CECS 347 Spring 2022 Project # 3

Space Invader Game

By

Abhishek Jasti, Anand Jasti

May 4, 2022

Design a space invader game with a minimum of three invaders and one spaceship. Using a potentiometer to control the movement of the spaceship and using onboard switches to start the game and to fire bullets.

CECS 347 Project 3 Report

Introduction

Making a space invader game wit a minimum of three invaders and one space ship. That uses potentiometer to control the movement of the space ship, onboard switch SW2 (right switch) is used to stared the game, and onboard switch SW1 (left switch) is used to fire a bullet. The space invader game will be displayed on the Nokia5110 LCD. We are using SSI to interface with the LCD. We are also using PLL, SysTick timer and edge-triggered interrupts, GPIO, and ADC to meet the required specifications for this space invaders game.

Operation

When the system is first powered on, the starting prompt will be displayed on the screen. When the player presses switch 2 the space invaders game will begin. There are three enemies that move from left to right, and when all the enemies are killed or when the last enemy leaves the screen the game is over. While the game is going on and switch 1 is pressed the player spaceship shoots a laser toward the enemy, and when the laser hits an enemy the enemy explodes and the player's score increments. When the game is over the ending prompt will be displayed which shows the player's score. After three seconds pass the starting prompt will be displayed again.

Switch 1:

When the game is on, if the player presses switch 1 the spaceship shoots a laser toward the enemies.

Switch 2:

Before the game starts, when switch 2 is pressed the game begins.

Potentiometer:

When the game is on, the potentiometer controls the player's spaceship's location on the LCD.

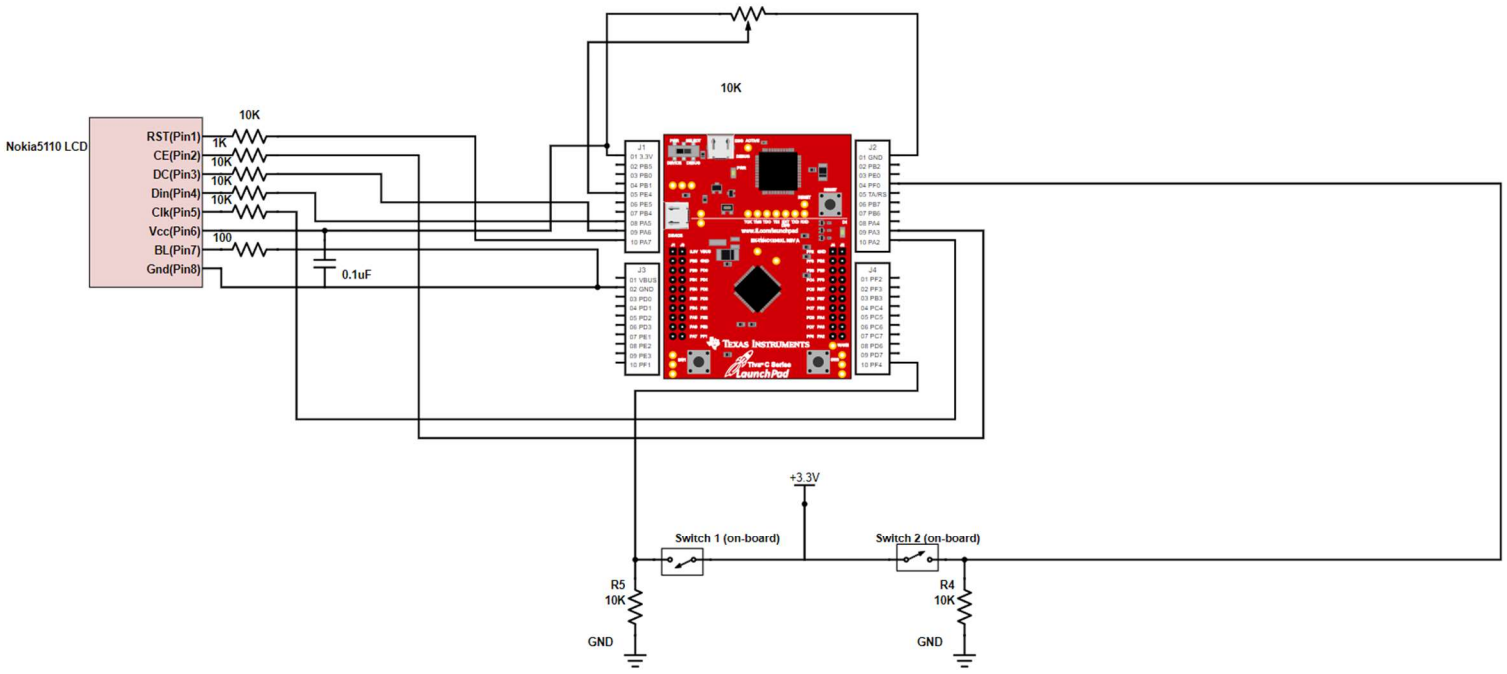
Link to Demonstration video:
https://drive.google.com/file/d/1MbmbD13is5nkLlk_LFAVxL6DhKNlsIc8/view?usp=sharing

Theory

This project uses ARM Cortex TM4C123GH6PM Microcontroller, more specifically we used three of the six General-Purpose I/O ports (PA, PF, PE). In port A, we used five pins (PA2, PA3, PA5, PA6, PA7) for synchronous serial interface to interface with the Nokia5110 LCD. In port F we used the onboard buttons, pin 0 and pin 4 for starting the game and for shooting the lasers at the enemy. In port E we used pin 2 to take in the input from a potentiometer which controls the player's space ship movement (more specifically defined in Operation and Hardware design). In this project we used basic hardware components like Nokia5110 LCD, GPIO pins, edge-triggered interrupts, ADC, systick timer interrupts, PLL, and SSI to implement this space invader game.

Hardware design

Schematic:



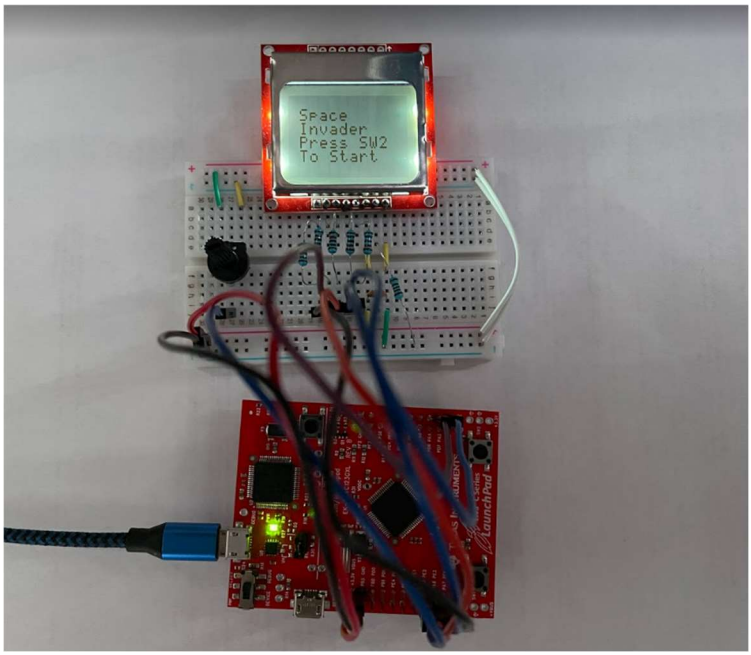
Outputs:

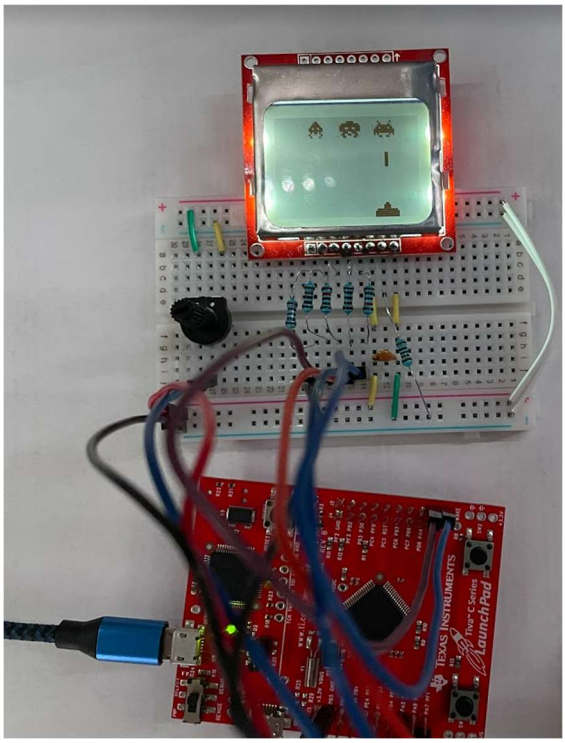
Reset for Nokia5110 LCD (Pin 1)	PA7
Chip Enable for Nokia5110 LCD (Pin 2)	PA3
Data/Command for Nokia5110 LCD (Pin 3)	PA6
Data in for Nokia5110 LCD (Pin 4)	PA5
Clock for Nokia5110 LCD (Pin 5)	PA2

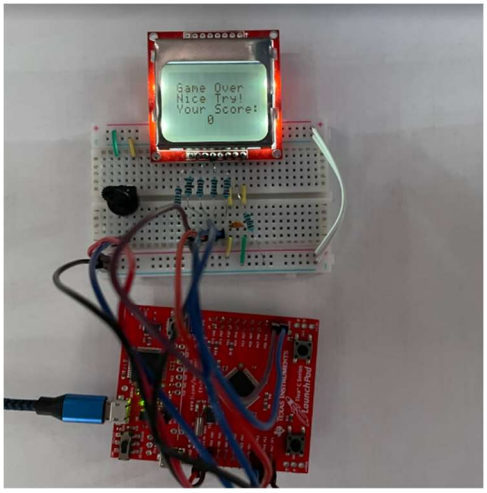
Inputs:

Switch 1 (on-board)	**PF4**
Switch 2 (on-board)	**PF0**
Potentiometer to control player space ship.	**PE2**

Pictures of Hardware System:







Software design

Software Source Code:

```
// SysTickInterrupt.c
// Documentation
// Description: Initialize SysTick timer for 204ms delay with
//    interrupt enabled and priority 1 assuming 80MHz clock
// Student Name: Abhishek Jasti, Anand Jasti

#include "tm4c123gh6pm.h"

// Initialize SysTick timer for 0.034s delay with interrupt enabled
void SysTickInterrupt_Init(void){
  NVIC_ST_CTRL_R = 0;                // disable SysTick during setup
  NVIC_ST_RELOAD_R = (16777216) - 1;        // number of counts to wait 204ms (assuming 80MHz clock)
  //NVIC_ST_RELOAD_R = (2720000*6) - 1;       // number of counts to wait 34ms (assuming 80MHz clock)
  NVIC_ST_CURRENT_R = 0;             // any write to CURRENT clears
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x1FFFFFFF)|0x20000000;  // priority 1
  NVIC_ST_CTRL_R = 0x07;             // enable SysTick with core and interrupts
}
```

In this bit of code we are initializing SysTick timer for 204ms delay with interrupt enabled and priority1. We are using the SysTick timer to sample ADC values for the potentiometer, and it is also used to implement the refresh rate for the Nokia5110 LCD.

```
// Initialize edge trigger interrupt for PF0 (SW2) and PF4 (SW1) falling edge
void PortF_EdgeTriggerInit(void){
  if ((SYSCTL_RCGC2_R &= SYSCTL_RCGC2_GPIOF) != SYSCTL_RCGC2_GPIOF){
      SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOF;   // activate port F
      while ((SYSCTL_RCGC2_R&SYSCTL_RCGC2_GPIOF)!=SYSCTL_RCGC2_GPIOF){} // wait for the clock to be ready
  }

  GPIO_PORTF_LOCK_R = 0x4C4F434B;     // unlock PortF PF0
  GPIO_PORTF_CR_R |= 0x01;            // allow changes to PF0
  GPIO_PORTF_AMSEL_R &= ~0x11;        // disable analog function
  GPIO_PORTF_PCTL_R &= ~0x000F000F;   // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R &= ~0x11;          // make PF0, PF4 input (built-in buttons)
  GPIO_PORTF_AFSEL_R &= ~0x11;        // disable alternate function on PF0, PF4
  GPIO_PORTF_PUR_R |= 0x11;           // enable pullup resistors on PF0, PF4
  GPIO_PORTF_DEN_R |= 0x11;           // enable digital pins PF0, PF4
  GPIO_PORTF_IS_R &= ~0x11;           // PF0, PF4 is edge-sensitive
  GPIO_PORTF_IBE_R &= ~0x11;          // PF0, PF4 is not both edges
  GPIO_PORTF_IEV_R &= ~0x11;          // PF0, PF4 Falling edge event
  GPIO_PORTF_ICR_R = 0x11;            // clear flag0, flag4
  GPIO_PORTF_IM_R |= 0x11;            // arm interrupt on PF0, PF4
  NVIC_PRI7_R = (NVIC_PRI7_R&0xFF1FFFFF)|0x00400000;  // priority 2
  NVIC_EN0_R |= 0x40000000;           // enable interrupt 30 in NVIC
}
```

In this code above we are initializing Port F pins 0 and 4 which are built-in push buttons to be inputs, using falling edge trigger interrupt. The push buttons are used as follows: sw1 will shoot a laser toward the enemies, sw2 will start the game at the beginning of the program.

```
// SS3 interrupts: flag set on completion but no interrupt requested
void ADC0_InitSWTriggerSeq3_Ch1(void){
  //volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000010;   // 1) activate clock for Port E
  while((SYSCTL_RCGC2_R&0x00000010) != 0x00000010){} // wait for clock to be ready
  //delay = SYSCTL_RCGC2_R;         //    allow time for clock to stabilize
  GPIO_PORTE_DIR_R &= ~0x04;      // 2) make PE2 input
  GPIO_PORTE_AFSEL_R |= 0x04;     // 3) enable alternate function on PE2
  GPIO_PORTE_DEN_R &= ~0x04;      // 4) disable digital I/O on PE2
  GPIO_PORTE_AMSEL_R |= 0x04;     // 5) enable analog function on PE2
  SYSCTL_RCGC0_R |= 0x00010000;   // 6) activate ADC0
  while((SYSCTL_RCGC0_R&0x00010000) != 0x00010000){} // wait for clock to be ready
  //delay = SYSCTL_RCGC2_R;
  SYSCTL_RCGC0_R &= ~0x00000300;  // 7) configure for 125K
  ADC0_SSPRI_R = 0x0123;          // 8) Sequencer 3 is highest priority
  ADC0_ACTSS_R &= ~0x0008;        // 9) disable sample sequencer 3
  ADC0_EMUX_R &= ~0xF000;         // 10) seq3 is software trigger
  ADC0_SSMUX3_R = (ADC0_SSMUX3_R&0xFFFFFFF0)+1; // 11) channel Ain1 (PE2)
  ADC0_SSCTL3_R = 0x0006;         // 12) no TS0 D0, yes IE0 END0
  ADC0_ACTSS_R |= 0x0008;         // 13) enable sample sequencer 3
}

// Busy-wait Analog to digital conversion
// Input: none
// Output: 12-bit result of ADC conversion
unsigned long ADC0_InSeq3(void){
  unsigned long result;
  ADC0_PSSI_R = 0x0008;            // 1) initiate SS3
  while((ADC0_RIS_R&0x08)==0){};   // 2) wait for conversion done
  result = ADC0_SSFIFO3_R&0xFFF;   // 3) read result
  ADC0_ISC_R = 0x0008;             // 4) acknowledge completion
  return result;
}
```

In this bit of code we are initializing ADC0 sequencer 3 Ain1(PE2) to be software trigger. We are using this pin to get ADC values from the potentiometer. We also provided a function to read the ADC value and return the value.

```
285  // initialize all sprites
286  void Init(void){
287    uint8_t i;
288    for(i=0;i<3;i++){
289      Enemy[i].x = 20*i;
290      Enemy[i].y = 10;
291      Enemy[i].image = SmallEnemyPointA[i];
292      Enemy[i].life = 1;
293    }
294
295    // add initialization for player ship
296    PlayerShip.x = MAX_X/2;
297    PlayerShip.y = MAX_Y-1;
298    PlayerShip.image = PlayerShip0;
299    PlayerShip.life = 1;
300
301    // add initialization for bullet and explosion
302    Laser.x = PlayerShip.x;
303    Laser.y = PlayerShip.y+8;
304    Laser.image = Laser0;
305    Laser.life = 0;
306
307    SmallExplosion.x = Laser.x;
308    SmallExplosion.y = Laser.y;
309    SmallExplosion.life = 0;
310  }
```

In this code we initialized all the in-game sprites that are needed for the space invaders game. This function initializes the enemies, the player spaceship, the laser, and the explosion.

```
382  // update the screen with new positions for all sprites
383  void Draw(void){
384    uint8_t i;
385    Nokia5110_ClearBuffer();
386    for(i=0;i<3;i++){
387      if(Enemy[i].life > 0){
388       Nokia5110_PrintBMP(Enemy[i].x, Enemy[i].y, Enemy[i].image, 0);
389      }
390    }
391
392    // draw player ship
393    Nokia5110_PrintBMP(PlayerShip.x, PlayerShip.y, PlayerShip.image, 0);
394
395    // draw bullet or explosion
396    if(Laser.life){
397      Nokia5110_PrintBMP(Laser.x, Laser.y, Laser.image,0);
398    }
399
400    if(SmallExplosion.life){
401      Nokia5110_PrintBMP(SmallExplosion.x, SmallExplosion.y, SmallExplosion.image,0);
402    }
403
404    Nokia5110_DisplayBuffer();      // draw buffer
405  }
406
```

In this bit of code we update the LCD screen with the new positions for all sprits. This function draws the enemies, player's spaceship, laser, and/or explosion.

```
311
312  // update the positions for all sprites
313 void Move(void){
314   uint8_t i;
315   unsigned long ADC_value;
316   unsigned int playershipPosition;
317   for(i=0;i<3;i++){
318     if(Enemy[i].x < MAX_ENEMYX){
319       Enemy[i].x += 1;
320       if(Enemy[i].image == SmallEnemyPointA[i]){
321           Enemy[i].image = SmallEnemyPointB[i];
322       }
323       else if(Enemy[i].image == SmallEnemyPointB[i]){
324           Enemy[i].image = SmallEnemyPointA[i];
325       }
326     }else{
327       Enemy[i].life = 0;
328     }
329   }
330
331   // read ADC value for player ship
332   ADC_value = ADC0_InSeq3();
333   playershipPosition = ADC_value*(SCREENW-18)/4095;
334
335   // update player ship
336   PlayerShip.x = playershipPosition;
337
338   // update bullet or explosion
339   if(SW1Pressed){
340     Laser.x = PlayerShip.x+8;
341     Laser.y = PlayerShip.y-7;
342     Laser.life = 1;
343     SW1Pressed = 0;
344   }
345
346   for(i=0;i<3;i++){
347     if(Laser.x+2>=Enemy[i].x && Laser.x<=Enemy[i].x+16 && Laser.y-9<=10 && Laser.life && Enemy[i].life){
348       Enemy[i].life = 0;
349       SmallExplosion.x = Enemy[i].x;
350       SmallExplosion.y = Enemy[i].y;
351       SmallExplosion.life = 1;
352       Laser.life = 0;
353       playerScore++;
354     }
355   }
356
357   if(Laser.life){
358     if(Laser.y>2){
359       Laser.y-=2;
360     }
361     else{
362       Laser.life = 0;
363     }
364   }
365
366   if(SmallExplosion.life){
367     counter++;
368     if(SmallExplosion.image == SmallExplosion0){
369       SmallExplosion.image = SmallExplosion1;
370     }
371     else{
372       SmallExplosion.image = SmallExplosion0;
373     }
374
375     if(counter==3){
376       SmallExplosion.life = 0;
377       counter = 0;
378     }
379   }
380 }
381
```

In this code above we are updating the positions of all the in-game sprites. The move function updated the position of the enemies, laser, and/or the explosion. Before moving the

player's spaceship we first read the ADC value of the potentiometer, and move the spaceship based on the ADC value. This function also detects if a laser has hit an enemy, and increments the player's score. This function will also detects when switch 1 is pressed and shoots the laser out of the spaceship.

```
220 int main(void){
221   DisableInterrupts();
222   PortF_EdgeTriggerInit();
223   PLL_Init(Bus80MHz);                          // set system clock to 80 MHz
224   Nokia5110_Init();
225   SysTickInterrupt_Init();
226   ADC0_InitSWTriggerSeq3_Ch1();
227   EnableInterrupts();
228   Nokia5110_Clear();
229   Nokia5110_OutString("             Space       Invader     Press SW2   To Start");
230
231   while(1){
232     // if game is off
233     // display beginning message
234
235     // if game is on
236     //  if time to draw:
237     //    draw/move
238     if(SW1Pressed){
239       Nokia5110_Clear();
240       Nokia5110_OutString("             Space       Invader     Press SW2   To Start");
241       SW1Pressed = 0x00;
242     }
243
244     if(SW2Pressed){
245       playerScore = 0;
246       Init();
247       while(Enemy[0].life >0 || Enemy[1].life >0 || Enemy[2].life >0){
248         if(Refr_tick){
249           Move();
250           Draw();
251           Refr_tick = 0x00;
252         }
253       }
254       Nokia5110_Clear();
255       Nokia5110_OutString("             Game Over   Nice Try!   Your Score: ");
256       Nokia5110_OutUDec(playerScore);
257       SW2Pressed = 0x00;
258       Delay_50ms(55);
259       SW1Pressed = 0x01;
260     }
261     //Delay_50ms(18);                    // delay ~0.5 sec at 80 MHz
262   }
263 }
264
```

In this bit of code above, the main function implements the game logic of the space invaders game.

Conclusion

Implementing this space invader game project was not hard as we thought it would be. Since the project was split up into four different parts, it made it easy for us to focus on one

aspect of the project at a time. Connecting the Nokia5110 LCD to the board was very easy. The coding/software design for this project was very straight forward because we were given the driver functions for the Nokia5110 LCD. This project was very helpful, we gained more understanding and reviewed all the topics of Nokia5110 LCD, GPIO pins, edge-triggered interrupts, ADC, SysTick timer interrupts, PLL, and SSI. We learned hot display text and simple image on Nokia5110 LCD, and how to use serial synchronous interface to interface with the LCD, this project was also a great review for PLL, SysTick timer interrupts, edge-triggered interrupts, and ADC. Overall, this was a very fun project to do.