PROJECT REPORT

Presented by:

TEAM 10

ESP32 CODE EXPLANATION

1. Included Libraries

```
#include <Arduino.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>
```

- Arduino.h → Base functions for ESP32 (pinMode, digitalWrite, etc.).
- DHT.h → Reads temperature and humidity from DHT11 sensor.
- LiquidCrystal_I2C.h & Wire.h → Communicate with the LCD via I²C protocol.
- ESP32Servo.h → Controls the servo motor.
- WiFi.h → Connects ESP32 to Wi-Fi.
- PubSubClient.h → Handles MQTT communication.
- WiFiClientSecure.h → Enables encrypted (SSL/TLS) connection for secure MQTT (HiveMQ Cloud).

2. Pin Assignments

```
#define PIN_MQ2    34        // MQ2 Gas sensor (Analog input)
#define PIN_DHT    15        // DHT11 Temp & Humidity sensor (Digital)
#define PIN_FLAME  35        // Flame sensor (Analog input)
#define PIN_SERVO  14        // Servo motor (PWM output)
#define PIN_LED    25        // LED indicator
#define PIN_BUZZ   26        // Buzzer alarm
```

Why these pins are chosen:

- PIN_MQ2 (34) → ESP32 pin 34 is ADC (Analog-to-Digital Converter). MQ2 outputs analog values depending on gas concentration → so we need an ADC pin.
- PIN_DHT (15) → Any digital GPIO can be used; here GPIO15 is chosen for the single-wire digital communication of DHT11.
- PIN_FLAME (35) → Flame sensor outputs analog intensity values, so GPIO35 (ADC pin) is used.
- PIN_SERVO (14) → Servo needs PWM (Pulse Width Modulation). GPIO14 supports PWM.
- PIN_LED (25) → General-purpose output pin to drive LED.
- PIN_BUZZ (26) → GPIO26 used for buzzer (digital HIGH = ON, LOW = OFF).

3. Sensor & Actuator Setup

```
#define DHTTYPE DHT11
DHT dht(PIN_DHT, DHTTYPE);
Servo myServo;
int servoPos = 0;
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

- Initializes DHT11 on pin 15.
- Sets up servo motor object.
- Stores servo angle (servoPos).
- Configures LCD (16x2) at I²C address 0x27.

4. Wi-Fi & MQTT Setup

```
const char* ssid = "";
const char* password = "";
const char* mqtt_server = "...hivemq.cloud";
const int mqtt_port = 8883;
const char* mqtt_user = "...";
const char* mqtt_password = "...";
```

- ESP32 connects to Wi-Fi and then to HiveMQ Cloud MQTT Broker (secure connection using TLS on port 8883).
- MQTT topics for communication:
- Subscriptions (control): led, servo, buzzer
- Publications (confirmations): led/confirm, servo/confirm, buzzer/confirm
- Sensor data publishing: sensors/data

5. MQTT Callback Function

```
void callback(char* topic, byte* payload, unsigned int length)
```

- Runs when ESP32 receives a message from MQTT.
- LED Control → If topic is led, turn LED ON/OFF.
- Servo Control → If topic is servo, rotate servo to requested angle.
- Buzzer Control → If topic is buzzer, turn buzzer ON/OFF.
- This allows remote control of devices via MQTT (e.g., from web app or Node-RED).

6. Reconnect Function

void reconnect()

- Ensures ESP32 reconnects to MQTT broker if connection drops.
- Re-subscribes to topics after reconnect.

7. Setup Function

void setup()

- Configures pins (INPUT/OUTPUT).
- Initializes sensors (DHT11, Servo).
- Initializes LCD display.
- Connects to Wi-Fi.
- Connects to MQTT broker.
- LCD shows “Smart Kitchen” at startup.

8. Main Loop

void loop()

Tasks inside loop:

- Keep MQTT connection alive → client.loop().
- Read sensors:

1. Gas → analogRead(PIN_MQ2)
2. Flame → analogRead(PIN_FLAME)
3. Temp/Humidity → dht.readTemperature(), dht.readHumidity()

Danger Detection Logic:
bool danger = (gasValue > 1500) || (flameValue > 3000) || (temp > 50);
If gas > 3942 OR flame > 3000 OR temperature > 50°C → Danger.
If Danger:

1. Turn ON LED & buzzer.
2. Move servo to 90° (open door/valve).
3. LCD shows “ALERT! DANGER!”

If Safe:

4. Turn OFF LED & buzzer.
5. Servo = 0° (closed).
6. LCD shows current sensor readings.

Publish Sensor Data to MQTT:

- Sends data in JSON format → can be used in web dashboards (Grafana, Node-RED, custom app).
- Delay → Wait 10 seconds before next cycle.

Summary of Pins & Their Purpose

Pin (GPIO)	Component	Purpose
34	MQ2 Gas Sensor	Reads analog gas concentration
15	DHT11	Reads digital temperature & humidity
35	Flame Sensor	Reads flame intensity (analog)
14	Servo Motor	Controls door/valve via PWM
25	LED	Visual alert (ON = danger)
26	Buzzer	Audible alarm for danger

Why This Design?

- MQ2 + Flame + DHT11 → Detects gas leaks, fire, and high temperature in kitchen.
- Servo Motor → Can open a window/door/valve automatically for ventilation.
- LED + Buzzer → Local alert for people inside kitchen.
- LCD → Real-time display of sensor readings & danger warnings.
- Wi-Fi + MQTT → Remote monitoring & control (smartphone/web dashboard).

SENSORS

&

THRESHOLDS

Sensor Specifications

DHT11 Temperature Humidity Sensor Board
Specifications:

- Supply voltage: 3 V to 5.5 V
- Current Consumption: ~0.2 mA
- Measuring range: 0 – 50 °C
- Resolution: 8-bit (1 °C)
- Accuracy: 1 °C
- Response time: 6 – 15 seconds (usually 10 seconds)
- Measuring range: 20 – 90% RH
- Resolution: 8-bit (±1 % RH*)
- Accuracy: ±4 RH* (at 25 °C)
- Measuring range: 6 – 30 sec

Gas Sensor (MQ-2)
Specifications:

- Power Supply: 5 V
- Current consumption: 150 mA
- Measuring range up to 10,000 ppm
- Operating temperature: -20 to 50 °C
- Analog and digital output
- Resistor which pulls the sensor’s output to ground with a value of 1 kΩ

Flame Sensor Module
Specifications :
ParameterValue
Operating Voltage: 3.3V – 5V
Current Draw: <7mA
Detection Wavelength: 700nm – 1100nm
Detection Range: Up to 100 cm (adjustable, depends on flame size)
Detection Angle: ~60 degrees
Response Time: Fast and accurate
Output Types: Digital (DO) and Analog (AO)

Sensor	Range	Threshold	Explanation
Gas (MQ2)	300–10,000 ppm	ADC value = 2000 (~1000 ppm)	Indicates a significant gas leak requiring immediate attention.
Flame	700–1100 nm	ADC value = 1000	Detects the presence of a flame.
Temperature	-40°C to 80°C	40°C	Indicates overheating or a potential fire risk

GAS SENSOR:

- Normal kitchen air (clean air) → < 200 ppm
- Gas leak alert → >1000 ppm
- Dangerous leak (explosive risk) → >2000 ppm

FLAME SENSOR:

- Ambient light (10-15)cm → ADC = 100-300
- Ambient light (20-80)cm → ADC = 2200-3000
- Flame at 10 cm → ADC = 50-200
- Flame at 20-80 cm → ADC = 100-1000 depend on degree

we will choose a threshold of 2000 or 2500, depending on the desired sensitivity.

SMART KITCHEN SAFETY & MONITORING SYSTEM

Q&A SUMMARY

1.What is the main idea of the project?

- It is a Smart Kitchen Safety & Monitoring System that uses sensors, ESP32, Supabase, MQTT, Node-RED, and Flutter to monitor gas leaks, flames, and temperature/humidity, trigger alarms, and provide real-time data on a mobile app.

2.Which sensors are used and why?

- MQ2 Gas Sensor → Detects gas leaks (LPG, methane, smoke) in range 300–10,000 ppm.
- Flame Sensor → Detects fire/IR radiation in range 700–1100 nm.
- DHT11 Sensor → Measures temperature (0–50°C) and humidity (20–90%) for monitoring kitchen environment.
- Benefit: Together, these sensors cover fire, gas, and heat risks.

3.What are the problems this project solves?

- Gas leaks that may cause explosions.
- Fire hazards from cooking flames.
- Overheating or unsafe temperature/humidity levels.
- Lack of real-time monitoring and alerts in traditional kitchens.

4.What challenges might we face?

- Sensor calibration (MQ2 & flame sensor need adjustment).
- False alarms from smoke/steam.
- Wi-Fi instability affecting cloud updates.
- Synchronizing ESP32, MQTT, Supabase, Node-RED, and Flutter.
- Designing a bilingual app (English/Arabic) that is user-friendly.

5. How does the system work (workflow)?

- Sensors collect data (gas, flame, temp, humidity).
- ESP32 processes readings, controls servo motor, LED, buzzer for local action.
- ESP32 publishes data via MQTT.
- Node-RED subscribes, processes data, and pushes alerts.
- Data stored in Supabase (cloud database).
- Flutter app shows real-time dashboard, alerts, history, and allows user login.

6.What technologies are used and why?

- ESP32 → Wi-Fi-enabled microcontroller.
- MQTT → Lightweight protocol, best for IoT communication.
- Node-RED → Automation workflows and dashboard.
- Supabase → Cloud storage + authentication.
- Flutter → Cross-platform mobile app with EN/AR language support.

7.What are the benefits of this system?

- Real-time hazard detection.
- Automatic gas valve control (via servo).
- Alerts via buzzer, LED, and mobile notifications.
- Cloud storage for history & analytics.
- User-friendly mobile dashboard.

8. What are the expected results?

- A working prototype of the smart kitchen system.
- Real-time readings on mobile app & Node-RED dashboard.
- Cloud database (Supabase) with logs of hazards/events.
- Automatic safety actions (gas shut-off, alarms).

9.What future improvements can be added?

- AI-based predictions of fire/gas risk.
- SMS/Email alert system.
- Integration with Google Home / Alexa.
- Automatic fire extinguisher trigger.