ESP32-S3 + Firebase Environmental Monitoring System

This project demonstrates how to build a real-time environmental monitoring system using the ESP32-S3 microcontroller and Firebase Realtime Database. The system continuously measures temperature, humidity, air pressure, and air quality using three sensors: **DHT22**, **BMP180 (GY-68)**, and **MQ-135**. These sensors are connected to the ESP32-S3, which reads the data every 5 seconds and uploads it to the cloud via Firebase, enabling remote access to live sensor data for dashboards, analysis, or AI prediction.

The ESP32-S3 is the central component of the system, acting as the main processing unit with built-in Wi-Fi capabilities. It connects to the internet using the provided Wi-Fi credentials and handles all communication with the Firebase Realtime Database. Firebase authentication is managed using an email and password, and the Firebase ESP Client library is used to interact with the database. Additionally, the ESP32 synchronizes time using an NTP (Network Time Protocol) server, so that each reading is accurately timestamped.

The system uses the **DHT22 sensor** to measure temperature and humidity. This sensor outputs digital data and is connected to **GPIO 4** on the ESP32. For air quality monitoring, the **MQ-135 sensor** is used. It outputs an analog voltage corresponding to the concentration of gases like CO_2 and is connected to **GPIO 5**. The ESP32 processes this voltage to estimate the CO_2 level using a conversion formula. The **BMP180 (GY-68)** sensor is used to measure atmospheric pressure and communicates with the ESP32 via I^2C, with **SDA connected to GPIO 8** and **SCL to GPIO 9**. All components share a common ground, and power is supplied through 3.3V or 5V pins depending on the module's requirements.

After collecting data from all sensors, the ESP32 formats the values and uploads them to Firebase under a fixed path: /sensor_data/latest. The uploaded dataset includes temperature, humidity, pressure, CO_2 estimate, and the current date-time string (e.g., 2025-06-26 15:45:10). With each new reading, the previous values are overwritten, keeping only the latest environmental snapshot.

To make the code clean and maintainable, it is divided into multiple modular header files. Each file handles a specific task: Wi-Fi connection (wifi_setup.h), Firebase setup (firebase_setup.h), time synchronization (time_sync.h), sensor initialization (sensor_setup.h), sensor

data reading (sensor_reading.h), and Firebase uploading (upload_firebase.h). The main sketch file (SensorUploader.ino) ties everything together using the setup() and loop() functions.

This IoT system is fully extendable. A backend service, such as one built with **FastAPI**, can retrieve data from Firebase to perform advanced processing or predictive modeling using AI, such as **LSTM neural networks**. On the frontend side, frameworks like **React** can be used to build a dashboard that visualizes real-time sensor data for end users.

In conclusion, this project offers a scalable, modular, and cloud-connected IoT solution for environmental monitoring. It combines reliable hardware, real-time cloud storage, and clean software architecture—making it an ideal foundation for future applications involving automation, analytics, or artificial intelligence.