Universidad del Valle de Guatemala
Facultad de Ingeniería
Digital 2
Primer Ciclo, 2020
Steven Josue Castillo Lou 17169
Vincenzo Musella 17194

Fecha:
viernes, 8 de mayo

PROYECTO NO.2
VIDEOJUEGO EN PANTALLA TFT - TivaC
SPACE INVADERS

En este documento se encuentran una serie de imágenes y explicaciones de cómo es que se logró llevar a cabo el proyecto del videojuego. También se explica las complicaciones que se presentaron de la lógica del código a lo largo del proyecto. El documento cuenta con una serie de screenshots extraídos de "energía" y "arduino" que es donde se trabajó la programación del proyecto. Es importante mencionar que el documento es puramente informativo de manera muy superficial ya que la explicación detallada se dio el día de la presentación. También al final de este documento se encuentra un LINK que redirige a un video que muestra el funcionamiento del proyecto ya finalizado.

1. Listado de materiales utilizados para el proyecto

Material	Cantidad por integrante	Propósito
Tiva C Series Launch Pad	1	Herramienta programable para cargar el proyecto y realizar la comunicación con el arduino y la pantalla +SD.
2.4" TFT LCD Shield	1	Pantalla para mostrar el funcionamiento del videojuego.
Arduino UNO	1	Se utilizó para guardar la musica y reproducirla con la Tiva.
Push-buttons	6	Toman el roll del control del videojuego. Son tres para cada jugador.
Resistencia (470 ohms)	6	Se usaron con la conexión de los push-buttons.
Resistencia (1K ohms)	1	Resistencia para el buzzer.
Buzzer	1	Reproducir la música del videojuego.
Transistor 2N3904	1	Se utilizó junto con el buzzer para mejorar la calidad de sonido.
Jumpers (h-h o h-m)	30 (tener extras)	Realizar las conexiones entre la pantalla y la tiva como también el arduino. Se recomienda jumpers cortos para evitar el mal contacto en el proyecto.

2. Circuitos cableados de las conexiones Tiva/TFT/Arduino.

Figura 1. Cableado final del proyecto

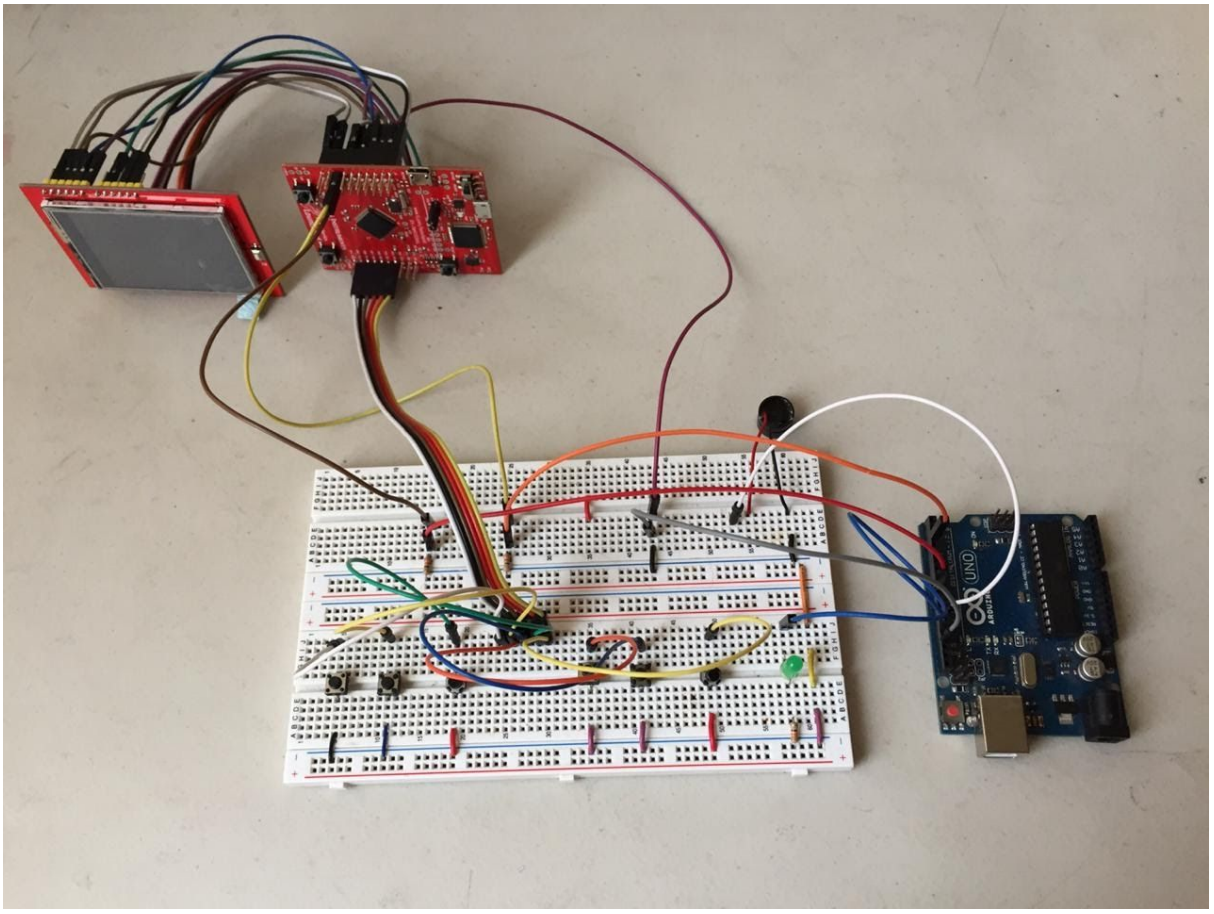
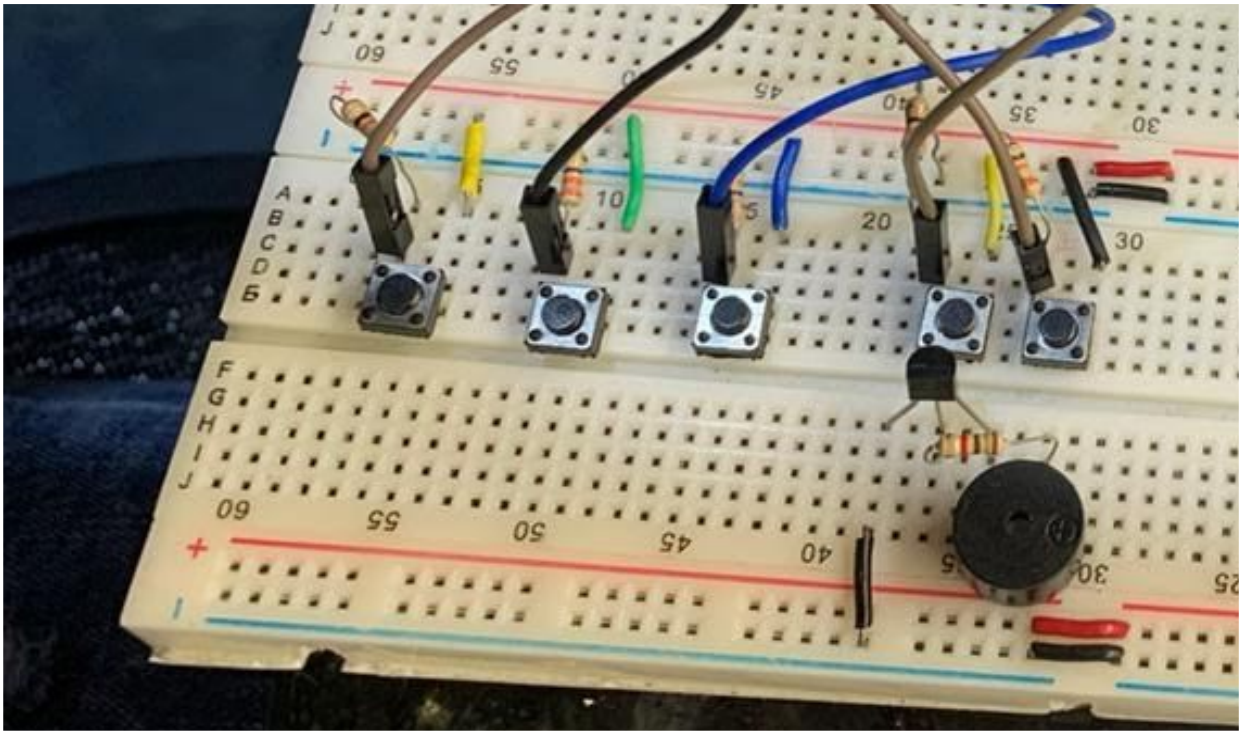


Figura 2. Cableado de los push-buttons y buzzer



3. Etapas del funcionamiento del videojuego en imágenes.

Figura 3. Juego inicializado (POWER ON)

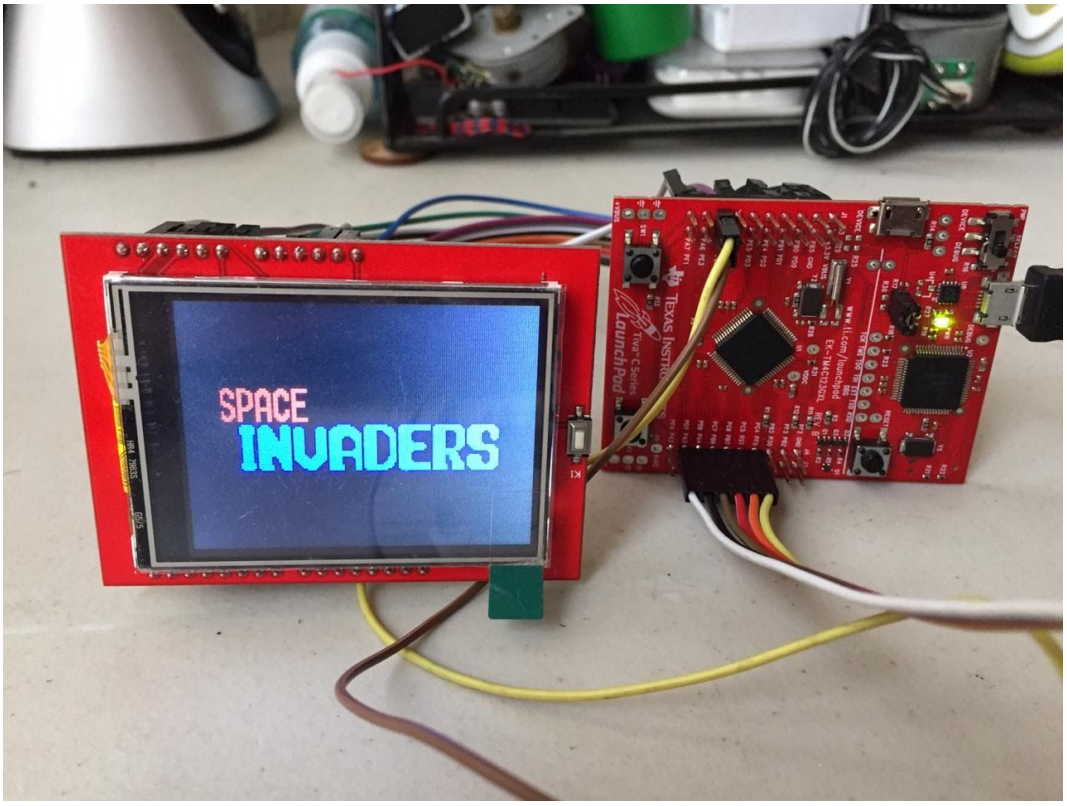


Figura 4. Menú de espera que muestra los aliens y naves de los jugadores

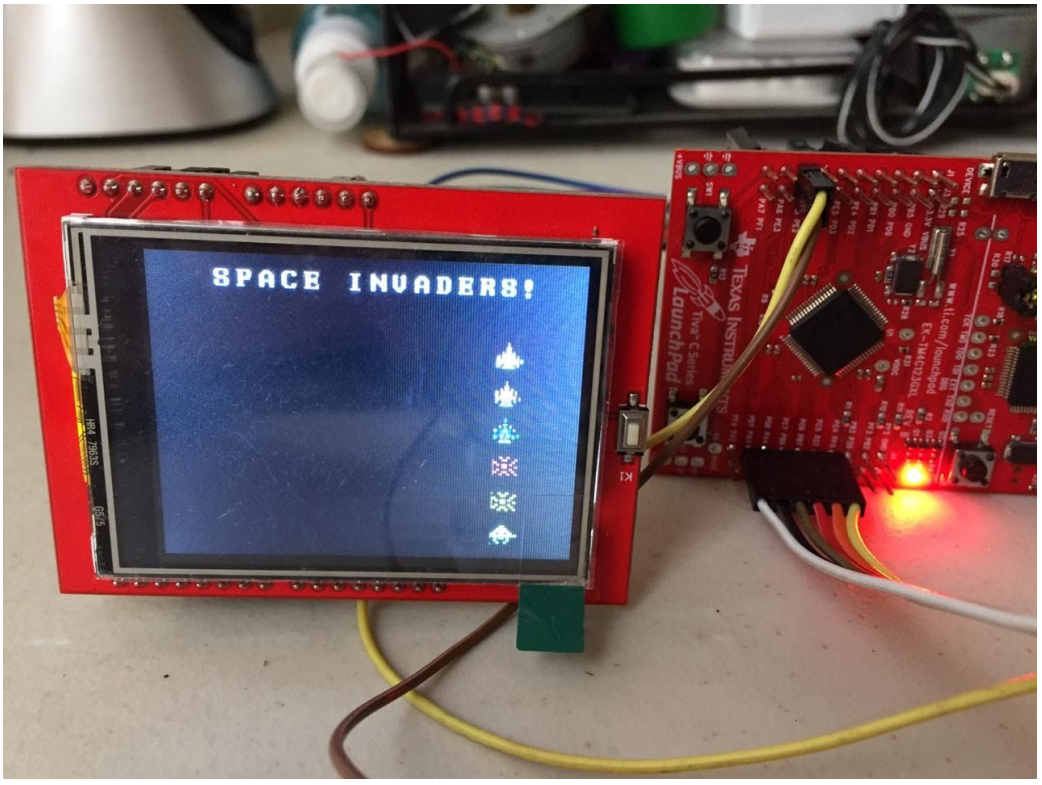


Figura 5. Pantalla de inicio del juego (comienza el juego)

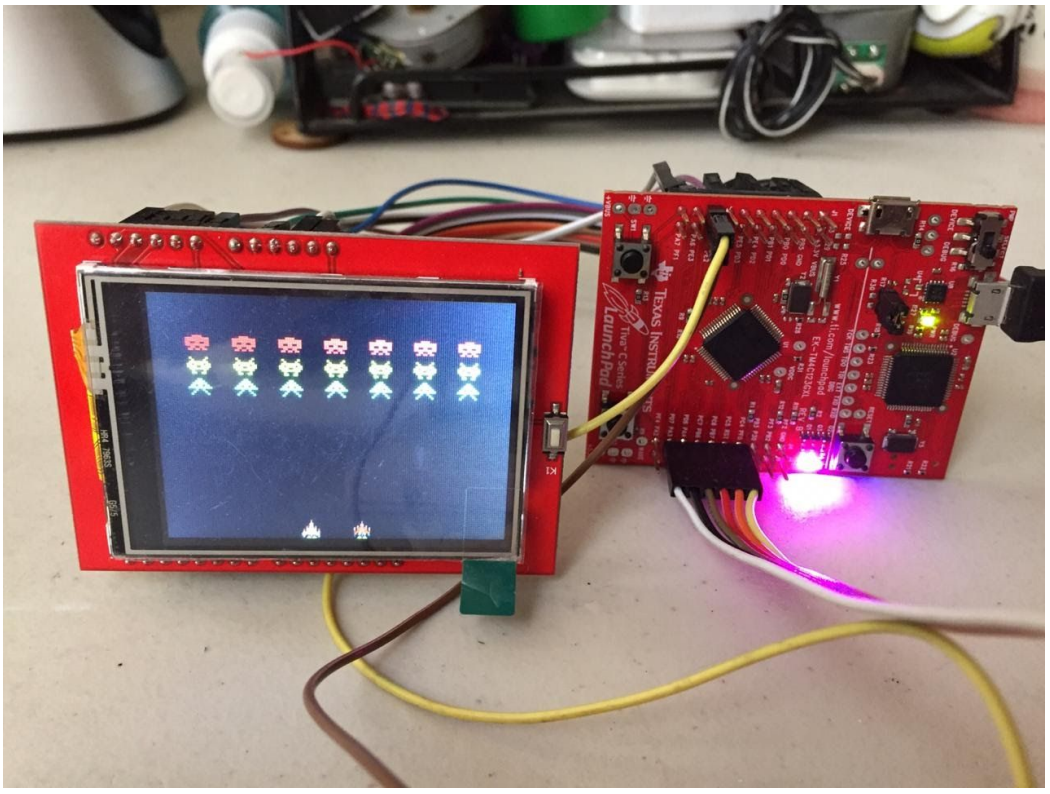


Figura 6. Movimiento y ataque de las naves

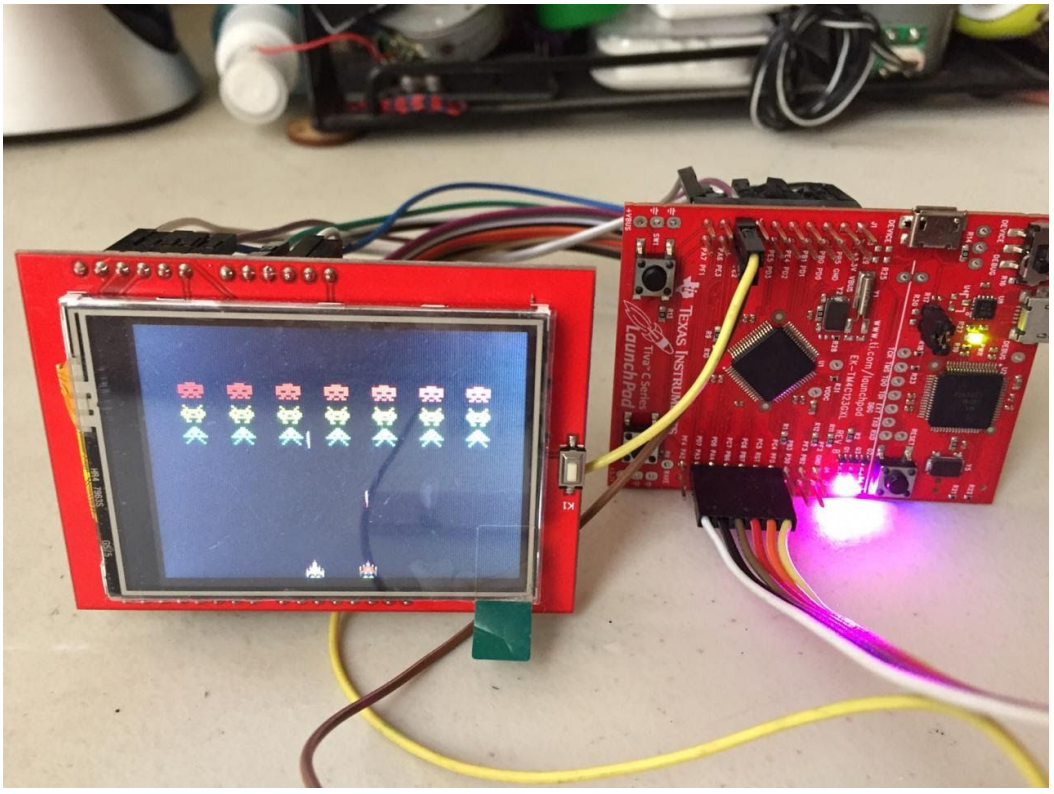


Figura 7. Impacto de la bala de una nave con un alien (enemigo)

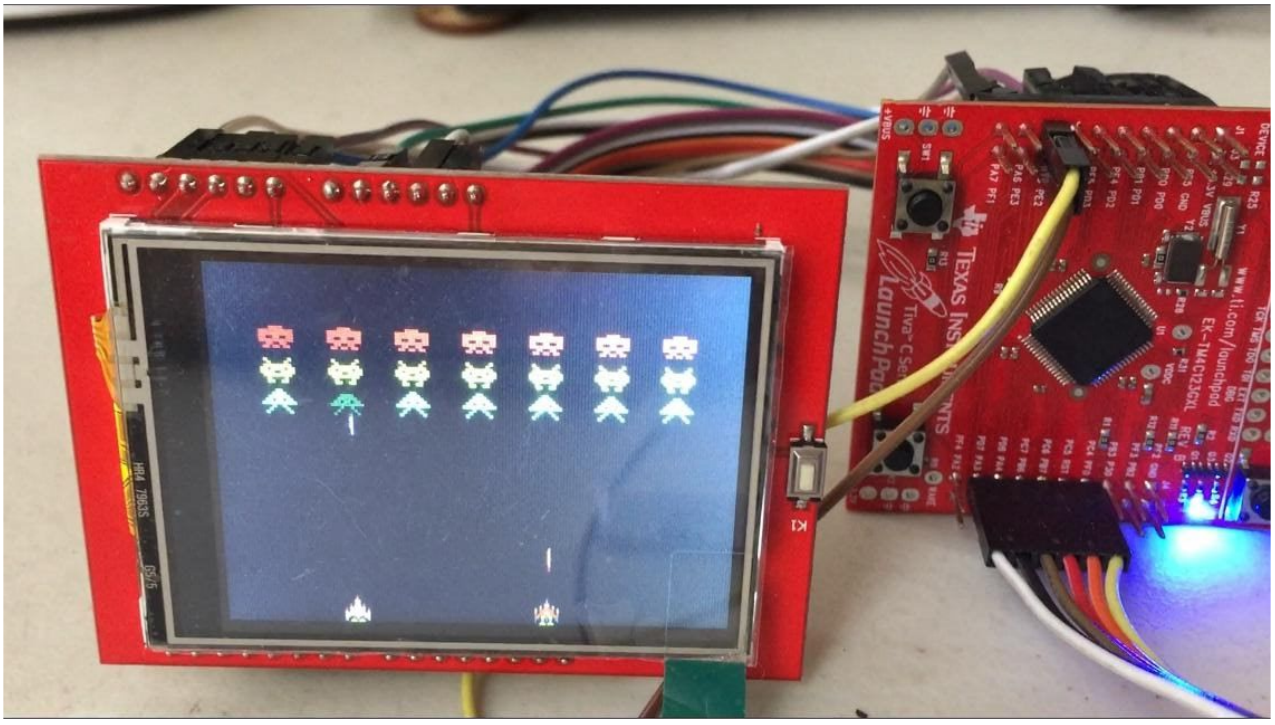
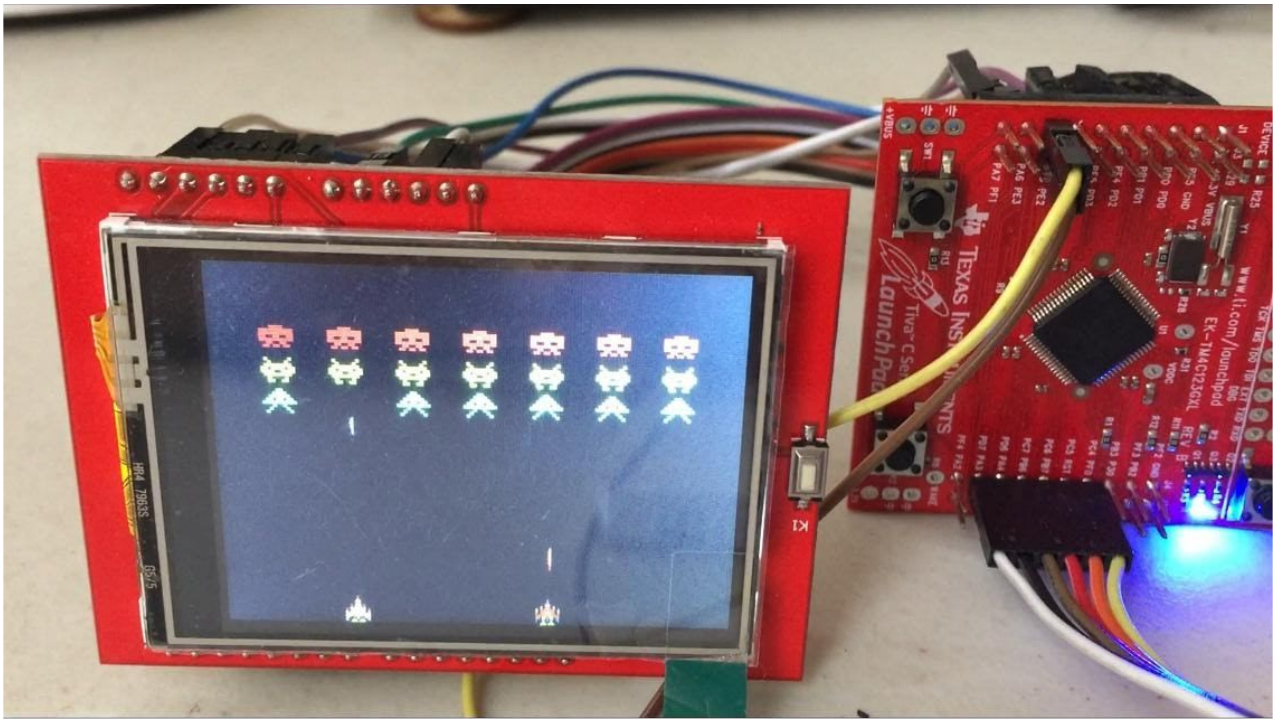


Figura 8. Eliminación de Alien (enemigo) luego del impacto de bala



4. Explicación del proceso que se muestra en las imágenes del numeral 3.

Como se observa en la figura número 3, el juego es encendido y despliega el fondo principal con el título del videojuego. Seguido a esto se presiona el botón SW1 de la tiva para proceder al siguiente menú que sería donde se despliegan todos los sprites de los aliens y las naves que usa cada jugador. En esta ventana se tiene un pequeño delay que es para permitir que el juego mientras se encuentra en dicha ventana pueda terminar de cargar todas las animaciones que se encuentran al finalizar el delay. Esto es tal y como se muestra en la figura 5, la pantalla del inicio del juego donde cada jugador ya puede comenzar a mover a su nave de izquierda a derecha como empezar a disparar a los aliens que es lo que se muestra en la imagen siguiente. En la figura 7 se observa como una bala de las de las naves choca con un alien y este se observa con un tono distinto al del resto. Esto le indica al jugador que el alien fue impactado por su bala y próximamente será eliminado. En la figura 8 observamos lo que ocurre después del impacto y es que el alien desaparece dejando ese espacio vacío y permitiéndole así al jugador seguir disparando a los aliens que siguen.

5. Complicaciones

A lo largo del proyecto nos encontramos con una serie de complicaciones tanto por cuestiones de trabajo a distancia como errores y bugs en la programación. Una de las complicaciones más extrañas que tuvimos fue que cuando estuvimos trabajando con la SD y guardar el fondo principal que en nuestro caso era el más pesado y nos liberaba bastante espacio en la memoria RAM de la TivaC. Al momento de juntarlo con la programación principal del juego este hacía que las naves de los jugadores se buggearan y no se pudieran mover libremente. Por lo que vimos que de forma separada si funcionaba es decir si en una tiva cargamos solo la SD este si desplegaba de forma adecuada el fondo y en la otra tiva se tenía todo el juego sin la SD este funcionaba sin problema. Pero como se dijo anteriormente era al momento de juntarlo, el programa dejaba de funcionar como no era deseado por lo que decidimos dejar fuera la SD y presentarla como parte individual.

Por otro lado la complicación más importante que se tuvo fue, la que afectó el funcionamiento de la parte fina del juego en sí. Dicha parte final trataba de comparar las coordenadas (xi,yi) de cada bala individual de las dos naves y así lograr que desaparecieran los aliens que a diferencia de las balas, los aliens enemigos solo variaba su movimiento en y. Entonces el problema fue que dicho movimiento de los aliens se hizo mediante la función “millis()” la cual iba a estar aumentando la coordenada “y1_mi” de los 21 aliens cada vez que el valor presente menos el valor pasado del millis del programa fuera mayor a 500ms, pero en sí el problema fue que como el movimiento y despliegue de las balas si se realizaron de forma individual por ende su comparación tenia que ser de forma individual con el objeto de interés con el que iba a chocar, es decir los aliens. Entonces la lógica del programa del movimiento de los aliens no fue la adecuada ya que cuando la bala chocaba con cada alien esta si comparaba bien sus coordenadas (xi,yi) y también limpiaba la imagen del alien con la función “FillRect” pero despues el codigo volvía a leer la función del millis y por lo tanto pintaba de nuevo el alien borrado. Por lo tanto este mal funcionamiento se hubiera corregido si cada allien se hubiera movido con funciones individuales. Dicha solución se discutió luego de platicar con Pablo Mazariegos y fue la misma de tener a los aliens desplegándose de forma individual y utilizando CASE así se puede eliminar uno por uno sin afectar el resto como se hizo con la comparación de las balas de cada nave.

6. Partes importantes del código empleado con su explicación

6.1) Función para mover y desplazar a los 21 aliens

```
// ---------------- Mov. Aliens ----------------------
//Funcion para Movimiento Aliens automático!!!

  unsigned long currentMillis = millis();
  if(currentMillis - previousMillis > 500) {
    yl_mi = yl_mi + 1;
    previousMillis  = currentMillis;
    previousMillis2 = previousMillis+5;

  if (yl_mi == 319){
      yl_mi = 0;
  }


//-----------------------------------------------
  LCD_Bitmap(30, yl_mi, 20, 20, ALIEN_uno);
    V_line( 30-2,yl_mi, 20, 0x00);

  LCD_Bitmap(30, yl_mi+20, 20, 20, ALIEN_dos);
    V_line( 30-2,yl_mi+20, 20, 0x00);

  LCD_Bitmap(30, yl_mi+40, 20, 20, ALIEN_tres);
    V_line( 30-2,yl_mi+40, 20, 0x00);
//-----------------------------------------------
  LCD_Bitmap(70, yl_mi, 20, 20, ALIEN_uno);
    V_line( 70-2,yl_mi, 20, 0x00);

  LCD_Bitmap(70, yl_mi+20, 20, 20, ALIEN_dos);
    V_line( 70-2,yl_mi+20, 20, 0x00);

  LCD_Bitmap(70, yl_mi+40, 20, 20, ALIEN_tres);
    V_line( 70-2,yl_mi+40, 20, 0x00);
//-----------------------------------------------
  LCD_Bitmap(110, yl_mi, 20, 20, ALIEN_uno);
    V_line( 110-2,yl_mi, 20, 0x00);

  LCD_Bitmap(110, yl_mi+20, 20, 20, ALIEN_dos);
    V_line( 110-2,yl_mi+20, 20, 0x00);

  LCD_Bitmap(110, yl_mi+40, 20, 20, ALIEN_tres);
    V_line( 110-2,yl_mi+40, 20, 0x00);
//-------------- ************** ---------------------

  LCD_Bitmap(150, yl_mi, 20, 20, ALIEN_uno);
    V_line(150-2,yl_mi, 20, 0x00);
```

Como ya se discutió anteriormente esta fue la función que nos causó problemas en la parte final de nuestro juego. El valor millis es un valor de tiempo que el microcontrolador da cuando ejecuta 1 ciclo de su programa. Por lo tanto dicha función “millis()” era la encargada de estar aumentando la coordenadas “y1_mi” de los 21 aliens cada vez que el valor presente menos el valor pasado del millis del programa fuera mayor a 500ms. Se escogió ese valor de comparación ya que con ese tiempo los 21 aliens se movían a la velocidad que queríamos, la cual no iba a ser rápida.

6.2) Función “collier” para comparar el choque de cada bala con los 21 aliens

```
// NUEVAS FUNCIONES ---------------------------------------------------------------------------

uint8_t collider(uint16_t objeto1[], uint16_t objeto2[], uint8_t d1[] , uint8_t d2[]) {     //FUNCION PARA COMPAR CHOQUE BALA VS ALIEN
  uint16_t dx;
  uint16_t dy;
  uint16_t objeto_x = abs((objeto1[0] + d1[0]) - (objeto2[0] + d2[0]));
  uint16_t objeto_y = abs((objeto1[1] + d1[1]) - (objeto2[1] + d2[1]));
  dx = abs(d1[0] + d2[0]);
  dy = abs(d1[1] + d2[1]);
  if (((objeto_x - dx) < 0) && ((objeto_y - dy) < 0)) {
    return 1; //chocaron ambos ejes
  } else {
    return 0; //no choco
  }
}
```

```
objeto1[1] ={y1_mi};      // Definicion de los objetos de las coord.y(coordenada variable) de LOS 21 ALIENS
objeto2[1] ={y1_mi+20};
objeto3[1] ={y1_mi+40};

objeto4[1] ={y1_mi};
objeto5[1] ={y1_mi+20};
objeto6[1] ={y1_mi+40};

objeto7[1] ={y1_mi};
objeto8[1] ={y1_mi+20};
objeto9[1] ={y1_mi+40};

objeto10[1]={y1_mi};
objeto11[1]={y1_mi+20};
objeto12[1]={y1_mi+40};

objeto13[1]={y1_mi};
objeto14[1]={y1_mi+20};
objeto15[1]={y1_mi+40};

objeto16[1]={y1_mi};
objeto17[1]={y1_mi+20};
objeto18[1]={y1_mi+40};

objeto19[1]={y1_mi};
objeto20[1]={y1_mi+20};
objeto21[1]={y1_mi+40};

posicion_bala[0]=xi_bala;   //Definicion de los objetos de las coordenadas de las balas de la nave 1
posicion_bala[1]=y1_atack1;
```

Esta es la función más importante del código, cabe mencionar que fue basada en un código encontrado en internet. Dicha función lo que hacia era ir comprando las coordenadas (x,y) de dos objetos de interés, estos objetos en nuestro caso son las balas de las naves con los aliens. Entonces a las balas se les asigna sus respectivos objetos según sus variables para las coordenadas de su movimiento, objeto[0] → coor. xi_bala y objeto[1] → y1_atack y en el caso de los aliens se les asigna el objeto[1] a sus coordenadas en y únicamente ya que solo se movían en el eje y. Entonces esta función restaba el valor de la coordenada en x del objeto_x con el dx que es la suma de las longitudes de ancho o alto de la imagen a comparar y si esa resta era negativa entonces quería decir que ambos objetos estaban superpuestos uno con otro y entonces la función regresaba un 1, en base a ese uno sabíamos o no si se habían chocado los 2 objetos.

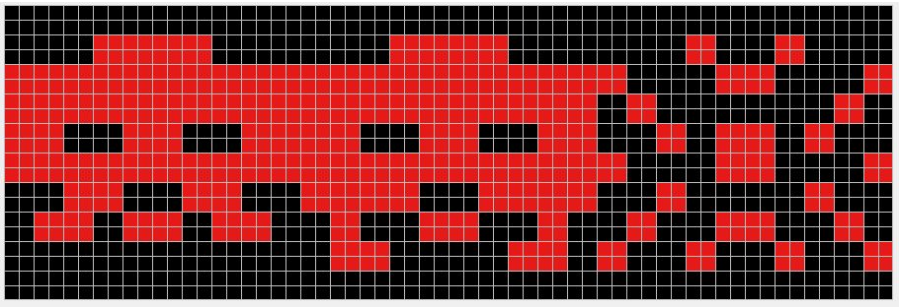
6.3) Función “naves_choque” , implementada para comparar cada choque de bala por alien individual.

```
1001 uint8_t naves_choque(uint8_t posicion){                    //FUNCION PARA CHOQUE C/NAVE INDIVIDUAL
1002   uint8_t temp;
1003   switch(posicion){
1004
1005     case 0:
1006     temp=collider(posicion_bala,objeto1,d1,d2);
1007     return temp;
1008     break;
1009
1010     case 1:
1011     temp=collider(posicion_bala,objeto2,d1,d2);
1012     return temp;
1013     break;
1014
1015     case 2:
1016     temp=collider(posicion_bala,objeto3,d1,d2);
1017     if(temp){
1018       FillRect(objeto3[0],objeto3[1],20,20,0xFF); //------------------- alien de la fila de hasta abajo
1019     }
1020     return temp;
1021     break;
1022
1023     case 3:
1024     temp=collider(posicion_bala,objeto4,d1,d2);
1025     return temp;
1026     break;
1027
1028     case 4:
1029     temp =collider(posicion_bala,objeto5,d1,d2);
1030     return temp;
1031     break;
1032
1033     case 5:
1034     temp=collider(posicion_bala,objeto6,d1,d2);
1035     if(temp){
1036       FillRect(objeto6[0],objeto6[1],20,20,0x00); //------------------- alien de la fila de hasta abajo
1037     }
1038     return temp;
1039     break;
```

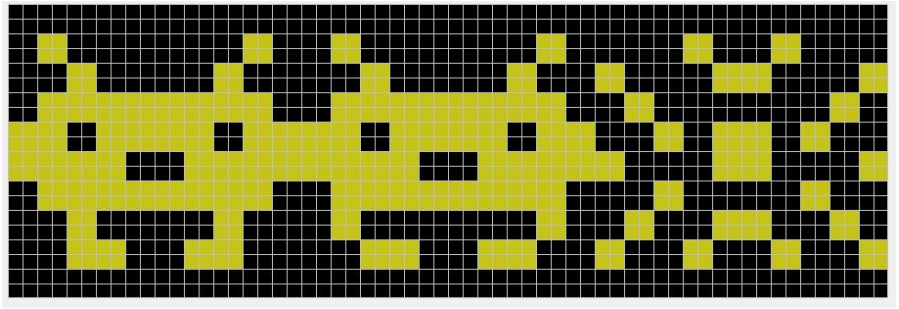
Esta función lo que hacía era ir comparando cada choque de las 21 naves si es que se daba. Es decir se definió un switch-case, con 21 casos posibles para los 21 choques posibles. a naves choque le entra el parámetro de posición, el cual es n y n2 que son los parámetros del contador de las balas dichos parámetros se establecieron con aritmética modular es decir → n++; y luego n=n%21; para los 21 casos luego se define una variable “temp” que guardaría el 1 (si hay un choque dada la función collier) o un 0 (si no hay choque) luego valor “temp” como se ve en la fila 1018 y 1036 de la imagen del código se compara si es verdadero o no y si es verdadero pues se realiza el FillRect para limpiar cada alien individualmente en sus coordenadas actuales dados los objeto”n” donde n → 0 < n < 20 (21 coordenadas en y de las 21 naves).

7. Gráficos de los aliens, naves y fondos

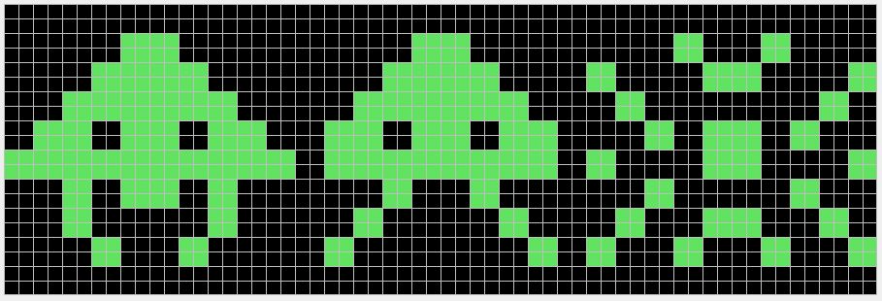
En el videojuego se muestra una serie de aliens distintos al igual que las naves de los jugadores. Para esto se utilizo el sitio web “piskelapp.com” que nos permite editar a nuestro estilo los píxeles de la imagen y así poder cambiar no solo colores sino movimientos. También se usó la aplicación “lcd-image-converter” donde se editó el tamaño de los sprites como extraer el código hexadecimal de estos para poder programarlos en la flash o SD según se trabajara.



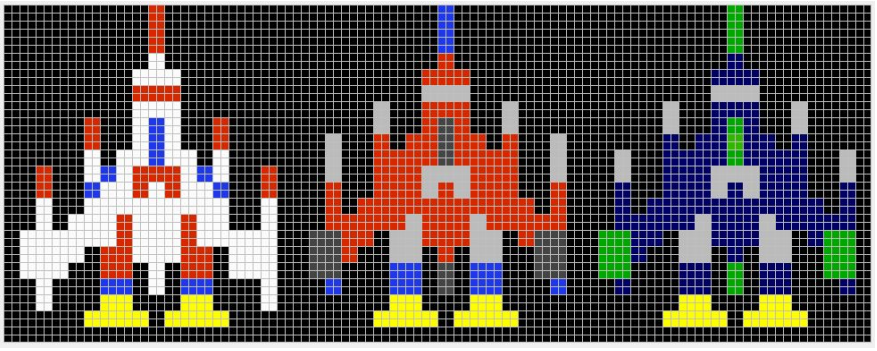
Alien 1



Alien 2

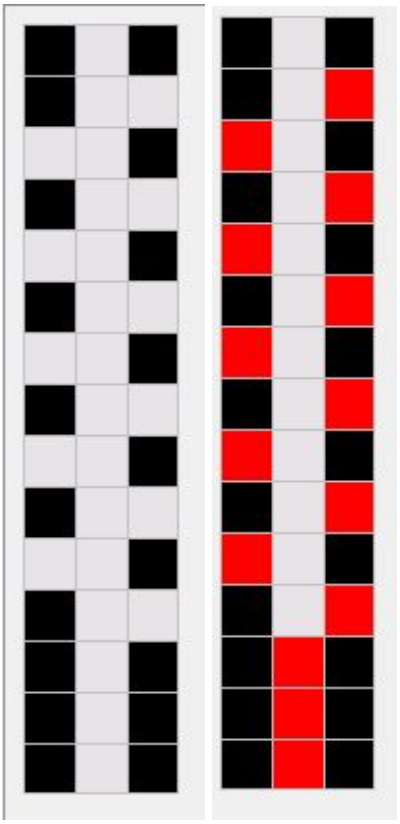


Alien 3



Naves: 1, 2 y 3

Fondo Pantalla Principal



Bala nave 1 y nave 2

- **Link del repositorio del proyecto**
  https://github.com/ChinoLou/Proyecto2_Digital2_TFT_SpaceInvaders

- **Link gráficos creados en PISKEL**
  https://www.piskelapp.com/user/6619960875417600/public

- **Link del video del proyecto funcionando:**
  https://www.youtube.com/watch?v=-EhB5ZCdVWA&t=47s