**GAS AND TEMPERATURE SENSOR DOCUMENTATION**

**OVERVIEW**

This project aims to create a sensor-based alert system using an Arduino Uno, which detects gas levels and temperature. The system is designed to alert occupants when dangerous gas levels are present or if the room's temperature reaches an intolerable level, indicative of a fire or overheating.

Simulation URL: <https://wokwi.com/projects/412811019007793153>

**OBJECTIVES**

- Detect and alert occupants to gas presence in a room.

- Monitor room temperature and issue warnings if it reaches potentially hazardous levels.

- Display visual alerts using LED indicators.

**COMPONENTS**

1. Arduino Uno: Acts as the central processor, reading analog inputs from the gas and temperature sensors and controlling LEDs based on these readings.

2. Gas Sensor (MQ2): Detects the presence of gas. When gas is detected at high levels, it triggers an LED to blink and sends a message.

3. Temperature Sensor: Reads the room's temperature, mapping it to an LED bar graph that increases with temperature. It also displays warning messages at critical temperature levels.

4. LED Bar Graph: Displays a visual representation of the temperature, increasing the number of illuminated LEDs as the temperature rises.

5. Power Supply: Supplies power to the Arduino and sensors.

**FUNCTIONALITY**

1. Pin and Constant Definitions

Define analog and digital pins for the sensors. Set the number of LEDs in the bar graph and assign their respective pins.

2. Setup Function

a. Initialize serial communication at 9600 baud.

b. Configure the MQ2 sensor's analog (MQ2\_ANA) and digital (MQ2\_DIG) pins as inputs.

c. Set the Arduino's built-in LED as an output.

d. Set all pins in the ‘ledPins’ array as outputs to control the LED bar graph.

3. Loop Function

a. Gas Detection:

- Read analog and digital values from the MQ2 sensor.

- If gas is detected (digital reading is HIGH), print "Gas detected" and blink the LED.

b. Temperature Reading:

- Read the analog value from the temperature sensor.

- Print a warning message if the temperature reaches a critical threshold.

c. LED Bar Graph Display:

- Map the temperature reading to an LED level.

- Illuminate LEDs up to the calculated level, representing the current temperature.

d. Loop Delay:

- Pause for 1000 ms before repeating the loop to ensure accurate readings and updates.

**ARDUINO CODE**

#define MQ2\_ANA A1

#define MQ2\_DIG 2

#define TEMP\_SENSOR A0

const int ledCount = 10;    // Number of LEDs in the bar graph

int ledPins[] = {3, 4, 5, 6, 7, 8, 9, 10, 11, 12}; // Pins for LEDs of bar graph

void setup() {

**Serial**.begin(9600);

  pinMode(MQ2\_ANA, INPUT);

  pinMode(MQ2\_DIG, INPUT);

  pinMode(LED\_BUILTIN, OUTPUT);

  // Initialize LED pins as OUTPUT

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

    pinMode(ledPins[i], OUTPUT);

  }

}

void loop() {

  // Read gas sensor values

  float gas\_analog = analogRead(MQ2\_ANA);

  int gas\_digital = digitalRead(MQ2\_DIG);

  if (gas\_digital == 1){

**Serial**.println("Gas detected.");

    digitalWrite(LED\_BUILTIN, HIGH);   // turn the LED on

    delay(10);

    digitalWrite(LED\_BUILTIN, LOW);    // turn the LED off

    delay(10);

  }

  // Read temperature

  float temperature = analogRead(TEMP\_SENSOR);

  if (temperature < 300){

**Serial**.println("Evacuate the room !");

  }

  else if (temperature < 450){

**Serial**.println("Temperature approaching dangerous levels...");

  }

  // Map temperature to LED levels

  int ledLevel = map(temperature, 115, 953, 0, ledCount);

  // Update LED bar graph

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

    if (i < ledLevel) {

      digitalWrite(ledPins[i], LOW); // Turn off LED

    } else {

      digitalWrite(ledPins[i], HIGH);  // Turn on LED

    }

  }

  delay(500);

}

**C CODE FOR GAS SENSOR**

#include "wokwi-api.h" // Include Wokwi API for hardware simulation

#include <stdio.h>

#include <stdlib.h>

// Structure to hold the state of the MQ2 chip

typedef struct {

  pin\_t pin\_ao;          // Analog output pin for gas concentration

  pin\_t pin\_do;          // Digital output pin for gas detection threshold

  pin\_t pin\_vcc;         // VCC pin for power supply

  pin\_t pin\_gnd;         // GND pin for ground connection

  uint32\_t gas\_attr;     // Attribute for gas concentration percentage

  uint32\_t threshold\_attr; // Attribute for clean air threshold

} chip\_state\_t;

// Forward declaration of the timer event handler

static void chip\_timer\_event(void \*user\_data);

// Function to initialize the MQ2 chip

void chip\_init(void) {

  chip\_state\_t \*chip = malloc(sizeof(chip\_state\_t)); // Allocate memory for chip state

  // Initialize pins

  chip->pin\_ao = pin\_init("AO", ANALOG); // Set AO pin as ANALOG input

  chip->gas\_attr = attr\_init("gas", 10); // Initialize gas attribute with default value

  chip->threshold\_attr = attr\_init("threshold", 50); // Initialize threshold attribute

  chip->pin\_do = pin\_init("DO", OUTPUT\_LOW); // Set DO pin as OUTPUT (initially LOW)

  chip->pin\_vcc = pin\_init("VCC", INPUT\_PULLDOWN); // Set VCC pin as INPUT\_PULLDOWN

  chip->pin\_gnd = pin\_init("GND", INPUT\_PULLUP); // Set GND pin as INPUT\_PULLUP

  // Configure timer for periodic updates

  const timer\_config\_t timer\_config = {

    .callback = chip\_timer\_event, // Set the callback function for the timer

    .user\_data = chip, // Pass the chip state to the callback

  };

  timer\_t timer\_id = timer\_init(&timer\_config); // Initialize timer

  timer\_start(timer\_id, 1000, true); // Start timer to call every 1000 ms (1 second)

}

// Timer event handler function

void chip\_timer\_event(void \*user\_data) {

  chip\_state\_t \*chip = (chip\_state\_t\*)user\_data; // Cast user\_data to chip state

  // Read gas and threshold attributes as float values

  float voltage = (attr\_read\_float(chip->gas\_attr)) \* 5.0 / 100; // Convert gas % to voltage

  float threshold\_v = (attr\_read\_float(chip->threshold\_attr)) \* 5.0 / 100; // Convert threshold % to voltage

  // Check if power is supplied correctly

  if (pin\_read(chip->pin\_vcc) && !pin\_read(chip->pin\_gnd)) {

    pin\_dac\_write(chip->pin\_ao, voltage); // Write voltage to AO pin

    // Check if the gas concentration exceeds the threshold

    if (voltage > threshold\_v)

      pin\_write(chip->pin\_do, HIGH); // Set DO pin HIGH if threshold is exceeded

    else

      pin\_write(chip->pin\_do, LOW); // Set DO pin LOW if below threshold

  }

}

**ADVANTAGES**

- Early Detection: Alerts occupants to hazards promptly.

- Cost-Effective: Utilizes inexpensive components.

- Customizable and Scalable: The system can be modified for additional sensors or displays.

- Real-Time Alerts: Provides instant notifications for safety.

- Low Power Consumption: Efficiently manages power usage.