File S1. Arduino Code for Air Quality Logger

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

// PMS5003 pins (UART)
#define RXD2 16  // PMS5003 TX → ESP32 RX
#define TXD2 17  // PMS5003 RX ← ESP32 TX

// MH-Z14A CO₂ sensor (PWM)
#define CO2_PWM_PIN 34  // MH-Z14A PWM output pin

// Onboard LED (optional heartbeat indicator)
#define LED_PIN 2

// SD card chip select
#define SD_CS 5

File logFile;
unsigned long lastLogTime = 0;
float lastCO2 = -1;
int lastPM1 = -1, lastPM25 = -1, lastPM10 = -1;

// Format milliseconds into human-readable time string
String formatElapsedTime(unsigned long millisElapsed) {
    unsigned long totalSeconds = millisElapsed / 1000;
    unsigned int days = totalSeconds / 86400;
    unsigned int hours = (totalSeconds % 86400) / 3600;
    unsigned int minutes = (totalSeconds % 3600) / 60;
    unsigned int seconds = totalSeconds % 60;
    char timeStr[20];
    snprintf(timeStr, sizeof(timeStr), "%dd → %02u:%02u:%02u", days,
hours, minutes, seconds);
    return String(timeStr);
}

void setup() {
    Serial.begin(115200);
    Serial.println("Initializing PMS5003, MH-Z14A, SD card, and
LED...");

    pinMode(LED_PIN, OUTPUT);
    pinMode(CO2_PWM_PIN, INPUT);
    Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);  // Initialize
PMS5003 on Serial2

    delay(3000);  // Allow sensors to warm up

    if (!SD.begin(SD_CS)) {
        Serial.println("SD card initialization failed!");
```

```
    } else {
        Serial.println("SD card initialized.");
        logFile = SD.open("/airlog.csv", FILE_APPEND);
        if (logFile) {
            logFile.println("=== ESP32 restarted ===");
            logFile.close();
        }
    }
}

// Read CO₂ concentration from MH-Z14A via PWM
float readCO2PWM() {
    while (digitalRead(CO2_PWM_PIN) == LOW);
    unsigned long highStart = micros();
    while (digitalRead(CO2_PWM_PIN) == HIGH);
    unsigned long highEnd = micros();

    float Th_ms = (highEnd - highStart) / 1000.0; // High pulse width
in ms
    const float T_ms = 1004.0; // Total period in ms
    return 5000.0 * (Th_ms - 2.0) / (T_ms - 4.0); // CO₂ formula per
sensor specs
}

// Read particulate matter data from PMS5003
bool readPMSData(int &pm1_0, int &pm2_5, int &pm10) {
    if (Serial2.available() >= 32) {
        uint8_t buffer[32];
        Serial2.readBytes(buffer, 32);
        while (Serial2.available()) Serial2.read(); // Clear remaining
bytes

        if (buffer[0] == 0x42 && buffer[1] == 0x4D) { // Valid PMS5003
header
            pm1_0 = (buffer[10] << 8) | buffer[11];
            pm2_5 = (buffer[12] << 8) | buffer[13];
            pm10  = (buffer[14] << 8) | buffer[15];
            return true;
        }
    }
    return false;
}

// Log data to SD card and serial monitor
void logData(float co2, int pm1, int pm25, int pm10, unsigned long
timeMs) {
    String timeStr = formatElapsedTime(timeMs);

    Serial.println(timeStr);
    Serial.print("CO₂: "); Serial.print(co2); Serial.println(" ppm");
```

```
    Serial.print("PM1.0: "); Serial.print(pm1);
    Serial.print(" | PM2.5: "); Serial.print(pm25);
    Serial.print(" | PM10: "); Serial.println(pm10);

    logFile = SD.open("/airlog.csv", FILE_APPEND);
    if (logFile) {
        if (logFile.size() == 0) {
            logFile.println("Time,CO2 (ppm),PM1.0,PM2.5,PM10");
        }
        logFile.print(timeStr); logFile.print(",");
        logFile.print(co2 >= 0 ? String(co2, 2) : "NA");
logFile.print(",");
        logFile.print(pm1 >= 0 ? String(pm1) : "NA");
logFile.print(",");
        logFile.print(pm25 >= 0 ? String(pm25) : "NA");
logFile.print(",");
        logFile.println(pm10 >= 0 ? String(pm10) : "NA");
        logFile.close();
        Serial.println("Data saved.\n");
    } else {
        Serial.println("Failed to write to SD card.");
    }
}

void loop() {
    int pm1 = -1, pm25 = -1, pm10 = -1;
    float co2 = -1;

    bool pmValid = readPMSData(pm1, pm25, pm10);
    co2 = readCO2PWM();

    lastPM1 = pmValid ? pm1 : -1;
    lastPM25 = pmValid ? pm25 : -1;
    lastPM10 = pmValid ? pm10 : -1;
    lastCO2 = (co2 > 0) ? co2 : -1;

    digitalWrite(LED_PIN, HIGH);
    delay(50);
    digitalWrite(LED_PIN, LOW);

    if (millis() - lastLogTime >= 60000) {  // Log every 60 seconds
        logData(lastCO2, lastPM1, lastPM25, lastPM10, millis());
        lastLogTime = millis();
    }
}
```