TAREA DE SISTEMAS EMBEBIDOS - TEÓRICO

TAREA #3

Informe Técnico: Snake con Comunicación entre Microcontroladores

Integrantes:

Fernandez Sanchez Kevin David

Garcia Carbo Kevin Joel

Guale Salazar Jesse Argenys

Moreno Maldonado Jose Augusto

Paralelo: 2

Docente: Solis Mesa Ronald David

Fecha: 18/06/2025

1. Objetivo General

- Desarrollar un sistema de juego interactivo de 3 niveles de dificultad que combine salidas visuales mediante una matriz LED 8x8 controlada por el microcontrolador ATmega328P, y salidas auditivas mediante un sistema de reproducción de melodías con el microcontrolador PIC16F887, y entradas mediante pulsadores, empleando comunicación entre ambos dispositivos y simulando el funcionamiento completo en Proteus, con el propósito de reforzar habilidades en programación y control de sistemas embebidos. El sistema deberá ser programado en C para ambos microcontroladores.

2. Objetivos Específicos

- Diseñar e implementar un sistema de control de una matriz LED 8x8 con el microcontrolador ATmega328P para desplegar caracteres, símbolos y animaciones del juego.
- Programar un sistema de reproducción de melodías utilizando el PIC16F887, capaz de interpretar comandos recibidos desde el ATmega328P.
- Integrar botones físicos como entradas para interactuar con el juego (selección, acción, reinicio, etc.).
- Implementar 3 niveles de dificultad ya sea modificando la velocidad, lógica o complejidad del juego.
- Establecer un canal de comunicación entre los microcontroladores que permita la sincronización entre efectos visuales y sonoros.
- Simular el juego de manera completa en el entorno Proteus, integrando los dos microcontroladores, la matriz LED y el sistema de audio.
- Documentar todo el proceso de diseño, desarrollo y simulación en un informe técnico, incluyendo repositorio con evidencias y código.

3. Descripción Técnica del Juego

El juego Snake se implementa utilizando dos microcontroladores que trabajan de manera conjunta. El ATmega328P se encarga de la lógica principal del juego, el manejo de una matriz LED 8x8 mediante el controlador MAX7219 y la lectura de los botones físicos. Por otro lado, el PIC16F887 se dedica a generar efectos de sonido a través de un buzzer. La comunicación entre ambos es unidireccional, donde el ATmega actúa como transmisor de eventos y el PIC como receptor.

En el caso del ATmega328P, la representación visual del juego se realiza con ayuda del MAX7219, un controlador especializado en matrices LED, que permite controlar eficientemente las 64 posiciones de la matriz (8x8). La comunicación entre el ATmega y el MAX7219 se establece mediante una interfaz SPI por software, utilizando los pines PB3 (datos), PB5 (reloj) y PB2 (carga). Esta interfaz permite enviar información a cada fila de la matriz para mostrar la posición de la serpiente, la comida y mensajes animados como “GAME OVER” o “DIFICULTAD”, que se desplazan dinámicamente.

El ATmega también interpreta las acciones del usuario a través de cuatro botones conectados a los pines PB0, PB1, PD6 y PD7, que permiten cambiar la dirección de la serpiente. El juego cuenta con

funciones específicas para distintos niveles de dificultad (funcionA, funcionB y funcionC), las cuales varían en la velocidad y la cantidad de comida presente. La detección de colisiones, el crecimiento de la serpiente y la aparición de nuevos objetivos se gestiona internamente en el ATmega, mientras que la matriz LED muestra en todo momento el estado del juego.

Cuando se produce un evento importante, como comer un objetivo o perder, el ATmega activa brevemente una señal digital: PD5 en caso de haber comido y PD4 si hay colisión. Estas señales se envían al PIC16F887, que reacciona produciendo sonidos específicos.

En el PIC16F887, un buzzer conectado al pin RD0 (configurado como salida) se utiliza para generar sonidos. Las señales recibidas desde el ATmega llegan a los pines RD1 y RD2 (configurados como entradas). Si el PIC detecta un pulso en RD1, ejecuta la función waka_waka_sound(), la cual produce dos tonos breves mediante la generación precisa de frecuencias. Si el pulso se recibe en RD2, se activa la función game_over_sound(), que genera una secuencia de tonos graves más largos. Además, cuando el sistema se enciende, el PIC reproduce automáticamente una melodía de bienvenida inspirada en Pac-Man (startup_melody()), lo que refuerza el estilo retro del juego.

El nuevo diseño del código en el PIC permite generar melodías a partir de frecuencias musicales definidas (como DO, RE, MI), utilizando retardos a nivel de microsegundos para lograr sonidos audibles. Estas frecuencias se logran manipulando directamente el estado del pin de salida en ciclos temporales controlados.

4. Simulación en Proteus

Componentes utilizados:

- ATmega328P (control de matriz y lógica del juego).
- PIC16F887 (generación de sonidos).
- MAX7219 (driver de la matriz LED)
- Matriz LED 8x8 (controlada directamente sin MAX7219).
- 4 pulsadores: para dirección (arriba, abajo, izquierda, derecha).
- Buzzer conectado al PIC.
- Comunicación entre ATmega y PIC mediante 4 bits de datos + línea de validación.

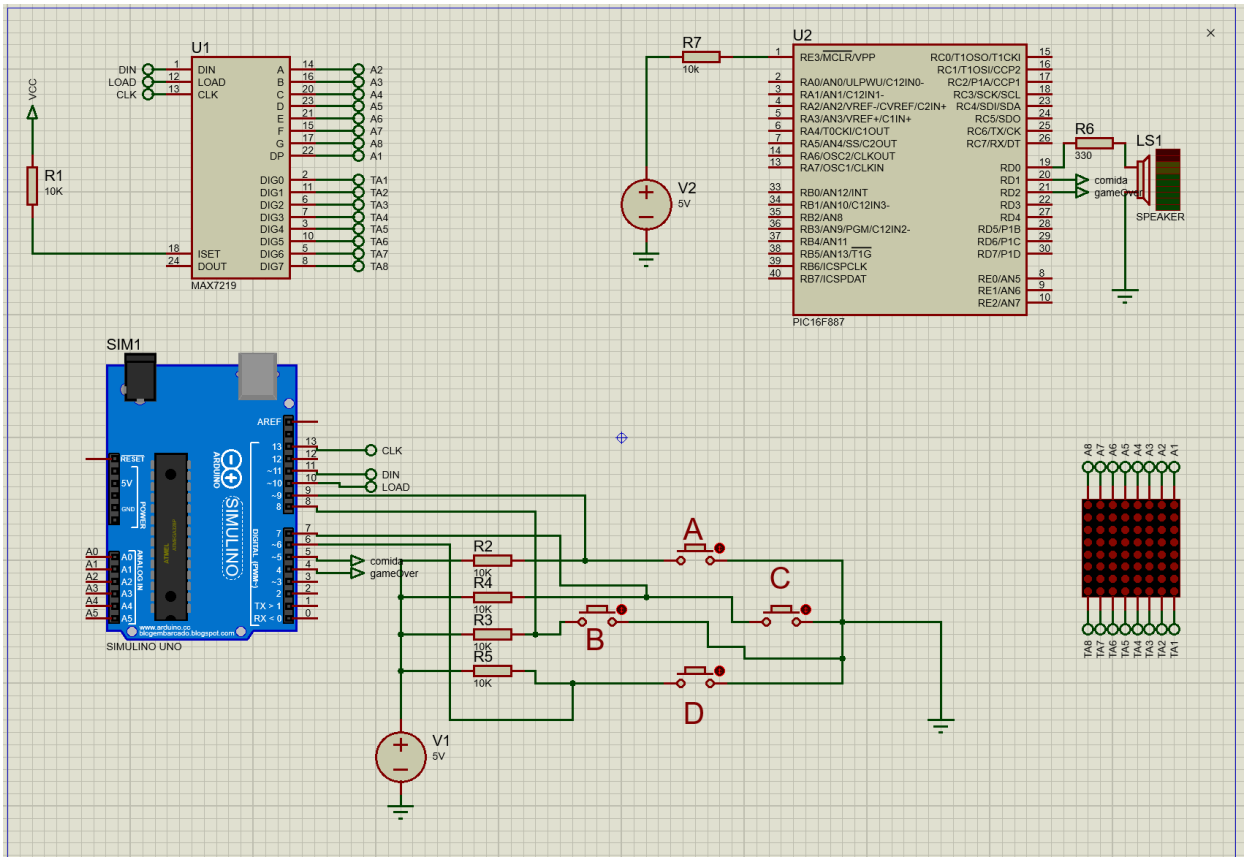

Imagen 6. Ejemplificación de la persistencia visual.

5. Explicación del código para cada microcontrolador ATMEGA328P:

El código implementa el juego Snake en una matriz LED 8x8 controlada mediante el chip MAX7219. El microcontrolador ATmega328P es responsable de la lógica del juego, la visualización en la matriz y la detección de entradas desde cuatro botones físicos conectados a los pines PB0, PB1, PD6 y PD7.

El MAX7219 se comunica con el ATmega por SPI software (pines PB2, PB3 y PB5). Se incluyen funciones para mostrar texto desplazado (scroll) y renderizar mensajes como “DIFICULTAD” y “GAME OVER”. La serpiente se representa como un conjunto de coordenadas que se actualan según la dirección indicada por los botones.

Hay tres funciones que representan los modos de dificultad:

- funcionA() – Modo fácil, velocidad baja y una comida.
- funcionB() – Modo medio, velocidad intermedia y una comida.
- funcionC() – Modo difícil, velocidad alta y dos comidas simultáneas.

Al iniciar el juego, se muestra un mensaje de bienvenida, y según el botón presionado, se ejecuta uno de los modos. Cuando la serpiente colisiona consigo misma, se muestra “GAME OVER” y se envía una señal digital al PIC (por PD4 o PD5) para reproducir un sonido.

PIC16F887:

Este código permite al microcontrolador PIC16F887 generar sonidos mediante un buzzer conectado al pin RD0, en respuesta a señales enviadas por el ATmega328P a través de los pines RD1 (evento de comida) y RD2 (evento de game over).

Se definen retardos precisos y constantes de tiempo asociadas a notas musicales (DO, RE, MI, etc.), que se usan para producir frecuencias audibles. La función principal playNote() alterna el estado del buzzer para generar ondas cuadradas, lo que crea los tonos deseados.

El programa incluye funciones simples como beep_short() y beep_long(), además de melodías completas (startup_melody(), retro_startup(), power_up_fanfare()), que se reproducen al encender el sistema. También hay sonidos específicos para eventos del juego: waka_waka_sound() al comer y game_over_sound() al perder.

En resumen, este módulo convierte al PIC en un generador de efectos sonoros musicales, sincronizado con los eventos del juego Snake controlado por el ATmega328P.

6. Comunicación entre Microcontroladores

La comunicación entre el ATmega328P y el PIC16F887 se implementa mediante un esquema de señalización digital simple, en el que el ATmega genera eventos y el PIC los interpreta. El sistema no necesita un protocolo de comunicación formal, ya que no se transmite información estructurada ni bytes de datos: solamente se notifican eventos binarios (comió o perdió).

Esta comunicación se logra mediante dos líneas digitales: PD5 del ATmega se conecta a RD1 del PIC y señala que el jugador ha comido; PD4 se conecta a RD2 y representa el fin del juego. El ATmega simplemente activa estas líneas durante un breve período (mediante `PORTD |= (1 << PIN_COMIDA)` o `PORTD |= (1 << PIN_GAMEOVER)`), y el PIC responde reproduciendo la melodía correspondiente. El PIC incluye protección contra rebotes al esperar que la línea se libere antes de permitir una nueva reproducción.

Este modelo de comunicación es eficiente, directo y fácil de escalar, ya que podría extenderse a nuevos eventos (como victoria, cambio de nivel, etc.) simplemente agregando nuevas líneas o multiplexando señales. Además, la separación funcional —visual para el ATmega con ayuda del MAX7219, y sonora para el PIC— permite que cada microcontrolador se enfoque en su tarea específica, lo cual reduce la carga computacional y facilita el mantenimiento del código.

En conjunto, el uso del MAX7219 para manejar la matriz LED, junto con la generación musical del PIC en base a frecuencias precisas, y una comunicación simple pero efectiva, da como resultado un sistema embebido bien estructurado, modular y funcional para el desarrollo del juego Snake.

7. Repositorio de GitHub

Enlace del repositorio: https://github.com/KevinFernandez21/MicrocontrollerComunitation/tree/main

8. Conclusiones

- La implementación del juego *Snake* permitió integrar diversos conceptos clave de los sistemas embebidos, como el manejo de periféricos visuales (matriz LED 8x8), generación de sonidos, comunicación entre microcontroladores y uso de entradas físicas mediante botones. El uso del controlador MAX7219 resultó fundamental para simplificar el manejo de la matriz, permitiendo realizar animaciones de texto, desplazamientos y representaciones del juego con eficiencia y claridad.
- El diseño modular basado en dos microcontroladores —el ATmega328P para la lógica visual y de juego, y el PIC16F887 para los efectos sonoros— permitió dividir responsabilidades de forma clara, mejorando el rendimiento general del sistema. Esta distribución funcional también facilitó la programación y depuración de cada componente de manera independiente.
- La implementación de niveles de dificultad ajustados únicamente por la velocidad demostró ser suficiente para aumentar el desafío del juego sin alterar su lógica fundamental. Además, la inclusión de melodías temáticas mediante la reproducción de frecuencias musicales definidas permitió enriquecer la experiencia del usuario, añadiendo un estilo retro atractivo y reconocible.
- La simulación en Proteus mostró un comportamiento estable y funcional del sistema, validando la interacción entre los microcontroladores y la sincronización entre los efectos visuales y auditivos. Se comprobó que la comunicación digital simple por pines fue adecuada para las necesidades del proyecto.

9. Recomendaciones

- Se recomienda considerar en futuras versiones la implementación de un protocolo de comunicación más estructurado como UART, I2C o SPI si se planea expandir el número de eventos o intercambiar información más compleja (por ejemplo, puntajes, tiempo de juego o selección de niveles personalizados).
- También sería beneficioso incorporar un sistema de puntuación en tiempo real y una función de almacenamiento de récords, así como una posible expansión hacia un modo multijugador, que enriquecería aún más la dinámica del juego.
- En cuanto a la interfaz sonora, se sugiere aprovechar la infraestructura actual para implementar más eventos auditivos, como sonidos para cambio de nivel, pausa o victoria. También sería útil explorar técnicas de PWM o DAC si se desea mayor fidelidad en las señales acústicas.
- Finalmente, se recomienda documentar y estructurar claramente el código fuente, separar funciones en archivos individuales (por ejemplo, lógica de juego, visualización, comunicación) y realizar pruebas en hardware real para validar tiempos, rebotes y ruido electromagnético en entornos físicos.