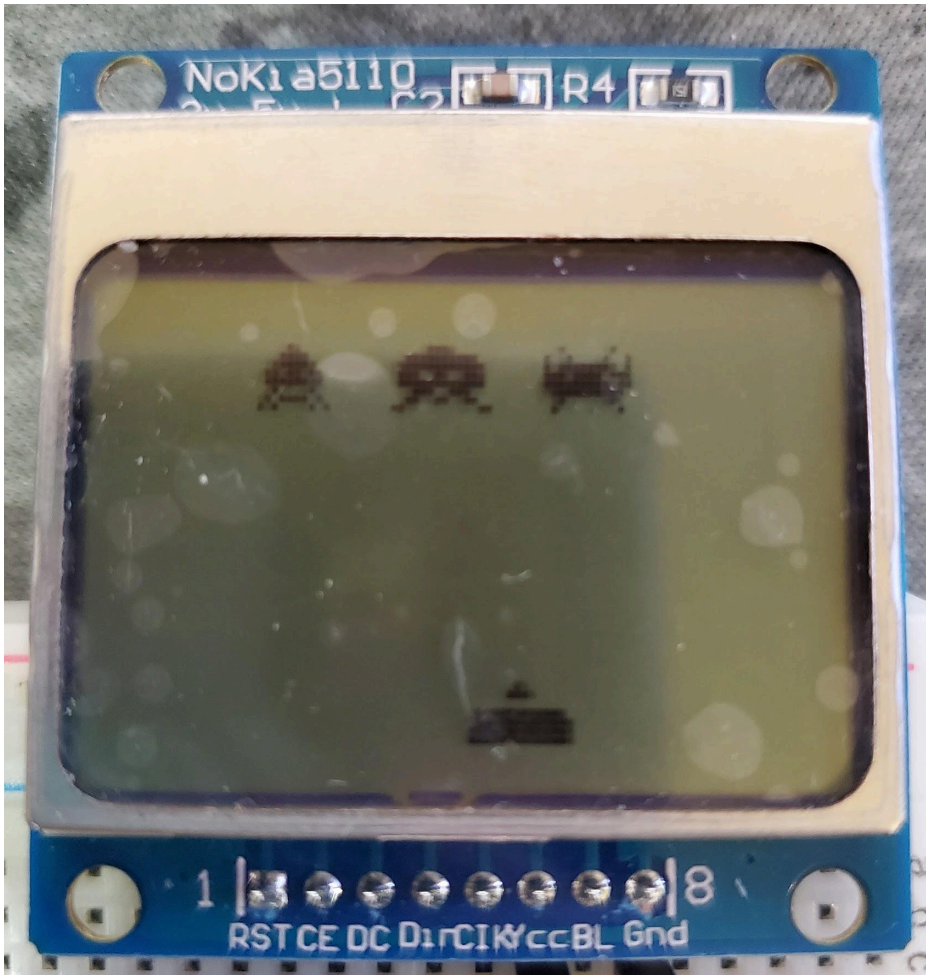


CECS 347 Spring 2023
Project 3: Space Invaders Game

Brandon Quach
Phuc Ngo

05/03/2023

Space Invaders Game on a Nokia 5110 LCD screen

Introduction:

In this project, we will be coding a space invaders game using the Nokia5110. The requirement will be to create a minimum of three different sprites for the space invaders and a single spaceship. The potentiometer will be used to move the spaceship left and right. 2 onboard switches will be used (one to start the game and the other to fire a bullet).

Operation:

In order for the system to work, the .c and .h files would have to be downloaded. Once all the files are downloaded, run the files on Keil V5 with the board connected. For this demo, you will need:

- LaunchPad
- Jumper Wires
- Nokia 5110 LCD screen
- Rotary Potentiometer
- BreadBoard

Connect the following wires of the Nokia 5110 to the LaunchPad and run the program.

Blue Nokia 5110

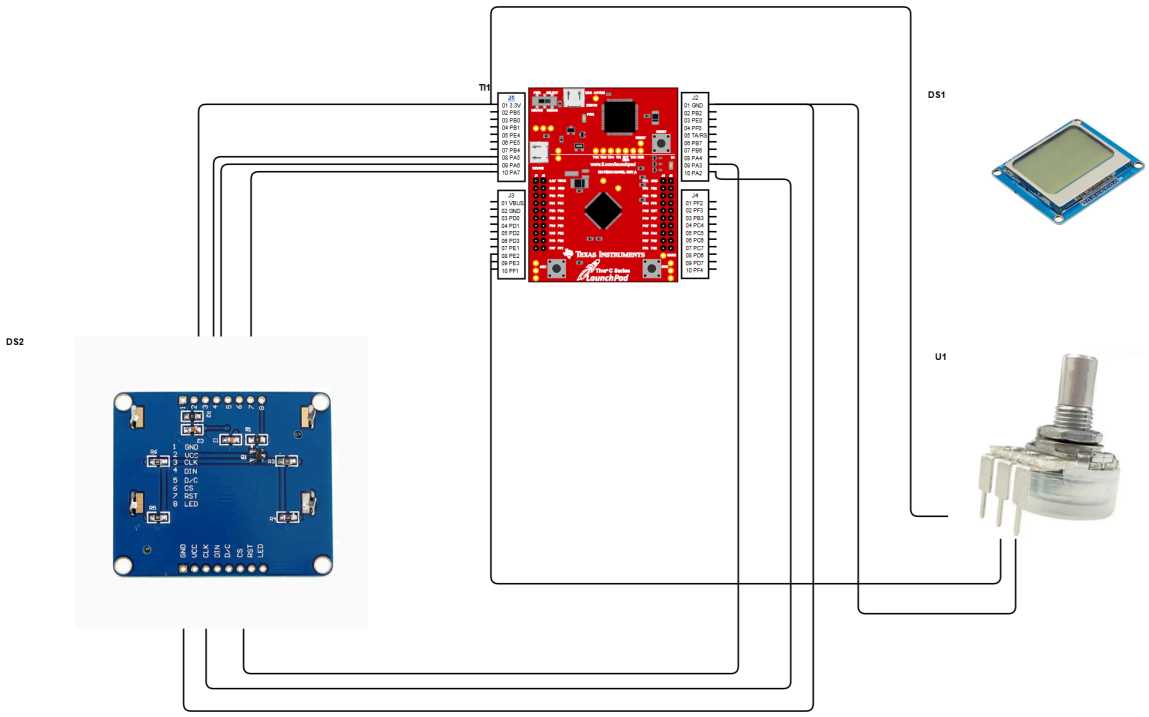
Signal	(Nokia 5110) LaunchPad pin
Reset	(RST, pin 1) connected to PA7
SSI0Fss	(CE, pin 2) connected to PA3
Data/Command	(DC, pin 3) connected to PA6
SSI0Tx	(Din, pin 4) connected to PA5
SSI0Clk	(Clk, pin 5) connected to PA2
3.3V	(Vcc, pin 6) power
back light	(BL, pin 7) not connected
Ground	(Gnd, pin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

Lab Demo Link - https://www.youtube.com/watch?v=iGHJD4FsCeM

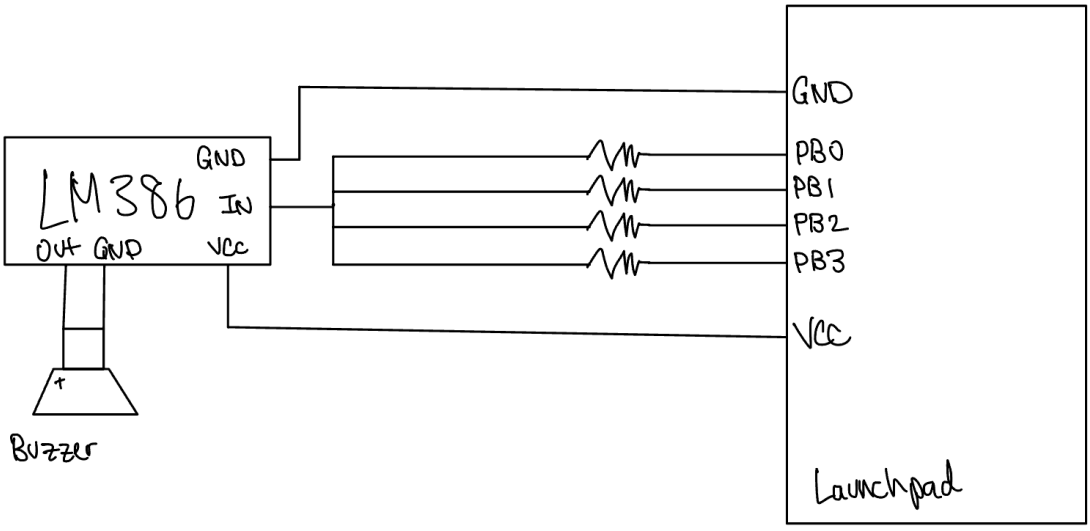
Theory:

The blue Nokia 5110 will display the game code programmed to run space invaders. A 10 Hz screen refresh will be implemented using SysTick timer and a 80 MHz clock system clock will be implemented by PLL. In order to move the ship, an ADC driver will be collecting samples and calculating the voltage. The software will convert that calculated voltage to position which makes the potentiometer move the spaceship. A bullet can be shot from the ship towards the space invaders using SW1. If there is already a bullet on the screen, pressing SW1 will do nothing. If there are no more invaders on the screen (enemies = DEAD), then the game will display the total score and the game will come to an end.

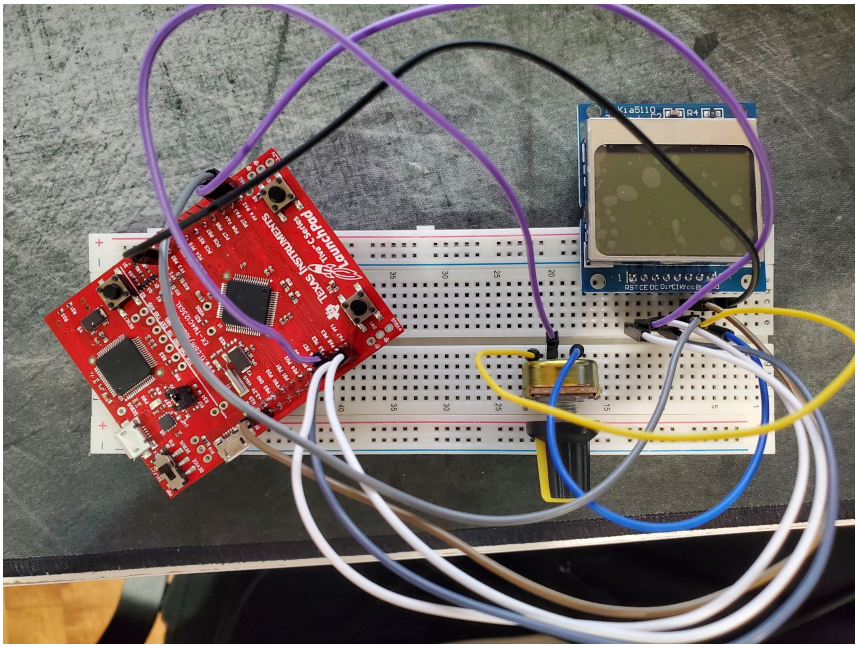
Hardware Design:



Schematic of hardware design



Schematic of DAC hardware design



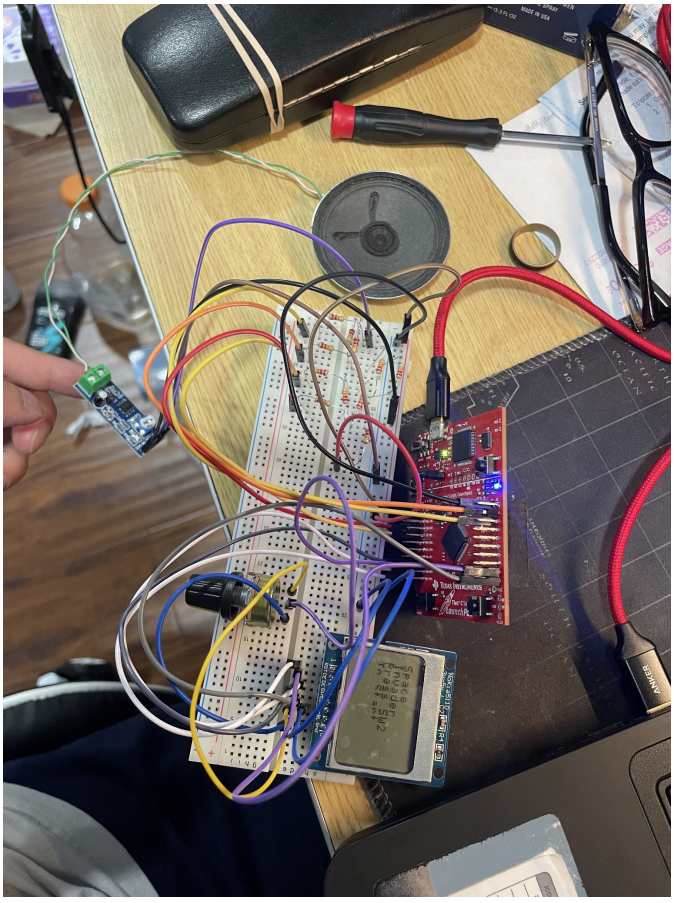


Photo of hardware

Software Design:

In this project, we separated it into two main functions for the LCD; Draw(); and Move();. Each function has its own function by creating the animation for characters we need which are space invaders, the spaceship, and the bullet. After the draw function, the next objective was how to make those characters move with 3 different types of sprites and to update the animation of the explosion once the enemy is hit. To make the enemy explode, we needed to use 2 different types of function. In other words, when the bullet is shot and it hits the enemy, the explosion function will be called and replace the enemy sprite and disappear.

```
//
  // Update the player ship position in display buffer
  Nokia5110_PrintBMP(PlayerShip.x, PlayerShip.y, PlayerShip.image, 0);

  // Update the bullet position in display buffer if there is one.
  if (Bullet.life==ALIVE) {
    Nokia5110_PrintBMP(Bullet.x, Bullet.y, Bullet.image, 0);
  }

  if (SmallExplosion.life == ALIVE){
    Nokia5110_PrintBMP(SmallExplosion.x, SmallExplosion.y, SmallExplosion.image, 0);
    Sound_Killed();
    Sound_Explosion();
    //Delay100ms(100);
    SmallExplosion.x = 0;
    SmallExplosion.y = 0;
  }

  //adding explosion
  //if hit the enemy
  for(i=0;i<3;i++){
    if(Bullet.x+2>=Enemy[i].x && Bullet.x<=Enemy[i].x+16 && Bullet.y-9<=10 && Bullet.life && Enemy[i].life){
      Enemy[i].life = 0;
      SmallExplosion.x = Enemy[i].x;
      SmallExplosion.y = Enemy[i].y;
      SmallExplosion.life = 1;
      Bullet.life = 0;
      score++;
    Bullet.life = 0;
    }

  }

  //if bullet is alive/dead
  if(Bullet.life){
    if(Bullet.y>2){
      Bullet.y-=2;
    }
    else{
      Bullet.life = 0;
```

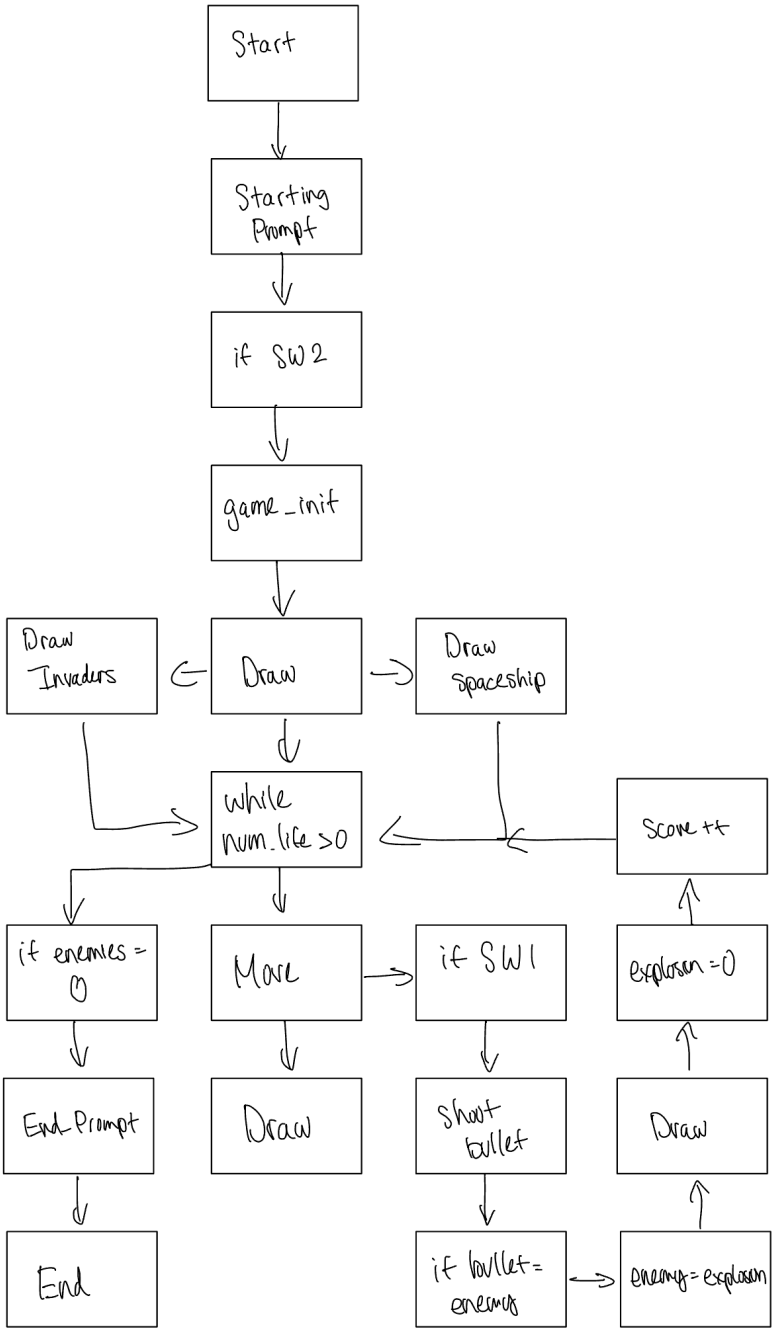
```
// Update positions for all alive sprites.
void Move(void){
  uint8_t num_life = 0;
  uint8_t i;
  unsigned long ADC_value;
  unsigned int playerPosition;
//  for(i=0;i<3;i++){
//    if(Enemy[i].x < MAX_ENEMYX){
//      Enemy[i].x += 1;
//    }else{
//      Enemy[i].life = 0;
//    }
//  }
  for(int i=0;i<3;i++){
      if(Enemy[i].x < MAX_ENEMYX){
        Enemy[i].image = Enemy[i].image == SmallEnemyPointA[i] ? SmallEnemyPointB[i] : SmallEnemyPointA[i];
        Enemy[i].x += 1;
        num_life++;
      } else {
        Enemy[i].life = 0;
      }
    }

  // Move enemies, check life or dead: dead if right side reaches right screen border or detect a hit

  // Read ADC and update player ship position: only x coordinate will be changed.
    ADC_value = ADC0_InSeq3();
    playerPosition = ADC_value*(SCREENW-18)/4095;
    PlayerShip.x = playerPosition;

  if (num_life==0) {
    game_s = OVER;
    End_Prompt();
    Delay100ms(300);
    Start_Prompt();
  }
}

// Update the screen:
```



Conclusion:

The lab was successful in reviewing topics : PLL, SysTick timer, edge-triggered interrupts, and ADC. The lab helped us understand what is required to implement a visual onto a LCD screen and how to use ADC in order to use a potentiometer as a joystick. The most interesting thing I learned from this lab is how an ADC driver can calculate voltage in order to translate that into position on the Nokia 5110. A difficulty of this project was translating SpaceInvadersV0.c(where the main can be changed) to the SpaceInvaders.c(where the main can't

be changed). We were stuck in this step for a while because we would forget really small lines of code that would make the game start.