Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería



**SISTEMAS EMBEBIDOS**

**Práctica 5: SPI**

**Docente: Evangelina Lara Camacho**

**Alumnos:**

Gómez Cárdenas Emmanuel Alberto - 01261509

Lizarraga Soto Diego - 01244912

## Objetivo

El alumno se familiariza 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

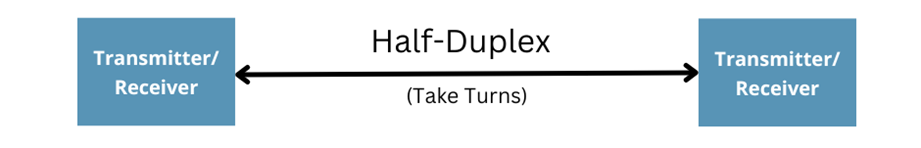
### Describa el modo SPI half-duplex del ESP32

**Definición:** El SPI (Serial Peripheral Interface) half-duplex es un modo de comunicación en el que los datos pueden fluir en una sola dirección a la vez, es decir, el maestro puede enviar datos al esclavo o recibir datos del esclavo, pero no ambos simultáneamente.

1. Uso compartido del bus de datos: en lugar de requerir pines dedicados para MOSI (Master Out, Slave In) y MISO (Master In, Slave Out), el modo half-duplex permite compartir un solo pin para ambas funciones.
2. Alternancia de transmisión y recepción: el ESP32 alterna entre transmitir y recibir en el mismo pin, lo que se controla mediante el código.
3. Reducción de velocidad efectiva de comunicación: debido a la alternancia entre las operaciones, la velocidad efectiva de transmisión de datos puede ser menor que en el modo full-duplex, ya que el dispositivo necesita cambiar el sentido de los datos en el bus.
4. Compatibilidad con dispositivos SPI de un solo cable: algunos sensores o dispositivos SPI que usan un solo cable para comunicación de datos se benefician de este modo half-duplex.

**Configuración de Hardware:** SPI usa cuatro líneas principales:

* MOSI (Master Out Slave In): Datos del maestro al esclavo.
* MISO (Master In Slave Out): Datos del esclavo al maestro.
* SCK (Clock): Sincroniza la transferencia de datos.
* CS (Chip Select): Habilita a esclavos específicos.

ESP-IDF permite configurar el modo half-duplex SPI configurando la estructura spi\_device\_interface\_config\_t. Se activa el modo half-duplex estableciendo el campo flags en la configuración con el valor SPI DEVICE HALF DUPLEX.

# Desarrollo

Implemente en un ESP 32 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, buscá 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.
* Cuándo 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 cantidad de veces indicada por el número.
* El mensaje puede tener varias capas de bloques.

Por ejemplo, el bloque [12AB] significa que se tiene que repetir 12 veces las letras AB.

* 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]

El programa imprime en pantalla: ABCDCDCD

**Ejemplo 2:**

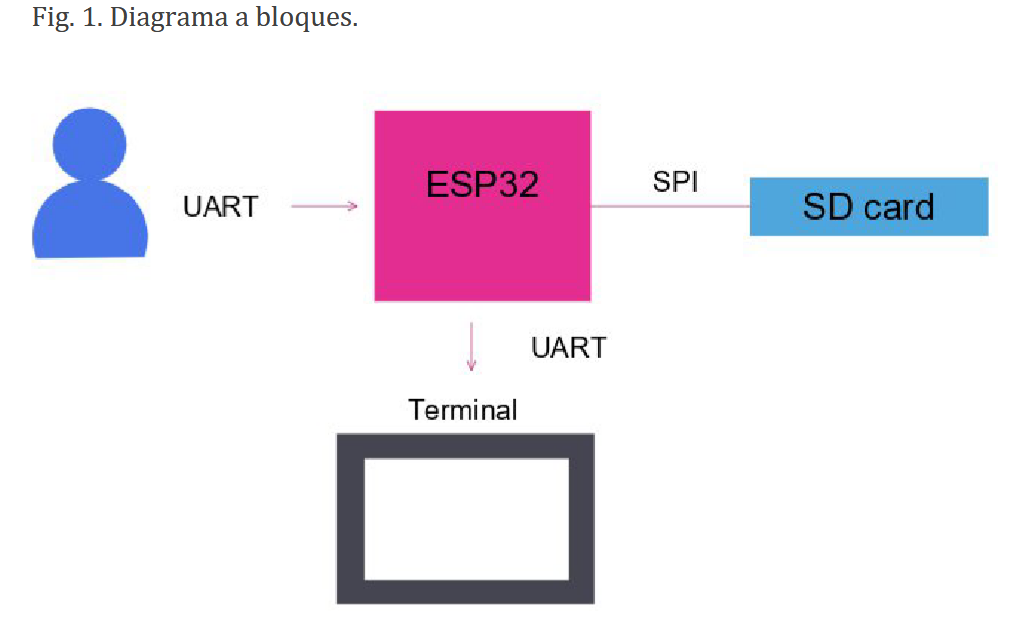
Mensaje en la SD Card: AB[2C[2EF]G]

El programa imprime en pantalla: ABCEFEFGCEFEFG

**Ejemplo 3:**

Mensaje en la SD Card: IF[2A]LG[5M]D

El programa imprime en pantalla: IFAALGMMMMMD



## Evidencia

## Inicio

## Ejemplo 1

## 

## Ejemplo 2

## 

## Ejemplo 3

## 

# Edge case testing

## 

## Conclusiones y Comentarios.

[Emmanuel Alberto Gómez Cárdenas](mailto:gomez.emmanuel@uabc.edu.mx): SPI es un protocolo de comunicación rápido y eficiente, la simplicidad con la que se maneja lo hace ideal para la transmisión de datos entre dispositivos a distancias cortas, como lo son los sensores, memorias, y demás.

[Diego Lizarraga Soto](mailto:diego.lizarraga@uabc.edu.mx): La práctica de comunicación SPI con una tarjeta SD permite transferir datos rápidamente entre un microcontrolador y el dispositivo de almacenamiento.Además, se destaca la importancia de comprender la configuración adecuada de pines y el manejo de comandos específicos para interactuar correctamente con la tarjeta SD.

## Dificultades en el Desarrollo

La recursión es una herramienta que nos ayudó a resolver el problema de la descompresión, sin embargo, al ser muy sensible a caer en los problemas como el stack overflow o un uso de memoria ineficiente, nos tomó tiempo implementar la recursión correctamente.

## Referencias

<https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/index.html>

Código  
El código fuente puede ser encontrado en el [Repositorio de GitHub “Sistemas Embebidos”](https://github.com/AlbGmx/SistemasEmbebidos)

#include <ctype.h>

#include <dirent.h>

#include <string.h>

#include <sys/stat.h>

#include <sys/unistd.h>

#include "driver/gpio.h"

#include "esp\_vfs\_fat.h"

#include "myUart.h"

#include "sdmmc\_cmd.h"

#define EXAMPLE\_MAX\_CHAR\_SIZE 64

#define MOUNT\_POINT "/sdcard"

#define PIN\_NUM\_MISO 19

#define PIN\_NUM\_MOSI 23

#define PIN\_NUM\_CLK 18

#define PIN\_NUM\_CS 5

#define START\_LOOP\_CHAR '['

#define END\_LOOP\_CHAR ']'

#define END\_LOOP\_CHAR\_TO\_IGNORE 1

static const char \*TAG = "sd\_card";

void list\_files\_in\_directory(const char \*path) {

struct dirent \*de;

DIR \*dr = opendir(path);

if (dr == NULL) {

ESP\_LOGE(TAG, "Could not open directory: %s", path);

return;

}

ESP\_LOGI(TAG, "\tImprimiendo FileSystem: %s", path);

ESP\_LOGI(TAG, "\t=============================================");

while ((de = readdir(dr)) != NULL) {

ESP\_LOGI(TAG, "\t\tFilename --> %s", de->d\_name);

}

ESP\_LOGI(TAG, "\t=============================================\n\n");

closedir(dr);

}

static esp\_err\_t s\_example\_write\_file(const char \*path, char \*data) {

ESP\_LOGI(TAG, "Abriendo archivo %s", path);

FILE \*f = fopen(path, "w");

if (f == NULL) {

ESP\_LOGE(TAG, "Fallo abrir el archivo para escritura");

return ESP\_FAIL;

}

fprintf(f, "%s", data); // Use "%s" to avoid formatting issues

fclose(f);

ESP\_LOGI(TAG, "Archivo escrito");

return ESP\_OK;

}

int getLoopEndingChar(const char \*str) {

int auxIndex = 0;

int openBrackets = 1;

while (str[auxIndex] && openBrackets > 0) {

char currentChar = str[auxIndex++];

if (currentChar == START\_LOOP\_CHAR) {

openBrackets++;

} else if (currentChar == END\_LOOP\_CHAR) {

openBrackets--;

}

}

return openBrackets == 0 ? auxIndex : -1;

}

int8\_t print\_recursion\_string(char \*str) {

int8\_t auxIndex = getLoopEndingChar(str);

if (auxIndex < 0) {

ESP\_LOGE(TAG, "Mismatched brackets.\n");

return -1;

}

int counter = 0;

int iterations = 0;

while (isdigit((unsigned char)str[counter]) && counter < auxIndex) {

iterations = iterations \* 10 + (str[counter] - '0');

counter++;

}

if (iterations == 0) iterations = 1;

char \*content\_start = &str[counter];

int content\_length = auxIndex - counter - END\_LOOP\_CHAR\_TO\_IGNORE;

for (int i = 0; i < iterations; ++i) {

int j = 0;

while (j < content\_length) {

if (content\_start[j] == START\_LOOP\_CHAR) {

j++;

int8\_t nestedIndex = print\_recursion\_string(&content\_start[j]);

if (nestedIndex < 0) return -1;

j += nestedIndex;

} else {

putchar(content\_start[j]);

j++;

}

}

}

return auxIndex;

}

void print\_expanded\_string(char \*str) {

while (\*str) {

if (\*str == START\_LOOP\_CHAR) {

int auxIndex = print\_recursion\_string(str + 1);

if (auxIndex < 0) {

ESP\_LOGE(TAG, "Error\n");

return;

}

str += auxIndex + 1;

} else {

putchar(\*str++);

}

}

putchar('\n');

}

static esp\_err\_t s\_example\_read\_file(const char \*path) {

ESP\_LOGI(TAG, "Abriendo archivo %s", path);

FILE \*f = fopen(path, "r");

if (f == NULL) {

ESP\_LOGE(TAG, "Fallo abrir el archivo para lectura");

return ESP\_FAIL;

}

char line[EXAMPLE\_MAX\_CHAR\_SIZE];

fgets(line, sizeof(line), f);

fclose(f);

// Strip newline

char \*pos = strchr(line, '\n');

if (pos) \*pos = '\0';

ESP\_LOGI(TAG, "RAW values: '%s'\n", line);

print\_expanded\_string(line);

printf("\n");

return ESP\_OK;

}

void app\_main(void) {

init\_UARTs();

// Give some time for the SD card to stabilize

vTaskDelay(pdMS\_TO\_TICKS(1000));

esp\_vfs\_fat\_sdmmc\_mount\_config\_t mount\_config = {

.format\_if\_mount\_failed = true,

.max\_files = 5,

.allocation\_unit\_size = 16 \* 1024,

};

sdmmc\_card\_t \*card;

sdmmc\_host\_t host = SDSPI\_HOST\_DEFAULT();

host.max\_freq\_khz = 5000;

spi\_bus\_config\_t bus\_cfg = {

.mosi\_io\_num = PIN\_NUM\_MOSI,

.miso\_io\_num = PIN\_NUM\_MISO,

.sclk\_io\_num = PIN\_NUM\_CLK,

.quadwp\_io\_num = -1,

.quadhd\_io\_num = -1,

.max\_transfer\_sz = 4000,

};

ESP\_ERROR\_CHECK(spi\_bus\_initialize(host.slot, &bus\_cfg, SDSPI\_DEFAULT\_DMA));

sdspi\_device\_config\_t slot\_config = SDSPI\_DEVICE\_CONFIG\_DEFAULT();

slot\_config.gpio\_cs = PIN\_NUM\_CS;

slot\_config.host\_id = host.slot;

ESP\_ERROR\_CHECK(esp\_vfs\_fat\_sdspi\_mount(MOUNT\_POINT, &host, &slot\_config, &mount\_config, &card));

ESP\_LOGI(TAG, "Filesystem montado");

sdmmc\_card\_print\_info(stdout, card);

char \*aux = (char \*)malloc(EXAMPLE\_MAX\_CHAR\_SIZE);

char \*filename\_path = (char \*)malloc(EXAMPLE\_MAX\_CHAR\_SIZE);

if (!aux || !filename\_path) {

ESP\_LOGE(TAG, "Memory allocation failed!");

} else {

while (true) {

list\_files\_in\_directory(MOUNT\_POINT);

put\_str(UART\_CONSOLE, "Leer o escribir archivo? (r/w): ");

char action = get\_char(UART\_CONSOLE);

if (action != 'r' && action != 'w') {

ESP\_LOGE(TAG, "Accion invalida");

continue;

}

snprintf(aux, EXAMPLE\_MAX\_CHAR\_SIZE, "\nIntroduce el nombre de archivo a %s: ", (action == 'r') ? "leer" : "ecribir");

put\_str(UART\_CONSOLE, aux);

char \*data = get\_line(UART\_CONSOLE);

if (data == NULL) {

ESP\_LOGE(TAG, "Error leyendo nombre de archivo!");

continue;

}

snprintf(filename\_path, EXAMPLE\_MAX\_CHAR\_SIZE, "%s/%s", MOUNT\_POINT, data);

esp\_err\_t ret;

if (action == 'r') {

ret = s\_example\_read\_file(filename\_path);

if (ret == ESP\_FAIL) {

ESP\_LOGE(TAG, "Error leyendo archivo!");

}

} else {

put\_str(UART\_CONSOLE, "\nIntroduce datos a escribir: ");

data = get\_line(UART\_CONSOLE);

if (data == NULL) {

ESP\_LOGE(TAG, "Error obteniendo data a escribir!");

continue;

}

ESP\_ERROR\_CHECK(s\_example\_write\_file(filename\_path, data));

}

ESP\_LOGI(TAG, "Hacer alguna otra accion? (y/n): ");

if (get\_char(UART\_CONSOLE) != 'y') {

break;

}

}

// Free allocated memory

free(filename\_path);

free(aux);

esp\_vfs\_fat\_sdcard\_unmount(MOUNT\_POINT, card);

ESP\_LOGW(TAG, "Tarjeta desmontada");

spi\_bus\_free(host.slot);

}

}