Sistema interactivo tipo Dodger con ESP32 y pantalla OLED

Luisa Castaño Sepúlveda, Anny Juliana Acosta, Juana Valentina Monsalve
Electrónica digital II

Lunes 20 de Octubre 2025

Índice

1. Identificación del proyecto

- **Nombre del proyecto:** Sistema interactivo tipo Dodger con ESP32 y pantalla OLED.
- **Integrantes del equipo:** Luisa Castaño Sepúlveda, Anny Juliana Acosta, Juana Valentina Monsalve

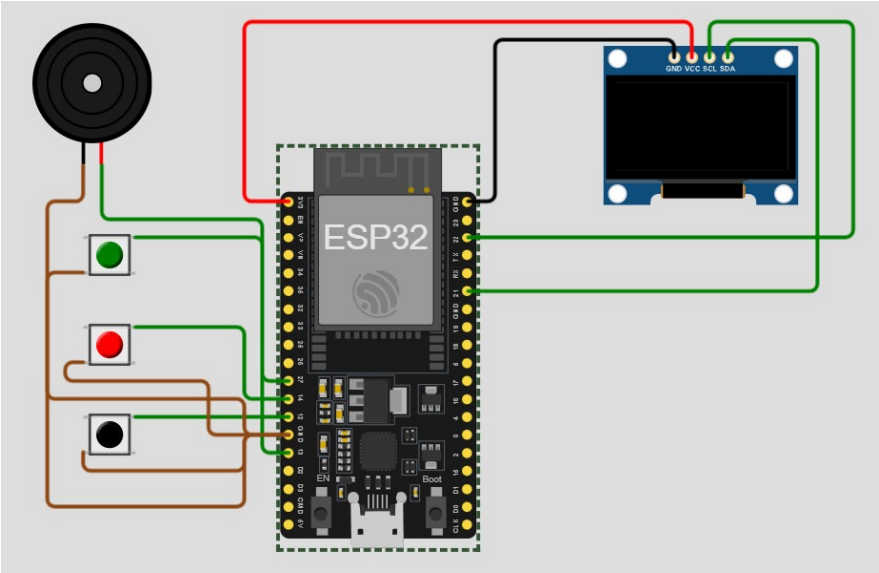
2. Resumen

En esta práctica se diseñó e implementó un videojuego tipo *Dodger* programado en MicroPython sobre el microcontrolador ESP32. El sistema permite controlar un jugador mediante botones físicos para esquivar obstáculos que se desplazan en la pantalla OLED SSD1306. Además, se incluyen efectos sonoros a través de un buzzer, diferentes modos de dificultad y un sistema de puntuación en tiempo real. El proyecto permitió afianzar conceptos de manejo de interrupciones, temporizadores, entradas digitales, generación de sonido mediante PWM y control gráfico por protocolo I2C.

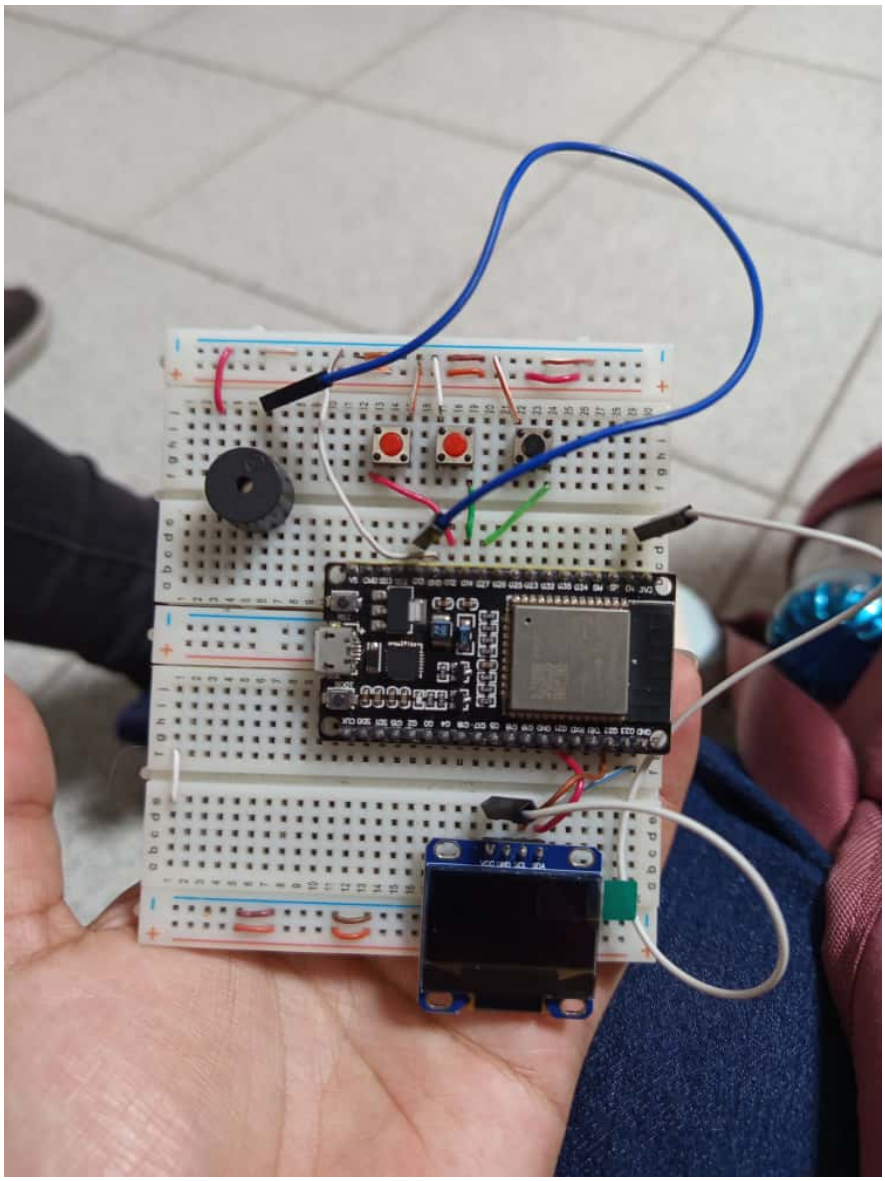
3. Descripción del hardware

- **ESP32:** microcontrolador principal encargado de la ejecución del juego y la lógica de control.
- **Pantalla OLED SSD1306:** interfaz visual donde se dibuja el jugador, los obstáculos, el puntaje y los menús.
- **Botones:**
  - **BTN_UP (GPIO 14):** mueve al jugador hacia arriba.
  - **BTN_DOWN (GPIO 27):** mueve al jugador hacia abajo.
  - **BTN_START (GPIO 12):** inicia el juego o pausa la partida.
- **Buzzer (GPIO 13):** genera efectos de sonido para los movimientos, colisiones y eventos del juego.
- **Conexión I2C:** líneas SCL (GPIO 22) y SDA (GPIO 21) para la comunicación con la pantalla OLED.
- **Fuente de alimentación:** 5V por conexión USB del computador.

Diagrama de conexión:



Circuito:



4. Descripción del software

- **Lenguaje usado:** MicroPython
- **Librerías:** `machine`, `time`, `random`, `ssd1306`
- **IDE:** Thonny

Flujo general del programa:

1. Inicialización de pines, pantalla OLED y buzzer.
2. Creación de estructuras para manejar el estado del juego y las variables globales (puntaje, obstáculos, modo, etc.).

3. Implementación de las funciones de sonido y animaciones mediante temporizadores (`Timer`).
4. Lógica del menú principal para seleccionar el modo de dificultad.
5. Ejecución del bucle principal del juego, donde se detectan los movimientos del jugador y se actualizan los obstáculos.
6. Detección de colisiones y gestión de los estados: `MENU`, `JUEGO`, `PAUSA`, `GAME_OVER`.

5. Estructura del código

- **Archivo principal:** `Seguimiento4f.py`
- **Variables globales:** controlan el estado del juego, posición del jugador, lista de obstáculos, puntaje y temporizadores.
- **Funciones principales:**
  - `dibujar_pantalla()`: actualiza el contenido del OLED con la información del juego.
  - `actualizar_juego()`: mueve los obstáculos, aumenta la dificultad y actualiza el puntaje.
  - `colisiona()`: detecta si el jugador ha chocado con algún obstáculo.
  - `play_beep()`: reproduce efectos de sonido mediante PWM.
  - `reset_game_state()`: reinicia los valores del juego al iniciar una nueva partida.
- **Bucle principal:** controla la transición entre los estados del juego y la interacción con los botones físicos.

6. Explicación de funciones principales

- `dibujar_pantalla()`: limpia y actualiza el contenido del OLED, mostrando jugador, obstáculos y HUD (puntaje, tiempo, modo).
- `actualizar_juego()`: genera nuevos obstáculos cada cierto tiempo, aumenta la velocidad con el progreso y controla la dificultad.
- `colisiona()`: compara las coordenadas del jugador con las de cada obstáculo para detectar colisiones.
- `play_beep()`: utiliza un temporizador para generar secuencias de tonos y crear efectos de sonido diferenciados.
- `reset_game_state()`: reinicia variables como puntaje, tiempo y lista de obstáculos al iniciar una nueva partida.

7. Comunicación

El sistema utiliza la comunicación **I2C** entre el ESP32 y la pantalla OLED SSD1306 para el envío de datos gráficos. Además, los botones y el buzzer interactúan directamente con los pines digitales del microcontrolador, generando una experiencia totalmente embebida sin necesidad de conexión a PC durante la ejecución.

8. Interfaz de usuario

- Menú principal que permite seleccionar el modo de juego (*Clásico*, *Contra-tiempo* o *Hardcore*).
- Pantalla OLED con representación gráfica del jugador y obstáculos.
- Indicadores en pantalla de puntaje, tiempo y estado del juego.
- Sonidos asociados a acciones específicas (inicio, movimiento, pausa, fin de juego).

9. Procedimiento de prueba

1. Conectar el ESP32 al computador mediante cable USB.
2. Abrir Thonny y cargar el archivo `Seguimiento4f.py`.
3. Ejecutar el código y verificar que el menú principal se muestre en pantalla.
4. Seleccionar el modo de juego con los botones y presionar `START`.
5. Observar el movimiento del jugador y la aparición de obstáculos.
6. Comprobar los efectos sonoros asociados a las acciones.
7. Verificar el funcionamiento correcto de pausa y reinicio de partida.

10. Manejo de errores y seguridad

- Se controla el rebote de botones mediante retardos por software.
- Los temporizadores se detienen y reinician correctamente para evitar bloqueos.
- Se manejan diferentes estados del juego para prevenir errores lógicos o lecturas simultáneas.
- Se recomienda usar una fuente de alimentación estable y evitar desconectar la pantalla durante la ejecución.