**Informe: Desarrollo de un Juego en Arduino con ESP32**

**Realizado por: David Alejandro Ortega Flórez**

**PARTE 1**

**Introducción:**

Este informe detalla el proceso de desarrollo de un juego interactivo utilizando un microcontrolador ESP32, una pantalla LCD, un buzzer, un potenciómetro y una protoboard. El proyecto surge como una oportunidad para aplicar conocimientos avanzados de programación y electrónica, combinando técnicas de programación orientada a objetos con el uso de periféricos para crear una experiencia de juego envolvente y educativa.

**Descripción General del Código:**

El código del juego está estructurado en archivos modulares que contienen clases y funciones especializadas para manejar la lógica del juego, la interacción con el usuario y la comunicación con los dispositivos periféricos. Se hace uso extensivo de conceptos de programación avanzada, como la herencia, el encapsulamiento y el polimorfismo, para facilitar la implementación y el mantenimiento del código.

**Componentes del Juego:**

Dinosaurio: Representado por la clase Dinosaurio, este personaje principal del juego tiene la capacidad de caminar y saltar para esquivar obstáculos. Se utiliza una máquina de estados para gestionar las diferentes acciones del dinosaurio, permitiendo una implementación modular y escalable de su comportamiento.

Cactus y Ave: Los obstáculos y elementos de interacción secundarios del juego están representados por las clases Cactus y Ave, respectivamente. Se emplean técnicas de animación y temporización para simular el movimiento de estos elementos a lo largo de la pantalla, añadiendo complejidad y desafío al juego.

Interacción con el Usuario: La interacción del usuario con el juego se realiza a través de un botón conectado al ESP32. Se utiliza un sistema de interrupciones para detectar el estado del botón y activar las acciones correspondientes del jugador, como saltar para esquivar obstáculos.

Pantalla LCD y Buzzer: La pantalla LCD y el buzzer se utilizan para proporcionar retroalimentación visual y auditiva al jugador durante el juego. Se implementa un sistema de renderizado en tiempo real para actualizar la pantalla con la posición y estado de los personajes, mientras que el buzzer se utiliza para reproducir efectos de sonido inmersivos, como el sonido del salto del dinosaurio o la señalización de fin de juego.

**Funciones Clave:**

void Dinosaurio::caminar() y void Dinosaurio::saltar(): Estas funciones controlan el movimiento del dinosaurio, utilizando técnicas de animación y temporización para lograr una representación realista de su movimiento en la pantalla.

void Cactus::mover() y void Ave::mover(): Estas funciones gestionan el movimiento de los obstáculos y elementos secundarios del juego, aplicando principios de física y geometría para calcular su trayectoria y posición en cada iteración del juego.

void mostrarEstadoFinal(): Esta función muestra en la pantalla LCD el puntaje final del jugador y un mensaje indicando el fin del juego. Se utiliza formateo de texto avanzado para presentar la información de manera clara y concisa.

**Configuración de Pines:**

Pantalla LCD: Se asignan los pines 22, 23, 5, 18, 19 y 21 para la conexión de la pantalla LCD al ESP32, utilizando una combinación de pines digitales y de control para garantizar una comunicación fiable y de baja latencia.

**Botón y Buzzer:** Los pines 4 y 25 se asignan para la conexión del botón y el buzzer al ESP32, respectivamente. Se implementan medidas de protección y filtrado de señales para garantizar un funcionamiento robusto y seguro en todo momento.

**PARTE 2**

**Explicación código principal**

Inclusión de bibliotecas y archivos de encabezado: Se incluyen las bibliotecas necesarias para el funcionamiento del juego, como LiquidCrystal.h para controlar la pantalla LCD, y se incluyen los archivos de encabezado Dinosaurio.h, Cactus.h y Ave.h, que contienen las clases que representan los personajes y elementos del juego.

Inicialización de componentes: Se crea una instancia del objeto LiquidCrystal llamado "lcd" con los pines de conexión especificados para la pantalla LCD. También se crean instancias de los objetos Dinosaurio, Cactus y Ave, que representan al personaje principal, los cactus y las aves respectivamente. Además, se define el pin para el buzzer.

Diseño de gráficos: Se definen matrices de bytes que representan los diseños de los personajes y elementos del juego, como el dinosaurio en reposo, caminando, saltando, así como los cactus y las aves.

Definición de estructuras y variables: Se define una estructura llamada "Button" para manejar la entrada del botón, con un pin y un indicador de si está presionado. Se declara una variable global para la velocidad del juego, un indicador de si se perdió el juego y una variable para almacenar la puntuación.

Función de interrupción: Se define la función "isr()" que se ejecuta cuando se detecta una interrupción en el pin del botón. Esta función establece la bandera "pressed" en true para indicar que se presionó el botón.

Configuración inicial en la función "setup()": Se inicializan los componentes del juego, incluyendo la pantalla LCD, el puerto serial, el botón y el buzzer. Se crean los caracteres personalizados en la pantalla LCD utilizando los diseños previamente definidos.

Bucle principal en la función "loop()": Este bucle se ejecuta continuamente durante el funcionamiento del juego. Comienza generando una semilla aleatoria para la función de generación de números aleatorios. Luego, verifica si el juego está en curso (es decir, si el jugador no ha perdido).

Control del jugador: Si se detecta que se ha presionado el botón, el dinosaurio realiza un salto y se reproduce un sonido a través del buzzer. De lo contrario, el dinosaurio continúa caminando.

Movimiento y dibujo de elementos: Se mueven los cactus y las aves hacia la izquierda en la pantalla, y se actualiza la posición y el estado del dinosaurio. Se dibujan los personajes y elementos en la pantalla LCD.

Actualización de la puntuación y ajuste de la velocidad: Se comprueba si el dinosaurio ha colisionado con algún elemento del juego. En caso afirmativo, se registra la colisión y se actualiza la puntuación. La velocidad del juego aumenta a medida que la puntuación aumenta.

Manejo del final del juego: Si se detecta que el jugador ha perdido, se muestra la puntuación final en la pantalla LCD y se espera a que se presione el botón para reiniciar el juego.

Funciones auxiliares: Se definen las funciones "mostrarEstadoFinal()" y "reiniciarJuego()" para mostrar el estado final del juego y reiniciar el juego respectivamente.

**DINOSAURIO.CPP**

Constructor: Inicializa el estado y la posición del dinosaurio.

Función caminar(): Cambia el estado del dinosaurio para simular su movimiento al caminar.

Función saltar(): Cambia el estado del dinosaurio para simular su acción de saltar.

Funciones getEstado(), getYPosicion() y getXPosicion(): Devuelven el estado actual, la posición vertical y la posición horizontal del dinosaurio respectivamente.

**CACTUS.CPP**

Constructor: Inicializa la posición inicial del cactus.

Función mover(): Decrementa la posición horizontal del cactus, simulando su movimiento hacia la izquierda en la pantalla.

Función getPosicion(): Devuelve la posición horizontal actual del cactus.

Función setPosicion(int pos): Establece la posición horizontal del cactus en la posición especificada.

**AVE.CPP**

Constructor: Inicializa la posición inicial y el estado del ave.

Función mover(): Decrementa la posición horizontal del ave, simula su movimiento hacia la izquierda en la pantalla y cambia su estado para simular la animación del vuelo.

Funciones getEstado() y getPosicion(): Devuelven el estado actual y la posición horizontal del ave, respectivamente.

Función setPosicion(int pos): Establece la posición horizontal del ave en la posición especificada.

**PARTE 3**

Se sabe que este proyecto fue obtenido gracias a Wokwi, pero fue modificado para obtener este proyecto, a continuación se habla de los cambios.

**CAMBIOS**

El Ave en el juego aparece de manera aleatoria utilizando un algoritmo que genera números aleatorios para determinar su posición inicial en la pantalla. Esto se logra mediante la función random() que selecciona un número aleatorio dentro de un rango específico. La posición inicial del Ave se elige dentro de los límites de la pantalla, asegurando que no aparezca encima del cactus.

Para evitar que el Ave aparezca encima del cactus, se calcula la diferencia entre la posición del cactus y la posición del Ave. Si la diferencia es menor o igual a un cierto umbral (en este caso, 2), se asigna una nueva posición al Ave que garantiza que no colisione con el cactus.

Los puntos se cuentan cada vez que el dinosaurio supera un obstáculo. Esto se detecta comparando las posiciones horizontales del dinosaurio y los obstáculos. Si la posición horizontal del dinosaurio coincide con la posición del cactus o del Ave y la posición vertical del dinosaurio está en la misma línea que el obstáculo, se incrementa la puntuación.

El sonido del buzzer se reproduce cuando el jugador presiona el botón para que el dinosaurio salte. Esto se realiza utilizando la función tone() que genera un tono audible en el pin del buzzer durante un tiempo específico. El tono y la duración están predeterminados en el código, lo que proporciona retroalimentación auditiva al jugador cada vez que realiza una acción.

**CONCLUSIONES**

Este informe destaca el desarrollo de un juego interactivo en Arduino con ESP32, combinando tecnología y creatividad. Se aplicaron conocimientos avanzados en programación y electrónica, organizando el código de manera modular y haciendo uso de conceptos como la herencia y el polimorfismo. La interacción con el usuario se logró mediante un botón y se implementó retroalimentación visual y auditiva. Se realizaron optimizaciones continuas, como ajustar la velocidad del juego según el puntaje del jugador. En conclusión, el proyecto demuestra cómo la tecnología puede ofrecer soluciones divertidas y educativas, promoviendo la innovación y el aprendizaje.

**REFERENCIAS**

**Código original wokwi**

- https://wokwi.com/projects/397802059690748929

**LINK Video funcionamiento ( No se incluyo el Buzzer por falta del componente)**

- Google Drive (https://drive.google.com/drive/folders/1PRTQinXAB09NqtWHdVLROPZNOUEkylvE?usp=sharing)

**LINK Simulación en Wokwi**

- Dino - Wokwi ESP32, STM32, Arduino Simulator (https://wokwi.com/projects/399709061184146433)

**LINK Repositorio en GitHub**

- NeoEzzio/Dino (github.com) (https://github.com/NeoEzzio/Dino)

**FIN DEL INFORME**