**Air Quality Monitoring System**

# **b. Team members:**

**Shaheed Basha Shaik -132614089, sshaik2680@sdsu.edu**

**Prasanth Rayudu – 132612321, vrayudu5728@sdsu.edu**

**c. Accomplishments:**

**i. Hardware Components and Status:**

| **Component** | **Function** | **Status** | **Notes/Changes** |
| --- | --- | --- | --- |
| MQ2 Sensor | Detects gases like CO₂, NH₃, LPG, and smoke | Working | Originally proposed MQ135; switched to MQ2 due to availability and ease of integration |
| DHT11 Sensor | Measures temperature and humidity | Working | No change from proposal |
| ESP32 Microcontroller | Collects data, controls actuators, and uploads to ThingSpeak | Fully Functional | Used as planned; stable performance with Wi-Fi connectivity |
| Cooling Fan (DC) | Activated when AQI exceeds threshold (controlled via PWM) | Working | Fan control tested and optimized for PWM response |
| Buzzer | Provides auditory alert when air quality is poor | Working | Used instead of HVAC and alarm system from proposal due to simplicity and scope control |
| Power Adapter (5V) | Powers the ESP32 and peripherals | Working | Ensured stable power supply without overheating |
| Breadboard & Jumper Wires | For prototyping circuit connections | Used | Standard prototyping setup |

**Changes and Issues in Hardware Development:**

**Sensor Substitution:**  
The originally proposed MQ135 gas sensor was replaced with the MQ2 sensor due to better availability and simpler calibration.

**Dropped LCD Display:**  
While an LCD was initially planned for local AQI display, it was removed from the final design. Real-time data visualization was instead achieved through the ThingSpeak cloud platform.

**Fan Control**  
Pulse Width Modulation (PWM) control is used for tuning to ensure smooth fan operation. During the initial testing, the fan would either not turn on or run at full speed. This was resolved by fine-tuning the PWM duty cycle and incorporating transistor switching.

**Buzzer Circuit**  
The buzzer was initially too sensitive and would activate at low AQI values. This was mitigated by adding threshold-based conditional logic in the microcontroller code

**ii. Software Frameworks, Implementation, and Issues:**

**Frameworks Used:**

* **Arduino IDE:** For coding and uploading to the ESP32.
* **C (Arduino language):** Handles sensor input, AQI logic, PWM, and data upload.
* **ThingSpeak:** Used for cloud-based monitoring of AQI, temperature, humidity, fan speed, and gas values.
* **Libraries:** WiFi.h, HTTPClient.h, DHT.h, and MQ2.h for communication and sensor interfacing.

**Software Design:**

The ESP32 runs a loop that:

1. Reads data from MQ2 and DHT11 and estimates AQI based on MQ2 output.
2. Controls fan speed and buzzer based on AQI.
3. Sends five parameters to ThingSpeak for real-time visualization.

**Status:**

* Fully implemented and tested.
* ThingSpeak updates reliably.
* PWM-based fan speed control tuned for accurate response.

**Changes and Challenges:**

* Mapping MQ2 analog output to AQI involved trial-and-error tuning.
* Initial PWM duty cycles for the fan were either too low or too high resolved through testing.
* Buzzer triggered early due to noise fixed using threshold-based conditions.

**Cyber-Physical System Description: Modeling, Design, and Analysis**

**iii. Modeling:**

The system is modeled using a **Finite State Machine (FSM)** with states based on AQI thresholds. Key variables and transitions are:

* **States**: IDLE, MONITORING, ALERT
* **Inputs**: AQI, Gas value, Temperature, Humidity
* **Outputs**: FAN speed, BUZZER, ThingSpeak Upload
* **Threshold logic**:
  + If AQI ≥ 100 or Gas ≥ 600 → enter ALERT (FAN + BUZZER ON)
  + If AQI < 100 and Gas < 600 → return to IDLE

A diagram of a system

AI-generated content may be incorrect.

Finite State Model

**Temporal Logic for Safety:**

We have defined a **Linear Temporal Logic (LTL)** safety formula:

**G ((AQI > 100 ∨ GAS > 600) → F (ALARM\_ON ∧ FAN\_ON))**

This means: “**Globally, if either AQI exceeds 100 or gas level exceeds 600, the system must eventually activate both the buzzer and the fan**.”

**Case Verification**:

| **Case** | **Condition** | **System Behavior** | **LTL Outcome** |
| --- | --- | --- | --- |
| **1. AQI > 100, Gas < 600, System ON** | AQI triggers condition | System enters ALERT → FAN + BUZZER activate | Formula holds |
| **2. Gas > 600, AQI < 100, System ON** | Gas triggers condition | System enters ALERT → FAN + BUZZER activate | Formula holds |
| **3. System OFF or Sensor Inactive** | SYSTEM\_ON = FALSE or SENSOR\_ACTIVE = FALSE | Stays in IDLE, AQI and Gas not evaluated | Precondition false → not violated |
| **4. AQI & Gas Fluctuate** | Values cross threshold multiple times | System enters/exits ALERT state accordingly | Formula holds consistently |

**Design:**

* **Microcontroller**: ESP32 (240 MHz dual core)
* **Sensors**: MQ2 (gas), DHT11 (temp + humidity)
* **Actuators**: DC Fan (PWM), Buzzer
* **Cloud**: ThingSpeak (Wi-Fi upload)
* **Programming**: Bare-metal style using setup() and loop() in Arduino IDE

The ESP32 executes a cyclic loop that reads sensors, computes AQI, activates outputs, and pushes data to the cloud.

A circuit board with wires

AI-generated content may be incorrect.

**Analysis:**

**Execution Environment and Real-Time Constraints:**

The system runs in a **bare-metal configuration** using the **Arduino framework** on an ESP32 microcontroller. All functionality is implemented inside the setup() and loop() functions without an operating system. This architecture provides predictable execution flow, with each cycle reading sensor data, making control decisions, and uploading them to the cloud.

The loop is expected to complete within a **15-second interval**, which is critical for:

* Timely activation of fan and buzzer in response to poor air quality.
* Consistent and accurate data upload to ThingSpeak.
* Maintaining periodic monitoring without drifting or delay
* This timing requirement forms the basis for the **Worst-Case Execution Time (WCET)** analysis.

**WCET Measurement and Results**

**Measured Code Block**:

unsigned long start = millis();

// Sensor reading, AQI calculation, actuator control, ThingSpeak upload

unsigned long end = millis();

unsigned long execTime = end - start;

Serial.println("Execution time: " + String(execTime) + " ms");

This block includes:

* Temperature and humidity readings (DHT11)
* Gas level input (MQ2)
* AQI computation
* Fan and buzzer control logic
* HTTP-based upload to ThingSpeak

**Timing Method**:

* Used Arduino’s built-in millis() function to measure execution time
* Logged multiple runs under different conditions:
  + Normal air quality
  + Simulated high gas
  + Weak Wi-Fi connection

| **Metric** | **Expected Target** | **Your System (626 cycles)** | **Comment** |
| --- | --- | --- | --- |
| Avg Loop Time | ≤ 15000 ms (soft limit) | 192 ms | Well within limits |
| Max Loop Time | ≤ 15000 ms | 244 ms | OK |

A graph of a graph with text

AI-generated content may be incorrect.

**iv. Experimental Design and Metrics of Success:**

**1.Success Metrics**

To evaluate the system’s performance, we defined the following metrics:

* **Accuracy:** Sensor readings (temperature, humidity, gas levels) should be within ±5% of known reference values.

**Standard Deviation Measurement:**

| **Parameter** | **Standard Reference Value** | **Measured Average** | **Deviation** | **Within ±5%?** |
| --- | --- | --- | --- | --- |
| Temperature (DHT11) | 25.0 °C (Digital Thermometer) | 24.7 °C | -0.3 °C | Yes |
| Humidity (MQ2) | 45% RH (Weather Station) | 46.1% RH | +1.1% | Yes |

* **Response Time:** Fan and buzzer should activate within 3–5 seconds of AQI threshold breach.
* **System Stability:** Continuous operation without crashes or data loss during extended monitoring.
* **Data Logging:** Successful upload of all five parameters (AQI, temperature, humidity, fan speed, gas value) to ThingSpeak at ~15-second intervals.

**Interpretation and Use of Metrics:**

We used key performance metrics to validate the real-time reliability and responsiveness of our system:

* **Sensor Accuracy**: During calibration, sensor values were compared against standard reference devices. Deviations within ±5% confirmed the sensors provided valid input for AQI computation.
* **Execution Time (WCET)**: The Arduino millis() function was used to measure the time taken to complete the sensing, control, and upload loop. The average WCET was ~191.87 ms, well under the 15-second update window.
* **Standard Deviation**: Over 626 measured cycles, a standard deviation of 16.81 ms indicated timing stability and consistency in system behavior.
* **Actuator Response Time**: System consistently responded within 3–5 seconds to poor air quality conditions, meeting the real-time control requirement.

**Experiments Performed:**

1. **Sensor Calibration:** Compared MQ2 and DHT11 readings with known values in controlled environments.
2. **Threshold Testing:** Verified actuator response across AQI/gas value ranges.
   * Low AQI → No fan/buzzer
   * High AQI → Fan (PWM) and buzzer activation
3. **ThingSpeak Integration:**
   * Verified real-time updates of all five fields.
   * Tested for delays and packet loss.
4. **Fan PWM Validation:**
   * Checked fan behavior at PWM 0, 140, 160, and 255.
5. **Loop Execution Timing:**
   * Measured loop completion under different conditions (normal, high gas, Wi-Fi drop).

### **2.Results:**

Through multiple test runs, the system consistently performed as intended. Key outcomes:

| **Test Condition** | **Expected Behavior** | **Observed Outcome** |
| --- | --- | --- |
| AQI < 100, Gas < 600 | Fan OFF, Buzzer OFF | Actuators stayed OFF |
| AQI ≥ 100 or Gas ≥ 600 | Fan ON (PWM), Buzzer ON | Fan and buzzer activated within ~4 seconds |
| Gas ≥ 1000 | Fan PWM = 255 (full speed) | Fan reached max speed |
| Wi-Fi interrupted | Data upload paused, local logic unaffected | System resumed cloud sync post reconnection |
| Long-duration test (3+ hours) | No resets or crashes | Stable performance |

A screenshot of a graph

AI-generated content may be incorrect.

[Thingspeak Ref](https://thingspeak.mathworks.com/channels/2907743)

**3.Possible Extensions to the Project:**

* **Improved Sensor Accuracy:** Replace MQ2 with MQ135 or dedicated CO₂ sensors for more precise AQI estimation.
* **Mobile Alert System:** Integrate IFTTT or a simple app to send notifications when AQI exceeds safe levels.
* **Battery Backup:** Add a rechargeable battery to ensure operation during power outages or for portable use.
* **Local Web Display:** Host a basic web interface on ESP32 to view real-time data over Wi-Fi without relying on ThingSpeak.
* **Sensor Health Check:** Include basic logic to detect and report faulty or disconnected sensors.

**d. Missing Milestones:**

**i. What We Have Not Delivered from the Milestone List:**

* **LCD Display Integration**: Originally planned for real-time local display of AQI and temperature but was removed to reduce hardware complexity and focus on ThingSpeak cloud visualization.
* **HVAC Relay Control**: Proposed in early planning but not implemented due to unavailability of compatible hardware and lack of real HVAC system for integration/testing.

**ii. Negative Results and Future Fixes:**

* **Buzzer Sensitivity**: Initially triggered at lower AQI levels due to noisy analog readings. This was resolved with threshold-based logic, but fine-tuning may still be needed in varied environments.
* **AQI Mapping from MQ2**: The mapping is an approximation; without pollutant-specific sensors, exact AQI values can't be guaranteed. Replacing MQ2 with gas-specific sensors (e.g., MQ135, MH-Z19) would improve reliability.
* **Cloud Upload Delay Under Weak Wi-Fi**: Occasionally delayed ThingSpeak updates when signal was weak. Could be addressed using data buffering or retries in future versions.