Media Center

COE718: Embedded Systems Design

Harrence Mathialagan(500559356)

I. Abstract

The purpose of this project was to design a media center on the Keil MCB1700 board. The requirement for the media center included displaying images in the photo gallery menu, streaming audio from the computer via USB, and implementing a video game. The on-board joystick was used to navigate through the main menu, photo gallery, and video game. The LCD was used to display the graphical representation of the media center. The software compiler used to program the media center was the Keil µVision IDE. The programming done in C language was loaded onto the board. The photo gallery consisted of three pictures that were scaled at 320 by 240 pixels. The audio stream consisted of a potentiometer that adjusted the volume of the sound. Lastly, the video game features five pictures that will be further discussed in the game implementation section.

II. Introduction

The objective of this project is to implement a media center on the MCB1700 that allows a user to access basic media functionalities. The functionalities include Music player, Photo gallery, and a game. The user should be able to play and adjust the volume of the music as well as being able to play a game. Due to a limited amount of memory on the board, there are restrictions on the number of pictures that can be uploaded onto the board since the built-in flash memory is 512 KB. The interim stage demonstrated successful outcomes of the menu photo gallery and audio stream. The final stage of this project was to implement the game successfully in addition to the interim stage and the outcome was successful.

The media center project was designed on the Keil MCB1760 board that was based on the NXP LPC1768 controller which features an ARM Cortex- M3. The ARM Cortex is known for features such as tail chaining, bit banding, and conditional executions. The only drawback that must be considered is the flash memory on the board. There is an SD card expansion slot that allows the user to add more pictures onto the board. Some of the main built-in hardware include a joystick with five buttons that were connected to pins P1.20, P1.23, P1.24, P1.25, and P1.26. The board consists five ports ranging from P0. XX to P4. XX, where the XX is the pin number. Based on the joystick, it is evident that the pins correspond to 20, 23,24, 25, and 26. There are 70 General Purpose Input Output (GPIO) pins on the board. The pins on port one and two were used to generate interrupts. A 320 by 240 Quarter Video Graphics array LCD was also connected to the board via 30-pin header. Other built-in hardware includes Potentiometer, LEDs, and speaker.

III. Past Work And Review

Prior to implementing the media center, there were certain embedded system programming that was learned throughout the course's labs and lectures. The Keil µVision IDE played a big role when it came to understanding how the debug tools

worked. It made the implementation of the media center possible to complete.

The Keil µVision IDE was thoroughly introduced in the labs. Lab1 was very informative when it came to understanding the software, and it's debugging tools. The concept of GPIOs in relation to LEDs were understood. Assigning LEDs states, reading joystick position, and displaying characters on the LCD display were learned throughout the lab sessions and demonstrated with clarity.

There were some key performance efficiency features of the ARM Cortex M3 processor that were also understood through the lab sessions. Bit-banding, Conditional branching, and barrel shifting were features that can be used to minimize the execution time of a program. Once again, these features were learned throughout the lab's session and demonstrated with clarity with the use of LED, LCD and joystick. The bit banding LED application that was learned in the lab sessions were implemented in the media center to make the program more efficient.

Understanding how to display strings of characters in various colors was an essential knowledge that was needed for the media center. Without graphical representation on the LCD, the media center project cannot be created with creativity.

The final concept that was learned in the labs were Real Time Operating System (RTOS) applications. These applications consist of task scheduling and multithreaded scheduling, and real-time scheduling. The concepts can also be applied to reduce computation times. However, these concepts were not implemented in this project as it was unnecessary

IV. METHODOLOGY

The project started off by creating a new project workspace and adding necessary headers and C files from previous labs into the workspace. The necessary files included files for the peripherals and music application.

Main menu:

The objective of the main menu was to design something user-friendly and easy to use. The main menu consists of the four options: "Photo Gallery", "Music Player," "Game 1", and "Game 2". The selection of each option is shown as an arrow pointing towards it. The user should be able to navigate through the menu options using the joystick. The main menu was the first thing that was created so that other options can branch through that main menu function.

Photo Gallery:

The photo gallery is accessed when the user presses the joystick. The photo gallery displays 320 by 240 pixel pictures. The user can view up to three different pictures using the left and right function of the joystick to navigate between pictures. Once the user presses the joystick, it will exit the application and go back to the main menu.

Music Player:

The next task was to implement a music player. A USB connection is required to stream audio from the computer and onto the board. Once, the music application has been initiated; the connection is set on the computer, the computer would be able to access the onboard speaker. The files that are required to enact the USB connection were obtained from the course directory. The files were then added to the IDE workspace and the project directory. To resolve the conflict of having two main functions in a project, the main function of the mp3 file is renamed to music. The music function is accessed once the user selects the music application in the main menu. After the application is selected, pressing the

joystick again would disable the mp3 player. An interrupt handling function is added to the mp3 file along with the music function, to execute the stop. The onboard potentiometer can be used to control the volume. Furthermore, the design section will discuss the execution of the stop method on the music player in depth.

Video game:

The video game implementation was the last stage of the completion of the project. This was also the most time consuming and complex part of the project. The two games that were implemented involved pixel manipulation and constant shift of images on the LCD. The video game uses the joystick to shift images accordingly.

The first game that was implemented was the fruit catch game. The purpose of this game was to catch the fruit with a basket. There were two images involved in the creation of this game. A picture of a bucket, cherry, and bomb were retrieved from the internet in bitmap format and converted to C files to be used in the game. The objective of the bucket was to move left and right ranging from the width of the LCD to catch the cherry and avoid the bomb. If the bomb was caught then the user loses. Also, if the cherry is missed the user loses as well. The joystick left, and right function allow the images to be shifted left and right. The cherry is the fruit that is falling towards the bucket at the bottom of the screen. Every time the bucket catches a cherry, the score is indicated with the onboard LEDs. The score system uses the bit-band function to turn the LEDs On and Off for score keeping purposes. When the bucket is able to catch the cherry, the next cherry will be falling with greater speed than the previous cherry. This is achieved by manipulating delays between drawing the images every time the vertical position of the image is being changed. Since the bucket, cherry, and bomb constantly change their position on the LCD, there would be multiple residual images displayed on the screen due to redrawing the images on the LCD. The previous images will be over written by the shifted images. Since the shifted images do not match the position of the previous image; there would be a residual version of the previous image. Hence a white blank picture was used on the left and right side of the bucket image. When the bucket image is shifted, the blank images will also shift, thus eliminating the residual image of the previously positioned image. The same concept applies for the cherry and the bomb but there would only be one white image being drawn from above them.

```
//bucket movement
if (buttons == KBD_RIGHT  && bucket!=290){
    bucket = bucket+10;
    buckclearleft=buckclearleft+10;
  buckclearright=buckclearright+10;
  GLCD_Bitmap (buckclearright, 200, 30,  29, whitebucket);
    GLCD_Bitmap (buckclearleft, 200, 30,  29, whitebucket);
    GLCD_Bitmap (bucket, 200, 30,  29, Bask);
}

if (buttons == KBD_LEFT  && bucket!=7){
    bucket = bucket-10;
  buckclearleft=buckclearleft-10;
  buckclearright=buckclearright-10;

  GLCD_Bitmap (buckclearright, 200, 30,  29, whitebucket);
GLCD_Bitmap (buckclearleft, 200, 30,  29, whitebucket);
    GLCD_Bitmap (bucket, 200, 30,  29, Bask);
}
//cherry movement
cherryy = cherryy +i;
cherryclear=cherryclear+i;
delay(5);
GLCD_Bitmap (cherryx, cherryclear, 30,  29, whitebucket);
GLCD_Bitmap (cherryx, cherryy, 20,  30, cherry);


//bomb movement
bomby = bomby +j;
bombclear=bombclear+j;
delay(5);
GLCD_Bitmap (bombx, bombclear, 30,  29, whitebucket);
GLCD_Bitmap (bombx, bomby, 20,  30, bomb);
```

Figure 1: Basket, Cherry, and Bomb Movement.

The second game that was implemented was the cup game. The objective of the cup game was to guess which cup had the ball. This game uses the same score keeping concept as the previous game where bit banding is incorporated onto the LEDs. There are three images involved in the making of the game. It includes a ball,

cup and an arrow. The ball is printed in a certain cup and it is up to the user to guess the cup that has the ball. The arrow is used to indicate which cup to select and the left and right joystick functions are used to point the arrow at the designated cup. The user chooses the corresponding cup with the joystick. The shifting concept from the above game was partially included in this game. Since the position of the images are fixed, a blank image is used to overwrite the corresponding image. This will ensure constant movement of the arrow and the upward movement of the cup once it is being selected. Once the joystick is pressed the corresponding cup will be lifted to see if the ball is there or not. Based on the outcome, the score will be shown on the on-board LED and the next round would start. There are two rounds in total.

V. Design

Image conversion:

The media center is composed of multiple pictures. There are three images in the photo gallery, and six images in the two games. As mentioned before, to incorporate the images on the board the image must be converted to C. The bitmap function in the GLCD.h file prints the images on to the LCD based on the hex values on the converted image file. The original file formats of the images varied from JPEG to PNG. Those file formats were initially converted to bitmap images prior to converting to C files. If the images were not converted to bitmap, the LCD would not be printing the image properly because the GLCD_Bitmap function can only process bitmap formatted images. Since the size of the LCD was 320x240, the images had to be scaled accordingly prior to C file conversion. The C conversion process is done through the GIMP software. Initially, the images are flipped before converting it due to the GLCD_Bitmap function. However, in the finalizing stage of the code, the GLCD function was modified accordingly to make sure the pictures aren't flipped.

```
void GLCD_Bitmap (unsigned int x, unsigned int y, unsigned int w, unsigned int h, unsigned char *bitmap) {
  int i, j;
  unsigned short *bitmap_ptr = (unsigned short *)bitmap;

  GLCD_SetWindow (x, y, w, h);

  wr_cmd(0x22);
  wr_dat_start();
  for (i = 0;i < (h-1)*w; i += w) {
    for (j = 0; j < w; j++) {
      wr_dat_only (bitmap_ptr[i+j]);
    }
  }
  wr_dat_stop();
}
```

Figure 2: Modified the outer for loop.

The above figure shows the modification that was done the ensure the image won't show as the flipped version on the LCD.

Main Menu:

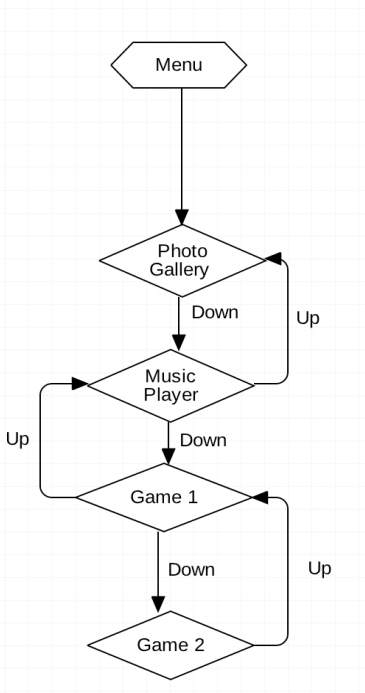

Figure 3: Flowchart for Main Menu.

The above figure shows the overall program design of the menu function. Moving the joystick up or down will navigate through the options.

```
if (joystick==KBD_DOWN && option != 4){
  option++;
}
if (joystick==KBD_UP && option != 1){
  option--;
}
```

Figure 4: Code for navigating through options on the main menu.

The above figure shows the code implementation of the menu navigation function. Two conditional statements are used. The first conditional statement checks if the joystick is pushed down and if it is not pointing to the last option (game2). If those conditions are true, then it goes to the next option below the current option. The second conditional is pretty much the opposite of the first conditional statement. The statement checks if the joystick is pushed up and if it is not pointing to the first option (photo gallery). If that statement is true, then it will go the next option above the current option.

Photo Gallery:

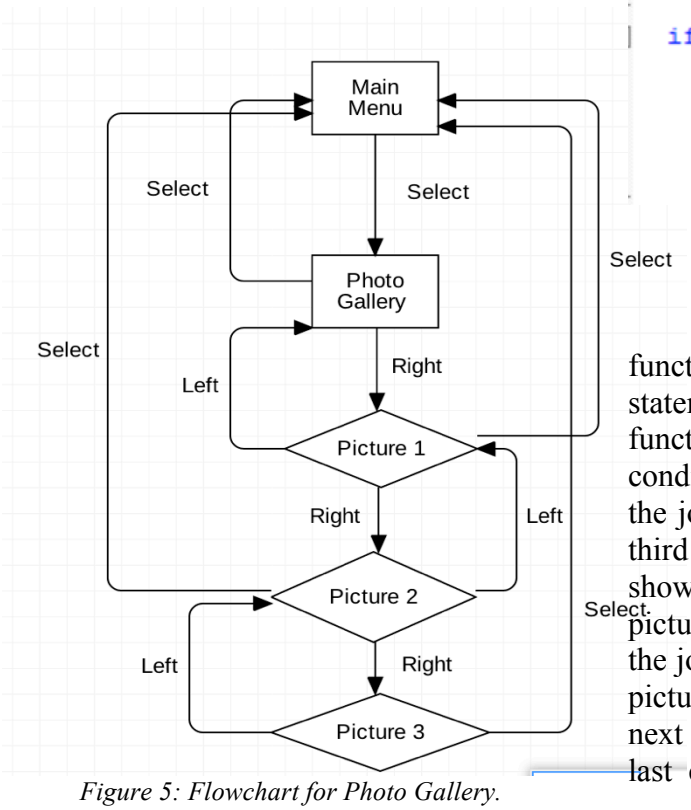

Figure 5: Flowchart for Photo Gallery.

The Figure shows the process of the photo gallery. The Photo gallery has a somewhat similar process as the Main menu option. Instead of the up and down function of the joystick, the left and right function are being used. The three pictures are scaled at 320 by 240 pixels and can be navigated by pushing the joystick left and right. At any moment, if the user decides to press the select option of the joystick, it will go back to the main menu.

```
void photogallery(){
  while(1){
        joystick = get_button();

    if (joystick==KBD_RIGHT && option != 3){
      option++;
    }
    if (joystick==KBD_LEFT && option != 0){
      option--;
    }

    if (option==1){
      GLCD_Bitmap(0, 0, 320, 240,BMW);
    }
  if (option==2){
      GLCD_Bitmap(0, 0, 320, 240,GTR);
    }
  if (joystick==KBD_SELECT ){
    delay (10);
      menu();
    }
```

Figure 6: Function of the photo gallery.

The above figure dictates the function of the code. Two conditional statements are used just like the main menu function in addition to the joystick select condition. The first condition implies that if the joystick is pushed to the right and if the third picture is not shown on the LCD, it will show the next picture below the current picture. The second condition implies that if the joystick is pushed to the left and the first picture is not shown, then it will show the next picture above the current picture. The last condition states that if the joystick is

pressed at any state the main menu screen will load up on the LCD.

Music Player:

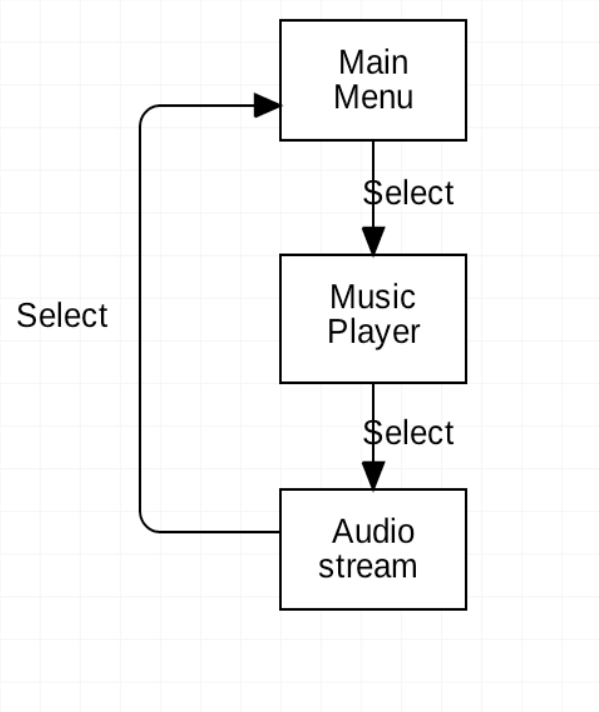

Figure 7: Flowchart for music player.

The music player was the simplest design to implement since all the files were given. The only thing that was implemented was the interrupt. Based on the above figure, it is evident that the select function of the joystick is used to navigate in and out of the music player. Pressing the music player option to initiate the music player screen. Pressing the joystick again will initiate the onboard USB audio stream. Lastly, pressing the joystick once more will make the menu screen load up.

```
void musicmenu (){
  GLCD_Clear(White);
while (start3==true){
  joystick = get_button();
      GLCD_SetBackColor(Black);
      GLCD_SetTextColor(White);
      GLCD_DisplayString(0, 0, __FI, "    MUSIC PLAYER            ");
      GLCD_SetBackColor(White);
      GLCD_SetTextColor(Black);

      GLCD_DisplayString(3, 4, __FI, "INSTRUCTIONS:");
      GLCD_DisplayString(5, 2, __FI, "ADJUST POTENTIO");
      GLCD_DisplayString(6, 4, __FI, "FOR VOLUME ");

  if (joystick == KBD_SELECT)
  {

    start3=false;
music();
  start3=true;
```

Figure 8: Music is being called from the main file.

The main function in the usbdmain file had to renamed to music because it created a conflict with the main function in the menu file.

```
  if(get_button()== KBD_SELECT){
  NVIC_DisableIRQ(TIMER0_IRQn);
    NVIC_DisableIRQ(USB_IRQn);
    USB_Connect(0);
    USB_Reset();
    USB_Connect(1);
    menu();


  }
```

Figure 9: Interrupt For music player.

The above figure represents the interrupt feature used to get out of the audio stream. It is placed at the end of the TIMER0 interrupt handle in the usbdmain.c file. The nested vector interrupt controller disables the interrupt request for TIMER0 and USB. Hence, the USB connection is disconnected, reset, and connected again. This occurs if the user decides to exit the music player application by pressing the joystick.

Video Game:

Game 1: Cup Game

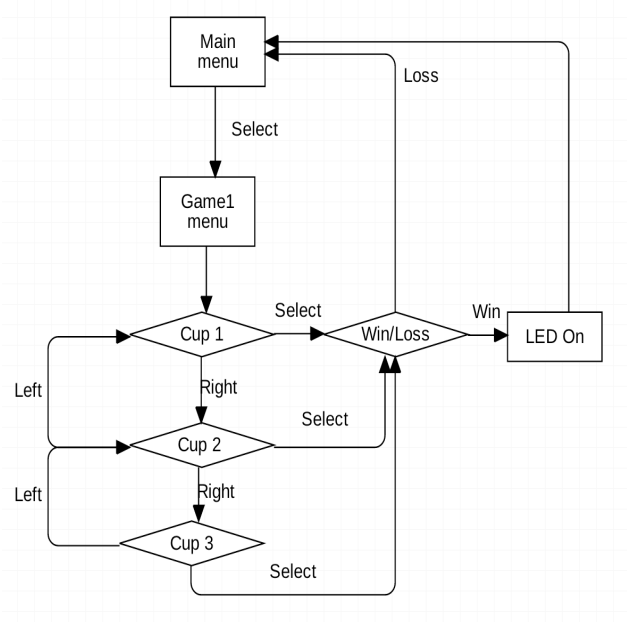

Figure 10: Flowchart for Cup Game.

This figure shows the implementation of the first game. Moving the joystick left and right will point an arrow at the corresponding cup. In each cup's decision-making process, if the joystick is pressed the screen will show winner if the ball is under the cup, if not it will say loser and will go to the next round. The next round follows the same process and will exit back to the main menu.

```
if (button==KBD_RIGHT && option1 != 3){

  option1++;
}
if (button==KBD_LEFT && option1 != 1){
  option1--;
}
```

Figure 11: Joystick Movement.

The same joystick movement conditions from the photo gallery and main menu are used to navigate through the cups.

```
    //option 2
  if (option1==2){
      GLCD_Bitmap (30, 100, 30,  29, whitebucket);
      GLCD_Bitmap (240, 100, 30,  29, whitebucket);
      GLCD_Bitmap (140, 100, 30,  29, arrow);
if (button==KBD_SELECT ){
    GLCD_Bitmap (140, 200, 30,  29, whitebucket);
    GLCD_Bitmap (140, 100, 30,  29, cup);
      d(50);
    GLCD_DisplayString(6, 0, __FI, "    Loser   ");
      d(100);
    GLCD_SetBackColor(White);
      GLCD_ClearLn  (6,__FI);
    GLCD_SetBackColor(Black);
       i++;


    }
    }
  //option 3
    if (option1==3){
      GLCD_Bitmap (140, 100, 30,  29, whitebucket);
      GLCD_Bitmap (240, 100, 30,  29, arrow);


    if (button==KBD_SELECT ){
    GLCD_Bitmap (240, 200, 30,  29, whitebucket);
    GLCD_Bitmap (240, 200, 30,  29,BASKETBALL_pixel_data);
    GLCD_Bitmap (240, 100, 30,  29, cup);
    d(50);
    GLCD_Clear(White);
      LEDP1_28=1;
    GLCD_DisplayString(6, 0, __FI, "    Winner!   ");
    score++;

    d(100);
    GLCD_SetBackColor(White);
      GLCD_ClearLn  (6,__FI);
    GLCD_SetBackColor(Black);
    i++;
    }
```

Figure 12: Conditional Statements of Cup Game.

This figure represents a sample of the game's first round. Simple If conditions are used to check which cup the arrow is being pointed at. If the select button is pressed then the corresponding outcome for the if condition will appear (either winner or loser).

Game 2: Fruit Catch Game

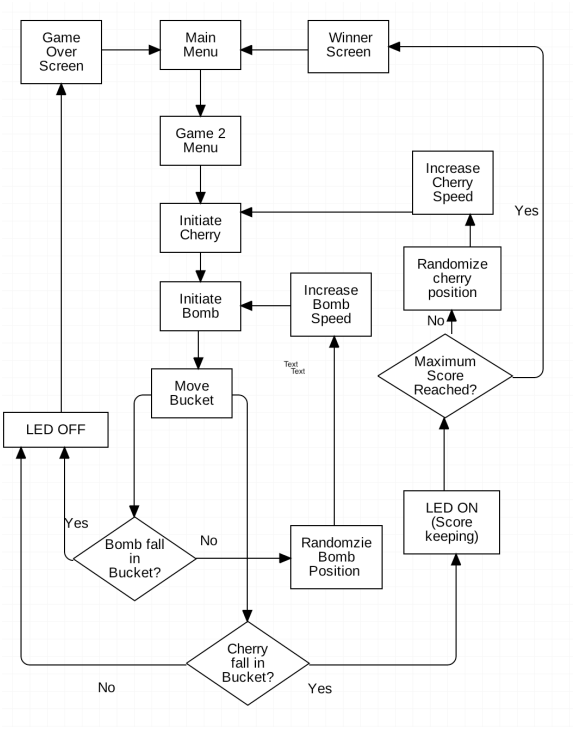

Figure 13: Flowchart for Fruit Catch Game.

Based on the figure, it was evident that the second game was more tedious the first game. The cherry will initiate from the top of the LCD and will move down vertically towards the bucket. If the bucket was able to catch the cherry, then the corresponding LED will turn on for score keeping purposes and the cherry will be initiated again with a greater falling speed and positioned randomly. If the bucket was not able to catch the cherry, then the screen will be redirected to the menu. It is also evident that if the bucket catches the bomb, then screen will also redirect to the menu. If not the game will continue and the bomb will be initiated again with a greater speed than before at a random position.

```
//bomb movement
bomby = bomby +j;
bombclear=bombclear+j;
delay(5);
GLCD_Bitmap (bombx, bombclear, 30,  29, whitebucket);
GLCD_Bitmap (bombx, bomby, 20,  30, bomb);
//bomb not caught
 if (bomby>164 &&(bombx!=(bucket)&&bombx!=(bucket+290))){
GLCD_Bitmap (bombx, bombclear+j+10, 30,  29, whitebucket);
bombx = rand()%320;
bomby=10;
bombclear=0;
j=j+1;

}

//cherry movement
cherryy = cherryy +i;
cherryclear=cherryclear+i;
delay(5);
GLCD_Bitmap (cherryx, cherryclear, 30,  29, whitebucket);
GLCD_Bitmap (cherryx, cherryy, 20,  30, cherry);

//win
if (cherryy>=164 &&(cherryx>=bucket&&cherryx<=(bucket+290))){
cherryx = rand()%320;
cherryy=10;
cherryclear=0;
i=i+1;
score1++;
```

Figure 14: Code for Cherry and Bomb movement along with their randomized location

The above figure shows how the movement, speed, and location of the cherry were implemented. The y coordinate of the cherry was displaced by variable “i” until the cherry has reached the bucket. Once the cherry has been caught by the bucket, the Rand()%320 function is called to randomize the position of the new cherry at the top and makes sure the cherry is within range of the LCD (0-320) hence, the modulus 320 attached to the Rand() function. The cherry is then cleared and initialized and the y coordinate displacement “i” of the new cherry is increased by one hence the falling rate of the cherry would be greater than the previous cherry. The bomb follows the same concept as the cherry but has the opposite purpose. The y coordinate of the bomb was displaced by “j” until the bomb has reached the bottom without touch the bucket. The Rand()%320 function is called for the bomb to randomize the position. The bomb is then cleared and initialized and the y coordinate displacement “j” of the new bomb will be set.

VI. Experimental Results

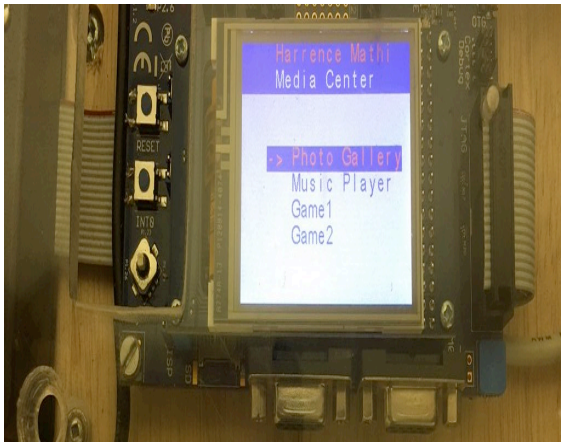

Figure 15: Main Menu pointing at Photo Gallery.

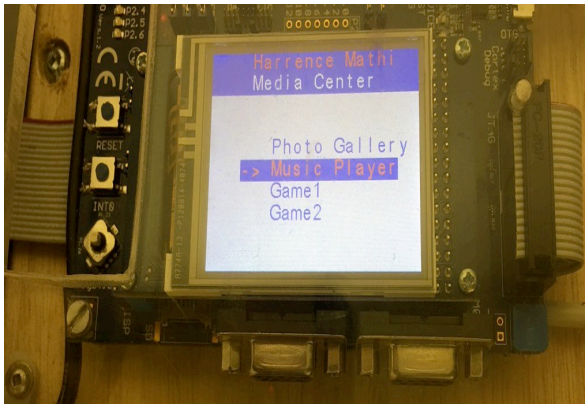

Figure 16: Main Menu pointing at Music Player.

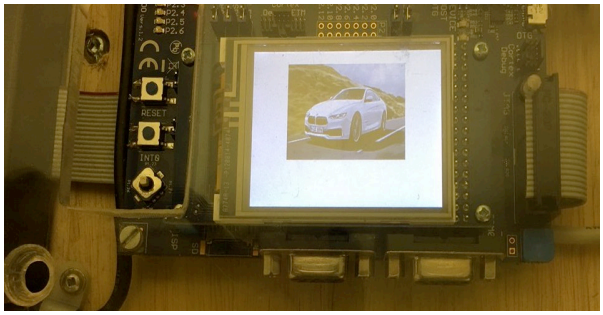

Figure 17: Photo Gallery First Image.

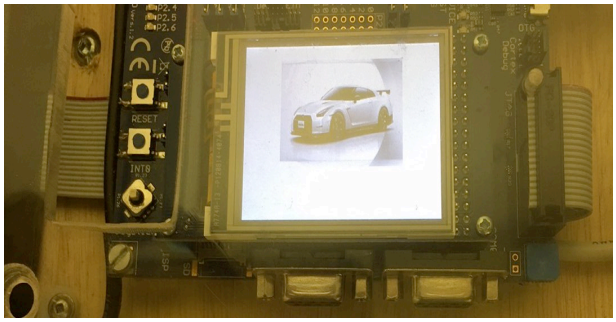

Figure 18: Photo Gallery Second Image.

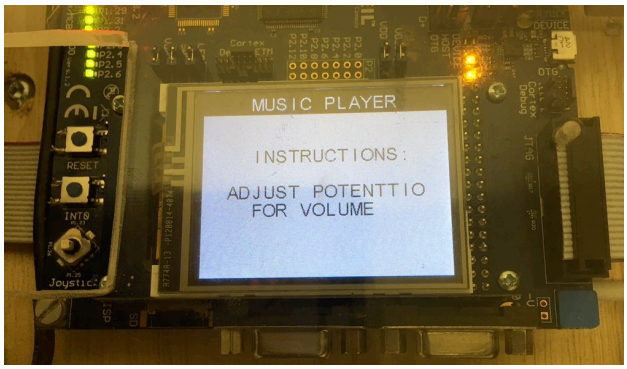

Figure 19: Music Player

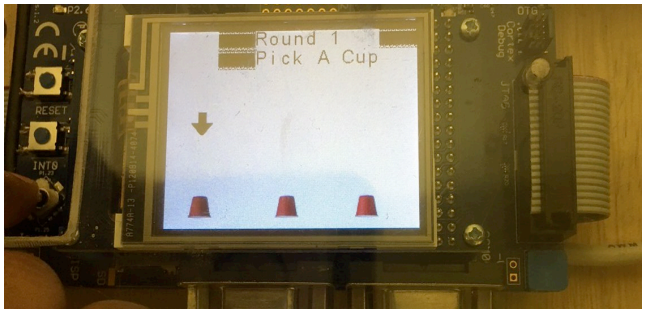

Figure 20: Cup Game

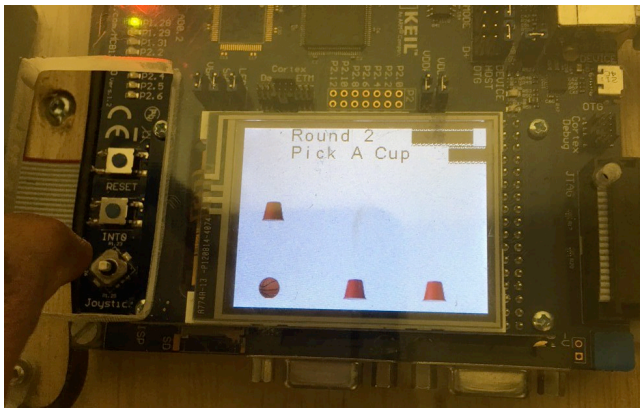

Figure 21: Cup Game Cup Selection.

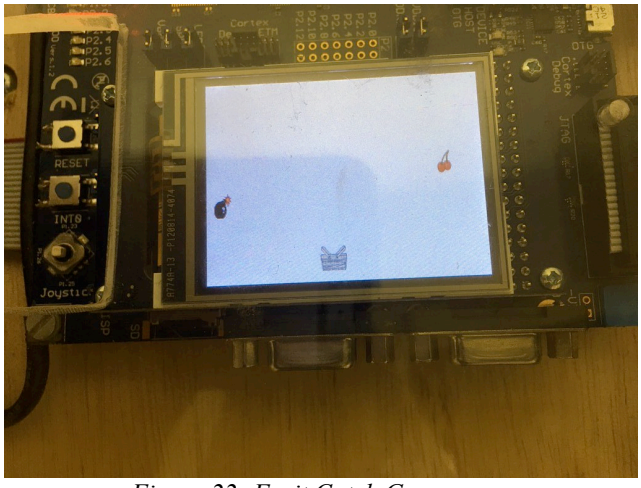

Figure 22: Fruit Catch Game.

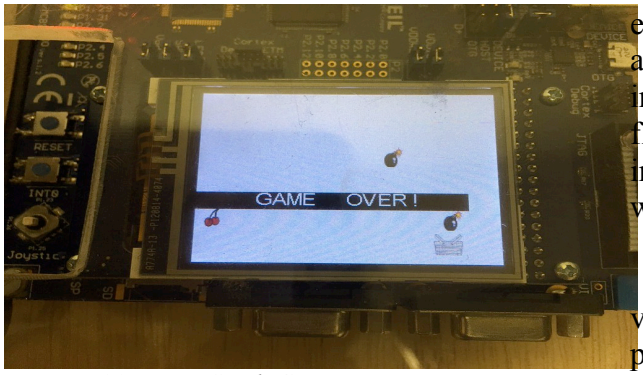

Figure 23: Fruit Catch Game at game over screen.

VII. Conclusion

The purpose of this project was to apply concepts that were taught throughout labs. Using various debugging tools in the Keil µVision IDE made the project easier to debug for errors. This course has highlighted some key features that make the Arm Cortex M3 superior to its predecessors. One feature that was implemented in this project was bit-banding. Majority of the project focused on the Graphic interface of the LCD, and the joystick function.

The photo gallery, music player, and the two video games were successfully implemented on the board. The transition between the images in the photo gallery was successfully implemented. The interrupt to exit out of the Music player function was also successfully implemented. In the game implementation, the animations worked flawlessly without leaving any residual images. Going in and out of applications worked successfully.

Embedded systems are used for various purposes in the technological purposes. Almost every single device or appliance use embedded systems. The Media center project has enhanced the knowledge of working with embedded systems. Overall the project was completed successfully given the allotted time.

References

[1] "Keil MCB1700 Evaluation Board Overview", *Keil.com, 2017.[Online].* Available: http://www.keil.com/mcb1700/. [Accessed: 21- Nov- 2017]

[2] Anita Tino, Ryerson University, Media Center Lab manual, COE718.

[3] Anita Tino, Ryerson University, COE 718 LAB MANUALS 1-4, 2016.

Appendix

{NextPage}