#include <DHT.h>

#include <Adafruit\_MPU6050.h>

#include <Adafruit\_Sensor.h>

#include <TinyGPS++.h>

#include <HardwareSerial.h>

#include <Wire.h>

#include <Adafruit\_BMP280.h>

#include <WiFi.h>

#include <Firebase\_ESP\_Client.h>

#include "addons/TokenHelper.h"

#include "addons/RTDBHelper.h"

// WiFi & Firebase Config

#define WIFI\_SSID "aaaaaa"

#define WIFI\_PASSWORD "123456781"

#define API\_KEY "AIzaSyCvF1XYU5B3sv09QGiRVRsk7tsPvcc4EYw"

#define DATABASE\_URL "https://home-automation-6f259-default-rtdb.firebaseio.com"

// DHT11 Settings

#define DHTPIN 4

#define DHTTYPE DHT11

const int TEMP\_SAMPLES = 5;

const int TEMP\_READ\_DELAY = 200;

const float TEMP\_OFFSET = 1.5;

const float HUM\_OFFSET = 0.0;

const float MAX\_TEMP\_CHANGE = 2.0;

// Voltage Monitoring

#define VOLTAGE\_IN\_PIN 34

#define VOLTAGE\_OUT\_PIN 35

#define DIVIDER\_IN\_RATIO 3.0

#define DIVIDER\_OUT\_RATIO 2.0

// Sensor Objects

DHT dht(DHTPIN, DHTTYPE);

Adafruit\_MPU6050 mpu;

Adafruit\_BMP280 bmp;

HardwareSerial SerialGPS(2); // UART2: RX=16, TX=17

TinyGPSPlus gps;

// Firebase Objects

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

bool signupOK = false;

// Global Variables

float lastValidTemp = 25.0;

float lastValidHum = 50.0;

float accelOffsets[3] = {0, 0, 0};

unsigned long lastSensorUpdate = 0;

bool sensorStable = false;

void setup() {

Serial.begin(115200);

// Initialize DHT11

dht.begin();

Serial.println("DHT11 Initialized (High Accuracy Mode)");

// Initialize MPU6050

if (!mpu.begin()) {

Serial.println("Failed to find MPU6050!");

while (1);

}

mpu.setAccelerometerRange(MPU6050\_RANGE\_8\_G);

mpu.setGyroRange(MPU6050\_RANGE\_500\_DEG);

mpu.setFilterBandwidth(MPU6050\_BAND\_21\_HZ);

Serial.println("MPU6050 Initialized");

// Initialize BMP280

if (!bmp.begin(0x76)) {

Serial.println("Could not find a valid BMP280 sensor!");

while (1);

}

bmp.setSampling(Adafruit\_BMP280::MODE\_NORMAL,

Adafruit\_BMP280::SAMPLING\_X2,

Adafruit\_BMP280::SAMPLING\_X16,

Adafruit\_BMP280::FILTER\_X16,

Adafruit\_BMP280::STANDBY\_MS\_500);

Serial.println("BMP280 Initialized");

// Initialize GPS

SerialGPS.begin(9600, SERIAL\_8N1, 16, 17);

Serial.println("GPS Initialized - Waiting for fix...");

// Initialize voltage monitoring

pinMode(VOLTAGE\_IN\_PIN, INPUT);

pinMode(VOLTAGE\_OUT\_PIN, INPUT);

Serial.println("Voltage Monitoring Ready");

// Connect to WiFi

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

Serial.print("Connecting to WiFi");

while (WiFi.status() != WL\_CONNECTED) {

Serial.print(".");

delay(300);

}

Serial.println("\nWiFi Connected");

// Initialize Firebase

config.api\_key = API\_KEY;

config.database\_url = DATABASE\_URL;

fbdo.setResponseSize(2048);

if (Firebase.signUp(&config, &auth, "", "")) {

signupOK = true;

Serial.println("Firebase signup OK");

}

config.token\_status\_callback = tokenStatusCallback;

Firebase.begin(&config, &auth);

Firebase.reconnectWiFi(true);

// Calibrate sensors

calibrateMPU();

calibrateDHT();

delay(2000); // Initial stabilization

}

void loop() {

unsigned long currentMillis = millis();

// Process GPS data continuously

while (SerialGPS.available() > 0) {

gps.encode(SerialGPS.read());

}

// Read all sensors every 2 seconds and send to Firebase

if (currentMillis - lastSensorUpdate >= 2000) {

lastSensorUpdate = currentMillis;

// Read DHT11

float currentTemp = getStableTemperature();

float currentHum = getStableHumidity();

// Apply corrections if stable

if (sensorStable) {

currentTemp += TEMP\_OFFSET;

currentHum += HUM\_OFFSET;

}

// Validate and update DHT readings

if (!isnan(currentTemp) && !isnan(currentHum)) {

if (abs(currentTemp - lastValidTemp) > MAX\_TEMP\_CHANGE) {

float verifyTemp = getStableTemperature();

if (!isnan(verifyTemp) && abs(verifyTemp - currentTemp) < 1.0) {

lastValidTemp = currentTemp;

} else {

currentTemp = lastValidTemp;

}

}

lastValidTemp = currentTemp;

lastValidHum = currentHum;

sensorStable = true;

}

// Read BMP280

float bmpTemp = bmp.readTemperature();

float pressure = bmp.readPressure() / 100.0; // Convert Pa to hPa

float altitude = bmp.readAltitude(1013.25);

// Read MPU6050

sensors\_event\_t a, g, temp;

mpu.getEvent(&a, &g, &temp);

float ax = a.acceleration.x - accelOffsets[0];

float ay = a.acceleration.y - accelOffsets[1];

float az = a.acceleration.z - accelOffsets[2];

// Read GPS

float latitude = gps.location.isValid() ? gps.location.lat() : 0;

float longitude = gps.location.isValid() ? gps.location.lng() : 0;

float gpsAltitude = gps.altitude.isValid() ? gps.altitude.meters() : 0;

int satellites = gps.satellites.isValid() ? gps.satellites.value() : 0;

float hdop = gps.hdop.isValid() ? gps.hdop.value() / 100.0 : 0;

// Read voltages

float inputVoltage = readVoltage(VOLTAGE\_IN\_PIN, DIVIDER\_IN\_RATIO);

float outputVoltage = readVoltage(VOLTAGE\_OUT\_PIN, DIVIDER\_OUT\_RATIO);

// Send all data to Firebase under "cansat" node

if (Firebase.ready()) {

FirebaseJson json;

// DHT11 data

json.set("dht/temperature", currentTemp);

json.set("dht/humidity", currentHum);

// BMP280 data

json.set("bmp/temperature", bmpTemp);

json.set("bmp/pressure", pressure);

json.set("bmp/altitude", altitude);

// MPU6050 data

json.set("mpu/accel\_x", ax);

json.set("mpu/accel\_y", ay);

json.set("mpu/accel\_z", az);

json.set("mpu/gyro\_x", g.gyro.x);

json.set("mpu/gyro\_y", g.gyro.y);

json.set("mpu/gyro\_z", g.gyro.z);

// GPS data

json.set("gps/latitude", latitude);

json.set("gps/longitude", longitude);

json.set("gps/altitude", gpsAltitude);

json.set("gps/satellites", satellites);

json.set("gps/hdop", hdop);

// Power data

json.set("power/input\_voltage", inputVoltage);

json.set("power/output\_voltage", outputVoltage);

// Send to Firebase

if (Firebase.RTDB.setJSON(&fbdo, "cansat", &json)) {

Serial.println("Data sent to Firebase successfully");

} else {

Serial.println("Failed to send data to Firebase: " + fbdo.errorReason());

}

}

// Print to serial for debugging

printSensorData(currentTemp, currentHum, bmpTemp, pressure, altitude,

ax, ay, az, g.gyro.x, g.gyro.y, g.gyro.z,

latitude, longitude, gpsAltitude, satellites, hdop,

inputVoltage, outputVoltage);

}

// Check Firebase token

if (Firebase.isTokenExpired()) {

Firebase.refreshToken(&config);

Serial.println("Token refreshed");

}

}

void printSensorData(float dhtTemp, float dhtHum, float bmpTemp, float pressure, float altitude,

float ax, float ay, float az, float gx, float gy, float gz,

float lat, float lon, float gpsAlt, int sats, float hdop,

float vin, float vout) {

Serial.println("\n--- Sensor Data ---");

Serial.print("DHT11 - Temperature: ");

Serial.print(dhtTemp, 1);

Serial.print("°C | Humidity: ");

Serial.print(dhtHum, 1);

Serial.println("%");

Serial.print("BMP280 - Temperature: ");

Serial.print(bmpTemp, 1);

Serial.print("°C | Pressure: ");

Serial.print(pressure, 1);

Serial.print(" hPa | Altitude: ");

Serial.print(altitude, 1);

Serial.println(" m");

Serial.print("MPU6050 - Accel: X=");

Serial.print(ax, 2);

Serial.print(" Y=");

Serial.print(ay, 2);

Serial.print(" Z=");

Serial.print(az, 2);

Serial.print(" m/s² | Gyro: X=");

Serial.print(gx, 2);

Serial.print(" Y=");

Serial.print(gy, 2);

Serial.print(" Z=");

Serial.print(gz, 2);

Serial.println(" rad/s");

if (lat != 0 && lon != 0) {

Serial.print("GPS - Lat: ");

Serial.print(lat, 6);

Serial.print(" Lon: ");

Serial.print(lon, 6);

Serial.print(" Alt: ");

Serial.print(gpsAlt, 1);

Serial.print(" m | Sats: ");

Serial.print(sats);

Serial.print(" | HDOP: ");

Serial.println(hdop, 2);

} else {

Serial.println("GPS - No fix");

}

Serial.print("Power - Input: ");

Serial.print(vin, 1);

Serial.print(" V | Output: ");

Serial.print(vout, 1);

Serial.println(" V");

Serial.println("-------------------");

}

// [Rest of your existing functions remain unchanged...]

// Improved DHT11 Functions

float getStableTemperature() {

float samples[TEMP\_SAMPLES];

int validSamples = 0;

for (int i = 0; i < TEMP\_SAMPLES; i++) {

float temp = dht.readTemperature();

if (!isnan(temp) && temp >= -10 && temp <= 60) {

samples[validSamples] = temp;

validSamples++;

}

delay(TEMP\_READ\_DELAY);

}

if (validSamples > 0) {

sortSamples(samples, validSamples);

return samples[validSamples / 2];

}

return NAN;

}

float getStableHumidity() {

float sum = 0;

int validSamples = 0;

for (int i = 0; i < TEMP\_SAMPLES; i++) {

float hum = dht.readHumidity();

if (!isnan(hum) && hum >= 0 && hum <= 100) {

sum += hum;

validSamples++;

}

delay(TEMP\_READ\_DELAY);

}

return (validSamples > 0) ? sum / validSamples : NAN;

}

void sortSamples(float arr[], int n) {

for (int i = 0; i < n-1; i++) {

for (int j = 0; j < n-i-1; j++) {

if (arr[j] > arr[j+1]) {

float temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

void calibrateDHT() {

Serial.println("Calibrating DHT11...");

float tempSum = 0;

float humSum = 0;

int validReadings = 0;

for (int i = 0; i < 10; i++) {

float temp = dht.readTemperature();

float hum = dht.readHumidity();

if (!isnan(temp) && !isnan(hum)) {

tempSum += temp;

humSum += hum;

validReadings++;

}

delay(300);

}

if (validReadings > 5) {

lastValidTemp = tempSum / validReadings;

lastValidHum = humSum / validReadings;

}

}

void calibrateMPU() {

Serial.println("Calibrating MPU6050... Keep sensor flat and still");

float axSum = 0, aySum = 0, azSum = 0;

for (int i = 0; i < 500; i++) {

sensors\_event\_t a, g, temp;

mpu.getEvent(&a, &g, &temp);

axSum += a.acceleration.x;

aySum += a.acceleration.y;

azSum += a.acceleration.z - 9.81;

delay(5);

}

accelOffsets[0] = axSum / 500;

accelOffsets[1] = aySum / 500;

accelOffsets[2] = azSum / 500;

}

float readVoltage(int pin, float ratio) {

int raw = analogRead(pin);

return (raw \* 3.3 / 4095) \* ratio;

}