



Dirección General de EDUCACIÓN TÉCNICA Y FORMACIÓN PROFESIONAL EDUCACIÓN

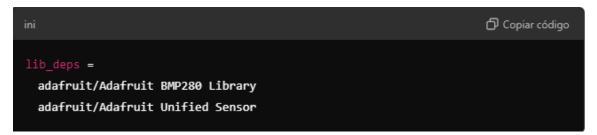


Profesor: Gonzalo Vera

Sensor de presión BMP280

1. Instalar las librerías necesarias en PlatformIO

- Abre PlatformIO y agrega la librería Adafruit BMP280 a tu proyecto. Esto te facilitará la lectura de datos del sensor.
 - En platformio.ini, agrega:



2. Conectar el sensor BMP280 al ESP32

El BMP280 se conecta al ESP32 utilizando los pines I2C:

- SDA (Serial Data Line): Conéctalo al pin 21 del ESP32.
- SCL (Serial Clock Line): Conéctalo al pin 22 del ESP32.
- VCC: Conéctalo al pin 3.3V del ESP32.
- GND: Conéctalo a GND del ESP32.

3. Código para la lectura del sensor

A continuación, te dejo un ejemplo para leer la presión y la temperatura desde el BMP280 y enviarlos a través de la API:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <WiFi.h>
#include <HTTPClient.h>

// Definir las credenciales Wi-Fi
const char* ssid = "Tu_SSID";
const char* password = "Tu_CONTRASEÑA";

// URL de la API
const char* serverName = "http://tu-api-url.com/post-data";
```

Librerías Utilizadas:

- Wire.h: Proporciona funciones para la comunicación I2C, que es cómo el ESP32 se comunica con el BMP280.
- Adafruit_Sensor.h y Adafruit_BMP280.h: Son librerías específicas para interactuar con el sensor BMP280. Nos permiten leer los valores de presión y temperatura.
- WiFi.h: Librería para conectar el ESP32 a una red Wi-Fi.
- HTTPClient.h: Librería que permite realizar solicitudes HTTP (enviar datos a la API RESTful).

Se crea un objeto bmp del tipo Adafruit_BMP280, que es necesario para interactuar con el sensor.

```
// Inicializar el sensor BMP280
Adafruit_BMP280 bmp;
void setup() {
 Serial.begin(115200);
 // Conectar a Wi-Fi
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(1000);
    Serial.println("Conectando a WiFi...");
  }
 Serial.println("Conectado a WiFi");
 // Inicializar BMP280
 if (!bmp.begin(0x76)) { // 0x76 es la dirección I2C predeterminada del BMP280
   Serial.println("Error al inicializar BMP280");
   while (1);
  }
```

- **bmp.readTemperature()**: Lee la temperatura desde el BMP280 en grados Celsius y almacena el valor en la variable temperatura.
- **bmp.readPressure()**: Lee la presión en Pascales. La dividimos por 100 para convertirla a hPa (hectopascales), que es una unidad más común para medir la presión atmosférica.

```
void loop() {
    // Leer presión y temperatura del BMP280
    float temperatura = bmp.readTemperature();
    float presion = bmp.readPressure() / 100.0F; // Convertir a hPa

Serial.print("Temperatura: ");
    Serial.print(temperatura);
    Serial.print(temperatura);
    Serial.print(" °C");

Serial.print("Presión: ");
    Serial.print(presion);
    Serial.println(" hPa");
```

Para enviar datos a la API

```
// Enviar datos a la API
sendData(temperatura, presion);

delay(60000); // Esperar 1 minuto antes de la próxima lectura
}

void sendData(float temp, float press) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverName);
    http.addHeader("Content-Type", "application/json");
```

- ✓ if (WiFi.status() == WL_CONNECTED): Primero, verifica si el ESP32 aún está conectado a la red Wi-Fi.
- ✓ **HTTPClient http**:: Crea un cliente HTTP para gestionar las solicitudes.
- ✓ http.begin(serverName);: Inicia una solicitud HTTP hacia el servidor especificado en serverName.
- ✓ http.addHeader("Content-Type", "application/json");: Agrega el encabezado HTTP para indicar que el contenido que estamos enviando es de tipo JSON.

```
// Formato JSON
String jsonPayload = "{\"sensor\":\"bmp280\",\"temperatura\":" + String(temp) + ",\"pr
int httpResponseCode = http.POST(jsonPayload);

if (httpResponseCode > 0) {
   String response = http.getString();
   Serial.println(httpResponseCode);
   Serial.println(response);
} else {
   Serial.print("Error en la conexión: ");
   Serial.println(httpResponseCode);
}

http.end();
} else {
   Serial.println("Error en la conexión WiFi");
}
```

- **String jsonPayload = ...**: Se crea un string en formato JSON que contiene los datos del sensor (temperatura y presion), listos para ser enviados a la API.
- int httpResponseCode = http.POST(jsonPayload);: Se envía la solicitud HTTP POST con el payload en formato JSON. httpResponseCode almacena el código de respuesta que envía el servidor.
- Si la solicitud es exitosa, se imprime el código de respuesta y el contenido de la respuesta.
- Si hay un error, se imprime un mensaje de error con el código de respuesta.
- http.end();: Cierra la conexión HTTP para liberar recursos.

4. Verificar la conexión I2C

Asegúrate de que el BMP280 esté correctamente conectado usando los pines SDA y SCL correctos. Si tienes problemas, puedes usar un **escáner I2C** para confirmar que el ESP32 detecta el sensor en la dirección correcta (por defecto 0x76 o 0x77).

5. Manejo de errores

- Si el sensor no se inicializa correctamente, muestra un mensaje en el monitor serie.
- Si falla la conexión a la API, maneja los errores y reintenta la conexión.

Este código configura el sensor BMP280, lee los datos de presión y temperatura, y envía esa información a una API RESTful para su procesamiento o almacenamiento.