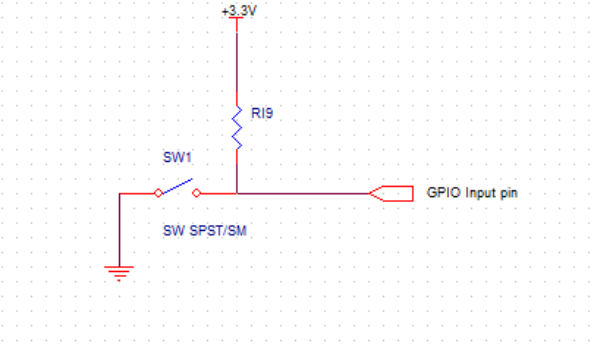
Universidad Del Valle De Guatemala
Electrónica Digital II
Integrantes:
Helder Ovalle; 18349
Rodrigo Diaz ; 18265

01/05/2021
Sección: 20

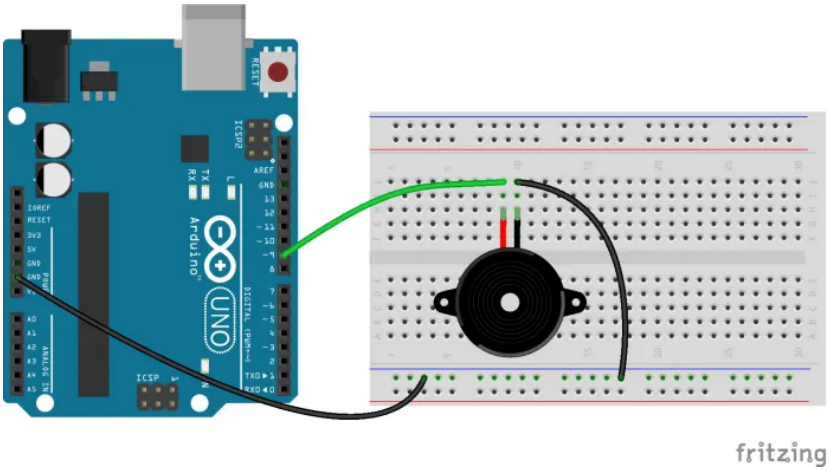
Video Juego Portable

- Trabajo escrito:

1. Circuitos utilizados:
   Los circuitos utilizados solamente fueron la configuración Pull-up para los botones y la conexión de un buzzer pasivo.



Circuito No.1 Pull-up



Circuito No.2 Buzzer Pasivo.

2. Datos:
   La planificación inicial de nuestro proyecto consistió en la idea de hacer un juego de peleas donde se tuviera una portada y pantalla de victoria para cada jugador, se pudiera escoger a distintos personajes, donde cada uno tenga un ataque de puños alternantes, patada, agachado, golpes de puño y patada para agachado, salto y golpes de puño y patada para salto. Se realizaron todas las animaciones para estos movimientos, así como un suelo que parecía un ring de pelea, la barra de vida, animación de muerte y textos para indicar el inicio de la pelea, los rounds ganados por cada jugador y la indicación de victoria al ganar 3 rounds.

Sin embargo, al realizar las pruebas nos dimos cuenta que la memoria flash se llenaba sin haber terminado las animaciones de un solo jugador. Entonces, tuvimos que desechar las animaciones de golpes de agachado, todo lo que tenía que ver con salto y las animaciones de muerte. Adicionalmente, movimos los textos de ganador y de comienzo de pelea a la sd para que los lea la RAM, donde el texto de ganador del jugador 2 se guardó completo y ya solo se sobreescribe el número 1 para el jugador uno. El suelo se convirtió en un pedazo de tierra pequeño que se repetía para cubrir todo el espacio. Los indicadores de victorias de rounds se convirtieron en cuadrados amarillos así como el indicador de quién ganó el último round. Ya no se pudo tener una pantalla de inicio ni las pantallas de victoria de cada jugador. Finalmente, no se quería seguir quitando animaciones entonces lo dejamos para que solo fueran dos personajes, uno para P1 y otro para P2.

La idea del juego cambió un poco porque ahora el agachado era la forma de cubrirse y no se podía saltar. Esto simplificó el programa original porque ya solo se tenía que revisar que no se chocara con las paredes de un lado y el contra el otro jugador del otro lado ya que nunca iban a cambiar de lado. Para no dejar el juego muy simple por los recortes de las animaciones, se decidió implementar combos en los golpes de izquierda y derecha alternantes y con la patada. Por último, se añadió que se escribirá en la SD cuando alguno de los jugadores ganará su tercer round.

3. Gráficos:
    - ❏ <u>Movimiento de ambos personajes:</u>

- Movimiento de Jin:
  Se utilizaron 3 frame's, posee una medida de 49x84.
- Movimiento de Kazuya:
  Se utilizaron 3 frame's, posee una medida de 49x84.

    - ❏ <u>Golpes de ambos personajes:</u>

- Golpe con derecha e izquierda de Jin:
  Está compuesto por 2 frame's ambos golpes, con una medida de 70x82(derecha) y 66x82(izquierda).
- Golpe con derecha e izquierda de Kazuya:
  Se compone de 2 frame's ambos golpes de nuevo, con una medida de 73x84(derecha) y 68x85(izquierda).

Para ambos jugadores se les tuvo que sumar +5 a la posición que tenían anteriormente para que el golpe se apreciara de mejor manera. Para el primer puñetazo(izquierda), que se conectara al contrario tendrá un valor de daño de 10, también luego de acertar el golpe empezaria el combo de los jugadores, que consiste en varios casos conectando el primero el segundo puñetazo conectado tendrá un valor de daño de 20. Si se falla el primer golpe y se acierta el segundo el valor de daño será de 10 ya que no hay combo, de igual manera pasaría si falla el primero y se conecta el segundo golpe.

    - ❏ Patadas de ambos personajes:

- Patada de Jin:
  Contiene dos frame's con una medida de 84x84.

- Patada de Kazuya:
  Contiene dos frame's con una medida de 72x92.

  La patada de ambos jugadores su daño es de 25, la cual también es parte del combo de ambos jugadores el "combo perfect" que tiene que cumplir acertando los dos puñetazos que se describió arriba, si no se lograra solo bajaría 25%, por último está el caso en el cual el jugador puede estar agachado y bajarle solamente 15% de daño ya que es una forma de evitar el "combo perfect".

  ❏ Agachado de ambos personajes:
- Agachado de Jin:
  2 frame's de 56x81.
- Agachado de Kazuya:
  2 frame's de 60x84.

  Para la función de agachado se tiene 2 propósitos recibir 15% de la patada y esquivar todos los puñetazos del otro jugador.

  ❏ Barra de vida de los personajes:
- Solamente era 1 frame de 234x15, la cual al ir recibiendo daño esta se iria pintando de color blanco hasta que se llenara.

4. Explicaciones:

   - Música:

   Se utilizó una TIVA solamente para la música. Se utilizó una función de beep donde se le especifica la frecuencia del sonido y que tanto se desea que dure. Se definieron las frecuencias de cada sonido como variables y se fue siguiendo la partitura para poder construir cada canción. Cuando se prende por primera vez, suena una canción de pelea de tekken y después ya solamente suena el tema de mortal kombat.

   - Jugabilidad:

Primero que todo, se incluyeron las librerías necesarias para utilizar la SD y la pantalla LCD. Se reservaron los pines correspondientes para poder conectarlas. Después se definieron los botones para los diferentes pushbuttons, variables para guardar la posición de los personajes, variables para guardar el digitalRead() de los botones, para guardar la vida, el combo, defensa, victoria, negar el movimiento al terminar una ronda y conteo de rounds. Se definieron todos los arrays, y los que estaban en la sd se debía especificar su tamaño por la forma en que se implementaron las funciones. Todos los botones se configuraron como input_pullup. En el setup se inicializa la SD y la LCD. Se utiliza un for para imprimir un montón de suelos, se imprime la barra de vida, variable de vida y la posición inicial de los jugadores. Para finalizar el setup, se cargan los textos de fight y de ganador para cada jugador. Se despliega el texto de fight por 1.5s antes de comenzar la pelea.

El loop principal primero setea la variable de no dejar movimiento si alguien gana en 0 y luego revisa todos los botones. Después se utilizan ifs para poder revisar si un jugador está

agachado y así poder esquivar los puños y reducir el daño de la patada. Luego se revisa si alguien ya ganó, para no dejar que se sigan moviendo. Se deshabilitan los botones y se setea nomov en 1 para que no se puedan mover. Hasta que P1 apache para moverse a la izquierda al mismo tiempo que P2 apache para moverse a la derecha se reinicia el juego. Al reiniciar, se repite el estado inicial definido en el setup.

Para poder mover a los jugadores, se revisa que su botón se haya apachado. Se realiza un for donde en cada vuelta se revisa si no se va a chocar contra el otro jugador o salirse de la pantalla. También se puede deshabilitar el movimiento con la variable nomov. Se utiliza la función Vline para poder borrar la imagen cuando se mueva y volver a pintar del color del fondo. Se utilizan las variables de x y Pos para cada jugador. Ambas tienen el mismo valor, pero una se utiliza como referencia para comparar y la otra para modificarla en los for.

Para poder agacharse, solamente se realiza el for de la animación porque anteriormente ya se definió la variable de defensa para simplificar el programa.

Para los puños, se utilizó una variable que cambia de 0 y 1 por cada botonazo y así alternar entre la animación de golpe de derecha y el de izquierda. Al final del ciclo for se utilizaba un frame de la animación del movimiento para que no terminara en una posición rara. La lógica de los combos ya fue explicada en la sección de gráficos.

Para las patadas, se realiza el for y de nuevo se utiliza un frame de movimiento para que no quede en posición rara. La lógica del daño también ya fue explicada en la pestaña de gráficos.

Todos los golpes, revisan si están a 75 pixeles de diferencia entre las variables de posición para saber si conecta el golpe. De lo contrario las animaciones no tienen efecto y se pierde el combo si lo hay.

Después se revisa si alguna de las variables de vida es igual o menor a 0. Esta se setea en 0 y se indica que un jugador ganó un round. La variable de win se setea en 1. Después se pinta de blanco cada barra de vida correspondiente dependiendo de la vida que tenga cada uno.

Finalmente se utiliza un switch-case para pintar los indicadores de rounds ganadas y borrarlas cuando se termina el juego.

Muchas de las funciones utilizadas ya estaban incluídas en el programa de ejemplo, pero se añadieron 3:

open_sd_bitmap: Lee un archivo de la sd, lo convierte a datos hexadecimales y los guarda en un array para luego ser abiertos con la función de bitmap.

ascii_to_hex: convierte un dato ascii a hex.

victory_sd: escribe a la SD qué jugador ganó, basado en el programa ReadWrite de los ejemplos.

5. Código debidamente comentado:
   Viene incluido en el zip, sería muy largo ponerlo aca. El proyecto de energia se llama ili9341-ejemplo. :)

- Links de github:

https://github.com/Helder1121/Labsdigitaldos/tree/main/Proyecto_3
https://github.com/RodDia2/Labs_Digital_2/tree/main/Mini_3

- Link de youtube:

https://youtu.be/96QkwIacPNE