

Video juego

1. Circuitos utilizados:

Para realizar el proyecto, se realizaron 7 circuitos diferentes independientes a las conexiones realizadas para la conexión de la pantalla a la tivac.

De estos 7 circuitos distintos, 6 son iguales entre ellos ya que fueron utilizados para los botones del juego todos en la configuración Pull up. 4 botones de movimiento izquierda derecha, 2 para cada jugador, los últimos dos botones son para que cada nave pudiera disparar. Por último el séptimo circuito fue utilizado para la conexión del buzzer a la tiva, para poder escuchar tanto la música del menú como la música del final.

Las conexiones realizadas de la tiva a la pantalla fueron realizadas utilizando las conexiones vistas en clase, se utilizó la conexión SPI del laboratorio #5 en el cual utilizamos la SD y además se utilizó la conexión vista en clase para poder desplegar imágenes en la pantalla.

Los 7 circuitos tienen una conexión de tierra al componente y el componente al pin de la tivac.

2. Gráficos:

Para los gráficos del juego todo los diseños fueron realizados en la página piskel. Luego la foto es pasada por un conversor el cual nos da el código en hexadecimal de cada color en el orden necesario para que se vea de la manera correcta en la pantalla. Se tienen los diseños de los enemigos, las dos naves como sprites, los asteroides y las dos balas, estos fueron desplegados como sprites. Luego el fondo del juego así como el título fueron desplegados como bitmaps. Para hacer la animación de las naves fue necesario hacer 3 diseños en piskel y descargarlos con 6 columnas. Además de esto para poder desaparecer a los enemigos se realizó otro fondo pero este siendo de 32x32 pixeles, esto fue desplegado encima de un enemigo cuando la bala los alcanzaba para poder hacerlos desaparecer.



3. Código:

Definimos las notas que utilizamos para la canción del final del juego,

```
41
42 #define note_cc 261
43 #define note_dd 294
44 #define note_ee 329
45 #define note_ff 349
46 #define note_g 391
47 #define note_gS 415
48 #define note_a 440
49 #define note_aS 455
50 #define note_b 466
51 #define note_cH 523
52 #define note_cSH 554
53 #define note_dH 587
54 #define note_dSH 622
55 #define note_eH 659
56 #define note_fH 698
57 #define note_fSH 740
58 #define note_gH 784
59 #define note_gSH 830
60 #define note_aH 880
61
```

En la configuración, realizamos un while, esto para poder mantener el juego en la pantalla de inicio, más adelante modificamos el start == 0 con un botón para poder comenzar a jugar. Dentro del while imprimimos el fondo del juego junto con los nombres y el logo de galaga

```
while (start == 0) {

    //midi();
    //FillRect(0, 0, 320, 240, 0x0000);
    LCD_Bitmap(0, 0, 320, 240, fondo);
    LCD_Bitmap(200, 20, 100, 75, gala);
    //LCD_Bitmap(50,50,50,50,gala);
    // LCD_Sprite(0, 0, 100, 100, galaga, 1, 0, 0, 0)
    String text1 = "GALAGA";
    String text2 = "Fernando Figueroa";
    String text3 = "Erick Aquino";
    String text4 = "Press 'START'";
    LCD_Print(text1, 20, 50, 2, 0x07E0, 0x0002);
    LCD_Print(text2, 20, 100, 2, 0x07E0, 0x0007);
    LCD_Print(text3, 20, 150, 2, 0x07E0, 0x0009);
    LCD_Print(text4, 50, 200, 2, 0xFFE0, 0x0002);
    midi();
```



Figura No.2 - Menu del juego

Realizamos un if revisando la variable enemigo, mientras esta sea 0 se mostrará en pantalla la nave enemiga, pero al momento de cambiar la variable enemigo a 1 esta desaparecerá ya que cambiamos la nave por un bitmap llamado explotar el cual es igual que el fondo, lo que causa el efecto que la nave enemiga desaparezca cuando la toca una bala. Este mismo if se hizo para cada enemigo con una variable enemigo distinta, y se debe cambiar la posición del enemigo y de el bitmap explotar

```
if (enemigo == 0) {  
    LCD_Sprite(18, 120, 32, 32, pulgon_verde, 1, 0, 0, 0);  
}  
else {  
    LCD_Bitmap(18, 120, 32, 32, explotar);  
}
```



Figura No.3 - enemigos antes y después de explotar

Estos dos if son utilizados para el movimiento del jugador 1. Revisamos que el botón que se encuentra al pin PF1 esté presionado. Cuando esto ocurre aumentamos la variable x en 3 unidades, luego revisamos si la variable x es mayor a 105, si esto es verdadero regresamos la variable x al valor 105, esto para fingir una pared, así

la nave no podrá pasarse al lado del jugador 2. El segundo if utilizado es igual que el primero, con la diferencia que no aumentamos el valor sino que lo disminuimos ya que la nave viene de regreso en el plano. En lugar de revisar 105 lo hacemos revisando 0, de la misma manera para que la nave no se pase de su espacio de movimiento. El código para la segunda nave será igual, la única diferencia sera el valor de la x que se revisa, para poder dejar al jugador 2 dentro de su espacio de juego

```
/*  
if (digitalRead(PF_1) == 0) {  
    x = x + 3;  
    antirebote1 = 0;  
    if (x > 105) {  
        x = 105;  
    }  
    delay(30);  
    int anim = (x / 5) % 8;  
    LCD_Sprite(x, 175, 32, 32, Nave, 3, anim, 1, 0);  
}  
  
*/
```

```
/*if(digitalRead(PF_2)==1){  
antirebote2 = 1;  
}*/  
if (digitalRead(PF_2) == 0) {  
    x = x - 3;  
    antirebote2 = 0;  
    if (x < 0) {  
        x = 0;  
    }  
    delay(30);  
    int anim = (x / 5) % 8;  
    LCD_Sprite(x, 175, 32, 32, Nave, 3, anim, 1, 0);  
}
```

Este código fue utilizado para la bala así como para comparar las posiciones de cada enemigo para poder desaparecer. Primero revisamos que el botón esté presionado así como la variable estado1 para evitar el anti-rebote y que salga más de una bala a la vez. Cambiamos la variable estado1 para el anti rebote, le asignamos a la variable yb1 150, esto sera la posicion inicial de la bala, para que salga justo enfrente de la nave y el sprite de la bala no interfiera con la nave por último le asignamos a la bala la posición de la nave, para que esta salga del mismo lugar de la nave y no en otra posición. En el siguiente if revisamos que la bala sea mayor a 0 para saber cuándo detener la bala, revisamos el estado de anti rebote y la variable contador bala. Cuando se cumplen todas estas condiciones, disminuimos la variable yb1 en 7 unidades y la volvemos a imprimir, debido a que la variable yb1 representa la posición en y de la bala, cada disminución será el movimiento de la bala ya que la estamos imprimiendo. En el siguiente if solo revisamos que ya la bala se haya detenido para poder regresar la variable estado1 a 0, para realizar la lectura de otro disparo. Por último en el if final, comparamos la posición de la bala con la posición del primer enemigo, si esta es igual, esto causará que la variable enemigo cambie a 1, causando que if de la variable enemigo explicado anteriormente cambie lo que desaparece la nave enemiga, además la variable contador bala cambiará a 1 indicando que el primer enemigo debe de quedarse desaparecido y regresamos el estado a 0 para el anti rebote. Se realizó el mismo código para todos los enemigos, únicamente cambiando la posición de cada enemigo y el nombre de la variable contbala y la de enemigo, para poder darle a cada enemigo su propia variable relacionada a su posición.

```
if ((digitalRead(PA_7) == 0) && (estadol == 0)) {  
    estadol = 1;  
    ybl = 150;  
    xbalal = x;  
}  
if (ybl > 0 && estadol == 1 && contBala == 0) {  
    ybl = ybl - 7;  
    delay(5);  
    LCD_Sprite(xbalal, ybl, 32, 32, bala, 1, 0, 1, 0);  
    if (ybl <= 0) {  
        estadol = 0;  
    }  
    if (ybl <= 120 && xbalal == 18) {  
        enemigo = 1;  
        contBala = 1;  
        estadol = 0;  
    }  
}
```

Utilizamos este if para reproducir música cuando gane un jugador, revisamos que todos los enemigos ya hayan desaparecido, esto lo hacemos mediante las variables contBala, que es la que se modifica cuando se desaparece cada enemigo, si todos los enemigos ya no están mostramos un mensaje junto con la música para indicar qué jugador ganó. La función beep, reproduce las notas que definimos al principio así como el tiempo del tono. Se realiza un if igual para el otro jugador únicamente se cambian las variables de controlar para que el if funcione con los enemigos del jugador 2.

```
if (contBala == 4 && contBala2 == 4 && contBala3 == 4 && contBala4 == 4) {  
    String text5 = "Jugador1 gana";  
    LCD_Print(text5, 50, 80, 2, 0x07E0, 0x0002);  
    imagen1();  
    beep(note_a, 500);  
    beep(note_a, 500);  
    beep(note_a, 500);  
    beep(note_ff, 350);  
    beep(note_cH, 150);  
    beep(note_a, 500);  
    beep(note_ff, 350);  
    beep(note_cH, 150);  
    beep(note_a, 650);  
  
    delay(150);  
    //end of first bit  
  
    beep(note_eH, 500);  
    beep(note_eH, 500);  
    beep(note_eH, 500);  
    beep(note_fH, 350);  
    beep(note_cH, 150);  
    beep(note_gS, 500);  
    beep(note_ff, 350);  
    beep(note_cH, 150);  
    beep(note_a, 650);  
  
    delay(150);  
    //end of second bit...  
  
    beep(note_aH, 500);
```

En la SD tenemos un archivo de texto, el cual es desplegado en energía, cuando un jugador gana, para realizar este código se utilizó el laboratorio#6 de almacenamiento sd. Se utiliza la conexión spi de la pantalla para poder leer la sd y así imprimir en pantalla el mensaje del ganador. Para el jugador se utiliza el mismo código solo se manda a llamar el segundo archivo de texto de que se encuentra en la sd

```
void image1 (void){  
    Serial.println("Opening Player1.txt");  
    myFile = SD.open("Player1.txt");  
    if (myFile) {  
        Serial.println("Player1.txt:");  
        while (myFile.available()) {  
            Serial.write(myFile.read());  
        }  
        myFile.close();  
    }  
    else {  
        Serial.println("File Could not be Opened/Found");  
    }  
}
```

Opening Player1.txt
Player2.txt:

A decorative graphic featuring a grid of small black dots arranged in a pattern that resembles a stylized tree or mountain range. The dots are concentrated in a central vertical column, with more dots branching out to the left and right, creating a symmetrical, organic shape.

Opening Player.txt
Player1.txt:

A decorative horizontal border at the bottom of the page. It features a repeating pattern of short diagonal lines and small circles, creating a textured, woven-like appearance.

Figuras 4 y 5 - Resultado mostrado en el monitor serial dependiendo del ganador

4. Anexos

[Link del video: https://youtu.be/ZfDASz8dOhU](https://youtu.be/ZfDASz8dOhU)

Link de GitHub Fernando Figueroa:

<https://github.com/Fernando14175/Fernando-Figueroa-14175.git>

Link de GitHub Erick Aquino:

https://github.com/TheDany4545/LABS_DIGITAL_2/tree/main/main/Videojuego_Galaga