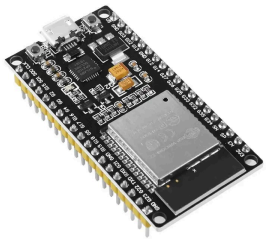
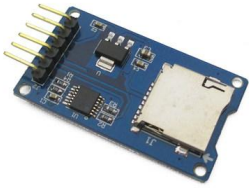
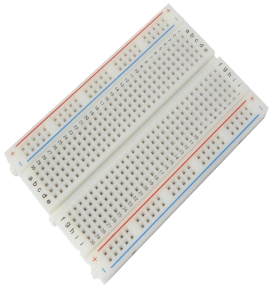
PRACTICA 6 : Buses de comunicación II (SPI)

MATERIAL

Para esta practica necesitaremos el microcontrolador ESP32, Tarjeta microSD y módulo lector, un Sensor lectura RFID, cables y una protoboard.

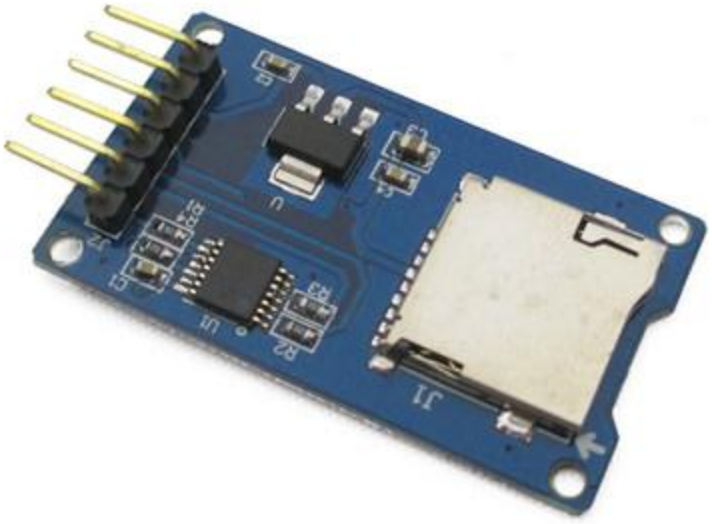

OBJETIVO Y FUNCIONALIDAD DE LA PRACTICA

El objetivo de la practica es comprender el funcionamiento del bus spi

Existen muchos dispositivos que tienen instalado el SPI para su control ; si bien en estas practicas vamos a poner
enfasis en aquellos dispositivos que a traves de comentarios de los alumnos son de facil obtencion .

Utilizaremos una lectura de SD

hardware : https://www.amazon.es/gp/product/B06XHJTGGC/ref=ppx_yo_dt_b_asin_title_o08_s00?ie=UTF8&psc=1



Utilizaremos una lectura RFID

hardware : https://www.amazon.es/BUYGOO-Keychain-Module-Reader-Arduino/dp/B07D9C82W8/ref=sr_1_7?adgrpid=64168290956&dchild=1&gclid=Cj0KCQjw9_mDBhCGARIsAN3PaFM9-kfIgN51lwJEzMFffK_oXlGh0EMOYz9he8RiqdjbXX8q_oYJGW0aAobkEALw_wcB&hvadid=320777693966&hvdev=c&hvlocphy=1005433&hvnetw=g&hvqmt=e&hvrand=2288607395559069836&hvtargid=kwd-302776364356&hydadcr=11857_1752977&keywords=rc522+rfid+module&qid=1618925460&sr=8-7


RFID-RC522

Ejercicio Practico 1 LECTURA/ESCRITURA DE MEMORIA SD

Código comentado

```
#include <SPI.h>     // Librería para comunicación SPI
#include <SD.h>      // Librería para usar tarjetas SD

// Definimos los pines personalizados para SPI
#define CS_PIN    5   // Pin de Chip Select (CS)
#define SCK_PIN   18  // Pin de reloj SPI (SCK)
#define MISO_PIN  2   // Pin de salida de datos del esclavo (MISO)
#define MOSI_PIN  15  // Pin de entrada de datos del esclavo (MOSI)

// Creamos una instancia de la clase SPI para el bus SPI1 (FSPI)
SPIClass spi = SPIClass(FSPI);

// Variable global para manejar archivos en la tarjeta SD
File myFile;

void setup() {
  Serial.begin(115200);  // Inicia la comunicación por puerto serie a 115200 baudios
  delay(2000);           // Espera 2 segundos para dar tiempo a que el puerto serie se co

  // Inicializamos el bus SPI con los pines personalizados
  spi.begin(SCK_PIN, MISO_PIN, MOSI_PIN, CS_PIN);

  Serial.print("Iniciando SD... ");

  // Intentamos inicializar la tarjeta SD usando el SPI personalizado y velocidad de 1MHz
  if (!SD.begin(CS_PIN, spi, 1000000)) {
    Serial.println("Fallo. Verifica:");
    Serial.println("- Pines y conexiones");
    Serial.println("- Formato FAT32");
    Serial.println("- Resistencia pull-up");
    while (1);  // Si falla, detiene el programa en un bucle infinito
  }

  Serial.println("Correcto");  // Si todo va bien, continúa

  // Intentamos abrir el archivo "myFile.txt" desde la tarjeta SD
  myFile = SD.open("/myFile.txt");
  if (myFile) {
    Serial.println("Contenido:");
```

```
    // Leemos y mostramos el contenido del archivo byte por byte
    while (myFile.available()) {
      Serial.write(myFile.read());
    }

    myFile.close();  // Cerramos el archivo
  } else {
    Serial.println("Error abriendo archivo");  // Si no se puede abrir el archivo, muestra
  }
}

void loop() {
  // Bucle principal vacío, ya que solo usamos setup()
}
```

Funcionamiento

1. Configura SPI personalizado usando pines específicos en lugar del SPI por defecto.
2. Inicia la comunicación serial para mostrar mensajes en el monitor.
3. Intenta inicializar la tarjeta SD con esos pines y velocidad de 1 MHz.
    - Si falla, muestra errores y se detiene.
4. Abre el archivo /myFile.txt:
    - Si lo encuentra, muestra su contenido en el monitor serial.
    - Si no, muestra un mensaje de error.
5. El loop() está vacío porque todo sucede una sola vez.

Salida monitor serie

Si todo funciona correctamente aparecera por pantalla:

```
Iniciando SD... Correcto
Contenido:
(contenido del archivo txt, en mi caso, Prueba practica 6 (part1))
```

En mi código he implementado diferentes pasos para verificar que todo funcione correctamente, o detectar más facilmente fallos. Estos errores te los mostrará de la siguiente

forma por el monitor serie:

```
Iniciando SD... Fallo. Verifica:
- Pines y conexiones
- Formato FAT32
- Resistencia pull-up
```

Video y foto demostración

https://drive.google.com/file/d/1d_OZs0m80aTNbgZWGsfb0BmogyupS0E2/view?usp=share_link
https://drive.google.com/file/d/1HCpVZHTdIxsG_Pt4plCRzjohp50I5vj5/view?usp=share_link

Ejercicio Practico 2 LECTURA DE ETIQUETA RFID

referencia: https://naylampmechatronics.com/blog/22_tutorial-modulo-lector-rfid-rc522.html

```
// Incluye la librería para comunicación SPI (protocolo que usa el RFID)
#include <SPI.h>
// Incluye la librería para manejar el lector RFID RC522
#include <MFRC522.h>
// Incluye funciones básicas de Arduino
#include <Arduino.h>

// Pines personalizados para ESP32-S3
// Define el pin 5 como SS (SDA) del lector RFID
#define SS_PIN 5
// Define el pin 6 para enviar datos SPI (MOSI)
#define MOSI_PIN 6
// Define el pin 7 para recibir datos SPI (MISO)
#define MISO_PIN 7
// Define el pin 8 como reloj SPI (SCK)
#define SCK_PIN 8
// Define el pin 4 como reset del lector RFID
#define RST_PIN 4
// Crea un objeto del lector RFID con los pines SS y RST definidos
MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup() {
  // Inicia la comunicación serial a 115200 baudios
  Serial.begin(115200);
  // Espera a que se abra el monitor serial (especialmente útil para algunas placas)
  while (!Serial);

  // Inicializa SPI con tus pines personalizados
  // Configura la comunicación SPI con los pines personalizados
  SPI.begin(SCK_PIN, MISO_PIN, MOSI_PIN, SS_PIN);

  // Inicializa el módulo RC522
  // Inicializa el lector RFID
  mfrc522.PCD_Init();
  // Muestra un mensaje en el monitor serial
  Serial.println("Lector RFID listo. Acerca una tarjeta...");
}

void loop() {
```

```
  // Revisa si hay una nueva tarjeta
  if (!mfrc522.PICC_IsNewCardPresent()) {
    // Si no hay tarjeta, termina la función y vuelve a empezar el loop
    return;
  }

  // Intenta leer la tarjeta
  if (!mfrc522.PICC_ReadCardSerial()) {
    // Si no se puede leer la tarjeta, termina la función
    return;
  }

  // Muestra el UID de la tarjeta
  // Muestra un mensaje antes del UID
  Serial.print("UID de la tarjeta: ");
  // Recorre cada byte del UID
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    // Agrega un cero si el byte es menor a 0x10 (para formato)
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");
    // Muestra el byte en formato hexadecimal
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    // Agrega ":" entre bytes, excepto al final
    Serial.print(i < mfrc522.uid.size - 1 ? ":" : "");
  }
  // Salta a la siguiente línea en el monitor serial
  Serial.println();

  // Finaliza comunicación con la tarjeta
  // Detiene la comunicación con la tarjeta leída
  mfrc522.PICC_HaltA();
}
```

Funcionamiento

Este código está diseñado para funcionar con una placa ESP32-S3 y un lector de tarjetas RFID modelo RC522. El propósito principal del programa es detectar cuando se acerca una tarjeta RFID al lector y mostrar en el monitor serial su identificador único (UID), que es como una huella digital de cada tarjeta.

Primero, el programa configura los pines del ESP32 que se usarán para comunicarse con el lector mediante el protocolo SPI. Luego, en la función de inicio (setup), se establece la comunicación serial para poder mostrar mensajes en pantalla y se inicializa el lector RFID.

En el ciclo principal (loop), el programa verifica constantemente si hay una nueva tarjeta presente. Si no la hay, simplemente espera. Si detecta una tarjeta, intenta leerla. Si la lectura es exitosa, muestra el UID de la tarjeta en el monitor serial.

Salida monitor serie

1. Al iniciar el programa:

```
Lector RFID listo. Acerca una tarjeta...
```

2. Cuando se acerca una tarjeta RFID válida:

```
UID de la tarjeta: XX:XX:XX:XX
```

Donde XX:XX:XX:XX representa el UID de la tarjeta en formato hexadecimal. La cantidad de bloques (XX) puede variar dependiendo del tipo de tarjeta (normalmente son 4, 7 o 10 dígitos hexadecimales separados por dos puntos).

Video demostración

https://drive.google.com/file/d/19LtfQhU7a8HSu9H82dNRn-NjvWdHbRe-/view?usp=share_link