U-Tetris: Tetris Controlado por UART

Joaquín Pérez Mauri, Juan Berniz, Felipe Morán, Julián Oreggioni
Universidad de la República, Montevideo, Uruguay
Email: {joaquin.perez.mauri,j.berniz.f,fmoran,juliano}@fing.edu.uy

***Resumen*—Se presenta la implementación del juego Tetris en el microcontrolador MSP430G2553 de Texas Instruments. La entrada de datos se realiza desde un teclado de una PC, mediante comunicación UART, y se visualiza en un display de tecnología OLED de 32 x 128 píxeles. El sistema diseñado es una prueba de concepto que fácilmente podría transformarse en un sistema portátil agregando botones y una batería, ya que solo ocupa 312 bytes de memoria RAM, y 4,8 kB de memoria Flash, y ofrece una autonomía mayor a 680 horas. El juego incluye funciones como el aumento de la dificultad, un sistema de puntuación y la pre-visualización de la próxima pieza.**

Palabras claves*— *entretenimiento, oled, bajo consumo

I. Introducción

La industria de los juegos electrónicos es una de las mayores industrias culturales, con más 2.500 millones de usuarios, que recauda cada año mas de 100.000 millones de dólares [1]. Uno de los juegos más conocidos es el Tetris [2]. Este juego, creado en 1984 en la Unión Soviética, no sólo revolucionó el mercado, sino que año tras año se siguen lanzando nuevas versiones, adaptándolo a las ultimas consolas y a nuevos equipos electrónicos [2].

Este trabajo presenta la prueba de concepto del juego cuya entrada de datos se realiza desde una PC para facilitar el desarrollo y el test automático desde la PC. El software desarrollado y su documentación está disponible en [3].

II. Solución Propuesta

El sistema está basado en el microcontrolador MSP430G2553 de Texas Instruments y un display de tecnología OLED (diodo orgánico de emisión de luz) de 32 x 128 píxeles que se comunican mediante I^2C (ver Fig. 1). El microcontrolador se conecta al puerto COM de una PC mediante UART. El usuario interactúa con el juego mediante una terminal serial y el teclado. De esta forma, se pueden mover y rotar las piezas que descienden en el display, llamadas tetraminós. Además de funciones básicas como gestión de colisiones entre tetraminós, eliminación de líneas completas y otras, la implementación incluye la pre-visualización del próximo tetraminó, que la aparición de los tetraminós sea pseudo-aleatoria, el aumento en la dificultad del juego modificando la velocidad de caída de los tetraminós, y un sistema de puntuación.

El display cuenta con memoria y contiene el controlador SSD1306 de Solomon Systech, que implementa la interfaz I^2C ofreciendo varios comandos que simplifican el uso de la pantalla. De [4] se obtuvieron funciones de inicialización y escritura de caracteres en el display.

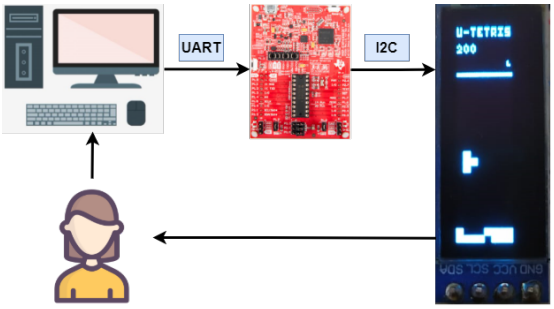

Figura 1. Diagrama funcional del sistema

III. Software Embebido

La arquitectura de software embebido utilizada es Round-Robin con interrupciones [5]. Éstas son generadas por dos periféricos: un timer utilizado para controlar la velocidad de caída de los tetraminós y una UART por donde llegan los comandos del usuario para interactuar con el juego. El main está compuesto por una inicialización general y por un loop infinito que se describe en la Fig. 2. En el bucle infinito se verifica si hubo alguna interrupción y se toma la acción correspondiente. En caso de que no llegue ninguna interrupción, se envía al microcontrolador a un modo de bajo consumo hasta que sea despertado por una interrupción.

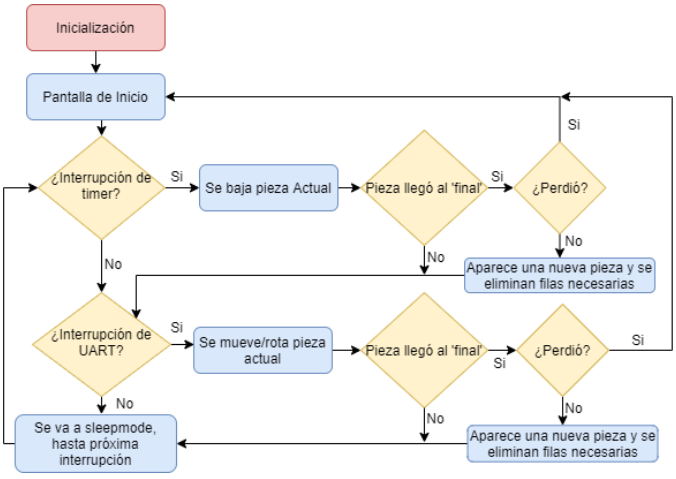

Figura 2. Diagrama de flujo del main

Al igual que el juego original, la presente implementación cuenta con 7 tetraminós. Cada uno es conformado por 4 *bloques* en distintas posiciones, donde cada bloque corresponde a un cuadrado de 4 x 4 píxeles. Para reducir la memoria

necesaria para almacenar el estado del juego, se toman los bloques como la unidad mínima de trabajo. Esto permite que el display de 128 x 32 píxeles pueda manejarse, a nivel del software embebido, como una matriz de 32 x 8 bloques. Ver Fig. 3 para mayor detalle.

Figura 3. Semántica utilizada en la programación del juego

Para representar cada tetraminó se utiliza una matriz de 4 x 4 bloques (ver Fig. 4), que permite incluir al tetraminó que ocupa más espacio tanto vertical como horizontal. La matriz se implementa como un vector de 4 bytes, donde cada byte indica qué bloque está presente en su respectiva columna. A modo de ejemplo, los números en azul abajo del bloque en Fig. 4 indican el valor que toma el vector para cada caso.

Para optimizar el uso de memoria, en lugar de generar un vector para cada tetraminó en cada posición se crearon funciones de *rotación* y *alineación*. La *rotación* se implementa intercambiando las columnas por las filas. Luego, es necesario realizar una *alineación* para evitar un desplazamiento indeseado (ver Fig. 4). Notar que el origen de coordenada utilizado para conocer la posición del tetraminó en el display se encuentra en la esquina inferior izquierda del mismo, pero como éste fue diseñado para ser usado en posición horizontal cuando se le envía la posición actual del tetraminó se debe alinear a la esquina superior izquierda.

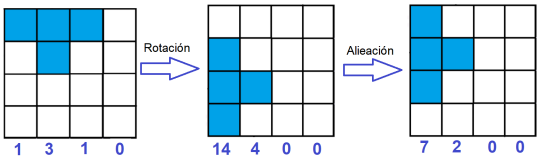

Figura 4. Diagrama explicativo de la función de rotación y de alineación

Para mostrar una secuencia de piezas diferente en cada juego se generan números pseudo-aleatorios con la función *rand()* de la librería *stdlib.h*.

IV. Resultados experimentales

En la Fig. 1 se puede ver una captura de pantalla del display. Las pruebas realizadas muestran que la aplicación funciona correctamente. Las funciones de aumento de la dificultad, el sistema de puntuación y la pre-visualización de la próxima pieza funcionan de acuerdo a lo esperado.

La Fig. 5 muestra el perfil de consumo de potencia medido con la herramienta *Energy Trace* de Texas Instruments. Se aprecian picos cuando el microcontrolador está en modo activo, y valles cuando no se registran interrupciones y se entra en un modo de bajo consumo. La primera sección de la gráfica señalada en rojo corresponde al consumo cuando se está en la pantalla de inicio, este consumo implica picos de 3,0 mA y un consumo promedio de 2,8 mA debido a que cada vez que el timer interrumpe, se verifica que no haya banderas encendidas y se vuelve al modo de bajo consumo. La segunda sección indicada en fucsia muestra el comienzo del juego sin recibir interrupciones por la UART, cuando llega la interrupción del timer solamente se borran los bloques de la pieza actual y se imprime la pieza mas abajo, lo que implica un consumo de 6,0 mA de pico y 3,7 mA de promedio. Y la tercer sección señalizada en azul corresponde al juego recibiendo continuamente mensajes por UART, que mueven o rotan el tetraminó actual, donde el consumo promedio es de 4,4 mA. Con estos resultados, si el sistema se alimentara de 2 baterías AA de 1500 mAh, se podría jugar en forma continua por mas de 682 horas. Los datos obtenidos del consumo son coherentes con lo esperado.

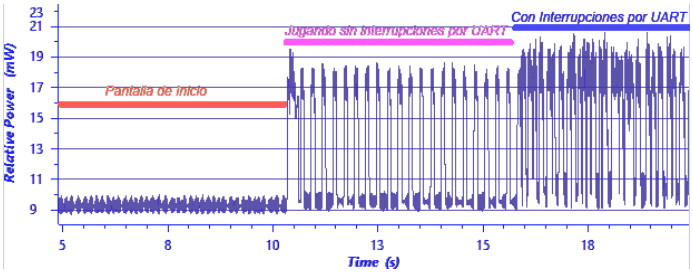

Figura 5. Consumo de potencia media.

Cuadro I
Comparación con otras soluciones que usan el mismo display

	Tiny Tetris [6]	U-Tetris
Plataforma	Arduino Uno	MSP430G2553
RAM (kB)	1.4 (68 %)	0.3 (60 %)
Flash (kB)	19.3 (61 %)	4.8 (29 %)
Autonomía	N/D	682 horas
Prestaciones	Próx pieza, sonido puntuación	Próx pieza, dificultad, puntuación

La implementación requirió 312 bytes de memoria RAM, que representa un 60 % del total, y ocupó 4,8 kB de memoria Flash, que es el 29 % del total. En el Cuadro I se resumen los principales resultados y se compara nuestra implementación con otra que usa el mismo display.

V. Conclusiones

Se implementó exitosamente el juego Tetris utilizando el microcontrolador MSP430G2553 y un display OLED. El juego es entretenido, la interacción lograda es correcta, lo que se logra mediante la variación de la dificultad, el sistema de puntuación y la pre-visualización de la próxima pieza. El sistema diseñado es una prueba de concepto que habilita el test automático desde el PC y fácilmente podría transformarse en un sistema portátil agregando botones y una batería. La implementación ocupa el 60 % de la memoria RAM, el 29 % de la memoria Flash, y ofrece una autonomía mayor a 680 horas.

REFERENCIAS

[1] A. García. (2019, Octubre) *Videojuegos: una industria de 135.000 millones*. Revista "Magazine Lifestyle", suplemento de "La Vanguardia", España. Consultado el 8/8/2021. [En linea]. Disponible en: http://www.magazinedigital.com/historias/reportajes/videojuegos-una-industria-135000-millones

[2] Electronic Arts. (2010, Enero) *Tetris Game Surpasses 100 Million Paid Mobile Downloads, is the Best-Selling Mobile Phone*. Consultado el 8/8/2021. [En linea]. Disponible en: https://www.ea.com/news/tetris-game-surpasses-100-million-mobile-downloads

[3] J. Berniz and J. Pérez-Mauri. (2021) UARTetris: Tetris controlado por UART. Proyecto asignatura Sistemas Embebidos para Tiempo Real. Facultad de Ingeniería, UdelaR. Montevideo, Uruguay. [En linea]. Disponible en: https://gitlab.fing.edu.uy/joaquin.perez.mauri/proyecto-sisem

[4] S. Perry. (2019, Mayo) *SSD1306 OLED display library for MSP430G2 TI launchpad*. Consultado el 8/8/2021. [En linea]. Disponible en: https://samueldperry.com/2019/05/24/ssd1306-oled-display-library-for-msp430g2-ti-lauchpad

[5] D. E. Simon, *An Embedded Software Primer*. Addison-Wesley Professional, 1999.

[6] A. J. Russell. (2016, Octubre) Tiny-tetris. Consultado el 8/8/2021. [En linea]. Disponible en: https://github.com/AJRussell/Tiny-Tetris