

## PRÁCTICA 5: Buses de comunicación II (SPI)

En esta práctica, utilizaremos otro bus de comunicación conocido como SPI. Tiene otras características diferentes al ya usado bus I2C. Por ejemplo, el bus SPI tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro (master) puede iniciar la comunicación con uno o varios dispositivos esclavos (slave), y enviar o recibir datos de ellos.

Para ver como funciona y su implementación, utilizaremos los siguientes códigos:

### Código 1: Lectura/escritura de memoria SD

```
#include <SPI.h>
#include <SD.h>

File myFile;

void setup()
{
  Serial.begin(115200);
  Serial.print("Iniciando SD ...");
  if (!SD.begin(39)) {
    Serial.println("No se pudo inicializar");
    return;
  }
  Serial.println("inicializacion exitosa");

  myFile = SD.open("a.txt");//abrimos el archivo
  if (myFile) {
    Serial.println("a.txt:");
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    myFile.close(); //cerramos el archivo
  } else {
    Serial.println("Error al abrir el archivo");
  }
}

void loop()
{
}
```

En este código, básicamente se busca mostrar un texto que está escrito en un texto .txt llamado "a" almacenado en una SD externa. La SD se lee a partir de un lector de tarjetas que se comunica con SPI. Aquí va la explicación:

### Inclusión de librerías

```
#include <SPI.h>
#include <SD.h>
```

- \* Estas líneas incluyen las bibliotecas necesarias para la comunicación con la tarjeta SD mediante el protocolo SPI (Serial Peripheral Interface) y sus dichas funciones.

## Declaración de variables

```
File myFile;
```

- \* Se declara una variable de tipo *File* llamada *myFile*. Esta variable se usará para manejar el archivo en la tarjeta SD durante el código.

## Función *setup()*

```
void setup()
{
  Serial.begin(115200);
  Serial.print("Iniciando SD ...");

  if (!SD.begin(39)) {
    Serial.println("No se pudo inicializar");
    return;
  }
  Serial.println("inicializacion exitosa");

  myFile = SD.open("a.txt");
  if (myFile) {
    Serial.println("a.txt:");
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    myFile.close();
  } else {
    Serial.println("Error al abrir el archivo");
  }
}
```

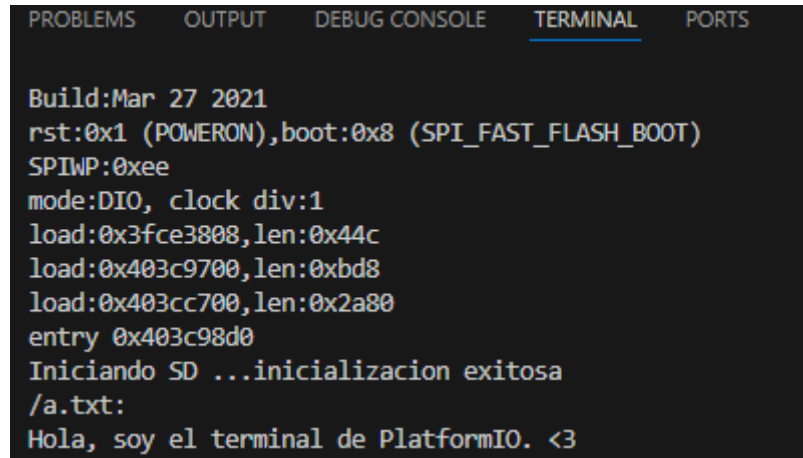
- \* ***Serial.begin(115200);*** inicializa la comunicación serie con una velocidad de 115200 baudios.
- \* ***Serial.print("Iniciando SD ...");*** imprime un mensaje indicando que se está iniciando la tarjeta SD.
- \* **En el *if*:**
  - o ***if (!SD.begin(39))***: intenta inicializar la tarjeta SD utilizando el pin 39 como el pin de selección de chip (CS). Si la inicialización falla, imprime "No se pudo inicializar" con un *Serial.println()* y termina la ejecución de la función *setup* con *return*;
  - o ***Serial.println("inicializacion exitosa");***: Indica que la inicialización de la tarjeta ha sido exitosa y muestra este mensaje.
- \* ***myFile = SD.open("a.txt");***: Intenta abrir el archivo *a.txt* en la tarjeta SD.
- \* ***if (myFile)***: Verifica si el archivo se abrió correctamente.
- \* ***Serial.println("a.txt:");***: Imprime el nombre del archivo.
- \* ***while (myFile.available()) { Serial.write(myFile.read()); }***: Con el *while* verifica que hay datos disponibles y con el *Serial.write*, lee y envía bit a bit el contenido del texto al puerto serie (Terminal).
- \* ***myFile.close();***: Cierra el archivo después de haber leído todo su contenido.

- \* **`else { Serial.println("Error al abrir el archivo"); }`** : Se ejecuta si el archivo no se pudo abrir, imprimiendo un mensaje de error.

## Función `loop()`

En este código, el `loop` está vacío. Igualmente, lo que hemos programado se ejecutará.

## Imagen del terminal



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Build:Mar 27 2021
rst:0x1 (POWERON),boot:0x8 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3808,len:0x44c
load:0x403c9700,len:0xbd8
load:0x403cc700,len:0x2a80
entry 0x403c98d0
Iniciando SD ...inicializacion exitosa
/a.txt:
Hola, soy el terminal de PlatformIO. <3
```

## Código 2: LECTURA DE ETIQUETA RFID

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN    8    //Pin 9 para el reset del RC522
#define SS_PIN     9    //Pin 10 para el SS (SDA) del RC522
MFRC522 mfrc522(SS_PIN, RST_PIN); //Creamos el objeto para el RC522

void setup() {
    Serial.begin(115200); //Iniciamos la comunicaci3n serial
    SPI.begin();         //Iniciamos el Bus SPI
    mfrc522.PCD_Init();  // Iniciamos el MFRC522
    Serial.println("Lectura del UID");
}

void loop() {
    // Revisamos si hay nuevas tarjetas presentes
    if ( mfrc522.PICC_IsNewCardPresent())
    {
        //Seleccionamos una tarjeta
        if ( mfrc522.PICC_ReadCardSerial())
        {
            // Enviamos serialamente su UID
            Serial.print("Card UID:");
            for (byte i = 0; i < mfrc522.uid.size; i++) {
                Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                Serial.print(mfrc522.uid.uidByte[i], HEX);
            }
            Serial.println();
            // Terminamos la lectura de la tarjeta actual
        }
    }
}
```

```

        mfr522.PICC_HaltA();
    }
}

```

En este código, se pretende leer la etiqueta que tienen, por ejemplo, las tarjetas de crédito o las tarjetas identificativas de una empresa la cual se utiliza para permitir el acceso a algún sitio o, en el caso de las tarjetas de crédito, para pagar inalámbricamente. Para ello utilizaremos un lector RFID, el cuál se comunica por SPI.

Primero inicializa un módulo RFID RC522 y espera a que una tarjeta RFID se acerque al lector. Cuando una tarjeta se detecta, lee su UID (Identificador Único) y lo imprime en el monitor serial en formato hexadecimal. Luego, termina la comunicación con la tarjeta para estar listo para leer otra tarjeta.

## Inclusión de librerías

```

#include <SPI.h>
#include <MFRC522.h>

```

- \* Estas líneas incluyen las bibliotecas necesarias para la comunicación SPI y para manejar el módulo RFID RC522, con sus dichas funciones.

## Definición de pines

```

#define RST_PIN 8
#define SS_PIN 9

```

- \* **RST\_PIN 8** : Se corresponde al pin de reset y se conectará al pin 8 de la placa
- \* **SS\_PIN 9** : Se corresponde al pin de selección del esclavo (SS) e irá conectado al pin 9 de la placa.

## Creación de objeto

```

MFRC522 mfrc522(SS_PIN, RST_PIN);

```

- \* Se crea un objeto *mfrc522* de la clase MFRC522, inicializándolo con los pines definidos anteriormente. Este objeto se utilizará para interactuar con el módulo RFID.

## Función *setup()*

```

void setup() {
    Serial.begin(115200);
    SPI.begin();
    mfrc522.PCD_Init();
    Serial.println("Lectura del UID");
}

```

El *setup()* se ejecuta una sola vez cuando el programa comienza y se utiliza para inicializar el hardware y configurar el entorno.

- \* **Serial.begin(115200)** : Inicializa la comunicación serial con una velocidad de 115200 baudios.
- \* **SPI.begin()** : Inicializa el bus SPI.
- \* **mfrc522.PCD\_Init()** : Inicializa el módulo RFID RC522.

- \* ***Serial.println("Lectura del UID");*** : Imprime un mensaje indicando que el sistema está listo para leer el UID de una tarjeta.

## Función *loop()*

```
void loop() {  
  
    if (mfrc522.PICC_IsNewCardPresent()) {  
  
        if (mfrc522.PICC_ReadCardSerial()) {  
  
            Serial.print("Card UID:");  
            for (byte i = 0; i < mfrc522.uid.size; i++) {  
                Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");  
                Serial.print(mfrc522.uid.uidByte[i], HEX);  
            }  
            Serial.println();  
  
            mfrc522.PICC_HaltA();  
        }  
    }  
}
```

- \* ***if (mfrc522.PICC\_IsNewCardPresent())***: Verifica si hay una nueva tarjeta presente.
  - ***if (mfrc522.PICC\_ReadCardSerial())*** : Selecciona la tarjeta y lee su número de serie (UID).
    - ***Serial.print("Card UID:");*** : Imprime "Card UID:" en el monitor serial.
    - ***for (byte i = 0; i < mfrc522.uid.size; i++)*** : Recorre cada byte del UID de la tarjeta.
      - ***Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");*** Imprime un espacio o un cero adicional si el byte es menor que 0x10, para formatear correctamente el UID.
      - ***Serial.print(mfrc522.uid.uidByte[i], HEX);*** : Imprime cada byte del UID en formato hexadecimal.
    - ***Serial.println();*** : Imprime una nueva línea después de imprimir el UID completo.
    - ***mfrc522.PICC\_HaltA();*** Termina la comunicación con la tarjeta actual para que se pueda leer una nueva tarjeta.

## Imagen del terminal

Desgraciadamente, para este código no he podido tener una captura de lo que sale en el terminal porque no dispongo del RFID y en el día de la práctica, olvidé hacerle una foto.

Lo que sí puedo asegurar, es que estuvimos probando y sí funcionaba. Acercamos al lector tarjetas de crédito y la propia tarjeta de la UPC y siempre nos salía el UID correspondiente.