Universidad Autónoma de Baja California

Facultad de Ciencias químicas e ingeniería

Asignatura: Sistemas embebidos

Práctica de Laboratorio #5

Alumno:

Rodríguez Contreras, Raul Arturo

1261510

Docente: Lara Camacho, Evangelina

Objetivo

El alumno se familiarizará con el uso del periférico SPI usando el sistema embebido ESP32 DevKit V1 para desarrollar aplicaciones para sistemas basados en microcontrolador para aplicarlos en la resolución de problemas de cómputo, de una manera eficaz y responsable.

Equipo

Computadora personal con conexión a internet.

Teoría

Describa el modo SPI half-duplex del ESP32:

Desarrollo

Implemente en un eSP32 ESP-IDF una aplicación que descomprime un mensaje comprimido almacenado en un archivo de texto en una SD card, haciendo uso de SPI y tareas. La implementación debe ser eficiente en el uso de recursos de cómputo (procesador, memoria y periféricos).

El ESP32 está conectado por SPI a un adaptador de tarjetas micro SD card. El esp32 recibe del usuario por uart el nombre de un archivo, busca el archivo en la SD card, lee el contenido, lo descomprime e imprime en pantalla el texto resultante. Si el archivo no existe, despliega en pantalla “Archivo no encontrado”.
El mensaje comprimido consiste en lo siguiente:

- Consta únicamente de letras, números y corchetes.
- Cuando hay un bloque de código dentro de los corchetes que consiste en un número y letras, significa que se tienen que repetir las letras la cnatiddad de veces indicada por el número. Por ejemplo, el bloque [12AB] significa que se tiene que repetir 12 veces las letras AB.
- El mensaje puede tener varias capas de bloques.

Puede basarse en el código “sd_card_example_main.c” sobre manejo de SD cards en el ESP32 ESP-IDF.
Ejemplo 1
Mensaje en la SD card: AB[3CD]
Impresión en pantalla: ABCDCDCD

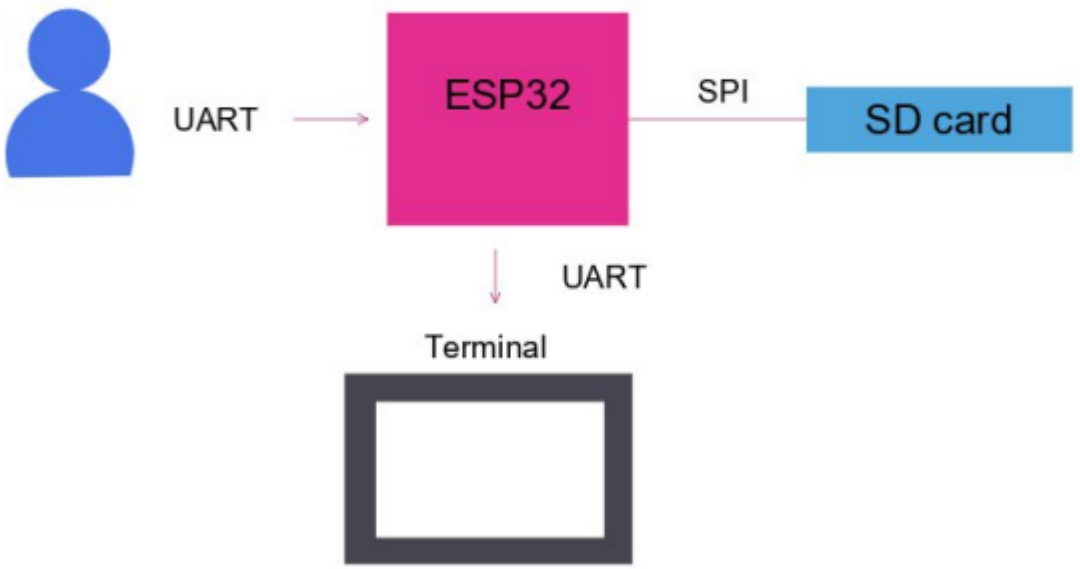
Ejemplo 2:
Mensaje en la SD card: AB[2C[2EF]G]
Impresión en pantalla: ABCEFEFGCEFEFG

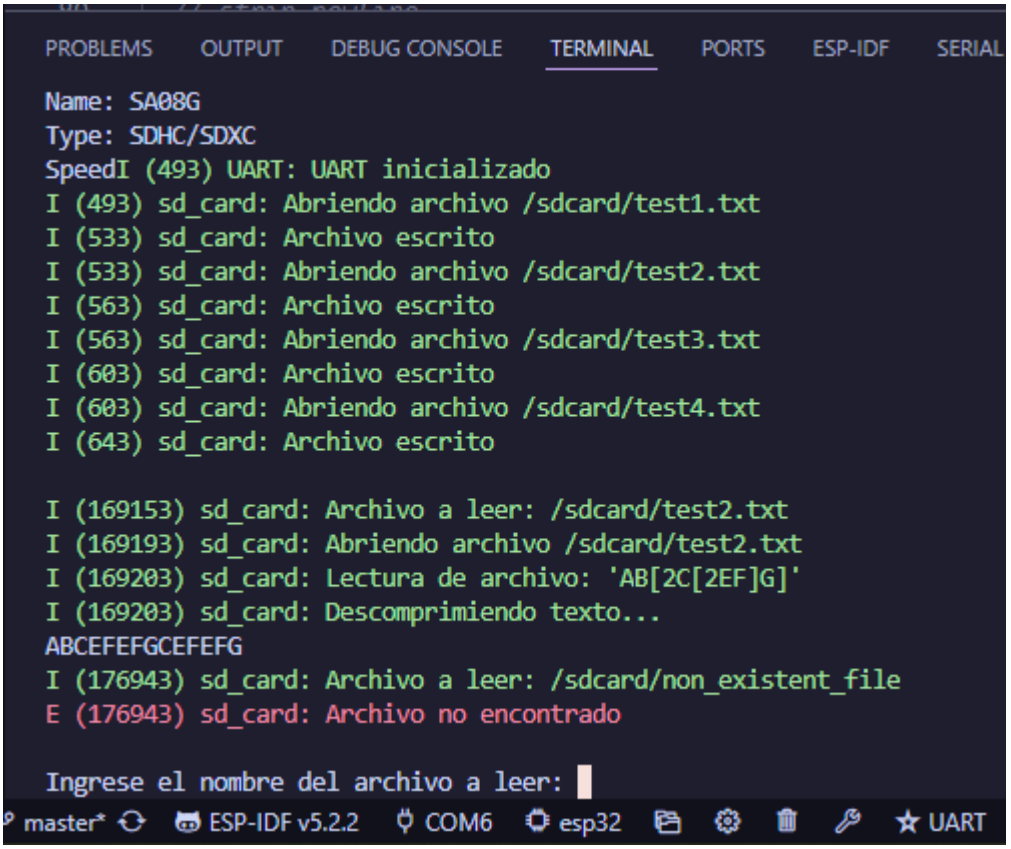
Ejemplo 3:
Mensaje en la SD card: IF[2A]LG[5M]D
Impresión en pantalla: IFAALGMMMMMD

Teoría

Describa a detalle la función xEventGroupSync la cual activa bits dentro de un grupo de eventos y luego espera a que se active una combinación de bits dentro del mismo grupo de eventos. Incluya un ejemplo de uso diferente al de la documentación. La función xEventGroupSync es utilizada para sincronizar múltiples tareas en un sistema de tiempo real mediante la activación y espera de ciertos bits dentro de un grupo de eventos. Esto es útil cuando varias tareas necesitan alcanzar un estado común antes de continuar su ejecución, logrando una coordinación precisa.

Fig. 1. Diagrama a bloques.




PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
ESP-IDF
SERIAL
Name: SA08G
Type: SDHC/SDXC
SpeedI (493) UART: UART inicializado
I (493) sd_card: Abriendo archivo /sdcard/test1.txt
I (533) sd_card: Archivo escrito
I (533) sd_card: Abriendo archivo /sdcard/test2.txt
I (563) sd_card: Archivo escrito
I (563) sd_card: Abriendo archivo /sdcard/test3.txt
I (603) sd_card: Archivo escrito
I (603) sd_card: Abriendo archivo /sdcard/test4.txt
I (643) sd_card: Archivo escrito
I (169153) sd_card: Archivo a leer: /sdcard/test2.txt
I (169193) sd_card: Abriendo archivo /sdcard/test2.txt
I (169203) sd_card: Lectura de archivo: 'AB[2C[2EF]G]'
I (169203) sd_card: Descomprimiendo texto...
ABCEFEFGCEFEFG
I (176943) sd_card: Archivo a leer: /sdcard/non_existent_file
E (176943) sd_card: Archivo no encontrado
Ingrese el nombre del archivo a leer:
master*
ESP-IDF v5.2.2
COM6
esp32
UART

Conclusiones y comentarios

El código conocido como “boilerplate” es bastante amplio en esta práctica, ya que realmente lo que se está procesando es solo el contenido de los archivos, que es donde el código estaba más complejo. Después de un tiempo de analizar el método de descompresión, se llegó a la conclusión de que era necesario implementar una solución recursiva, pero sin crear copias del buffer, por lo que la función se implementó con apuntadores, lo cual hace bastante eficiente el uso de procesamiento y memoria.

Dificultades de desarrollo

La dificultad no fue como tal dentro de la parte del protocolo SPI, ya que el SDK lo maneja de una manera sencilla, a diferencia del cambio de bibliotecas de I2C, el cual sí llevaba un trabajo detrás. Y el algoritmo al final resultó un ejercicio interesante que requirió un buen análisis de lo que se está intentando lograr. Tomó varias iteraciones pulir el diseño general, pero al final se completó de manera eficiente.