Problema de diseño: Road Fighter

Adriana María Velásquez Medina* and Maria José Castilla +**
Universidad de los Andes, Bogotá, Colombia.
(Dated: 6 de abril de 2024)

I. ESPECIFICACIONES

1.1 Descripción del producto

El producto realizado corresponde a un prototipo de un sistema digital basado en el funcionamiento básico del juego "Road Fighter", diseñado e implementado por medio de una matriz de LEDs de 8×8 donde los LEDs representan un carro. El objetivo del juego consiste en esquivar los carros que caen por la matriz. El juego se divide en 3 niveles, cada uno con mayor dificultad que el anterior, aumentando el número de carros por fila y su velocidad. Se obtiene la victoria si se superan los 3 niveles.

El prototipo implementado cuenta con 4 señales de control: Reset, donde el juego llega al estado inicial; Start, se usa para comenzar el juego; Right y Left; se utilizan para controlar el movimiento del carro en sentido horizontal. El carro principal manejado por el usuario no puede salir de los bordes. Además, se utiliza un contador de puntaje por niveles, que corresponde al número de filas superadas por el jugador. En cada nivel se obtienen 10, 15, 20 puntos respectivamente para un total de 45 en la victoria.

El juego cuenta con 5 diferentes **restricciones** para garantizar su buen funcionamiento. Primero, las secuencias de lineas de juego no se repiten. Por otro lado, los obstáculos descienden de la matriz a una razón de $1/3$ de unidad de tiempo τ y para cada nivel esta razón aumenta $1/3$. Además, cuando se selecciona start, la primera fila comienza a descender. Cuando el carro del usuario se encuentra en una esquina este no puede moverse hacia afuera del la matriz, llegando a la esquina contraria, sino que debe permanecer en el borde. Por último, este juego se basa en una aproximación de hardware. De esta manera utiliza las **máquinas de estado** para los estados: start, reset, Nivel 1, Nivel 2, Nivel 3, State Lost (derrota) y State Win (Victoria). De la misma manera, se utilizan **registros** para controlar el juego. Cada registro es un arreglo (una lista de ceros y unos, donde los unos representan un obstáculo o un LED prendido.

El jugador pierde cuando choca contra un obstaculo/carro, en esté caso la pantalla muestra una carita triste. Por otro lado, cuando el jugador supera los 3 niveles, la victoria se marca con un corazón. Las transiciones entre los niveles se marcan por los números 1, 2 y 3 respectivamente.

1.2 Diagrama de Caja Negra

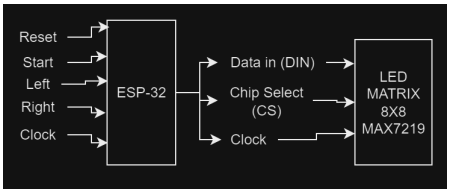

Figura 1. Diagrama de caja negra.

.

1.3 Macroalgoritmo general de solución

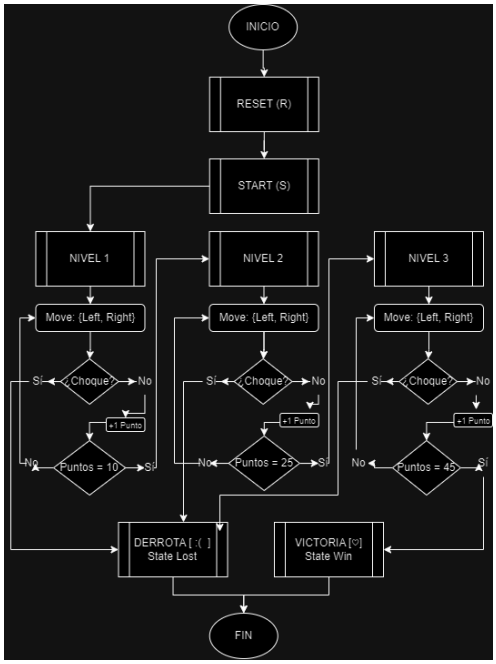

Figura 2. Macroalgoritmo.

.

* Correo institucional: am.velasquezm1@uniandes.edu.co
** Correo institucional: l.aristizabalg@uniandes.edu.co

1.4 Arquitectura del Sistema (PARTE CIRCUITAL): Diagrama de Bloques, Señales e Inter-conexiones.

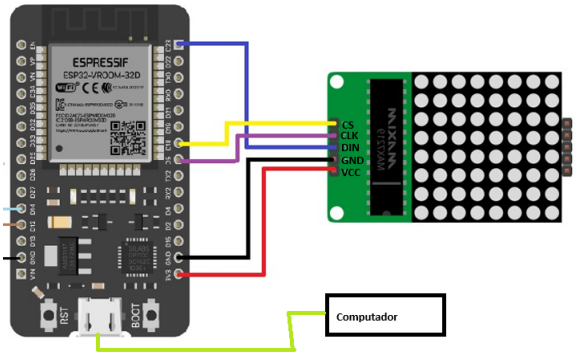

Figura 3. Diagrama Circuital. En la figura se observan 3 elementos: La matriz LED 8X8, el microcontrolador ESP-32 y un computador las señales de entrada a la matriz: CLK, Clock; CS, Chip select; DIN, Data In; VCC, fuente; GND, tierra. El computador está conectado con la ESP-32 por medio de un cable de datos. En el computador se utilizan las teclas R, S, D, A respectivamente. Tomado de [2] y modificado posteriormente.

.

1.6 Arquitectura del Sistema: (PARTE SOFTWARE): Diagrama de funciones

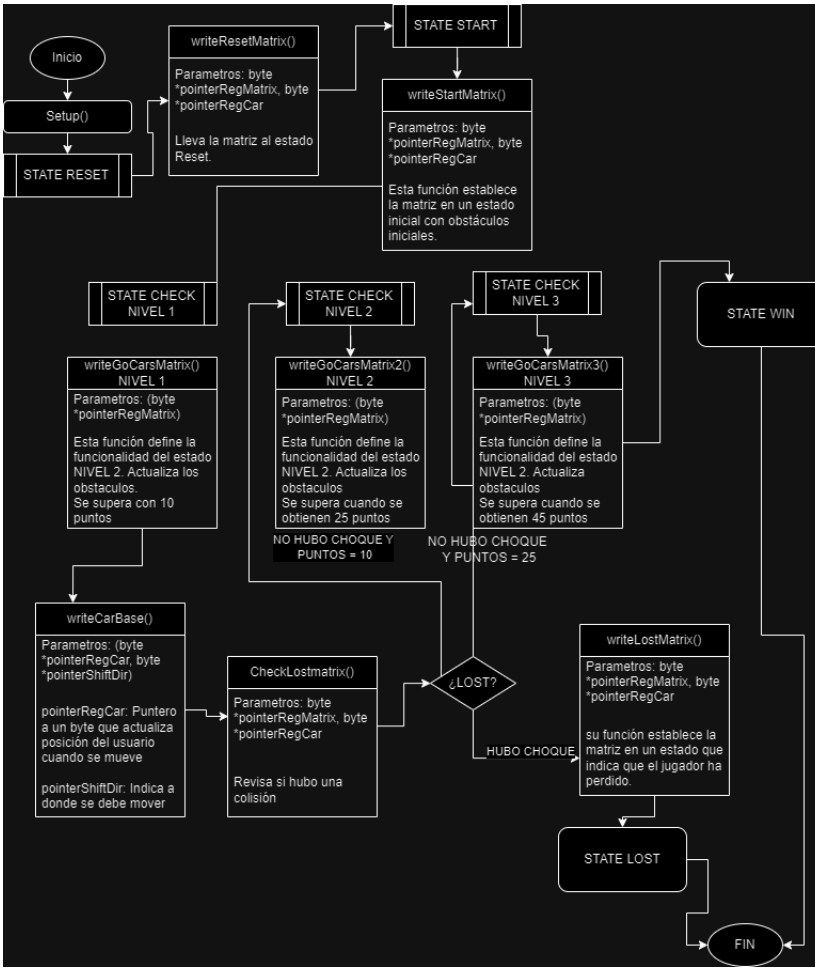

Figura 4. Diagrama de funciones.

.

II. REPORTES TÉCNICOS

2.1 Memorias de Cálculo

De acuerdo con [1] el microcontrolador ESP- 32 tiene un Procesador Tensilica Xtensa 32bits LX6 hasta 240MHz. A partir de esta información se puede utilizar la siguiente relación para encontrar el tiempo característico τ que utilizamos para definir los tiempos por los cuales se demora en bajar un píxel.

$$T = 1/f$$

$$T = 1/240MhZ$$

$$T = 1,17\dot{1}0^{-9}s$$

Esto quiere decir que en un ciclo sucede en $T = 1,17\dot{1}0^{-9}s$. Siendo este un tiempo muy pequeño, la función se ajustó para que cumpliera con las restricciones del juego. En primer lugar cada nivel sucede a $1/3$ de razon de tiempos comparado con el anterior y además tiene que ser jugable. De esta manera, y considerando la frecuencia del microcontrolador se eligió un tiempo característico en delay de $\tau = 7000$. de tal manera que el segundo nivel es $\frac{\tau}{3} = 7000/3$ y el tercer nivel es $\frac{\tau}{9} = 7000/9$.

2.2 Definición de vectores de prueba y simulaciones parciales de respaldo

- **Transición entre niveles:** Para asegurarse de que la transición entre niveles funcionara de manera correcta, se redujeron los carros a modo de prueba para que fuera más fácil pasar el nivel y así observar como se comportaba el sistema cuando se obtenía el número de puntos necesarios para pasar el nivel.
- **Velocidad de los carros:** Se intentaron diferentes tiempos para analizar que tan factible era la

velocidad más rápida y evitar que el juego fuera imposible de jugar. Para esto, se tuvo que reducir la velocidad de los carros y se definió un tiempo τ característico de tal forma que los niveles posteriores fueran n*3 veces más rápidos que la velocidad asociada a τ.

- **Dificultad por nivel: Número de carros por fila:** Para aumentar la dificultad entre niveles definimos cierta cantidad fija de carros por fila de tal forma que a medida que el nivel avanza hay más carros por fila. Esto lo hicimos eligiendo bytes cuya suma de sus dígitos fuera máximo 2, máximo 3 y máximo 4 respectivamente y eligiendo estos bytes de forma random dentro del juego. Esto quiere decir que el primer nivel solo va a tener máximo 2 carros por fila, el segundo máximo 3 y el tercero máximo 4. Para asegurarnos de que esta cantidad de carros correspondiera a la dificultad deseada, intentamos con varios carros por fila y así determinamos que máximo 2, máximo 3 y máximo 4 eran números que aseguraban que el juego fuera jugable. Por ejemplo, si se elige 7 u 8 carros por fila la dificultad sería excesivamente alta y el juego no sería jugable.

- **Restricción: Esquinas** Para asegurarnos que al llegar a una esquina el carro no pudiese avanzar más, planteamos el código y llevamos dichos carros a los bordes, eliminando los obstáculos a modo de prueba. al observar que los carros se mantenían quietos volvimos a añadir los carros al juego.

- **Estado Derrota:** Probamos distintas derrotas en los diferentes niveles para asegurarnos de que en cada nivel, si el usuario choca contra un carro entonces se llega al estado derrota y sale una carita triste.

- **Estado Victoria:** Realizamos casos de victoria para asegurarnos de que el corazón saliera siempre que se superaran la cantidad de puntos sin choque como lo indica el macroalgoritmo.

2.3 Indicadores de utilización de recursos y rendimiento de los dispositivos utilizados

Tiempo de ejecución 2s ,Uso de memoria RAM actual: 256 MB, Velocidad de transferencia de datos: 1.5 Mbps, Promedio de carga de la CPU: 0.35

III. IMPLEMENTACIÓN

3.1 Descripción en lenguajes SW

3.2 Funcionalidad modular

La funcionalidad modular se define por medio de cada uno de los estados:

- STATE Reset: writeResetMatrix() ->Lleva la matriz al estado Reset.
- STATE Start: writeStartMatrix() ->Esta función establece la matriz en un estado inicial con obstáculos iniciales.
- STATE NIVEL 1 writeGoCarsMatrix() ->Esta función define la funcionalidad del estado NIVEL 2. Actualiza los obstáculos. Se supera con 10 puntos
- STATE NIVEL 2 () writeGoCarsMatrix2() ->Esta función define la funcionalidad del estado NIVEL 2. Actualiza los obstáculos. Se supera cuando se obtienen 25 puntos
- STATENIVEL 3 () writeGoCarsMatrix3(): ->Esta función define la funcionalidad del estado NIVEL 2. Actualiza obstaculos Se supera cuando se obtienen 45 puntos
- STATE WIN:
- STATE LOST (State Lost): writeGoCarsMatrix3() y writeLostMatrix(): su función establece la matriz en un estado que indica que el jugador ha perdido.
- State win: Imprime un corazón cuando se superan 45 puntos.

3.2.1 Links a Videos y fotos de demostración de módulos del sistema.

El link se encuentra en el repositorio.

3.3 Funcionalidad global del sistema

En cuanto a la funcionalidad global del sistema, el diseño implementado es capaz de integrar todos los componentes en directamente en el prototipo del juego, cumpliendo con toda las especificaciones y restricciones para evidenciar una funcionalidad global de la solución. En otras palabras, fue posible interconectar los procesos dentro de cada modulo descrito en la sección anterior para lograr un juego que logra integrar los estados de:

START, RESET, NIVEL 1, NIVEL 2, NIVEL 3 STATE LOST Y STATE WIN, obteniendo un juego funcional.

3.3.1 Links a Videos y fotos de demostración del prototipo final.

IV. REFLEXIÓN

4.1 Resultados y lecciones aprendidas

En el proyecto Street Fighter, fue posible implementar y diseñar un juego por medio de un microcontrolador ESP-32, una matriz led y un computador, con un paradigma de hardware haciendo uso de registros y máquina de estados. El resultado principal es el prototipo del juego, cuyo fin es superar 3 niveles diferentes, que varían en número de puntos, velocidad de los carros y número de carros por fila. El juego se gana si se obtienen 45 puntos consecutivos. Hay diferentes estados dentro del juego: Reset, Start, Nivel 1, Nivel 2, Nivel 3 y derrota y victoria, por medio de los cuales se desarrolla el juego globalmente.

Las principales lecciones aprendidas son el uso de registros y las máquinas de estado. Para este proyecto se utilizaron registros que eran listas de dígitos o bytes en vez de utilizar una matriz directamente desde el software. Además se utilizaron estados para asegurar la interconexión de procesos para cada modulo de juego y así llegar a un producto con funcionalidad global. Por otro lado, aprendimos a realizar diagramas de caja negra y macroalgoritmos, que simplifican los procesos y resumen los estados de nuestro prototipo.

V. MATERIALES

Dispositivos Hardware

- ESP32 con cable de datos (ESP-WROOM-32)
- Matriz Led 8x8 (MAX7219)
- Computador para utilizar sus teclas (A y D) como inputs Left y Right respectivamente
- Jumpers macho-macho

Herramientas Software

- PaltformIO
- IDE Visual Studio Code
- lenguaje de programación C++

[1] Sigma Electronica. Esp-32. `https://www.sigmaelectronica.net/producto/esp-32/`, 2024.

[2] Fredy Segura-Quijano. Problema de diseño: Road fighter. 2024.