Documentación Técnica: Monitor de calidad de aire FIBA

Integración de
BME688, SPS30, TFT ST7789 y Firebase usando Python

Proyecto *FIBA*

12 de diciembre de 2025

Índice

1. Introducción

El presente documento detalla la implementación del firmware para el **Monitor Ambiental FIBA**. El sistema está diseñado para medir variables ambientales clave como temperatura, humedad (convertida a estimación de CO2) y material particulado (PM2.5 y PM10), visualizarlas en una pantalla TFT y transmitirlas en tiempo real a una base de datos en la nube (Firebase).

2. Hardware y Conexiones

El sistema utiliza un microcontrolador ESP32 como núcleo de procesamiento. A continuación se detallan las conexiones de periféricos.

Cuadro 1: Mapa de Conexiones (Pinout)

Componente	**Pines ESP32 / Protocolo**
SPS30 (MP)	I2C: SDA (21), SCL (22)
BME688 (Gas/Temp)	I2C: SDA (21), SCL (22) (Addr: 0x76/0x77)
TFT Display ST7789	SPI: MOSI (23), SCLK (18), CS (5), DC (2), RST (15)
Buzzer	GPIO 4 (Salida Digital)

3. Arquitectura de Software

3.1. Librerías Principales

Se utilizan las siguientes librerías para el manejo de hardware y servicios:

- `Adafruit_GFX` y `Adafruit_ST7789`: Control gráfico de pantalla.
- `SensirionI2cSps30`: Control del sensor de partículas SPS30.
- `Adafruit_BME680`: Control del sensor ambiental BME688.
- `WiFi.h`: Gestión de conexión inalámbrica.
- `Firebase_ESP_Client`: Comunicación con Firebase RTDB.
- `time.h`: Sincronización de hora vía NTP.

3.2. Ciclo Principal (Loop)

El `loop()` está diseñado de manera *no bloqueante* utilizando temporizadores basados en `millis()`.

- **Lectura BME688**: Se ejecuta cada 3 segundos. Lee temperatura, presión y resistencia al gas. Calcula el CO2 estimado.
- **Lectura SPS30**: Se ejecuta tan rápido como el sensor tenga datos disponibles ("Data Ready Flag"). Lee PM2.5 y PM10.
- **Envío a Nube**: Se ejecuta cada 60 segundos. Empaqueta los últimos valores leídos en un JSON y los envía a Firebase.

4. Algoritmia Clave

4.1. Estimación de CO2

Dada la complejidad de la librería BSEC, se implementó una estimación lineal simplificada basada en la resistencia del gas (en Ohmios). Cuanto mayor es la resistencia, más limpio es el aire (menos VOCs).

$$CO2_{ppm} = 400 + (-0{,}0355 \times (Gas_{Ohms} - 50000)) \quad (1)$$

Nota: Se asume una línea base de 50kΩ para aire limpio (400ppm) y 5kΩ para aire viciado (2000ppm).

4.2. Clasificación de Calidad de Aire (IBOCA)

Se utiliza una escala de colores inspirada en IBOCA para clasificar visualmente los contaminantes:

- **BAJO (Verde)**
- **MODERADO (Amarillo)**
- **REGULAR**
- **ALTO (Rojo)**
- **PELIGROSO**

5. Integración IoT

5.1. Sincronización de Hora (NTP)

El dispositivo se conecta a `pool.ntp.org` para obtener la hora UTC y aplica un offset de -18000 segundos (-5 horas) para sincronizar con la hora local de Colombia.

5.2. Firebase Realtime Database

Los datos se estructuran en formato JSON y se envían al nodo `/medidas`. Estructura del paquete:

```
{
  "timestamp": "YYYY-MM-DD HH:MM:SS",
  "temperature": 25.4,
  "gas_kohm": 45.2,
  "co2_ppm": 550,
  "pm25": 12.0,
  "pm10": 18.0
}
```

6. Configuración Inicial

En la función `setup()`, se realiza una secuencia de diagnóstico visual:

1. Conexión WiFi.
2. Inicialización de pantalla.
3. Verificación de sensores (Muestra **BME680 OK** y **SPS30 OK** en pantalla).

7. Sistema de Big Data (Python)

Como complemento al firmware, se desarrolló un script en Python (`BIGDATAFIBA.py`) para la descarga y almacenamiento persistente de los datos desde la nube.

7.1. Funcionalidad

El script se ejecuta independientemente del ESP32 (en un PC o servidor) y realiza las siguientes tareas:

1. Se autentica en Firebase utilizando una llave de servicio privada (.json).
2. Escucha cambios en el nodo `/medidas` o consulta periódicamente (Polling) cada 60 segundos.
3. Convierte los paquetes JSON recibidos en registros estructurados.
4. Almacena la información en un archivo Excel local llamado `fiba_data.xlsx`.

7.2. Estructura del Script

El código utiliza las librerías `firebase_admin` para la conexión segura y `pandas` para la manipulación de datos y exportación a Excel.

```
# Fragmento del ciclo principal
while True:
    try:
        # Obtener datos de Firebase
        ref = db.reference('/medidas')
        data = ref.get()

        # Convertir a DataFrame y guardar
        df = pd.DataFrame(data_list)
        df.to_excel(OUTPUT_FILE, index=False)

    except Exception as e:
        print(f"Error: {e}")

    time.sleep(60)
```