```
#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#include <DHT.h>

#include <SimpleKalmanFilter.h>


// Sensor Definitions

#define DHTPIN D6

#define DHTTYPE DHT11

#define MQ2_PIN A0

#define PIR_PIN D5

#define TRIG_PIN D7

#define ECHO_PIN D8

#define BUZZER_PIN D3

#define RELAY_PIN D4


// Gas Thresholds (customizable)

#define GAS_WARNING_THRESHOLD 280  // Red alert if above this

#define GAS_MODERATE_THRESHOLD 150 // Yellow alert if between 150-280


// OLED Display

#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);


// DHT Sensor

DHT dht(DHTPIN, DHTTYPE);


// Global variable declarations
```

```
SimpleKalmanFilter gasKalman(10, 10, 0.5);  // Initialize Kalman filter
float filteredGasValue;  // Declare the variable to store filtered value

// WiFi Credentials
const char* ssid = "Bhanu";
const char* password = "Letmein1001";

ESP8266WebServer server(80);

// Variables
float temperature = 0;
float humidity = 0;
int gasValue = 0;
bool motionDetected = false;
float distance = 0;
bool relayState = false;
bool buzzerState = false;
bool buzzerEnabled = true;
unsigned long lastSensorUpdate = 0;
unsigned long lastBuzzerTime = 0;
const long sensorInterval = 1000; // Update sensors every 1 second
const long buzzerDuration = 3000; // Buzzer duration in ms (3 seconds)

// Function declarations
void handleRoot();
void handleRelay();
void handleBuzzer();
void handleSensorData();
void handleNotFound();
void readSensors();
void checkAlarms();
void updateDisplay();

void setup() {
  Serial.begin(115200);
  Serial.println("\nStarting Home Automation System...");

  // Initialize DHT sensor FIRST with a delay
  dht.begin();
  delay(2000);  // Crucial delay for DHT sensor stabilization

  // Initialize OLED
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("OLED allocation failed"));
    for(;;);
  }
  display.display();
  delay(2000);
  display.clearDisplay();

  // Initialize pins
  pinMode(PIR_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
  digitalWrite(BUZZER_PIN, LOW);
  Serial.println("Pins initialized");

  // Connect to WiFi
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  // Web Server Routes
  server.on("/", handleRoot);
  server.on("/relay", handleRelay);
  server.on("/buzzer", handleBuzzer);
  server.on("/sensorData", handleSensorData);
  server.onNotFound(handleNotFound);

  server.begin();
  Serial.println("HTTP server started");

  // Initial sensor read
  readSensors();
  updateDisplay();
}

void loop() {
  server.handleClient();

  // Update sensors at regular intervals
  if (millis() - lastSensorUpdate >= sensorInterval) {
    readSensors();
    checkAlarms();
    updateDisplay();
    lastSensorUpdate = millis();
  }

  // Turn off buzzer after duration
  if (buzzerState && (millis() - lastBuzzerTime >= buzzerDuration)) {
    digitalWrite(BUZZER_PIN, LOW);
    buzzerState = false;
    Serial.println("Buzzer timeout - turned OFF");
  }
}

void readSensors() {
  // Add delay between readings (DHT22 needs at least 2 seconds)
  static unsigned long lastDHTRead = 0;
  if (millis() - lastDHTRead >= 2000) {  // Only read every 2 seconds
    lastDHTRead = millis();

    // Reading temperature or humidity takes about 250ms
    float newTemp = dht.readTemperature();
    float newHum = dht.readHumidity();

    // Check if any reads failed
    if (isnan(newTemp) || isnan(newHum)) {
      Serial.println("Failed to read from DHT sensor!");
      temperature = NAN;
      humidity = NAN;
    } else {
      temperature = newTemp;
      humidity = newHum;
      Serial.print("Temp: "); Serial.print(temperature);
    Serial.print("°C, Hum: "); Serial.print(humidity); Serial.println("%");
   }
  }

  // Read MQ2
   int rawGas = analogRead(MQ2_PIN);
  filteredGasValue = gasKalman.updateEstimate(rawGas);
  gasValue = filteredGasValue;

  // Read PIR
  motionDetected = digitalRead(PIR_PIN);

  // Read Ultrasonic
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;
}

void checkAlarms() {
  // Check all alarm conditions
  bool shouldBuzz = false;

  if (gasValue > 280) {
    Serial.println("High gas level detected!");
    shouldBuzz = true;
  }
  if (motionDetected) {
    Serial.println("Motion detected!");
    shouldBuzz = true;
  }

  if (distance < 20 && distance > 0) {
    Serial.println("Object too close!");
    shouldBuzz = true;
  }

  // Activate buzzer if needed and enabled
  if (shouldBuzz && buzzerEnabled && !buzzerState) {
    digitalWrite(BUZZER_PIN, HIGH);
    buzzerState = true;
    lastBuzzerTime = millis();
    Serial.println("Buzzer activated");
  }
}

void updateDisplay() {
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("Home Automation");

  // Temperature
  display.setCursor(0, 16);
  display.print("Temp: ");
  if (isnan(temperature)) {
```

```
  display.print("Error");
} else {
  display.print(temperature, 1);
  display.print(" C");
}

// Humidity
display.setCursor(0, 26);
display.print("Hum: ");
if (isnan(humidity)) {
  display.print("Error");
} else {
  display.print(humidity, 1);
  display.print(" %");
}

// Gas Level
display.setCursor(0, 36);
display.print("Gas: ");
display.print(gasValue);
if(gasValue > 250) {
  display.print(" !");
}

// Motion
display.setCursor(0, 46);
display.print("Motion: ");
display.println(motionDetected ? "YES" : "NO");

// Distance
display.setCursor(0, 56);
```

```
  display.print("Dist: ");
  display.print(distance, 1);
  display.println(" cm");

  display.display();
}

void handleRoot() {
  String html = R"=====(
<!DOCTYPE html>
<html>
<head>
<title>Home Automation System</title>
<meta name='viewport' content='width=device-width, initial-scale=1.0'>
<link
href='https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600&display=swap'
rel='stylesheet'>
<style>
body { font-family: 'Poppins', sans-serif; background-color: #f8fafc; margin: 0; padding: 0; color:
#1e293b; }
.container { max-width: 1200px; margin: 0 auto; padding: 20px; }
h1 { color: #0f172a; text-align: center; margin-bottom: 30px; font-weight: 600; }
.dashboard { display: grid; grid-template-columns: repeat(auto-fit, minmax(300px, 1fr)); gap: 25px; }
.card { background-color: white; border-radius: 12px; padding: 20px; box-shadow: 0 4px 6px
rgba(0,0,0,0.05); transition: all 0.3s ease; border-top: 4px solid; }
.card:hover { box-shadow: 0 10px 15px rgba(0,0,0,0.1); transform: translateY(-5px); }
.card.temperature { border-top-color: #ef4444; }
.card.humidity { border-top-color: #3b82f6; }
.card.gas { border-top-color: #f59e0b; }
.card.motion { border-top-color: #10b981; }
.card.distance { border-top-color: #8b5cf6; }
.card.relay { border-top-color: #ec4899; }
```

```
.card.buzzer { border-top-color: #6366f1; }
.sensor-value { font-size: 28px; font-weight: 600; text-align: center; margin: 10px 0; }
.sensor-label { font-size: 16px; color: #64748b; text-align: center; }
.status { display: flex; align-items: center; justify-content: center; margin: 15px 0; }
.status-indicator { width: 14px; height: 14px; border-radius: 50%; margin-right: 10px; }
.status-on { background-color: #10b981; }
.status-off { background-color: #ef4444; }
.btn-group { display: flex; justify-content: center; gap: 10px; margin-top: 20px; }
.btn { padding: 10px 20px; border-radius: 8px; text-decoration: none; font-weight: 500; transition: all
0.2s; border: none; cursor: pointer; }
.btn-primary { background-color: #3b82f6; color: white; }
.btn-primary:hover { background-color: #2563eb; }
.btn-success { background-color: #10b981; color: white; }
.btn-success:hover { background-color: #059669; }
.btn-danger { background-color: #ef4444; color: white; }
.btn-danger:hover { background-color: #dc2626; }
.btn-warning { background-color: #f59e0b; color: white; }
.btn-warning:hover { background-color: #d97706; }
.alert { padding: 12px; border-radius: 8px; margin: 15px 0; text-align: center; font-weight: 500; }
.alert-warning { background-color: #fef3c7; color: #92400e; }
.alert-danger { background-color: #fee2e2; color: #b91c1c; }
.alert-info { background-color: #dbeafe; color: #1e40af; }
@media (max-width: 768px) { .dashboard { grid-template-columns: 1fr; } }
</style>
</head>
<body>
<div class='container'>
<h1>Home Automation Dashboard</h1>
<div class='dashboard'>
<div class='card temperature'>
<h2>Temperature</h2>
```

```
<div class='sensor-value' id='temp-value'>)=====";
  html += isnan(temperature) ? "Error" : String(temperature, 1) + "°C";
  html += R"=====(</div>
<div class='sensor-label'>Current Temperature</div>
</div>
<div class='card humidity'>
<h2>Humidity</h2>
<div class='sensor-value' id='hum-value'>)=====";
  html += isnan(humidity) ? "Error" : String(humidity, 1) + "%";
  html += R"=====(</div>
<div class='sensor-label'>Current Humidity</div>
</div>
<div class='card gas'>
<h2>Air Quality</h2>
<div class='sensor-value' id='gas-value'>)=====";
  html += String(gasValue);
  html += R"=====(</div>
<div class='sensor-label'>Gas Level (0-1023)</div>
<div class='alert' id='gas-alert'>)=====";
  if(gasValue > 280) {
  html += "<div class='alert alert-danger'>⚠ WARNING: High Gas Level!</div>";
}
else if(gasValue > 150) {
  html += "<div class='alert alert-warning'>Moderate Gas Level</div>";
}
else {
  html += "<div class='alert alert-info'>Air Quality Normal</div>";
}
  html += R"=====(</div>
</div>
<div class='card motion'>
```

```
<h2>Motion Detection</h2>
<div class='status'>
<div class='status-indicator' id='motion-indicator'>)=====";
  html += motionDetected ? "status-on" : "status-off";
  html += R"=====(</div>
<span id='motion-status'>)=====";
  html += motionDetected ? "Motion Detected" : "No Motion";
  html += R"=====(</span>
</div>)=====";
  if(motionDetected) {
    html += "<div class='alert alert-warning' id='motion-alert'>Movement detected in the area!</div>";
  }
  html += R"=====(
</div>
<div class='card distance'>
<h2>Distance Sensor</h2>
<div class='sensor-value' id='dist-value'>)=====";
  html += String(distance, 1);
  html += R"=====( cm</div>
<div class='sensor-label'>Object Distance</div>)=====";
  if(distance < 20 && distance > 0) {
    html += "<div class='alert alert-warning' id='distance-alert'>Object too close!</div>";
  }
  html += R"=====(
</div>
<div class='card relay'>
<h2>Relay Control</h2>
<div class='status'>
<div class='status-indicator' id='relay-indicator'>)=====";
  html += relayState ? "status-on" : "status-off";
  html += R"=====(</div>
<span id='relay-status'>)=====";
  html += relayState ? "Relay: ON" : "Relay: OFF";
  html += R"=====(</span>
</div>
<div class='btn-group'>
<button onclick="controlDevice('relay', 'on')" class='btn' id='relay-on-btn'>)=====";
  html += relayState ? "btn-success" : "btn-primary";
  html += R"=====(">Turn ON</button>
<button onclick="controlDevice('relay', 'off')" class='btn' id='relay-off-btn'>)=====";
  html += !relayState ? "btn-danger" : "btn-primary";
  html += R"=====(">Turn OFF</button>
</div>
</div>
<div class='card buzzer'>
<h2>Buzzer Control</h2>
<div class='status'>
<div class='status-indicator' id='buzzer-state-indicator'>)=====";
  html += buzzerState ? "status-on" : "status-off";
  html += R"=====(</div>
<span id='buzzer-status'>)=====";
  html += buzzerState ? "Buzzer: ACTIVE" : "Buzzer: INACTIVE";
  html += R"=====(</span>
</div>
<div class='status'>
<div class='status-indicator' id='buzzer-enabled-indicator'>)=====";
  html += buzzerEnabled ? "status-on" : "status-off";
  html += R"=====(</div>
<span id='buzzer-enabled'>)=====";
  html += buzzerEnabled ? "Alarms: ENABLED" : "Alarms: DISABLED";
  html += R"=====(</span>
</div>
```

```
<div class='btn-group'>
<button onclick="controlDevice('buzzer', 'on')" class='btn' id='buzzer-on-btn'>)=====";
  html += buzzerEnabled ? "btn-success" : "btn-primary";
  html += R"=====(">Enable Alarms</button>
<button onclick="controlDevice('buzzer', 'off')" class='btn' id='buzzer-off-btn'>)=====";
  html += !buzzerEnabled ? "btn-danger" : "btn-primary";
  html += R"=====(">Disable Alarms</button>
</div>
<div class='btn-group'>
<button onclick="controlDevice('buzzer', 'test')" class='btn btn-warning'>Test Buzzer</button>
</div>
</div>
</div>
<script>
function controlDevice(device, state) {
  fetch('/' + device + '?state=' + state)
    .then(response => updateSensorData());
}

function updateSensorData() {
  fetch('/sensorData')
    .then(response => response.json())
    .then(data => {
      // Update all sensor values
      document.getElementById('temp-value').textContent = data.temp + '°C';
      document.getElementById('hum-value').textContent = data.hum + '%';
      document.getElementById('gas-value').textContent = data.gas;
      document.getElementById('motion-status').textContent = data.motion ? 'Motion Detected' : 'No
Motion';
      document.getElementById('dist-value').textContent = data.dist + ' cm';
      document.getElementById('relay-status').textContent = 'Relay: ' + (data.relay ? 'ON' : 'OFF');
      document.getElementById('buzzer-status').textContent = data.buzzerState ? 'Buzzer: ACTIVE' :
'Buzzer: INACTIVE';
      document.getElementById('buzzer-enabled').textContent = data.buzzerEnabled ? 'Alarms:
ENABLED' : 'Alarms: DISABLED';


      // Update status indicators
      document.getElementById('motion-indicator').className = 'status-indicator ' + (data.motion ?
'status-on' : 'status-off');
      document.getElementById('relay-indicator').className = 'status-indicator ' + (data.relay ? 'status-
on' : 'status-off');
      document.getElementById('buzzer-state-indicator').className = 'status-indicator ' +
(data.buzzerState ? 'status-on' : 'status-off');
      document.getElementById('buzzer-enabled-indicator').className = 'status-indicator ' +
(data.buzzerEnabled ? 'status-on' : 'status-off');


      // Update gas alert
      const gasAlert = document.getElementById('gas-alert');
      if(data.gas > 250) {
        gasAlert.className = 'alert alert-danger';
        gasAlert.textContent = '⚠ WARNING: High Gas Level Detected!';
      } else if(data.gas > 150) {
        gasAlert.className = 'alert alert-warning';
        gasAlert.textContent = 'Moderate Gas Level';
      } else {
        gasAlert.className = 'alert alert-info';
        gasAlert.textContent = 'Air Quality Normal';
      }


      // Update motion alert
      const motionAlert = document.getElementById('motion-alert');
      if(data.motion) {
        if(!motionAlert) {
          const motionCard = document.querySelector('.card.motion');
          const newAlert = document.createElement('div');
          newAlert.className = 'alert alert-warning';
          newAlert.id = 'motion-alert';
          newAlert.textContent = 'Movement detected in the area!';
          motionCard.appendChild(newAlert);
        }
      } else if(motionAlert) {
        motionAlert.remove();
      }

      // Update distance alert
      const distAlert = document.getElementById('distance-alert');
      if(data.dist < 20 && data.dist > 0) {
        if(!distAlert) {
          const distCard = document.querySelector('.card.distance');
          const newAlert = document.createElement('div');
          newAlert.className = 'alert alert-warning';
          newAlert.id = 'distance-alert';
          newAlert.textContent = 'Object too close!';
          distCard.appendChild(newAlert);
        }
      } else if(distAlert) {
        distAlert.remove();
      }

      // Update button states
      document.getElementById('relay-on-btn').className = 'btn ' + (data.relay ? 'btn-success' : 'btn-
  primary');
      document.getElementById('relay-off-btn').className = 'btn ' + (!data.relay ? 'btn-danger' : 'btn-
  primary');
      document.getElementById('buzzer-on-btn').className = 'btn ' + (data.buzzerEnabled ? 'btn-
  success' : 'btn-primary');
```

```
      document.getElementById('buzzer-off-btn').className = 'btn ' + (!data.buzzerEnabled ? 'btn-
danger' : 'btn-primary');
    });
}

// Update data every 500ms for real-time feel
setInterval(updateSensorData, 500);
updateSensorData();
</script>
</div>
</body>
</html>
)=====";

  server.send(200, "text/html", html);
}

void handleRelay() {
  if (server.arg("state") == "on") {
    digitalWrite(RELAY_PIN, HIGH);
    relayState = true;
  } else if (server.arg("state") == "off") {
    digitalWrite(RELAY_PIN, LOW);
    relayState = false;
  }
  handleSensorData(); // Return updated sensor data
}

void handleBuzzer() {
  String state = server.arg("state");
  if (state == "on") {
    buzzerEnabled = true;
  } else if (state == "off") {
    buzzerEnabled = false;
    digitalWrite(BUZZER_PIN, LOW);
    buzzerState = false;
  } else if (state == "test") {
    digitalWrite(BUZZER_PIN, HIGH);
    buzzerState = true;
    lastBuzzerTime = millis();
  }

  handleSensorData(); // Return updated sensor data
}

void handleSensorData() {
  String json = "{";
  json += "\"temp\":" + String(isnan(temperature) ? "0" : String(temperature, 1)) + ",";
  json += "\"hum\":" + String(isnan(humidity) ? "0" : String(humidity, 1)) + ",";
  json += "\"gas\":" + String(gasValue) + ",";
  json += "\"motion\":" + String(motionDetected ? "true" : "false") + ",";
  json += "\"dist\":" + String(distance, 1) + ",";
  json += "\"relay\":" + String(relayState ? "true" : "false") + ",";
  json += "\"buzzerState\":" + String(buzzerState ? "true" : "false") + ",";
  json += "\"buzzerEnabled\":" + String(buzzerEnabled ? "true" : "false");
  json += "}";

  server.send(200, "application/json", json);
}

void handleNotFound() {
  server.send(404, "text/plain", "Not found");
}
```