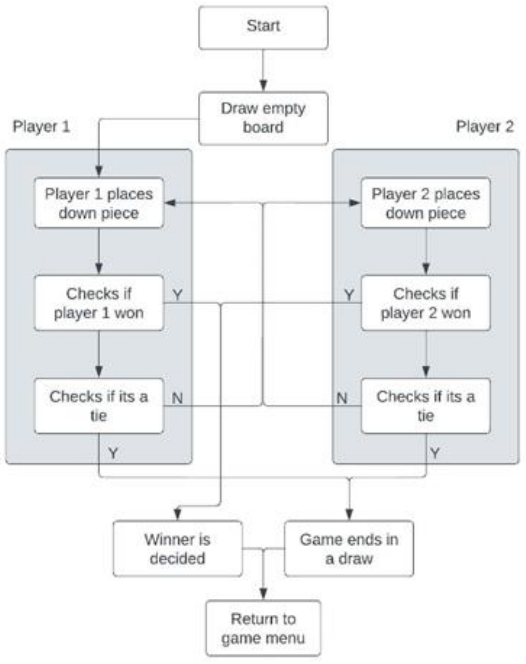

Figure 3: Block Diagram of Tic Tac Toe Game

IV. Methodology

C. *Main Menu*

The menu consists of 3 function names: Photo Gallery, Audio Player, and Game. It also includes instructions on how to navigate with the joystick. Up and down allows the user to cycle through each function, and right to select the current function that is being hovered over. This is indicated by highlighting the function name in green.

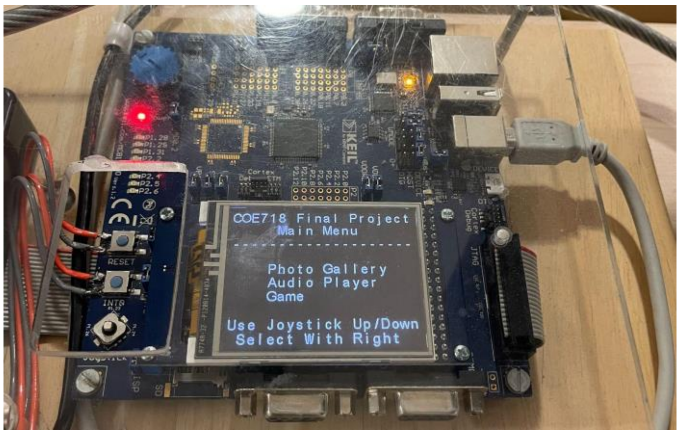

Figure 4: Main Menu when the application is first booted up

D. *Photo Gallery*

When the Photo Gallery is selected, it calls a new method called photo_gallery() which contains the code for creating the gallery menu and displaying the image. Using a method instead of having all the code in one file makes the program more readable, and easier to understand, maintain, and debug. The Photo Gallery works similarly to the menu, allowing the user to navigate through the 3 images and select one. When an image is selected, it is displayed on the LCD using the function, and LEDs 1-3 are turned on respective to which image is being viewed. In addition, to return to the photo gallery, the user would move the joystick to the left. When returning to the main menu, the user selects to return to home, and the photo_gallery() function returns control to the main c file.

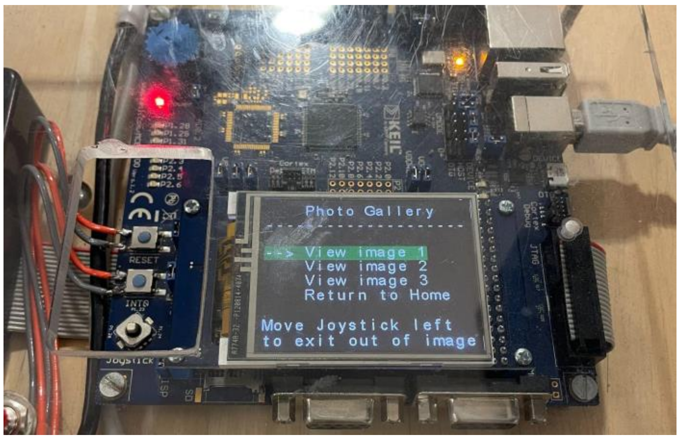

Figure 5: Photo Gallery Menu

Several steps were taken to import an image and display it on the MCD1700 Board. First, the image was loaded into the program called GIMP (GNU Image Manipulation Program). Within this program, it was rotated by image 180 degrees since the board draws the image upside down and scaled it down to about 225x225 pixel images for the full image to display correctly. Next, the image was exported as a c source file and added to the project directory. To use this file, it had to be included in the file in which the image wanted to be drawn in. Finally, the function GLCD_Bitmap (width position, height position, width dimension, height dimension) is called to draw the image.

To place all of the images in the middle of the LCD, the image dimension was used to calculate how much it needed to be offset in both the x and y direction. To center the image horizontally, the maximum width of the LCD, 320, was subtracted from the width of the image, then divided the result by 2. To center the image vertically, the maximum height of the LCD, 240, was subtracted from the height of the image, then divided the result by 2. This aligned the image directly in the middle of the LCD

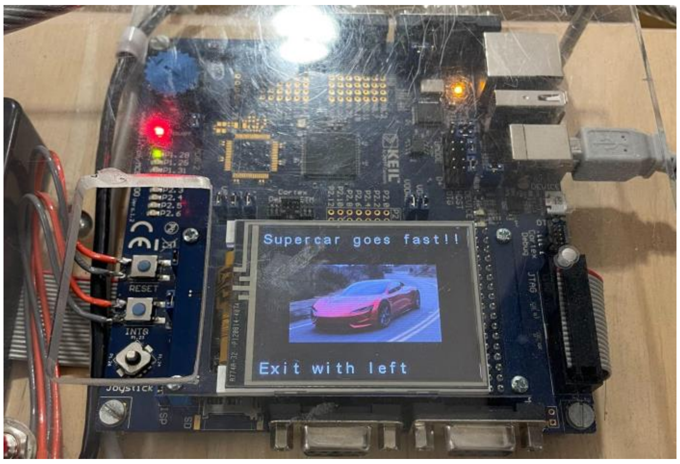

Figure 6: Displaying Image 1

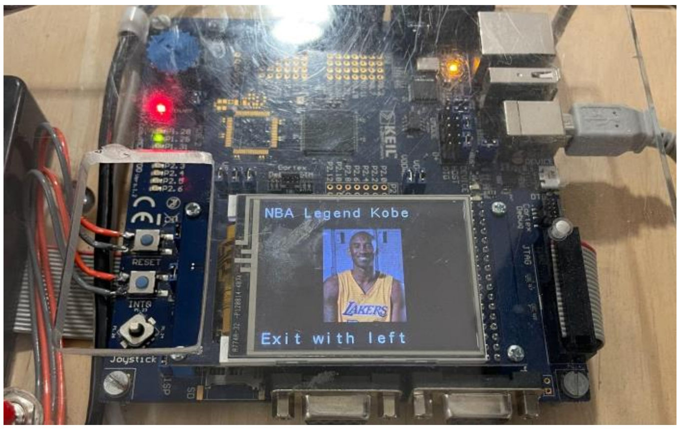

Figure 7: Displaying Image 2

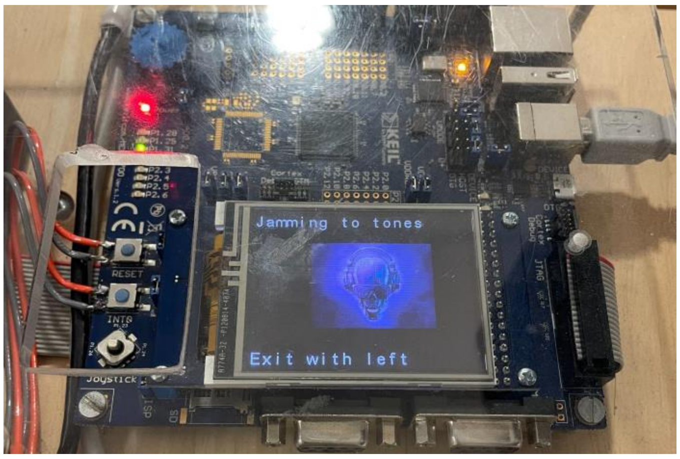

Figure 8: Displaying Image 3

E. *Audio Player*

When the audio function is selected, it calls a function called audio_player() which streams audio via USB from the PC to the board, for example playing music through a YouTube video. The audio is streamed using the onboard speakers connected to it. The potentiometer can increase the volume of the audio when turned counter-clockwise and decrease the volume when turned clockwise. T ability for the 8 LEDs on the board to indicate how loud or low the volume is when the potentiometer is turned was also added. LEDs are turned on as the volume increases and turned off as the volume decrease. This was implemented by reading the value of the potentiometer and turning the LED off or on based on the range of the value.

To turn off the audio, the board reads the speakers from the computer as the board's speakers and enables the USB connection to allow media to play through the board instead of the computer. To stop the board from using the speakers, the user must move the joystick to the left, and a function called NVIC_SystemReset() is initialized. This function triggers a soft rest in the AIRCR register. This includes resetting the method to turn on and use the speakers. Once the audio is turned off, it breaks out the function audio_player() and returns to the main menu

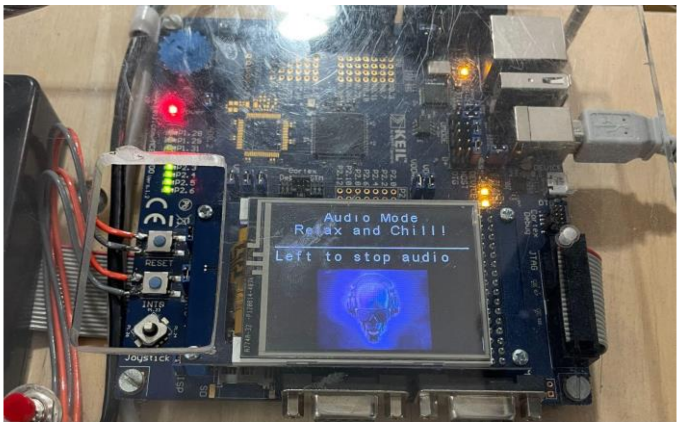

Figure 9: Audio Player Screen

F. *Game (Tic-Tac-Toe)*

When the game is selected, it calls the function game_menu() which brings the user to the menu for the game. This menu has a start game and a return-to-home option.

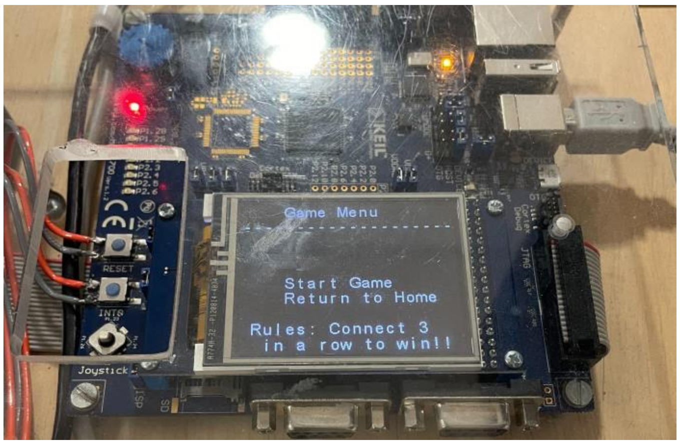

Figure 10: Game Menu

1) *Game Rule*

The game is played on a 3 by 3 square grid. Two players take turns placing down game pieces, X and O, in the empty squares. The winner is the first player to get 3 pieces in a row either up, down, across, or diagonally. This comes out to a total of 8 possible ways to win. If all 9 squares are full and no player connects 3 of their pieces in a row, the game ends in a tie.

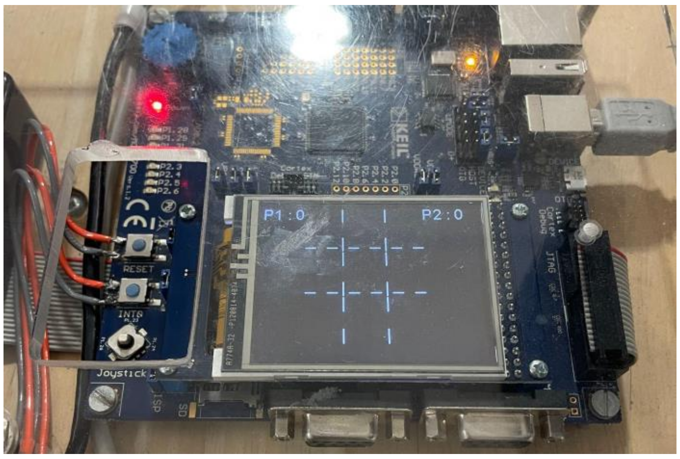

Figure 11: Empty Tic Tac Toe board

2) *Tic Tac Tow Table*

Once a game starts, an empty 3 by 3 square grid is drawn on the board. To make sure that the pieces are placed in the correct spots, a grid of the coordinates was created which contains the spaces in which only the players can place a game piece down.

(1, 5)	(1, 9)	(1, 13)
(4, 5)	(4, 9)	(4, 13)
(7, 5)	(7, 9)	(7, 13)

Table 1: Defined coordinate space for Tic-Tac-Toe

Table 1 represents the exact places on the LCD where the game pieces could only be placed in. By default, the game starts at the coordinate (1, 5) and is defined by setting the values 5 and 1 to x and y respectively. The values of x or y are then incremented or decremented by a value that will move the cursor to the next defined coordinate space. In this case, y increases or decreases by a value of 3, and x increases or decreases by a value of 4. To prevent the cursor from leaving the defined coordinate space, conditions were made to ensure that the value of y and x do not exceed their maximum values or go below their minimum value. This would stop incrementing or decrementing the values of x or y.

3) *Keeping track of the pieces*

A two-dimensional array was utilized to track where each game piece is placed on the tic-tac-toe board. The two-dimensional array is like a matrix with rows and columns. When declaring the array, a 2D array size was set that would contain all defined coordinate space for the Tic-Tac-Toe board. At the being of each game, the array is initialized with all zeros to indicate that it is empty. Each time a game piece is drawn on the board, the same x and y coordinates are stored within the 2D array with a placeholder value greater than 0. Player 1 has a placeholder value of 1 and Player 2 has a placeholder value of 2. This placeholder value keeps an update on when a game piece is placed in one of the defined coordinate spaces and indicates which player placed it. Now an if statement can be used to then check whether the defined coordinate space is already occupied with a placeholder value of 1 or 2. This essentially prevents either player from trying to play a game piece in an already occupied defined coordinate space and overwriting it.

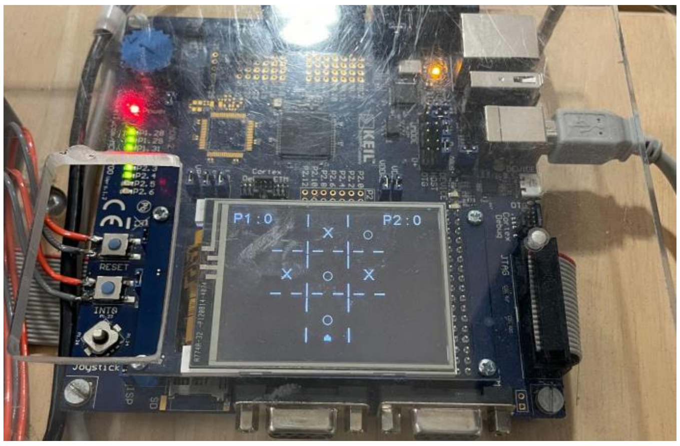

Figure 12: Example game 1

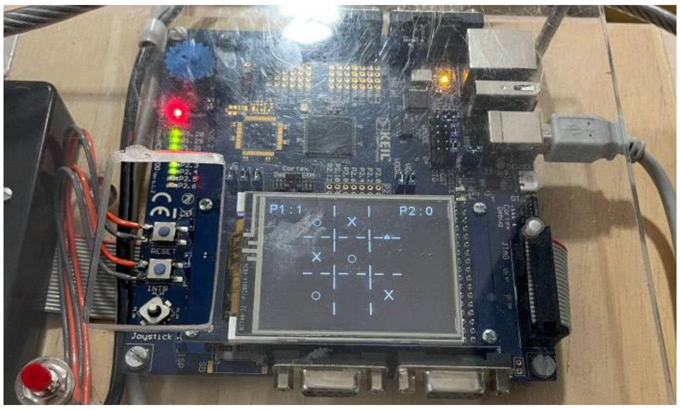

Figure 13: Example game 2

4) Determining win/lose/draw

Now that a two-dimensional array is implemented to keep track of where each game piece is placed and by which player, the game can now determine when a player wins/loses and when there is a draw. The placeholder values are assigned each time a game piece was placed. An “if” statement was created that checks all 8 possible ways of winning for each player. A player is declared a winner when the placeholder value is the same in 3 rows for any of the possible ways to win. Once a player wins, the game returns to the game menu and displays a message on the board stating who the winner is. In the case where no player connects 3 of their pieces in a row, the game returns to the game menu and states that it’s a draw.

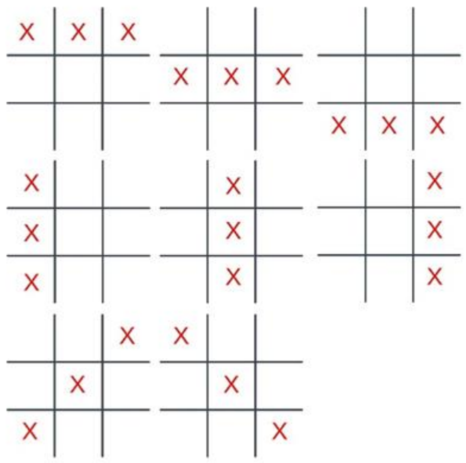

Figure 14: Tic Tac Toe winning algorithm (8 arrangements)

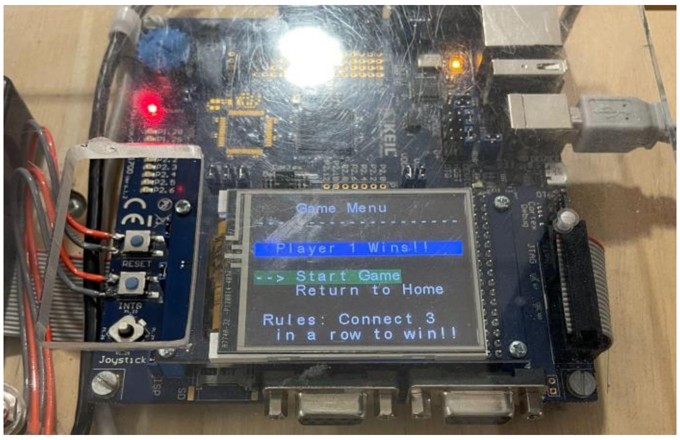

Figure 14: Player 1 wins

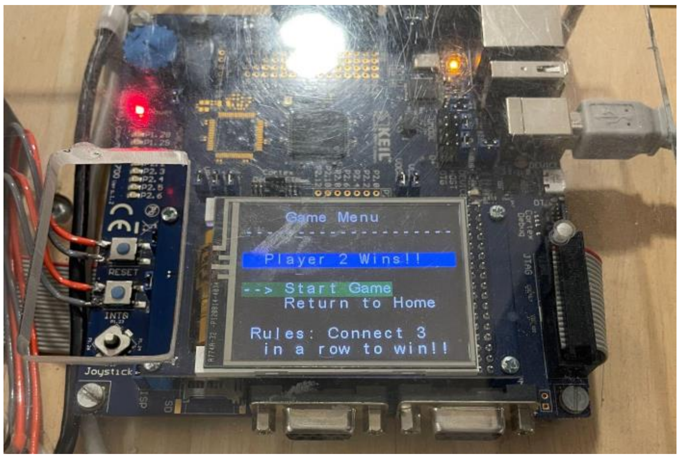

Figure 15: Player 2 wins

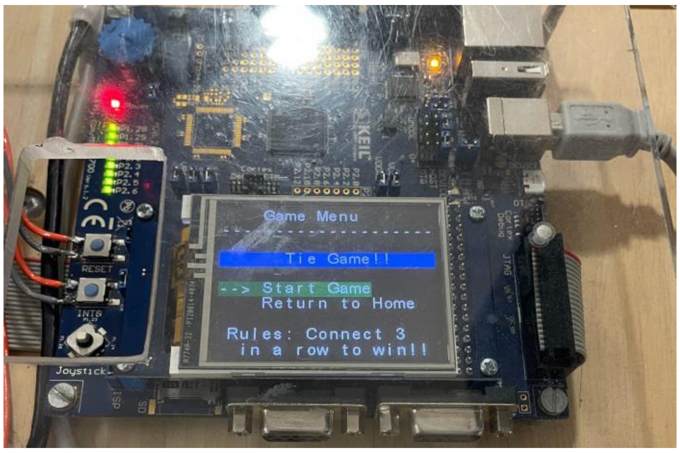

Figure 16: Resulting in a Tie

5) *Tracking scores*

To keep track of the score, separate variables were created, P1 and P2, and equated to 0 for player 1 and player 2 when the program begins. The score can be found in the game in the top left for player 1 and the top right for player 2. When a player wins, their variable increases by 1. If there is a tie, the variables are unchanged. The score of each player is kept tracked throughout the entire program as it is stored as a global variable and is only reset when clicking the reset button on the board.

6) *Hovering features*

The hover feature is used to indicate where the player is about to place their game piece. Some initial issues occurred when first creating the hover feature. The main purpose of the hovering feature is to draw a char that implied a cursor in the defined coordinate space every time the joystick was moved left, right, up, and down and cleared the previously placed cursor. This gave it the effect of moving the joystick throughout the game board. The issue with this was that it would overwrite the drawn game pieces that were placed on the board. This design implementation was flawed as it made it impractical to keep track of where the pieces were placed on the board using the LCD and were still tracked with the 2d placeholder matrix. To fix this issue, decided to draw the cursor directly 1 line underneath the defined coordinate space. To define the places where the cursor can be drawn, the defined coordinate space was received and added 1 to the y direction. This new approach allowed the game to indicate to the player where they are offering over on the game board and still place their game pieces without getting them overwritten.

V. CONCLUSION

In conclusion, the program had a fully functional media center that uses both hardware and software. The knowledge gained through the semester lectures and labs helped tremendously with developing the project. This includes understanding the MCB1700 Board features, uVision concepts, debugging, multi-threading, using divide and conquer to solve big problems, and hardware and software integration. The media center included an mp3 player, a photo gallery, and an animated game that fully represented a media center that was user-friendly, simple, and fun to interact with. This project gave me insight into real-life projects and it was a great experience overall.

VI. Reference

[1] "CMSIS-RTOS Version 1.03," *RTX implementation*. [Online]. Available: https://www.keil.com/pack/doc/CMSIS/RTOS/html/rtxImplementation.html. [Accessed: 01-Dec-2022].

[2] "COE 718: Embedded Systems Design," *Labs and Project*. [Online]. Available: https://www.ecb.torontomu.ca/~courses/coe718/lab-project.html. [Accessed: 01-Dec-2022].

[3] H. K. Shaik, "How to create a tic-tac-toe game in python?," *Geekflare*, 08-Nov-2021. [Online]. Available: https://geekflare.com/tic-tac-toe-python-code/. [Accessed: 01-Dec-2022].

[4] *C Tic Tac Toe game*. YouTube, 2021.