XO GAME

Team member

Hamed Essam Hamed

Goda Saber Goda

Ahmed Osama Ibrahim

Introduction

The XO game, also known as Tic Tac Toe, is a classic two-player game where players alternately mark spaces in a 3x3 grid. The goal is to align three of their marks vertically, horizontally, or diagonally before their opponent. In this project, we will implement and deploy the XO game using the TM4C123G microcontroller from Texas Instruments, leveraging its powerful ARM Cortex-M4 architecture.

This report provides an overview of the microcontroller, the hardware and software components needed, and a detailed explanation of the implementation process. Additionally, we will discuss how the project can be executed on a simulator and using actual hardware.

Overview of the TM4C123G Microcontroller

Features and Specifications

The TM4C123G is a high-performance 32-bit microcontroller built around the ARM Cortex-M4 processor. Key features include:

- **Clock Frequency:** Up to 80 MHz, providing ample computational power.
- **Floating-Point Unit (FPU):** Enhances mathematical calculations and improves performance.
- **Nested Vector Interrupt Controller (NVIC):** Supports efficient interrupt handling.

- **Debugging Interface:** JTAG and Serial Wire Debug (SWD) are available for programming and debugging.
- **GPIOs:** The microcontroller offers a versatile GPIO interface that allows interaction with transducers, sensors, actuators, displays, and other peripherals.

The Cortex-M4F processor supports tail chaining functionality, ensuring efficient interrupt handling. GPIOs on the TM4C123G are more advanced than those on typical 8-bit microcontrollers, providing numerous configuration options.

Tiva-C LaunchPad

The Tiva-C LaunchPad development board includes the TM4C123G microcontroller and is designed for prototyping and development. It provides onboard debugging support and is compatible with various sensors, displays, and other external devices.

GPIO Usage

The TM4C123G's GPIO interface allows for seamless communication with external devices. For this project, the GPIOs will be used to:

- Connect a Nokia 5110 LCD for game display.
- Interface with input switches for player controls.
- Drive RGB LEDs for visual indicators.

Required Hardware and Connections

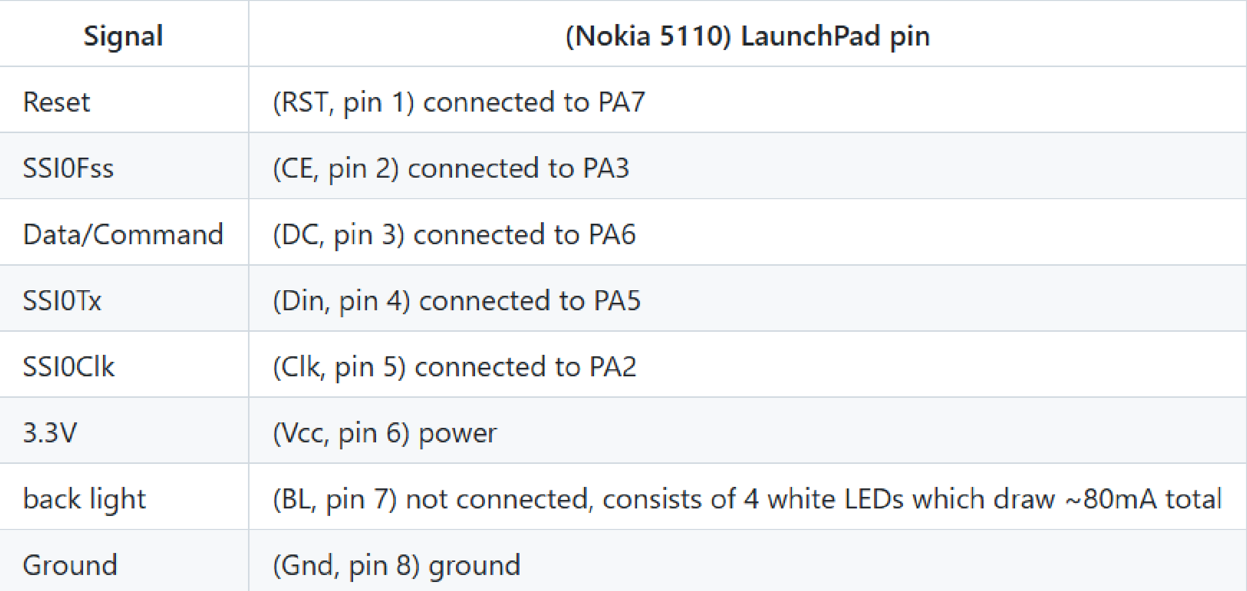
Components

To implement the XO game using hardware, the following components are required:

- **Tiva-C LaunchPad:** The primary microcontroller (TM4C123GH6PM) board.
- **Nokia 5110 LCD (Blue):** For displaying the game grid and status.
- **2 Buttons:** For player inputs.
- **3 RGB LEDs:** For visual feedback (e.g., indicating player turns or game outcomes).
- **Male-Female Jumper Wires:** For making electrical connections.
- **Resistors:** 470 Ω and 10 k Ω resistors for circuit stability.
- **Breadboard:** For prototyping and organizing connections.
- **Potentiometer:** 10 k Ω for ADC application to control on the light of the led.

Nokia 5110 LCD Connections

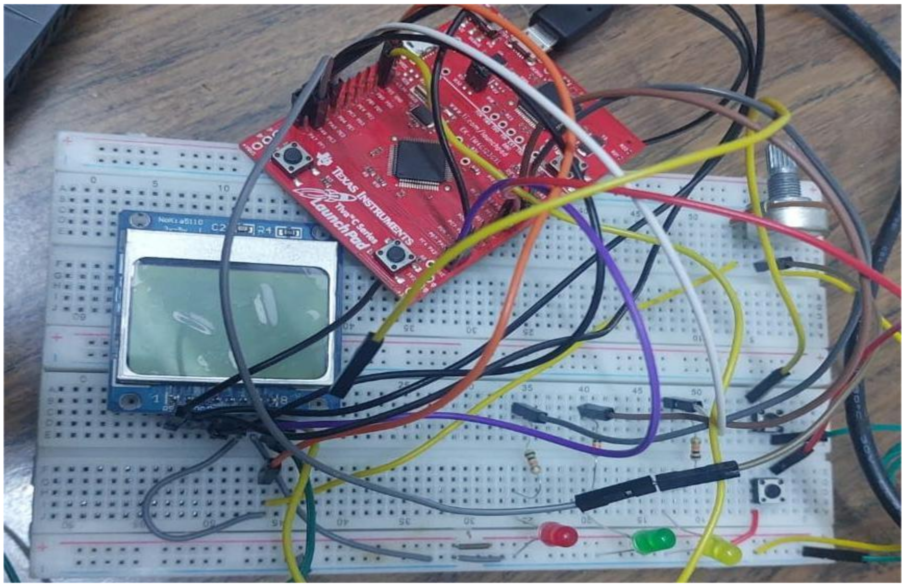
The Nokia 5110 LCD connects to the Tiva-C LaunchPad as follows:



Signal	(Nokia 5110) LaunchPad pin
Reset	(RST, pin 1) connected to PA7
SSI0Fss	(CE, pin 2) connected to PA3
Data/Command	(DC, pin 3) connected to PA6
SSI0Tx	(Din, pin 4) connected to PA5
SSI0Clk	(Clk, pin 5) connected to PA2
3.3V	(Vcc, pin 6) power
back light	(BL, pin 7) not connected, consists of 4 white LEDs which draw ~80mA total
Ground	(Gnd, pin 8) ground

Circuit Diagram

A detailed circuit diagram is essential for understanding the connections. The diagram includes the microcontroller, Nokia 5110 LCD, switches, RGB LEDs, and resistors.



Software Implementation

Development Environment

The software for this project is developed using:

- **IDE:** Keil uVision.
- **Programming Language:** C.
- **Libraries:** Peripheral Driver Library (TivaWare) for interacting with the microcontroller peripherals.

Program Structure

The program is structured into the following modules:

1. **Initialization:** Configures the GPIOs, Timers, and other peripherals.
2. **Display Management:** Controls the Nokia 5110 LCD to render the game grid and display messages.
3. **Input Handling:** Reads player inputs from the buttons.
4. **Game Logic:** Implements the XO game rules and checks for winning conditions.
5. **LED Control:** Manages the RGB LEDs for visual feedback.
6. **LED Control:** Using ADC (Potentiometer) to control led.

Game Logic

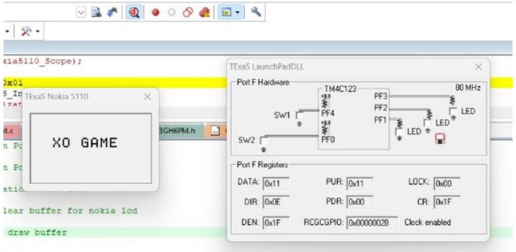
The game grid is represented as a 2D array. The program tracks player moves, updates the grid, and checks for:

- Three consecutive marks horizontally, vertically, or diagonally.
- Full grid without a winner (draw).
- After each round ask player if want play again or not (End game).

Testing

Simulator

The project can be tested using a simulation tool such as Proteus or Tiva-C's integrated simulator. This eliminates the need for physical hardware during the initial development phase.



```
kia5110_Scope);
0x01
lear buffer for nokia lcd
 draw buffer
```

TExaS Nokia 5110

XO GAME

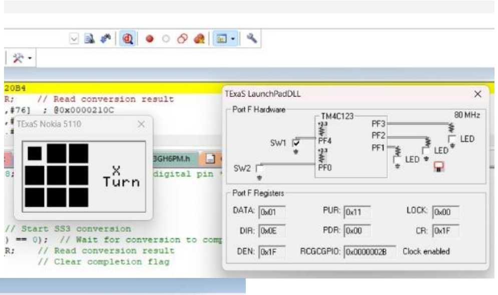
TExaS LaunchPadDLL

Port F Registers

DATA: 0x11 PUR: 0x11 LOCK: 0x00

DIR: 0x0E PDR: 0x00 CR: 0x1F

DEN: 0x1F RCGCGPIO: 0x0000002B Clock enabled



```
20B4
R;     // Read conversion result
,#76]  ; @0x0000210C
8;                                digital pin
// Start SS3 conversion
) == 0);  // Wait for conversion to comp
R;     // Read conversion result
       // Clear completion flag
```

TExaS Nokia 5110

X
Turn

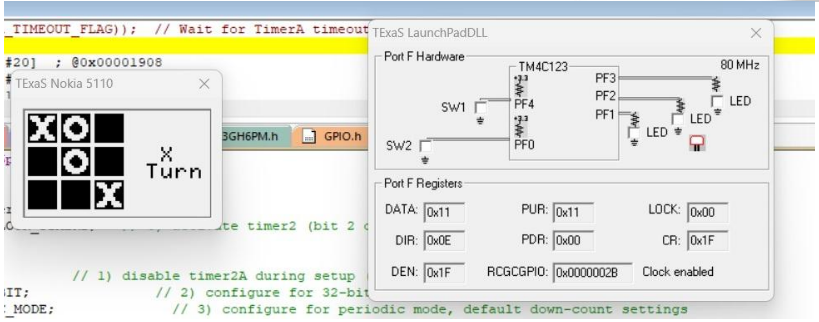
TExaS LaunchPadDLL

Port F Registers

DATA: 0x01 PUR: 0x11 LOCK: 0x00

DIR: 0x0E PDR: 0x00 CR: 0x1F

DEN: 0x1F RCGCGPIO: 0x0000002B Clock enabled



```
_TIMEOUT_FLAG));  // Wait for TimerA timeout
#20]  ; @0x00001908
                        te timer2 (bit 2 
        // 1) disable timer2A during setup
IT;                    // 2) configure for 32-bit
_MODE;                   // 3) configure for periodic mode, default down-count settings
```

TExaS Nokia 5110

X
Turn

TExaS LaunchPadDLL

Port F Registers

DATA: 0x11 PUR: 0x11 LOCK: 0x00

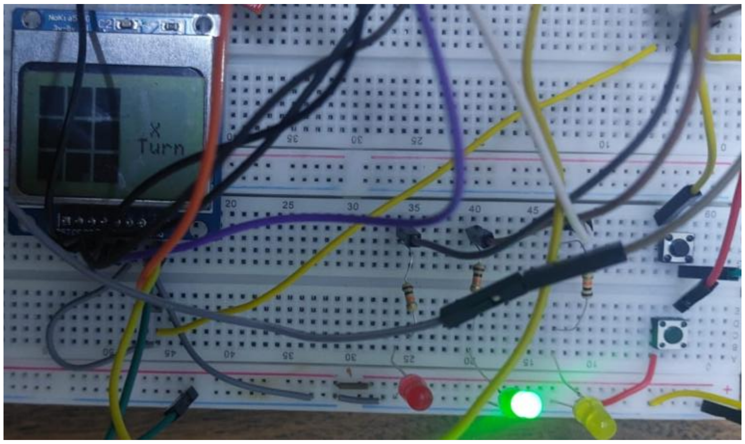
DIR: 0x0E PDR: 0x00 CR: 0x1F

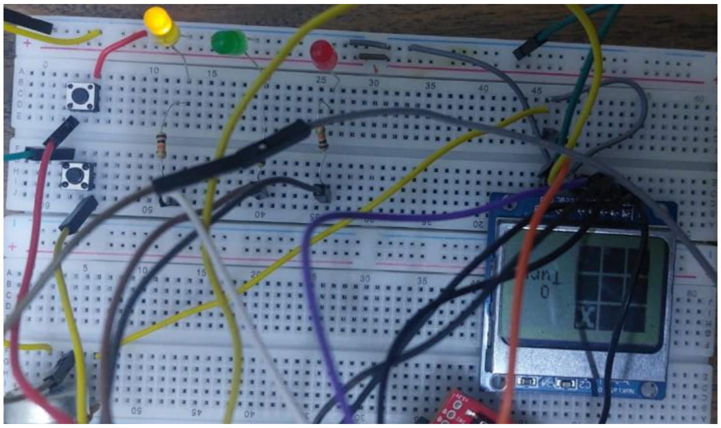
DEN: 0x1F RCGCGPIO: 0x0000002B Clock enabled

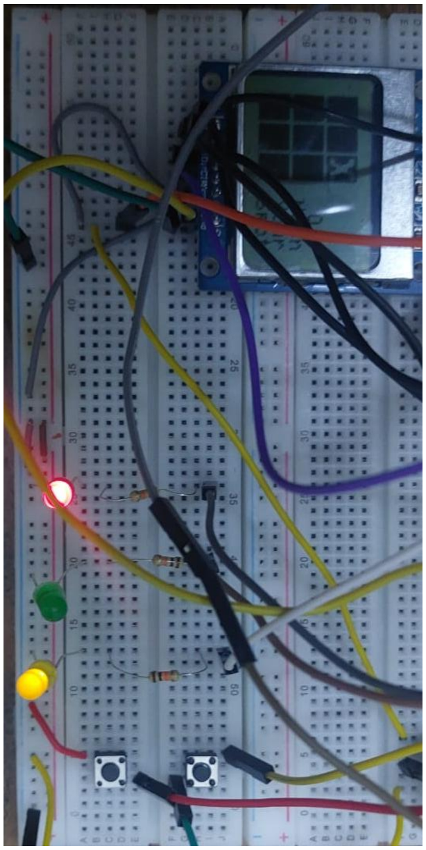
Hardware Implementation

To deploy the project on hardware:

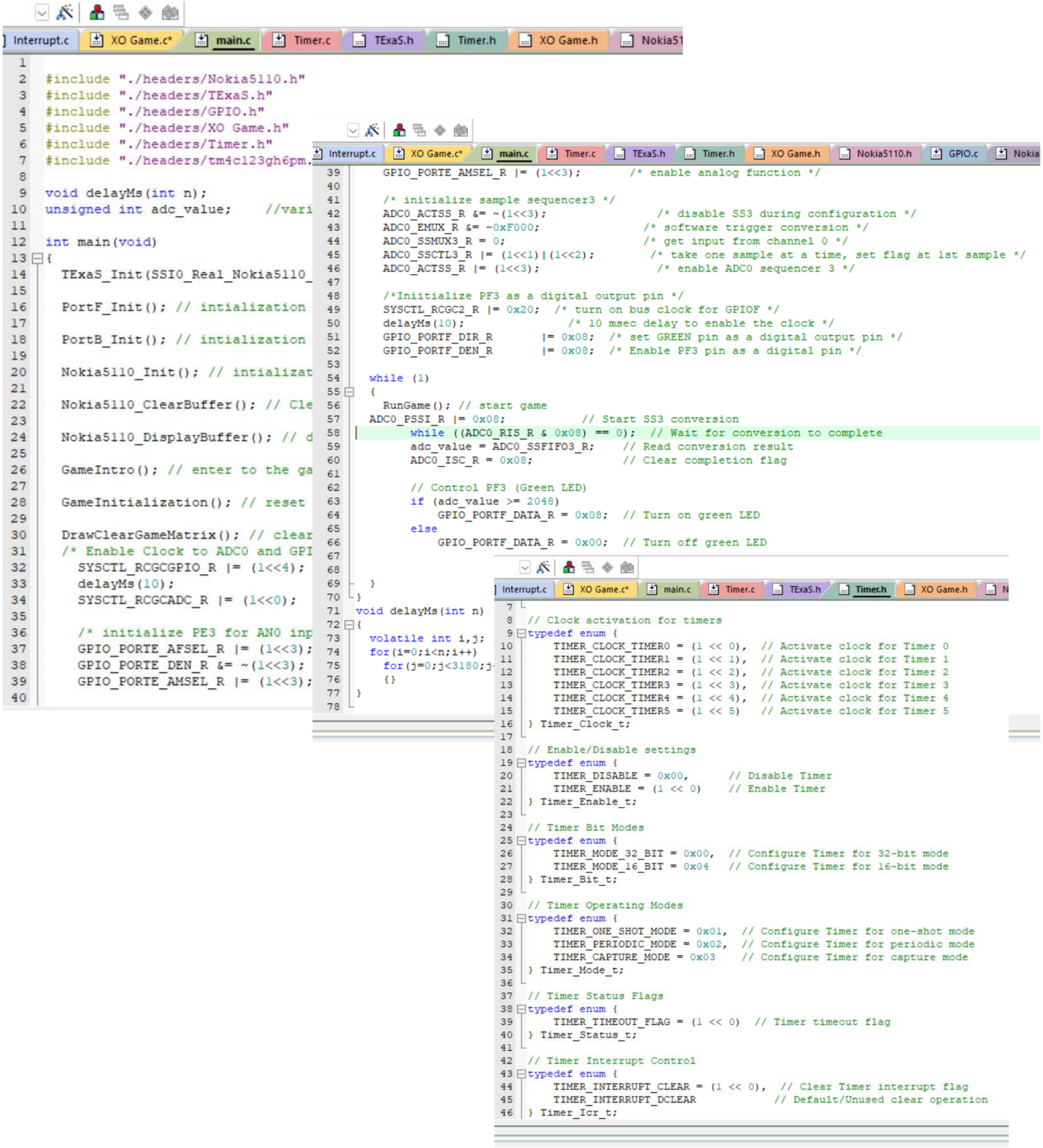
1. Assemble the components on a breadboard.
2. Connect the Nokia 5110 LCD, buttons, and LEDs to the LaunchPad as per the circuit diagram.
3. Upload the program to the microcontroller using the debugging interface.
4. Test the game functionality.







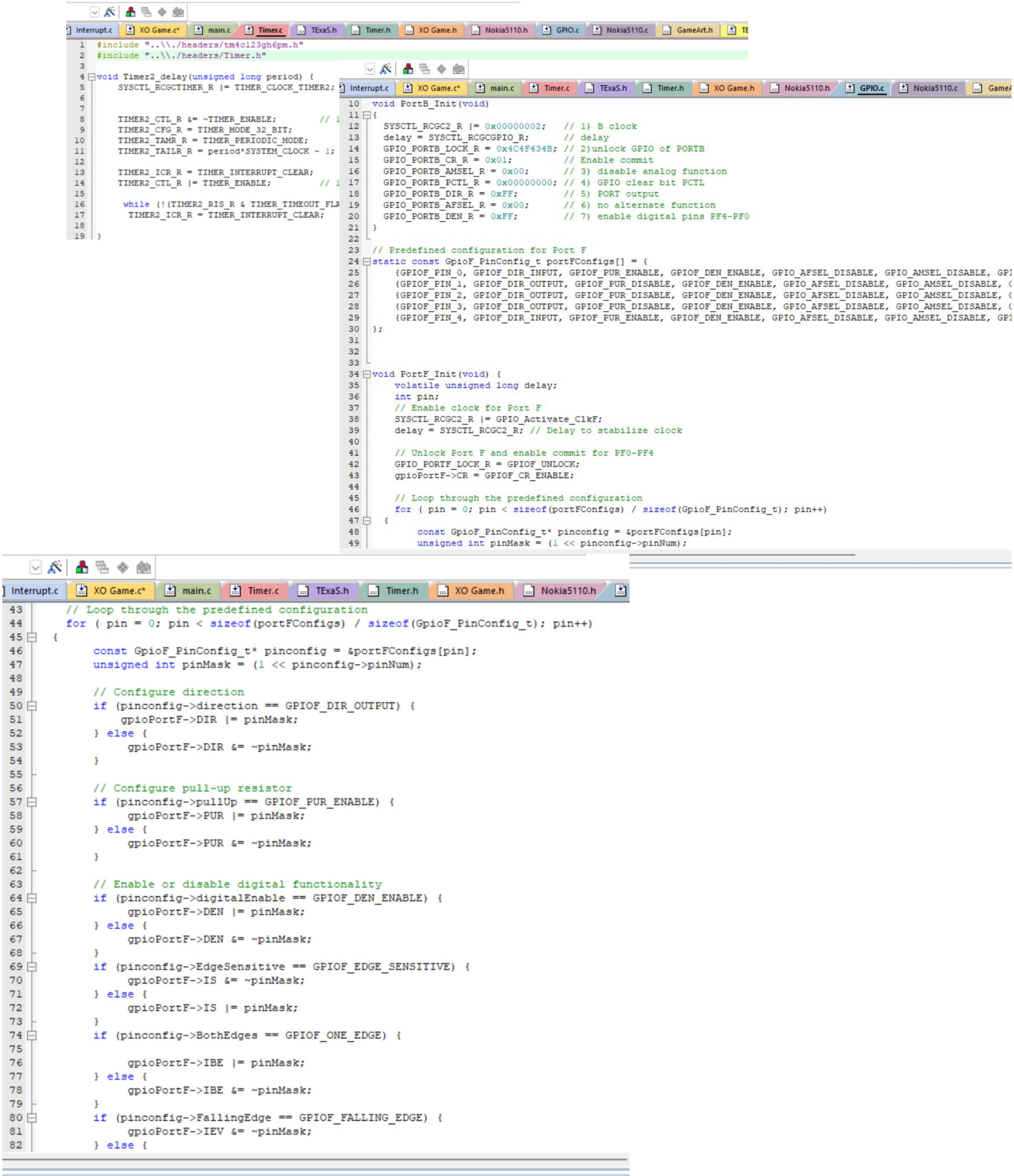
Source Code



```

#include "./headers/Nokia5110.h"
#include "./headers/TExaS.h"
#include "./headers/GPIO.h"
#include "./headers/XO Game.h"
#include "./headers/Timer.h"
#include "./headers/tm4c123gh6pm.

void delayMs(int n);
unsigned int adc_value;     //vari

int main(void)
{
  TExaS_Init(SSI0_Real_Nokia5110_

  PortF_Init(); // intialization

  PortB_Init(); // intialization

  Nokia5110_Init(); // intializat

  Nokia5110_ClearBuffer(); // Cle

  Nokia5110_DisplayBuffer(); // d

  GameIntro(); // enter to the ga

  GameInitialization(); // reset

  DrawClearGameMatrix(); // clear
  /* Enable Clock to ADC0 and GPI
    SYSCTL_RCGCGPIO_R |= (1<<4);
    delayMs(10);
    SYSCTL_RCGCADC_R |= (1<<0);

    /* initialize PE3 for AN0 inp
    GPIO_PORTE_AFSEL_R |= (1<<3);
    GPIO_PORTE_DEN_R &= ~(1<<3);
    GPIO_PORTE_AMSEL_R |= (1<<3);

```

```
    GPIO_PORTE_AMSEL_R |= (1<<3);       /* enable analog function */

    /* initialize sample sequencer3 */
    ADC0_ACTSS_R &= ~(1<<3);                 /* disable SS3 during configuration */
    ADC0_EMUX_R &= ~0xF000;                /* software trigger conversion */
    ADC0_SSMUX3_R = 0;                     /* get input from channel 0 */
    ADC0_SSCTL3_R |= (1<<1)|(1<<2);         /* take one sample at a time, set flag at 1st sample */
    ADC0_ACTSS_R |= (1<<3);                  /* enable ADC0 sequencer 3 */

    /*Iniitialize PF3 as a digital output pin */
    SYSCTL_RCGC2_R |= 0x20;  /* turn on bus clock for GPIOF */
    delayMs(10);                 /* 10 msec delay to enable the clock */
    GPIO_PORTF_DIR_R        |= 0x08;  /* set GREEN pin as a digital output pin */
    GPIO_PORTF_DEN_R        |= 0x08;  /* Enable PF3 pin as a digital pin */

  while (1)
  {
    RunGame(); // start game
  ADC0_PSSI_R |= 0x08;           // Start SS3 conversion
        while ((ADC0_RIS_R & 0x08) == 0);  // Wait for conversion to complete
        adc_value = ADC0_SSFIFO3_R;     // Read conversion result
        ADC0_ISC_R = 0x08;              // Clear completion flag

        // Control PF3 (Green LED)
        if (adc_value >= 2048)
            GPIO_PORTF_DATA_R = 0x08;  // Turn on green LED
        else
            GPIO_PORTF_DATA_R = 0x00;  // Turn off green LED


  }
}
void delayMs(int n)
{
  volatile int i,j;
  for(i=0;i<n;i++)
    for(j=0;j<3180;j
    {}
}

```

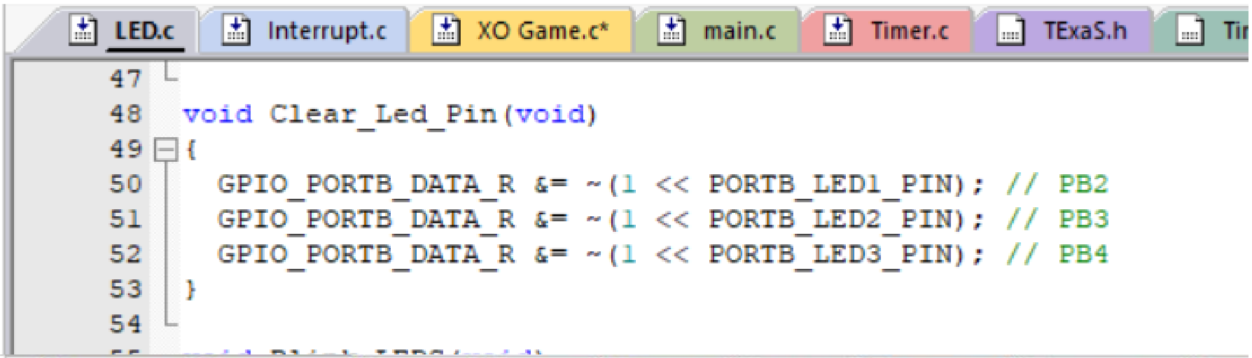
```

// Clock activation for timers
typedef enum {
    TIMER_CLOCK_TIMER0 = (1 << 0),  // Activate clock for Timer 0
    TIMER_CLOCK_TIMER1 = (1 << 1),  // Activate clock for Timer 1
    TIMER_CLOCK_TIMER2 = (1 << 2),  // Activate clock for Timer 2
    TIMER_CLOCK_TIMER3 = (1 << 3),  // Activate clock for Timer 3
    TIMER_CLOCK_TIMER4 = (1 << 4),  // Activate clock for Timer 4
    TIMER_CLOCK_TIMER5 = (1 << 5)   // Activate clock for Timer 5
} Timer_Clock_t;

// Enable/Disable settings
typedef enum {
    TIMER_DISABLE = 0x00,      // Disable Timer
    TIMER_ENABLE = (1 << 0)    // Enable Timer
} Timer_Enable_t;

// Timer Bit Modes
typedef enum {
    TIMER_MODE_32_BIT = 0x00,  // Configure Timer for 32-bit mode
    TIMER_MODE_16_BIT = 0x04   // Configure Timer for 16-bit mode
} Timer_Bit_t;

// Timer Operating Modes
typedef enum {
    TIMER_ONE_SHOT_MODE = 0x01,  // Configure Timer for one-shot mode
    TIMER_PERIODIC_MODE = 0x02,  // Configure Timer for periodic mode
    TIMER_CAPTURE_MODE = 0x03    // Configure Timer for capture mode
} Timer_Mode_t;

// Timer Status Flags
typedef enum {
    TIMER_TIMEOUT_FLAG = (1 << 0)  // Timer timeout flag
} Timer_Status_t;

// Timer Interrupt Control
typedef enum {
    TIMER_INTERRUPT_CLEAR = (1 << 0),  // Clear Timer interrupt flag
    TIMER_INTERRUPT_DCLEAR             // Default/Unused clear operation
} Timer_Icr_t;
```



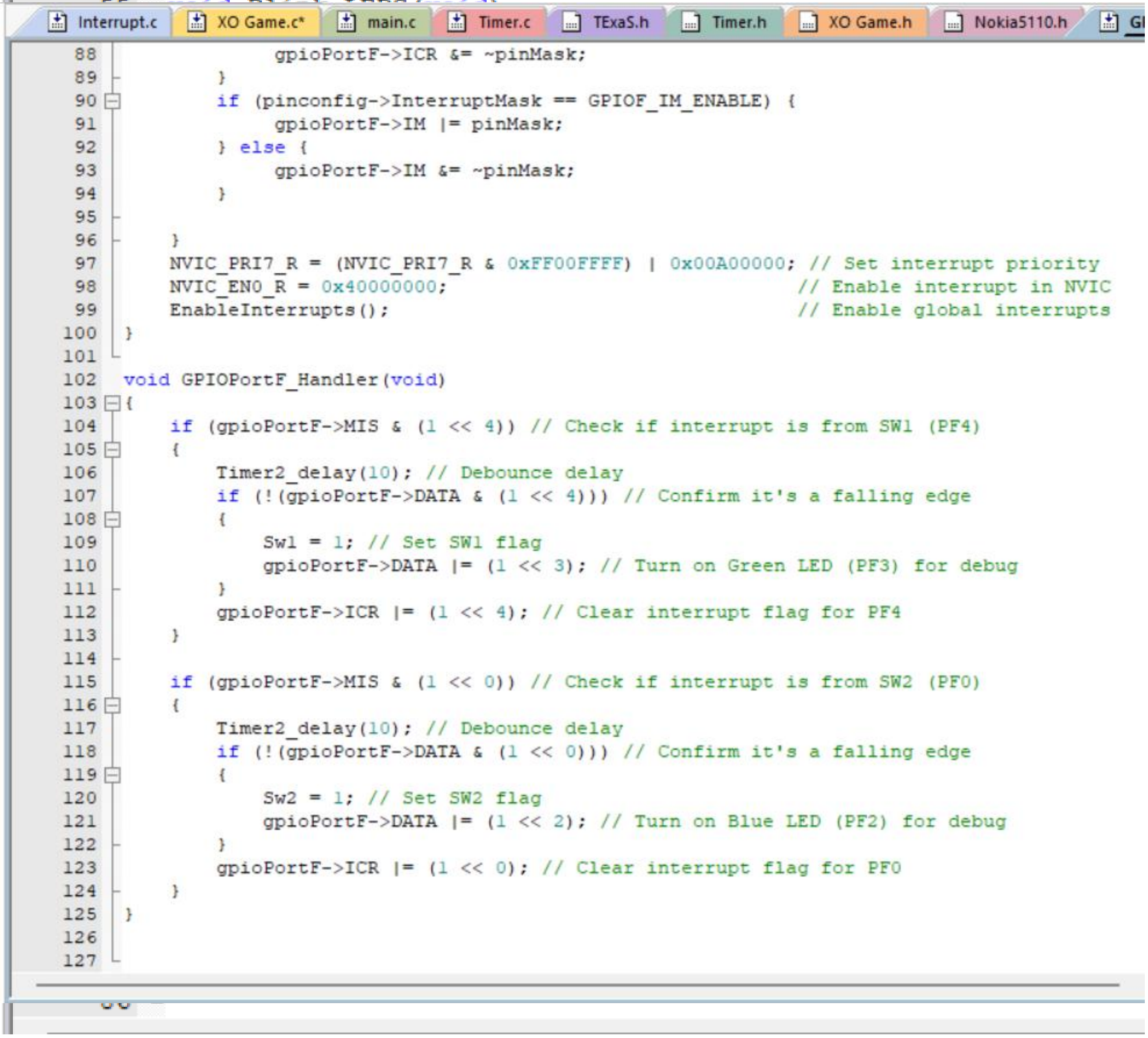
```c
#include "..\\./headers/tm4c123gh6pm.h"
#include "..\\./headers/Timer.h"

void Timer2_delay(unsigned long period) {
    SYSCTL_RCGCTIMER_R |= TIMER_CLOCK_TIMER2;


    TIMER2_CTL_R &= ~TIMER_ENABLE;          // 1
    TIMER2_CFG_R = TIMER_MODE_32_BIT;
    TIMER2_TAMR_R = TIMER_PERIODIC_MODE;
    TIMER2_TAILR_R = period*SYSTEM_CLOCK - 1;

    TIMER2_ICR_R = TIMER_INTERRUPT_CLEAR;
    TIMER2_CTL_R |= TIMER_ENABLE;           // 1

     while (!(TIMER2_RIS_R & TIMER_TIMEOUT_FLA
      TIMER2_ICR_R = TIMER_INTERRUPT_CLEAR;

}
```

```c
void PortB_Init(void)
{
  SYSCTL_RCGC2_R |= 0x00000002;    // 1) B clock
  delay = SYSCTL_RCGCGPIO_R;       // delay
  GPIO_PORTB_LOCK_R = 0x4C4F434B; // 2)unlock GPIO of PORTB
  GPIO_PORTB_CR_R = 0x01;          // Enable commit
  GPIO_PORTB_AMSEL_R = 0x00;       // 3) disable analog function
  GPIO_PORTB_PCTL_R = 0x00000000; // 4) GPIO clear bit PCTL
  GPIO_PORTB_DIR_R = 0xFF;         // 5) PORT output
  GPIO_PORTB_AFSEL_R = 0x00;       // 6) no alternate function
  GPIO_PORTB_DEN_R = 0xFF;         // 7) enable digital pins PF4-PF0
}

// Predefined configuration for Port F
static const GpioF_PinConfig_t portFConfigs[] = {
    {GPIOF_PIN_0, GPIOF_DIR_INPUT, GPIOF_PUR_ENABLE, GPIOF_DEN_ENABLE, GPIO_AFSEL_DISABLE, GPIO_AMSEL_DISABLE, GPI
    {GPIOF_PIN_1, GPIOF_DIR_OUTPUT, GPIOF_PUR_DISABLE, GPIOF_DEN_ENABLE, GPIO_AFSEL_DISABLE, GPIO_AMSEL_DISABLE, G
    {GPIOF_PIN_2, GPIOF_DIR_OUTPUT, GPIOF_PUR_DISABLE, GPIOF_DEN_ENABLE, GPIO_AFSEL_DISABLE, GPIO_AMSEL_DISABLE, G
    {GPIOF_PIN_3, GPIOF_DIR_OUTPUT, GPIOF_PUR_DISABLE, GPIOF_DEN_ENABLE, GPIO_AFSEL_DISABLE, GPIO_AMSEL_DISABLE, G
    {GPIOF_PIN_4, GPIOF_DIR_INPUT, GPIOF_PUR_ENABLE, GPIOF_DEN_ENABLE, GPIO_AFSEL_DISABLE, GPIO_AMSEL_DISABLE, GPI
};



void PortF_Init(void) {
    volatile unsigned long delay;
    int pin;
    // Enable clock for Port F
    SYSCTL_RCGC2_R |= GPIO_Activate_ClkF;
    delay = SYSCTL_RCGC2_R; // Delay to stabilize clock

    // Unlock Port F and enable commit for PF0-PF4
    GPIO_PORTF_LOCK_R = GPIOF_UNLOCK;
    gpioPortF->CR = GPIOF_CR_ENABLE;

    // Loop through the predefined configuration
    for ( pin = 0; pin < sizeof(portFConfigs) / sizeof(GpioF_PinConfig_t); pin++)
  {
        const GpioF_PinConfig_t* pinconfig = &portFConfigs[pin];
        unsigned int pinMask = (1 << pinconfig->pinNum);
```

```c
    // Loop through the predefined configuration
    for ( pin = 0; pin < sizeof(portFConfigs) / sizeof(GpioF_PinConfig_t); pin++)
  {
        const GpioF_PinConfig_t* pinconfig = &portFConfigs[pin];
        unsigned int pinMask = (1 << pinconfig->pinNum);

        // Configure direction
        if (pinconfig->direction == GPIOF_DIR_OUTPUT) {
            gpioPortF->DIR |= pinMask;
        } else {
             gpioPortF->DIR &= ~pinMask;
        }

        // Configure pull-up resistor
        if (pinconfig->pullUp == GPIOF_PUR_ENABLE) {
             gpioPortF->PUR |= pinMask;
        } else {
             gpioPortF->PUR &= ~pinMask;
        }

        // Enable or disable digital functionality
        if (pinconfig->digitalEnable == GPIOF_DEN_ENABLE) {
             gpioPortF->DEN |= pinMask;
        } else {
             gpioPortF->DEN &= ~pinMask;
        }
        if (pinconfig->EdgeSensitive == GPIOF_EDGE_SENSITIVE) {
             gpioPortF->IS &= ~pinMask;
        } else {
             gpioPortF->IS |= pinMask;
        }
        if (pinconfig->BothEdges == GPIOF_ONE_EDGE) {

             gpioPortF->IBE |= pinMask;
        } else {
             gpioPortF->IBE &= ~pinMask;
        }
        if (pinconfig->FallingEdge == GPIOF_FALLING_EDGE) {
             gpioPortF->IEV &= ~pinMask;
        } else {
```



```
void Clear_Led_Pin(void)
{
  GPIO_PORTB_DATA_R &= ~(1 << PORTB_LED1_PIN); // PB2
  GPIO_PORTB_DATA_R &= ~(1 << PORTB_LED2_PIN); // PB3
  GPIO_PORTB_DATA_R &= ~(1 << PORTB_LED3_PIN); // PB4
}
```



```
                gpioPortF->ICR &= ~pinMask;
            }
            if (pinconfig->InterruptMask == GPIOF_IM_ENABLE) {
                gpioPortF->IM |= pinMask;
            } else {
                gpioPortF->IM &= ~pinMask;
            }

        }
        NVIC_PRI7_R = (NVIC_PRI7_R & 0xFF00FFFF) | 0x00A00000; // Set interrupt priority
        NVIC_EN0_R = 0x40000000;                              // Enable interrupt in NVIC
        EnableInterrupts();                                   // Enable global interrupts
    }

    void GPIOPortF_Handler(void)
    {
        if (gpioPortF->MIS & (1 << 4)) // Check if interrupt is from SW1 (PF4)
        {
            Timer2_delay(10); // Debounce delay
            if (!(gpioPortF->DATA & (1 << 4))) // Confirm it's a falling edge
            {
                Sw1 = 1; // Set SW1 flag
                gpioPortF->DATA |= (1 << 3); // Turn on Green LED (PF3) for debug
            }
            gpioPortF->ICR |= (1 << 4); // Clear interrupt flag for PF4
        }

        if (gpioPortF->MIS & (1 << 0)) // Check if interrupt is from SW2 (PF0)
        {
            Timer2_delay(10); // Debounce delay
            if (!(gpioPortF->DATA & (1 << 0))) // Confirm it's a falling edge
            {
                Sw2 = 1; // Set SW2 flag
                gpioPortF->DATA |= (1 << 2); // Turn on Blue LED (PF2) for debug
            }
            gpioPortF->ICR |= (1 << 0); // Clear interrupt flag for PF0
        }
    }
```

Conclusion

This project demonstrates the deployment of the XO game using the TM4C123G Tiva-C LaunchPad. By leveraging the microcontroller's powerful features and versatile GPIOs, we created an engaging application that highlights both software and hardware integration. The implementation is suitable for both simulation and physical hardware, making it an excellent educational project for learning embedded systems development.