Instituto Politécnico Nacional

Escuela Superior de Cómputo

Practica 6.- comunicación SPI con Arduino y sensor BMP280

Sistemas en chip
Profesora: Ana Luz Barrales Lopez

Luciano Hernández Jonathan
Rodriguez Morales Iñaki

17 de noviembre de 2024

1 Introducción

SPI es una interfaz de comunicación serial sincrónica que permite el intercambio rápido de datos entre un microcontrolador (maestro) y uno o más periféricos (esclavos). Utiliza cuatro líneas de comunicación principales:

- **MOSI (Master Out Slave In):** Línea para los datos enviados del maestro al esclavo.
- **MISO (Master In Slave Out):** Línea para los datos enviados del esclavo al maestro.
- **SCK (Serial Clock):** Reloj generado por el maestro para sincronizar la transferencia de datos.
- **SS (Slave Select):** Línea de selección de esclavo; permite activar o desactivar un dispositivo esclavo.

1.1 Objetivo

Aprender cómo funciona la comunicación SPI mediante la configuración y uso de esta interfaz en un microcontrolador, intercambiando datos entre el microcontrolador y un periférico.

1.2 Materiales

Los materiales utilizados en la práctica son los siguientes:

- Arduino Uno
- Arduino IDE
- Cable para conexiones
- 2 Sensores de presion y temperatura (BME280)

2 Desarrollo de la Práctica

Se conecta el pin VCC del Arduino al VCC de ambos sensores y del LCD, y el GND al GND de todos los dispositivos. Para el bus SPI, conecta el pin SCK del LCD y de los sensores al pin 13 del Arduino, el pin MISO de los sensores al pin 12, y el pin MOSI del LCD al pin 11. Cada dispositivo tendrá un pin de selección (CS/CE): asigna un pin digital del Arduino (por ejemplo, pin 9 para el CS del primer sensor, pin 10 para el segundo sensor, y pin 8 para el LCD).

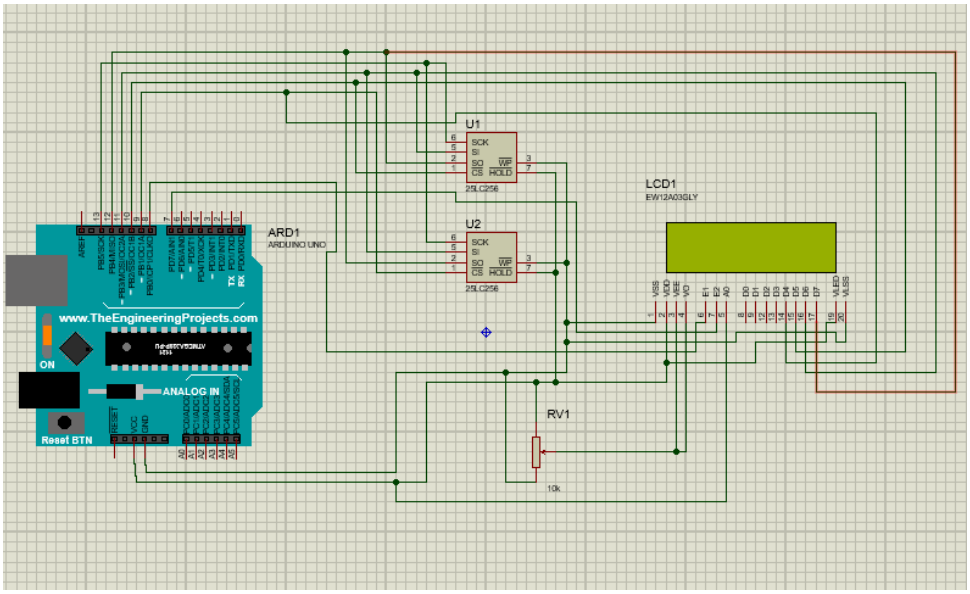

Figure 1: Diagrama del circuito: UART simplex

2.1 Desarrollo del código

2.2 Librerias necesarias

Este bloque incluye las librerías necesarias para el programa. SPI.h permite utilizar el protocolo SPI, Wire.h es útil para la comunicación I2C (aunque no se usa directamente en este caso), y las librerías $Adafruit_Sensor.hyAdafruit_BMP280.hpermiteninteractuarconlossensoresBMP2$

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
```

2.3 Pines digitales

En este bloque, se definen los pines digitales para los pines de selección de chip (CS) de los dos sensores BMP280. Además, se crean los objetos bmp1 y bmp2 para manejar cada sensor de forma independiente.

```
#define SENSOR1_CS 9
#define SENSOR2_CS 10

Adafruit_BMP280 bmp1;
Adafruit_BMP280 bmp2;
```

2.4 Comunicación serial

Este bloque inicializa la comunicación serial para la depuración, el protocolo SPI, y configura los pines CS de los sensores como salidas. Además, se asegura de que ambos sensores BMP280 estén correctamente conectados. Si alguno de los sensores no responde, el programa muestra un mensaje de error en el monitor serial y se detiene.

```
void setup() {
```

```
  Serial.begin(9600);
  SPI.begin();

  pinMode(SENSOR1_CS, OUTPUT);
  pinMode(SENSOR2_CS, OUTPUT);
  digitalWrite(SENSOR1_CS, HIGH);
  digitalWrite(SENSOR2_CS, HIGH);

  if (!bmp1.begin(SENSOR1_CS)) {
    Serial.println("Sensor 1 no encontrado.");
    while (1);
  }
  if (!bmp2.begin(SENSOR2_CS)) {
    Serial.println("Sensor 2 no encontrado.");
    while (1);
  }
}
```

2.5 Ciclo principal de activación de los sensores

En el ciclo principal, se activan los sensores de forma secuencial mediante el manejo de los pines CS (LOW para activar y HIGH para desactivar). Se leen la temperatura y la presión de cada sensor utilizando las funciones proporcionadas por la librería Adafruit$_BMP280.Luego, losdatosobtenidosseimprimenenelmonitorserial.Finalmente, elprogramaesp$

```
    void loop() {
      digitalWrite(SENSOR1_CS, LOW);
      float temp1 = bmp1.readTemperature();
      float pres1 = bmp1.readPressure();
      digitalWrite(SENSOR1_CS, HIGH);

      digitalWrite(SENSOR2_CS, LOW);
      float temp2 = bmp2.readTemperature();
      float pres2 = bmp2.readPressure();
      digitalWrite(SENSOR2_CS, HIGH);

      Serial.print(temp1); Serial.print("  C , Pres: ");
      Serial.print(pres1); Serial.println(" Pa");

      Serial.print(temp2); Serial.print("  C , Pres: ");
      Serial.print(pres2); Serial.println(" Pa");

      delay(2000);
    }
```

2.6 Simulación

A continuación se muestra el funcionamiento de esta práctica en el software de proteus y usando un display lcd que se mostró en el primer esquema de conexión.

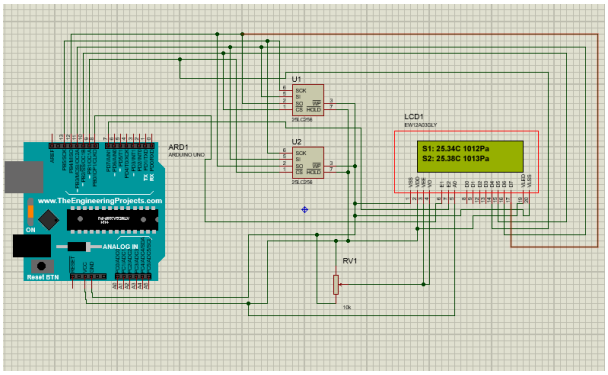

Figure 2: Funcionamiento del circuito

Primera imagen donde se puede observar la conexion que se siguió para el correcto funcionamiento de la practica donde el arduino controla de manera correcta ambos esclavos(sensores) y muestra los datos en el display LCD.

2.7 Implementación

Una vez implementado la conexion y desarrollo del codigo dentro de la simulación, se elaboró el circuito de manera practica con componentes reales para observar su funcionamiento, a continuación, se muestran las imagenes del citcuito en funcionamiento conectado a a un arduino uno.

Para esto se utilizó un codgio ligeramente diferente, pues en este casos se recibirán los datos en el monitor serial del IDE del arduino. El codigo es el siguiente.

```
#include <SPI.h>

// Pines de selecci n de esclavo para los dos sensores
const int CSB_PIN1 = 10;
const int CSB_PIN2 = 9;

// Variables para los coeficientes de calibraci n
uint16_t dig_T1_1, dig_T1_2;
int16_t dig_T2_1, dig_T2_2;
int16_t dig_T3_1, dig_T3_2;
uint32_t t_fine_1, t_fine_2;

void setup() {
  // Configurar pines de esclavo y comunicaci n SPI
  SPI.begin();
  pinMode(CSB_PIN1, OUTPUT);
  pinMode(CSB_PIN2, OUTPUT);
  digitalWrite(CSB_PIN1, HIGH);
  digitalWrite(CSB_PIN2, HIGH);

  Serial.begin(9600);

  // Inicializar sensores
  initBMP280(CSB_PIN1);
  initBMP280(CSB_PIN2);
}

void loop() {
```

```
29      // Leer temperatura y presi n del primer sensor
30      float temperatura1 = readTemperature(CSB_PIN1, 1);
31      float presion1 = readPressure(CSB_PIN1, 1);
32
33      // Leer temperatura y presi n del segundo sensor
34      float temperatura2 = readTemperature(CSB_PIN2, 2);
35      float presion2 = readPressure(CSB_PIN2, 2);
36
37      // Mostrar datos del primer sensor
38      Serial.print("Temperatura 1: ");
39      Serial.print(temperatura1);
40      Serial.println("  C ");
41      Serial.print("Presi n 1: ");
42      Serial.print(presion1);
43      Serial.println(" Pa");
44
45      // Mostrar datos del segundo sensor
46      Serial.print("Temperatura 2: ");
47      Serial.print(temperatura2);
48      Serial.println("  C ");
49      Serial.print("Presi n 2: ");
50      Serial.print(presion2);
51      Serial.println(" Pa");
52      Serial.println("\n--------------\n ")
53
54      delay(1000);
55    }
56
57    void initBMP280(int csbPin) {
58      digitalWrite(csbPin, LOW);
59      writeRegister(csbPin, 0xF4, 0x27); // Configurar control (modo
    normal, oversampling)
60      writeRegister(csbPin, 0xF5, 0xA0); // Configurar tiempo de espera
     y filtro
61
62      // Leer coeficientes de calibraci n
63      dig_T1_1 = readRegister16(csbPin, 0x88);
64      dig_T2_1 = readRegister16Signed(csbPin, 0x8A);
65      dig_T3_1 = readRegister16Signed(csbPin, 0x8C);
66
67      dig_T1_2 = dig_T1_1; // Usar mismas calibraciones para ambos
    sensores (ajustar si necesario)
68      dig_T2_2 = dig_T2_1;
69      dig_T3_2 = dig_T3_1;
70
71      digitalWrite(csbPin, HIGH);
72    }
73
74    float readTemperature(int csbPin, int sensor) {
75      digitalWrite(csbPin, LOW);
76      int32_t rawTemp = ((uint32_t)readRegister(csbPin, 0xFA) << 12) |
77                        ((uint32_t)readRegister(csbPin, 0xFB) << 4) |
78                        ((readRegister(csbPin, 0xFC) >> 4));
79      digitalWrite(csbPin, HIGH);
80
81      int32_t var1, var2;
82      if (sensor == 1) {
83        var1 = ((((rawTemp >> 3) - ((int32_t)dig_T1_1 << 1))) * ((
```

```
int32_t)dig_T2_1)) >> 11;
84        var2 = (((((rawTemp >> 4) - ((int32_t)dig_T1_1)) * ((rawTemp >>
      4) - ((int32_t)dig_T1_1))) >> 12) * ((int32_t)dig_T3_1)) >> 14;
85        t_fine_1 = var1 + var2;
86        return ((t_fine_1 * 5 + 128) >> 8) / 100.0;
87      } else {
88        var1 = ((((rawTemp >> 3) - ((int32_t)dig_T1_2 << 1))) * ((
    int32_t)dig_T2_2)) >> 11;
89        var2 = (((((rawTemp >> 4) - ((int32_t)dig_T1_2)) * ((rawTemp >>
      4) - ((int32_t)dig_T1_2))) >> 12) * ((int32_t)dig_T3_2)) >> 14;
90        t_fine_2 = var1 + var2;
91        return ((t_fine_2 * 5 + 128) >> 8) / 100.0;
92      }
93    }
94
95    float readPressure(int csbPin, int sensor) {
96      digitalWrite(csbPin, LOW);
97      int32_t rawPress = ((uint32_t)readRegister(csbPin, 0xF7) << 12) |
98                         ((uint32_t)readRegister(csbPin, 0xF8) << 4) |
99                         ((readRegister(csbPin, 0xF9) >> 4));
100     digitalWrite(csbPin, HIGH);
101
102     return rawPress / 256.0; // Ajustar seg n la conversi n del
    sensor espec fico
103   }
104
105   void writeRegister(int csbPin, byte reg, byte value) {
106     digitalWrite(csbPin, LOW);
107     SPI.transfer(reg & 0x7F); // Escribir en el registro
108     SPI.transfer(value);
109     digitalWrite(csbPin, HIGH);
110   }
111
112   uint16_t readRegister16(int csbPin, byte reg) {
113     return (readRegister(csbPin, reg) | (readRegister(csbPin, reg +
    1) << 8));
114   }
115
116   int16_t readRegister16Signed(int csbPin, byte reg) {
117     return (int16_t)readRegister16(csbPin, reg);
118   }
119
120   byte readRegister(int csbPin, byte reg) {
121     digitalWrite(csbPin, LOW);
122     SPI.transfer(reg | 0x80); // Leer del registro
123     byte result = SPI.transfer(0x00);
124     digitalWrite(csbPin, HIGH);
125     return result;
126   }
```

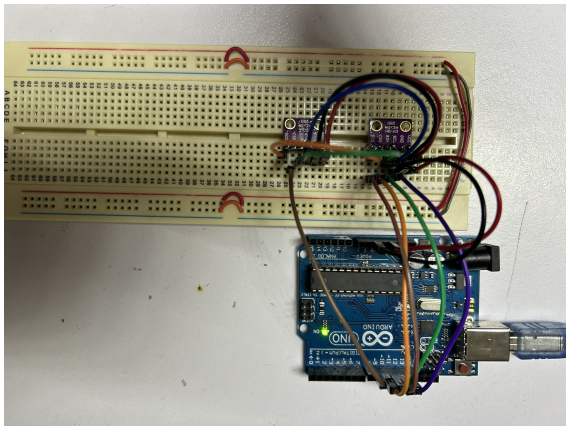

Figure 3: Funcionamiento del cricuito

Primera fotografía donde se puede observar la conexion que se siguió patra el correcto funcionamiento de la practica. Donde se pueden apreciar ambos sensores conectados de manera correcta.

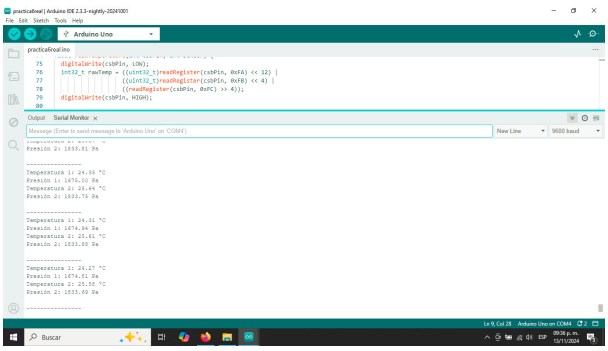

Figure 4: Funcionamiento del cricuito

Segunda fotografía donde se puede observar los datos siendo recibidos de manera correcta por los sensores BME280 en la terminal serial del IDE del arduino,

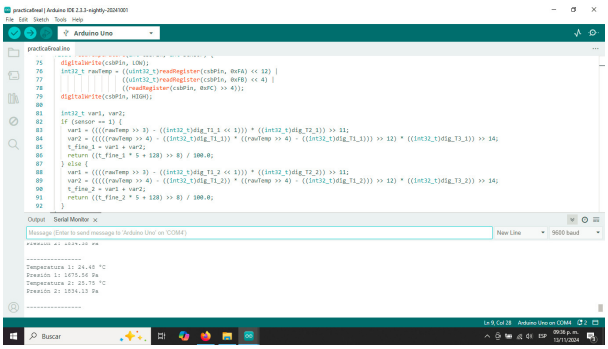

Figure 5: Caption

Ultima fotoggrafía donde se puede apreciar que al cambiar los valores recibidos en los sensores, estos se actualizan de manera correcta mediante la utilización del protocolo SPI en la terminal serial del ArduinoIDE.

3 Conclusiones

- **Morales Rodriguez Iñaki** La integración de un display LCD para mostrar los datos de los sensores BMP280 en tiempo real refuerza la capacidad de monitoreo y retroalimentación del sistema. Los datos de temperatura y presión se presentan de manera clara en el LCD, lo que facilita la interpretación inmediata de las condiciones medidas por los sensores. Además, esta práctica subraya la importancia de la visualización de datos en proyectos de adquisición de información, permitiendo que el usuario pueda verificar el estado de los sensores sin necesidad de un ordenador adicional. En conjunto, la correcta implementación de la comunicación SPI, la gestión de sensores, y la visualización en el LCD brindan una solución práctica y eficiente para sistemas de monitoreo y control en tiempo real.

- **Luciano Hernández Jonathan** Esta práctica demuestra cómo utilizar el protocolo SPI para gestionar múltiples sensores BMP280 mediante un Arduino. Al emplear pines de selección de chip (CS), se habilita la lectura independiente de los sensores, lo que permite que el Arduino actúe como maestro y controle dos sensores BMP280 como esclavos. La correcta configuración de la comunicación SPI, así como la manipulación de los pines CS para activar y desactivar cada sensor, es importante para obtener datos precisos sin interferencias. Esto no solo optimiza el uso de pines del microcontrolador, sino que también garantiza un flujo de datos eficiente para proyectos que requieren la integración de varios sensores, como sistemas de monitoreo ambiental o control de condiciones atmosféricas.

4 Referencias

1 Vaayusastra, "Title of the video," YouTube, Feb. 22, 2022. [Online]. Available: https://shorturl.at/j2RBT [Accessed: Nov. 17, 2024].

2 Wokwi, "Arduino BMP280 sensor SPI example," Wokwi, [Online]. Available: https://wokwi.com/projects/380577657556930561. [Accessed: Nov. 17, 2024].