Berkay GÖKSU 2030757
Caner POTUR 2031250

20/01/2019

EE447 – Laboratory Project Final Report
Battleship Game Console

Introduction

In this project, our main purpose is to build a platform to play a different version of the famous "Battleship" game. It will be a 2-Player turn-based strategy game and it will test visual memory of players. Firstly, Player 1 will place the battleships and civilian ships, then second player will try to find the battleships without hitting the civilian ships.

Overall system includes TM4C123 Microcontroller unit, 2 buttons of it, 2 potentiometers and Nokia 5110 LCD screen, also win screen can be seen in Fig. 1. In addition, we started to solve this multi-task problem by understanding completely. Therefore, we drew the Flow chart of this game and this flow in detail can be seen in Fig. 2.

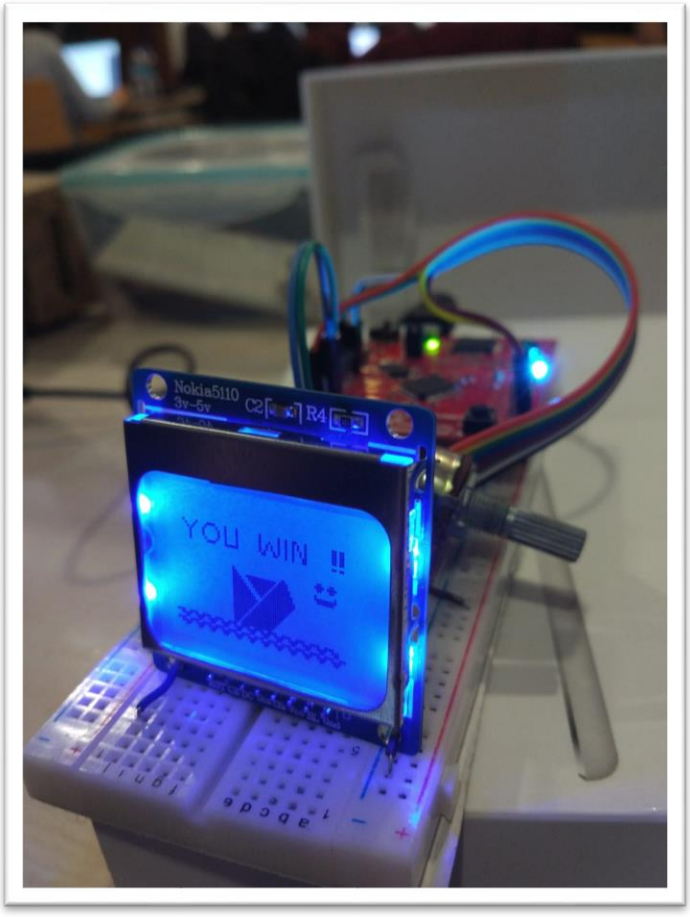

Figure 1: WIN Screen and overall system

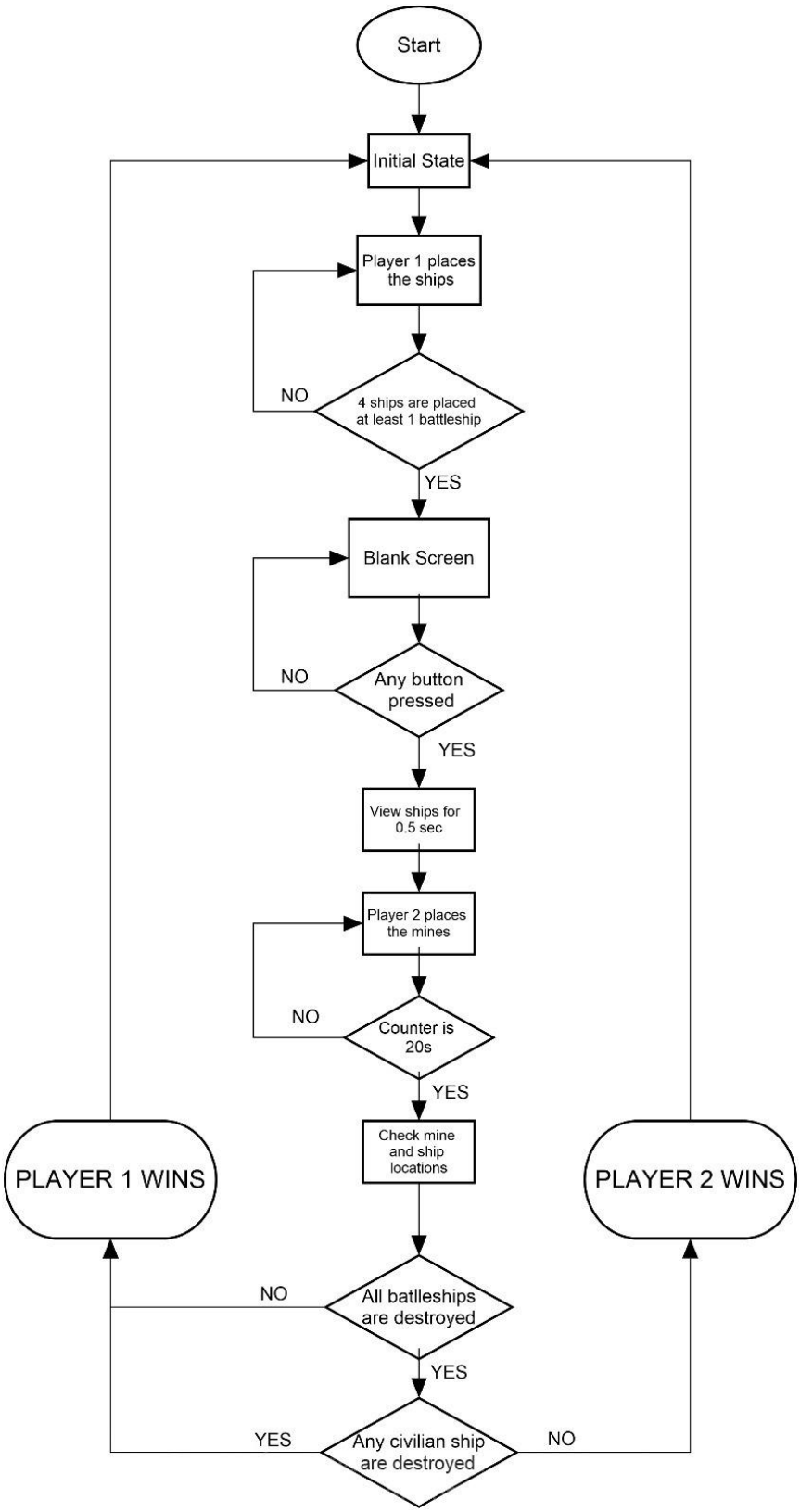

Figure 2: Flow Chart of Battleship Game

In order to better understand the project, we divided the game flow into 5 main parts. These parts are Beginning of Game, Deployment Phase, Attack Phase, Control Phase and End of Game". In these parts of the report, we discussed the difficulties we encountered and the solutions we found.

Also the below table shows the pin assignment of our game console.

Pinout Diagram of the Overall System

NOKIA 5110 Pin 1 (RST)	MCU (PA7)
NOKIA 5110 Pin 2 (CE)	MCU (PA3)
NOKIA 5110 Pin 3 (DC)	MCU (PA6)
NOKIA 5110 Pin 4 (Din)	MCU (PA5)
NOKIA 5110 Pin 5 (Clk)	MCU (PA2)
NOKIA 5110 Pin 6 (Vcc)	MCU (3.3V)
NOKIA 5110 Pin 7 (BL)	MCU (3.3V)
NOKIA 5110 Pin 8 (Gnd)	MCU (GND)
POT 1	MCU (PE2)
POT 2	MCU (PE3)
SW 1	MCU (PF0)
SW 2	MCU (PF4)

BEGINNING OF GAME

At the beginning of the game, it was necessary to make the initializations. Therefore, GPIO, SPI, NOKIA_CONFIG and SysTick initializations were performed. The control of the two buttons in the project is done by polling and all operations outside of it are located in the SysTick handler as interrupt. At this stage, it was very important to check the Data / Command (DC) pin correctly for the SPI to work correctly. When the DC signal is low, the Nokia interprets the incoming SPI bits as a command. Similarly, if the DC signal is high, the SPI bits are interpreted as data to be displayed. While doing initialization, we arranged command mode and made the display settings, then turned it to data mode and control the screen.

Once all the adjustments have been made, the game is ready to go into the deployment phase. Ready phase can be seen in Fig. 3.

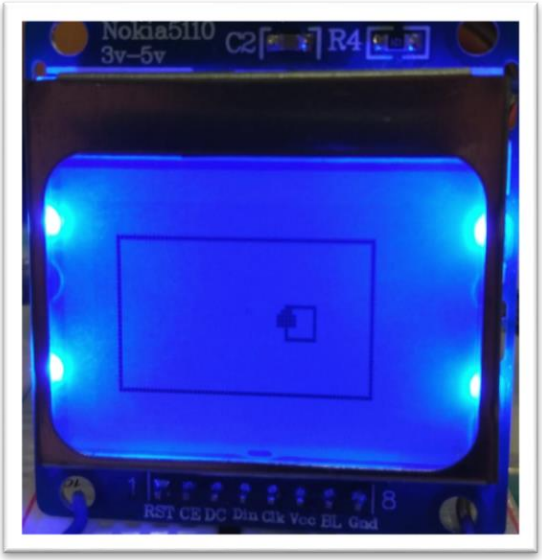

Figure 3: Nokia 5110 LCD Screen Display

DEPLOYMENT PHASE

One of the most important and challenging stages of the game is the Deployment Phase. In this phase, ships are placed into the map as civilian ships or battleships type, with a total number of up to 4 by Player 1. There were 2 major difficulties we encountered at this stage. First, the location of the ships had to be stored in a way that could be called later, and the second was the problem we had because we sent the data to the Nokia screen as 8 bits.

The solution we found to solve our first problem was keeping the place of the ships in memory. To do this, we stored 2 POT's values in separate addresses according to ship type and how many times the button has been pressed. We used SRAM part of MCU unit for this store operation. Moreover, when the button was pressed, R9, R10, R11, R12 was written as 1 or 0 one by one. R9 represents 1st ship, R10 represents 2nd ship, R11 represents 3rd ship, and R12 represents 4th ship. Also, "1" represents that this is a battleship and "0" represents this is a civilian ship.

The solution we found to solve our second problem was using shift operations in horizontal addressing mode. Our ships are in 8 x 8 box and while moving cursor in Y direction, there were problem. While drawing cursor or ships for every bits in Y, we used LSL operation for upper y value and LSR operation for below y value. By doing these operations, we can control every bits in map accurately.

After the deployment of 4 ships, cursor were disappointed. And final view of deployment can be seen by Player 1. When Player 1 was ready, the button was pressed again and Nokia screen were cleared. Then game console passed to the second player and the game continued with the attack phase.

Also the game console doesn't allow the deployment 4 civilian ships in a game. Hence, if first 3 ships are civilian, 4th one must be battleship as a bonus feature.

The last version of deployment phase example can be seen in Fig. 4. White box represents civilian ships and black box represents battleships.

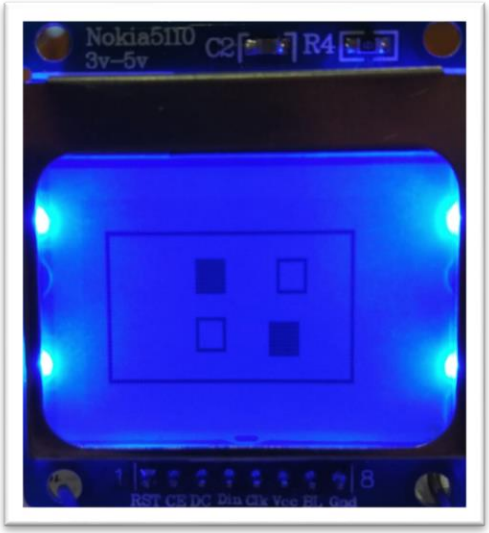

Figure 4: Nokia 5110 LCD Screen Display

ATTACK PHASE

After deployment phase, Player 2 takes the microcontroller and initiates the attack phase by pressing a button. In our algorithm, when this button is pressed, we initiated a GPIO timer interrupt working at 16 MHz and set its "_TAILR" as 8.000.000. In this way, GPIO TimerHandler starts after 0.5 second. Meanwhile, player 2 can see where the ships are placed. When GPIO TimerHandler starts, the "_TAILR" is updated as 16.000.000. Thus, we obtained a counter that counts every second. Player 2 must place the mines in 20 seconds He can see the remaining time right upper corner of the screen. . The mine is placed to the pointing pixel of cursor. When a button is pressed, we store the pot values at SRAM. For example, we store the POT values of second mine at "MINE2X =EQU 0x20000458 and MINE2Y = EQU 0x2000045C". It can be placed up to 4 mines. When time is up attack phase ends and check algorithm starts.

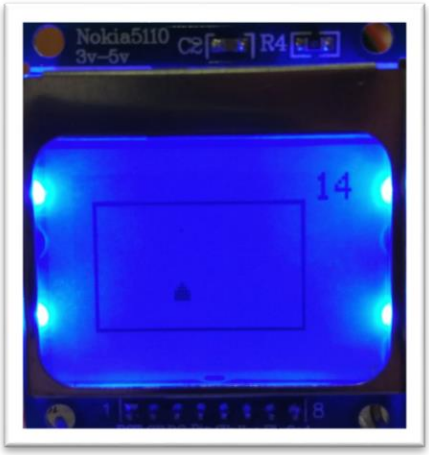

Figure 5: Nokia 5110 LCD Screen Display

CONTROL PHASE

After 20 seconds, the control phase checked the ships and mines. Every ships were called from their addresses according to ship types one by one. After the calling operation of ships, POT value which showed X direction were divided by 75 and POT value which showed Y direction were divided 169. Then we can get the real position of X and Y coordinates of ships in map. We did same division for mines with 66 for X and 132 for Y value.

After the first ship was called, mines were called in order. Firstly we checked X position of 1st ship and 1st mine. If the ship's X position was greater than 1st mine x position, control operation continue with other mines. If the X position of ship lower than mine the system add 7 and check again. We add 7 because we stored top left corner coordinate of ships and it had 8 bit length. Therefore, if the new X position value of ship is greater than mine position, it means that the ship can be destroy according to X values, hence control operation will continue for Y coordinates. We did same operation in Y and if this was matched also, it means that the ship is destroyed absolutely. If this ship is war ship R7 increase 1. And if all mines were not hit civilian ships, R7 increase 1. If all war ships are destroyed without hitting any civilian ship, R7 was 4.

After the all operation if R7 was 4, it means that Player 2 win the game. If R7 was different from 4, it means that Player 1 win the game. This decision will transfer to the end of the game operation.

END OF GAME

Control algorithm determined the winner of the game. If Player 2 wins the game, we call the "DRAW_WIN" subroutine as shown in Fig. 6. Otherwise, we call the "DRAW_LOSE" subroutine as shown in Fig. 7. Any of these images stay at the LCD screen until button 1 or button 2 is pressed.

If players determine to make a rematch, they can press any button. After that, the program clears all the registers (R0-R12), the ship addresses (i.e. SHIPWAR1X, SHIPWAR4Y) and the mine addresses (i.e. MINE1X, MINENUMBER). After clearing all addresses and registers, the program returns to the beginning of the code. A new game is ready!

Figure 6: Nokia 5110 LCD Screen Display

Figure 7: Nokia 5110 LCD Screen Display

Conclusion

We used different techniques that we have learnt in EE447 course such that sensing a button press with polling, using Systick Handler Interrupt, using GPIO Timer Interrupt, sensing the POT voltage values with ADC, sending data to a slave(Nokia 5110 LCD) with SPI. Moreover, we searched and learnt about configuration and usage of Nokia 5110 LCD. In this report, the configuration of the Nokia 5110 LCD, deployment phase, attack phase, main check systems were explained. We used different algorithms to overcome the difficulties that we faced in this project. Firstly, we wrote a code using shifting techniques to move cursor vertically. Secondly, we assigned some addresses at memory to store the positions of ships and mines. Thirdly, we used Systick Handler to adjust the refresh speed of screen. Fourthly, we used an interrupt of a GPIO timer which starts after 0.5 second and counts down every second. Finally, we wrote "LTORG" instruction to create large subroutines. After finishing the overall system, we tested the game many times and updated the required parts. In the end, we revealed our final design successfully..

References

[1] Texas Instruments, Inter-Integrated Circuit (I2C) Interface, Tiva TM4C123GH6PM Microcontroller datasheet, June 2014

[2] Project manual of Battleship Game Console