Project Description

The goal of this project is to create a connect 4 game that will be played using the buttons and touch screen. The user will be able to select between two modes, one player mode and two player mode. The game will conclude when a player successfully gets 4 in a row/column/diagonal. After the game concludes there will be a screen to display the time for that round and a score between the two colors. The user can restart the game and reselect the mode after the game concludes.

Project Scope

This project will utilize the following peripherals:

- RNG peripheral
- System tick peripheral (for the timer)
- Touch Screen peripheral
- Button peripheral and interrupts for it
- Additionally it will utilize a scheduler for the various events.

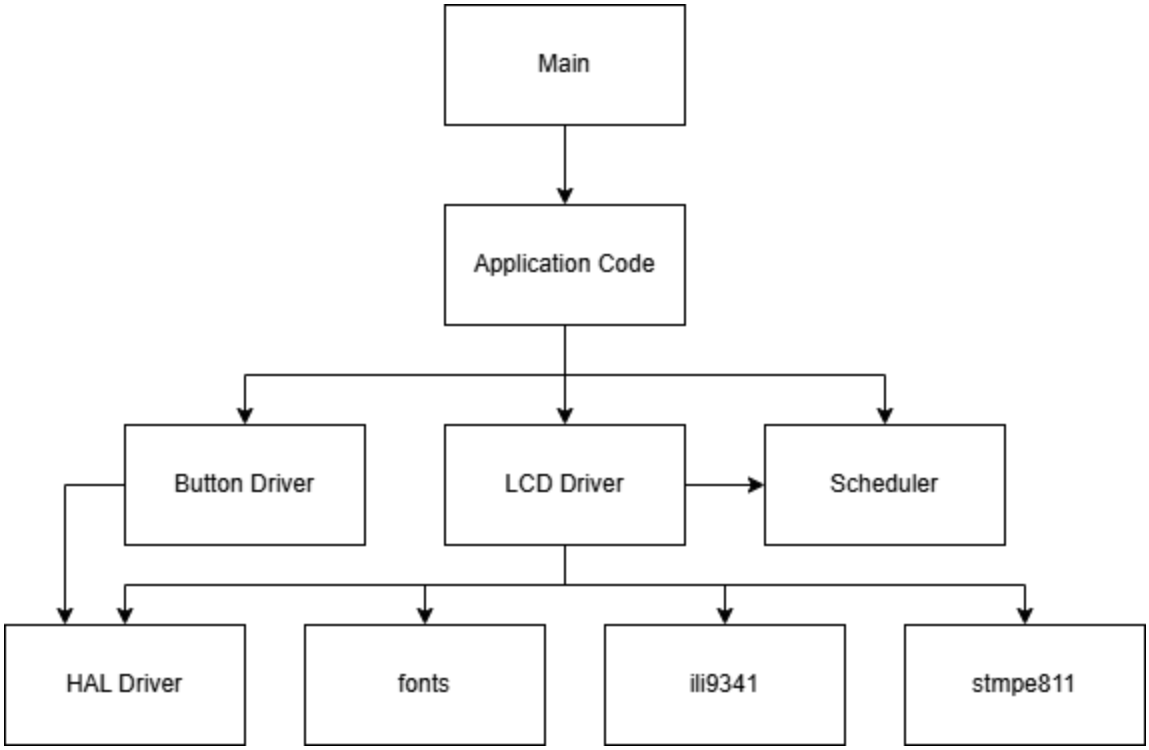
Timeline

I first started on displaying the connect 4 grid. After that was accomplished I moved onto displaying the chip dictating the players turn and incorporating touch to control the movement. When that was accomplished I moved onto making it so the button could drop a chip and making the correct logic for detecting where to place the chip when dropped and whether the user is actually able to drop the chip. Next, I worked on making the logic for detecting whether a game ends or not. When that was finished I moved onto the end screen, allowing the user to reset the game and correctly displaying the scores. When I completed that screen I worked on having the startup screen to select the mode (I did not implement the modes yet). After having the complete loop between all 3 screens I went back to add the timer. When that was completed I added an additional screen to select the player color. I forgot to do tie detection so I added this next. Then, I made it so the chips would animate when they are dropped. When that was all completed I made one player mode and had the ai drop the chip randomly using the random number generator. Finally, I had a complete project so I turned it in.

Testing Strategy

I will plan on testing each part of the project as I am going rather than waiting until the end. I will test piece by piece, so after I think I created the displays I like I will flash it to the code and see if it displays the information I actually want. I will do the same for the inputs to make sure that part of the project works before moving onto the next. I will use git and version control to quickly revert any changes that I make which break the project.

Coding Hierarchy



LCD Driver was already pointing to fonts ili9341 and stmpe811 and these files were not modified.

Code Overview

Important Functions: Most of the code utilizes functions that are created in the LCD_Driver file. In the future I would split up these functions into various files but at the time I did not.

In the ApplicationCode.h we have the interrupt service routine for the button. This is EXTI0_Handler() and will check if the game is in progress, and if it is, insert a chip if there is a spot available. Depending on whether the user is in 1 player mode, this will additionally call for the AI to perform its turn.

The most important function of the code is LCD_Insert_Chip_Game_Grid() which:

- Detects if space is available by calling LCD_Space_Available_Game_Grid()
    - This function checks whether the top row of a given column is empty, if it is then we are able to place a piece; otherwise, we can't for that column
- If there is space it will drop the piece to the lowest available empty slot in that column
- It will proceed to animate it by calling LCD_Animate_Falling
    - This redraws the chip until it reaches the correct y position
- We then update the matrix containing the information of whether is a spot has a piece to house the color of the player who just dropped a chip
- After that is accomplished, we update the hovering chip to be the other players color by calling LCD_Draw_Chip_To_Drop()
- We then want to check if the game ends by checking whether the newly placed piece results in a connect 4. This is done by calling LCD_Game_Won

    - LCD_Game_Won will return true if there is a connection 4. Since a game can only conclude on the row,column, or diagonal of the most recently placed chip, we just need them.

The other important function implemented is LCD_Find_Best_Move() which in reality is a misnomer as it just randomly picks a spot. This function:

- Uses the HAL to generate a random number then takes the modulo of that number in order to correlate it to the 7 different columns.
- It then checks whether that column is full and if it is, it will continue to generate a random number until we get a column that is not full

The LCD_Game_Tie() function:

- Returns true if there is a tie. It detects if there is a tie by looking at the top row for each column. If this entire row is filled, then there must be a tie.
- This function is called immediately after detection of whether is a player won or not

Many of the remaining functions just set up visual information for the various different screens and the code controls which of these functions are currently being called through the scheduler.

Struggles and Obstacles

One struggle I encountered was coming up with how to code the game win logic detection. Originally, I thought to loop through all possible diagonal, rows, and columns but ultimately decided not to as it was inefficient. This led me to try to come up with code that would detect all adjacent pieces to the most recently placed chip to detect a win. This would continue until it encountered a piece that was of the opposite color or an empty spot, in which case it would return to the start and go the other direction and see if both directions lead to a connect 4. Ultimately, I was unable to conceptualize how to code this so I landed on an in between in that I would only just the diagonals (starting from a corner), rows, and columns of the most recently placed chips.

Other than that, I struggled on animating the piece falling down as It was overriding my blue background for the grid. Eventually I came up with the solution that I would check whether an individual pixel was blue and if it was then I would not paint over it, otherwise I would animate the circle falling down as normal.

When doing this project I did not face that many challenges and everything felt relatively straightforward. At the start it took some time to understand how the LCD display worked, but once I got that figured out everything went smoothly.

What I learned

I learned how to incorporate all the various ideas that we learned in this class into one project. This project also gave me practice in understanding code that other people wrote. We had to understand how the LCD code worked and had to understand some new functions in the HAL such as Systick.

What I would do differently

When doing this project, I learned that keeping the code organized is extremely important. In the end I had the majority of my code in my LCD_Display file. This was because I

felt at the start that I was going to be calling a lot of functions in the LCD_Display and did not want to have a file above it in the hierarchy and below ApplicationCode. This was because I did not wish to have wrapper functions as I felt I would end up confusing myself. Now I see that these wrapper functions could help clear up some confusion as I would be able to have functions that aren't wrappers but would modify the 2D array of the grid without updating the display. Having a modified coding hierarchy would definitely be something I would change in the future.

Additionally, I would have tried to explore layers as I saw that was something the LCD display had. This would have helped me a lot when animating my pieces to fall and could allow for my project to appear even nicer.

How could the project be improved

I feel that the project could be improved by moving the initializations of some peripherals that we had in main into different files. For example, in main we were given an initialization of the RNG peripheral; however, we could not access this peripheral in ApplicationCode as it was lower on the hierarchy. The only way to access it would be to use the keyword extern which would violate coding hierarchies. Besides that, I feel that the project directions were clear and I do not have any ideas of modifications in the directions besides suggesting in the directions to use SysTick.