Escuela Superior Politécnica del Litoral

Sistemas Embebidos

Tarea # 2

Comunicación entre microcontroladores
ATMEGA328P – PIC 16F887

Integrantes:

- Córdova Flores Luis Felipe
- Gallegos Orellana Alexander Joao
- Vera Castro Josué Matías

Docente:

Solís Mesa Ronald David

PAO II

2025-2026

Objetivo General

Desarrollar un videojuego interactivo que cuente con 3 niveles de dificultad, el cual se mostrará en una matriz LED 8x8 controlada por un microcontrolador ATmega328P, que, además, cuente con salidas auditivas que serán reproducidas con el PIC16F887, y utilizando entradas mediante pulsadores, al mismo tiempo que se emplea comunicación entre ambos dispositivos y simulando el funcionamiento completo en Proteus.

Objetivos Específicos

- Implementar un sistema visual dinámico usando una matriz LED 8×8 para mostrar obstáculos en movimiento y la posición del jugador.
- Permitir que el jugador controle su movimiento mediante botones físicos, desplazándose a izquierda o derecha.
- Incorporar distintos niveles de dificultad que modifiquen la velocidad del juego.
- Establecer una comunicación funcional entre el ATmega328P y el PIC16F887 para indicar eventos del juego (inicio, victoria, derrota).
- Diseñar efectos de sonido y melodías asociadas al juego, incluyendo música de fondo, sonidos de interacción, victoria y derrota.
- Sincronizar la lógica del juego con la respuesta auditiva proveniente del PIC.
- Simular el funcionamiento de todos los elementos en el entorno de Proteus.

Descripción técnica completa del Videojuego

El videojuego está compuesto por dos microcontroladores el ATmega328P el cual controla la parte visual del juego además de seleccionar la dificultad del videojuego mediante la entrada de botones. Y el PIC16F887 el cual controla el audio del juego ejecutando melodías durante el juego, al momento de la victoria y de la derrota.

El juego consiste en un camino vertical en el cual se van desplazando hacia abajo obstáculos de 8 bits que representan columnas llenas o vacías. El jugador se encuentra en la fila inferior con una mascara de bits que representan su posición. El ATmega328P mueve estos obstáculos hacia abajo y si un obstáculo coincide con la posición actual del jugador será la derrota para el jugador.

En el caso de que uno de los meteoros alcance a la nave espacial, esto terminará el juego mostrando una “X” en la matriz LED. En el caso de lograr esquivar todos los obstáculos, la nave llegará a la línea de meta y terminará el juego mostrando una “V”.

El videojuego posee 3 niveles de dificultad, el cual esta enfocado en la velocidad de caída de los obstáculos, siendo el nivel 1 el fácil con una velocidad de caída de 400ms, el 2 y el 3 siendo medio y difícil, con una velocidad de caída de 200ms y 100ms respectivamente. Esto se ve reflejado en la siguiente parte del código:

```
switch (dificultad) {
```

```
  case 1: step_delay_ms = 400; break;
  case 2: step_delay_ms = 200; break;
  case 3: step_delay_ms = 100; break;
}
```

El juego posee una pantalla inicial en la cual se muestra un icono de bienvenida. Ahora enfocándonos en el PIC16F887 funciones como **Melodia()** reproduce la canción principal. **Melody_Win()** reproduce secuencias de notas específicas. **Melody_Lose()** reproduce tonos descendentes y repetitivos.

Explicacion del Codigo ATmega328P

Antes de proceder al código se debe empezar declarando que se tienen los siguientes periféricos: Configuracion de pines de la matriz led 8x8 como pines de salida, Configura botones como entradas con pull-up, PC4 es la señal de salida de victoria y PC5 es la señal de salida de derrota.

Ahora centrándonos en las partes más importantes del código

La matriz se maneja por multiplexación:

```
PORTD = PORT_ROW[j];     // Selecciona fila
PORTB = ~matriz[j];      // Enciende columnas
```

El movimiento del jugador:

- **Botón izq.:** *if (!(PINC & (1 << BOTON_MOVER_IZQ))) player_pos <<= 1;*
- **Botón derecha**: *if (!(PINC & (1 << BOTON_MOVER_DER))) player_pos >>= 1;*

Actualización de obstáculos:

Se verifican cada cierto tiempo y dependiendo de la dificultad

```
for (int i = 7; i > 0; i--)
   game_data[i] = game_data[i - 1];
```

La primera fila recibe el siguiente byte del patrón:

```
game_data[0] = OBSTACULOS_PATRON[obstaculo_indice++];
```

Detección de colisión y envío de señal de partida perdida:

```
if (game_data[6] & player_pos) {
  PORTC |= (1 << PIN_PERDIDA);
}
```

Señal de victoria:

```
if (game_data[6] == 0xFF) PORTC |= (1 << PIN_GANADA);
```

Ahora explicaremos el código del PIC16F887

Cada nota es llamada a ***Sound_Play(frecuencia, tiempo),*** ademas se tienen las melodías principales:

- Melodia principal

```
void Melodia(){
  Sol_uno(); Sol_uno(); Mi_medio(); ...
}
```

- Melodia de victoria

```
void Melody_Win(){
  do_prime(); la_prime(); sol(); ...
}
```

- Melodia de derrota

```
void Melody_Lose(){
 ToneA(); ToneC(); ToneE();
}
```

Comunicación entre microcontroladores

El sistema usa comunicación digital simple por líneas de control, no serial.

Señal de Win: Desde el pin PC4 del ATmega al PIC

Señal de Lose: Desde el pin PC5 del ATmega hacia el PIC

Durante el juego el ATmega mantiene win o lose en estado bajo, en caso de derrota el ATmega pone a PC5= 1, entonces el PIC lo detecta y ejecuta Melody_Lose(); lo mismo sucede si es Win el ATmega pone a PC4= 1 y el PIC reproduce Melody_Win();

Cabe recalcar que la comunicación es unidireccional basada en señales digitales y basadas en acciones por flancos altos y no requiere protocolos como I2C o UART ni SPI.

Simulación

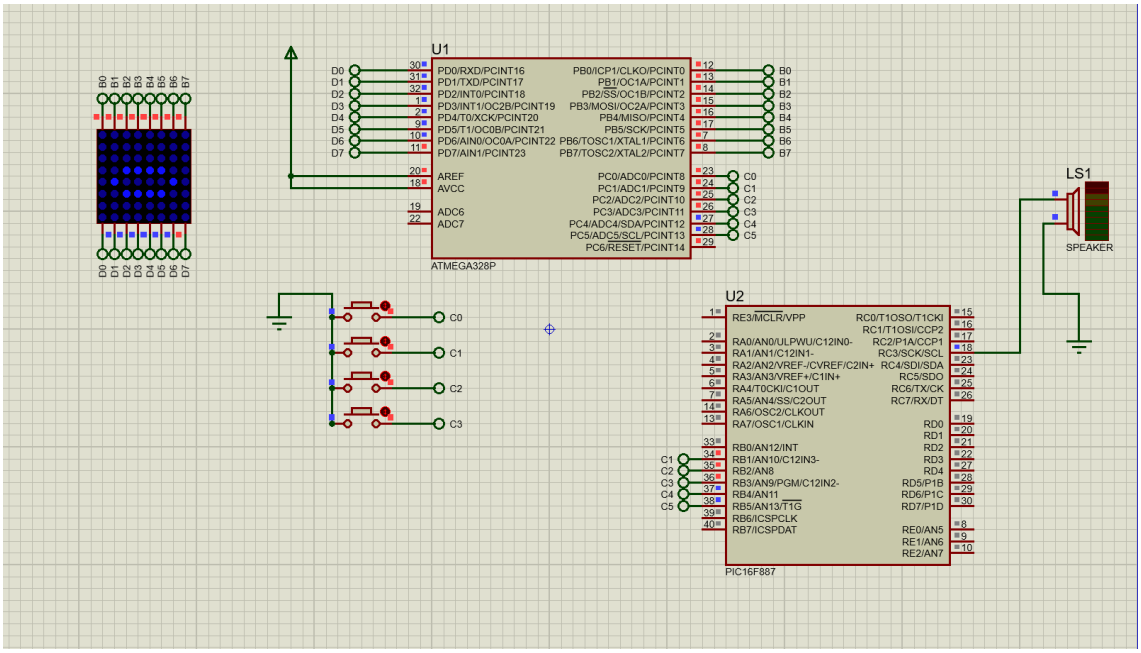

Figura 1. Inicio de simulación.

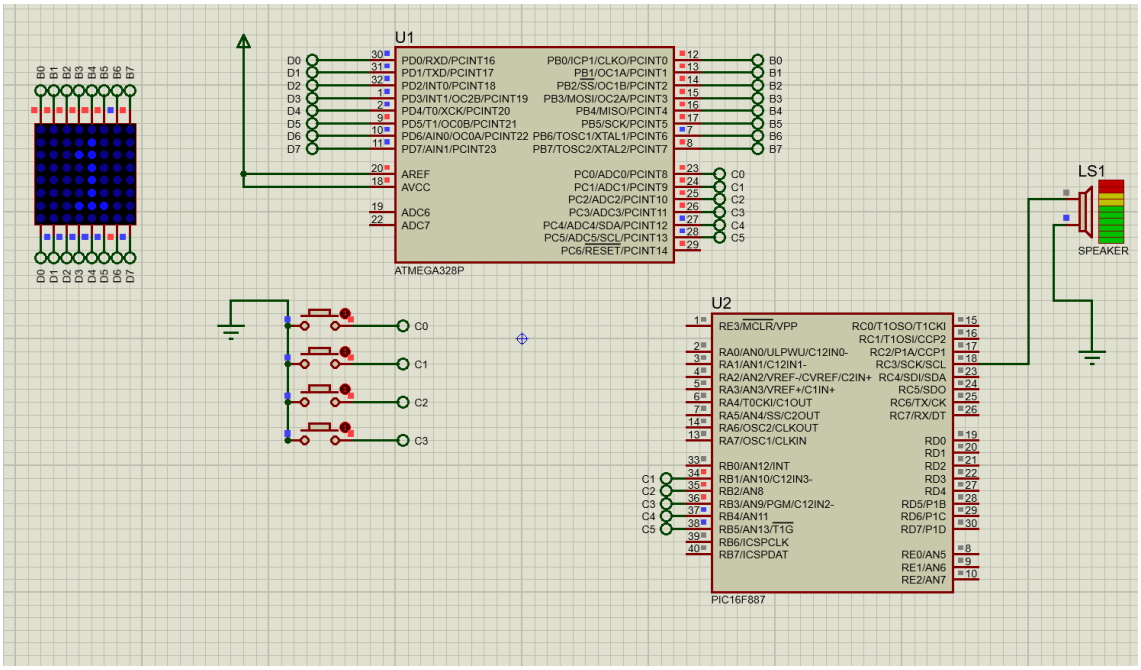

Figura 2. Selección nivel 1.

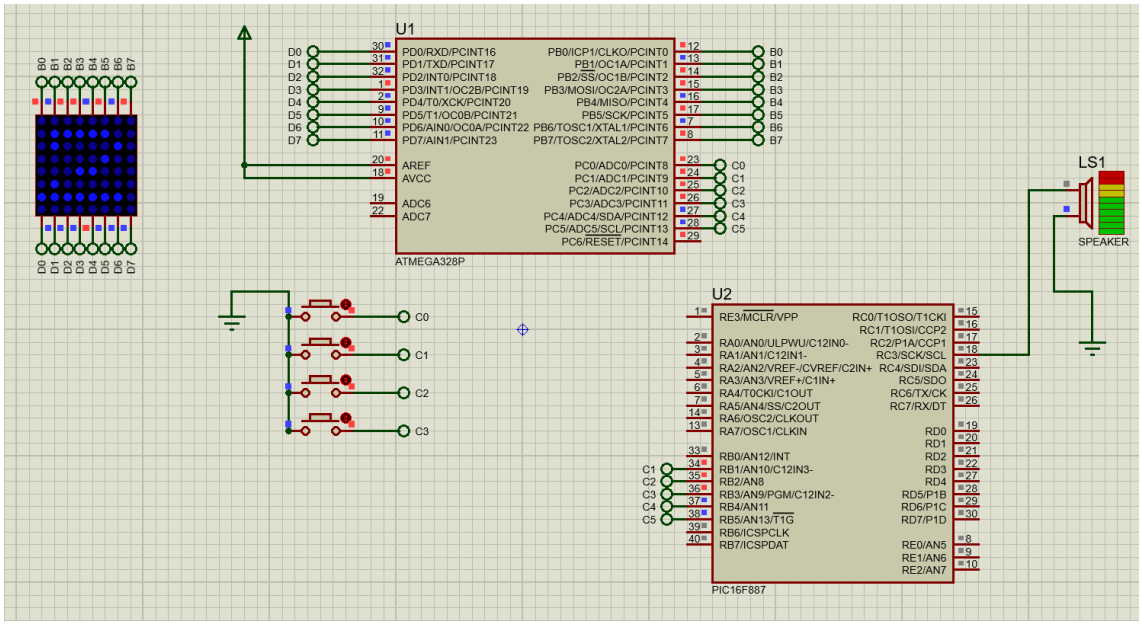

Figura 3. Selección nivel 2.

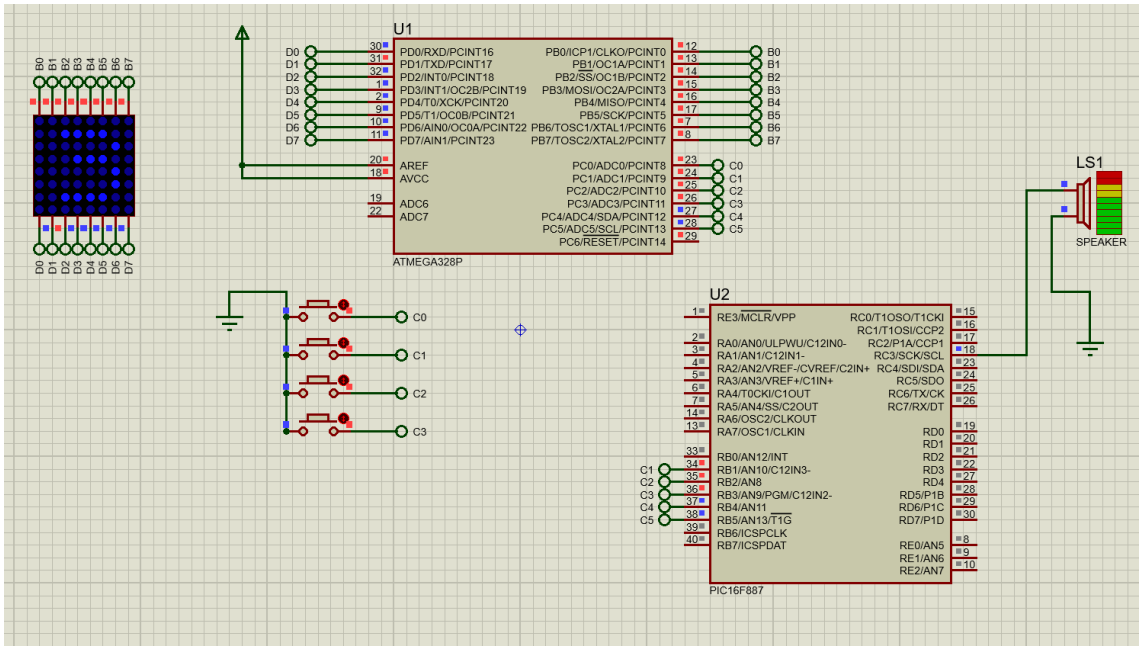

Figura 4. Selección nivel 3.

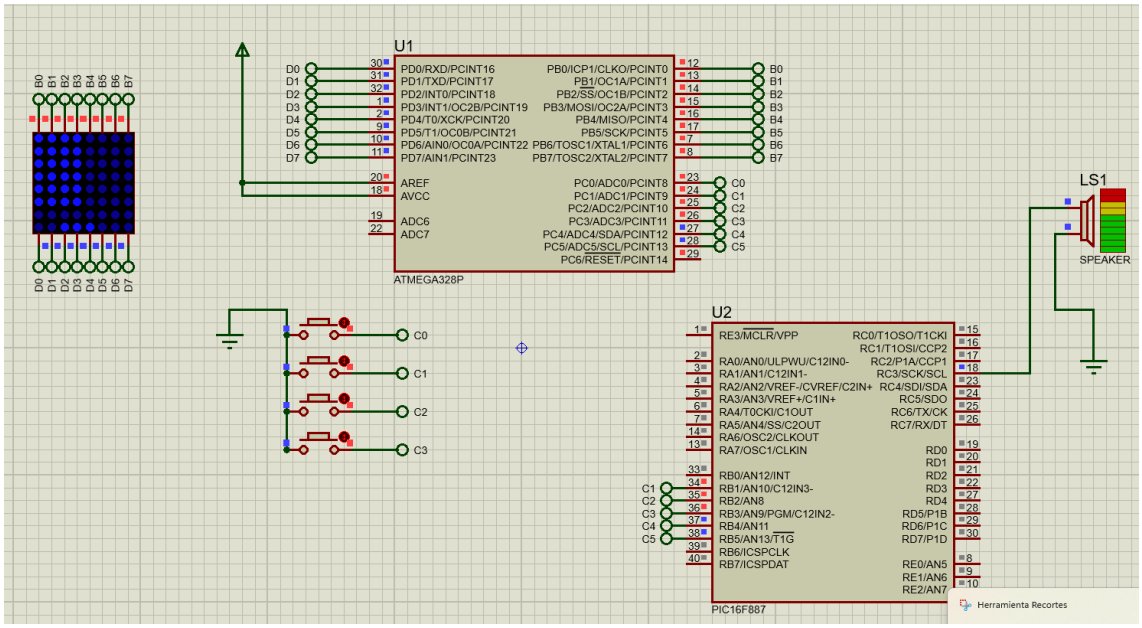

Figura 5. Inicio de juego.

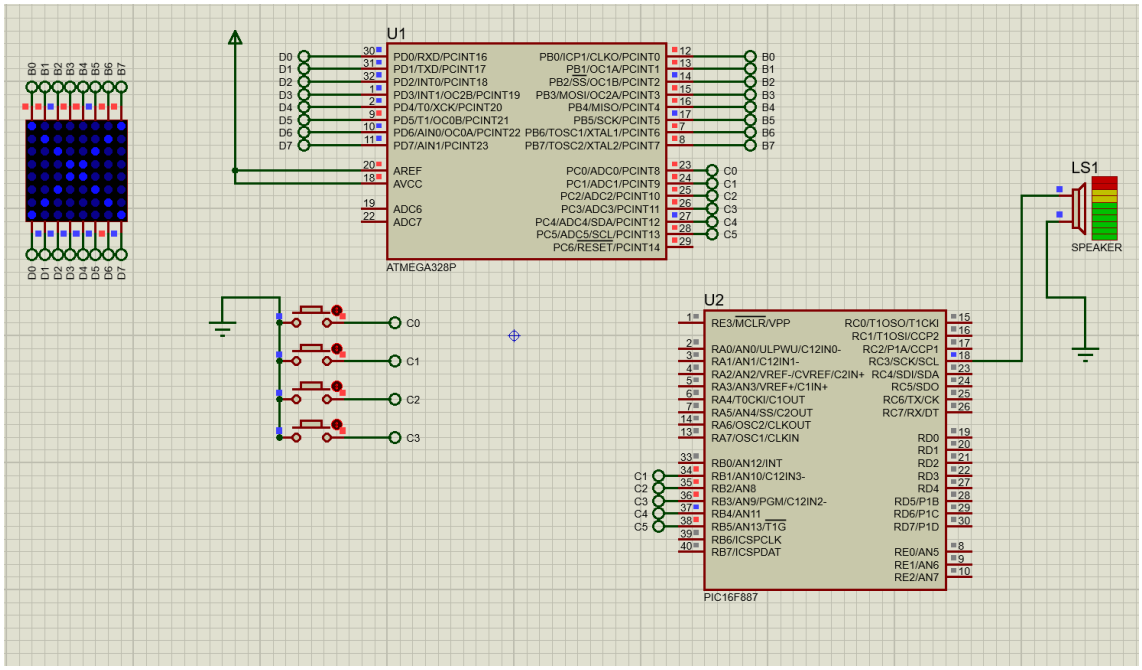

Figura 6. Derrota.

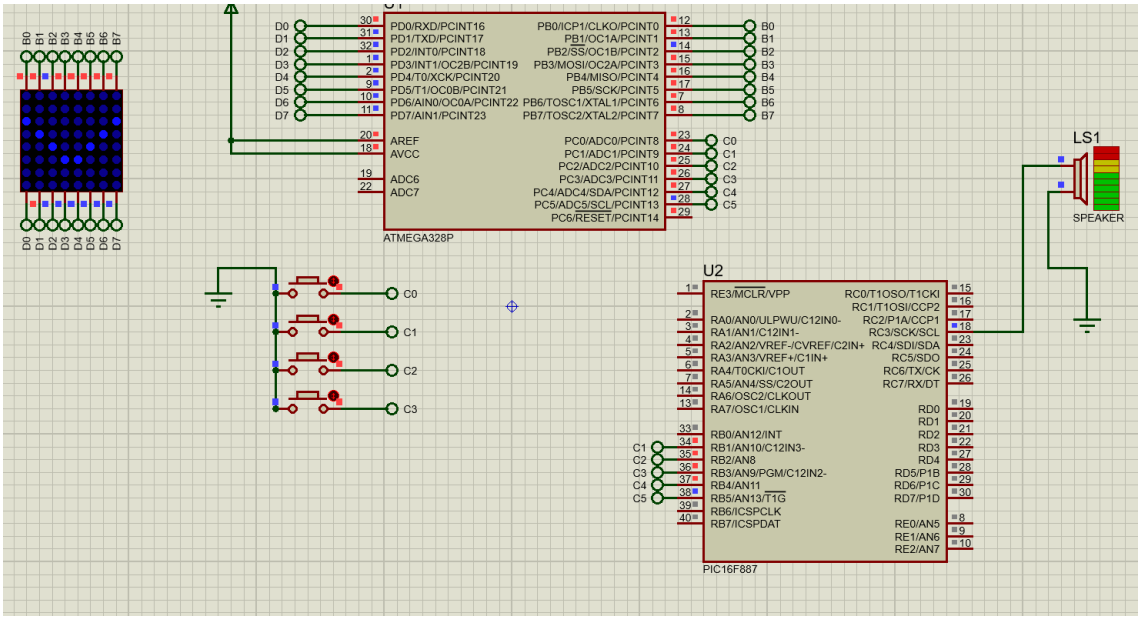

Figura 7. Victoria.

Enlace al repositorio de Github

https://github.com/TheDevil21/Videojuego.git

Conclusiones:

- El sistema combina correctamente la lógica de juego del ATmega328P con el sistema de audio del PIC16F887, logrando una interacción coordinada entre la jugabilidad (LED matrix) y los efectos sonoros (melodías de victoria y derrota).

- El proyecto permite a los usuarios interactuar con hardware real, comprender manejo de matrices LED, comunicación entre dispositivos y generación de audio digital. El sistema es adecuado para validar conceptos de electrónica digital y programación de microcontroladores.

- El ATmega328P implementa correctamente desplazamiento de obstáculos, detección de colisiones, selección de dificultad y gestión de pantalla final. Esto da como resultado un juego estable y funcional.

RECOMENDACIONES:

- Actualmente el refresco está ligado a retardos de software. Usar timers y/o interrupciones del ATmega328P permitiría una animación más estable y suave.

- Usar debouncing por software o hardware en botones
  Para mejorar la precisión en lecturas de movimiento y selección de dificultad.
- Agregar un sistema de puntuación
  Para hacer el juego más motivante, se recomienda mostrar en la matriz LED: puntos obtenidos o tiempo de supervivencia