ABSTRACT

The project focuses on the IOT based smoke alarm detection system which alarms the users about the increase in the level of smoke at a particular place and acts as a security management system. The use of MQT gas sensor and ESP8266 wifi module helps the system note the value of the smoke present in the room and using the ESP8266 wifi module the data is transferred to Thing Speak where the data is plotted in the graph. The ThingSpeak on reaching the maximum limit calls the Twilio API and sends the message to the administrator to warn them about the estimated value of gas inside the room.

TABLE OF CONTENTS

INTRODUCTION:

The increase in industrialization has caused an increase in hazards in daily life. One of the most concerned factor is House Fires. According to an estimated data, in the year 2019, more than a million houses were reported to have caught fire. In India, most of the cases are unreported owing to poverty. Our system is designed keeping in mind the money and efficiency of the device. The IOT BASED SMOKE ALARM would serve as a First aid to any house fire and limit the tragic incident to a minimal level.

MQ2 gas sensor detects smoke and leakage of combustible gas Methane, Butane, LPG etc. ESP8266 reads the sensor data and push the same to ThingSpeak (IoT analytics platform) using MQTT. If the gas concentration exceeds the preset threshold value (calibrated), ESP8266 triggers the alarm (buzzer) with a flashing red LED. On the other hand, ThingSpeak "React"s and initiates the action "ThingHTTP" to call Twilio (Cloud communications platform) API to send SMS notification. ThingSpeak also provides option to visualize (barchart, gauge) the data pushed by ESP8266.

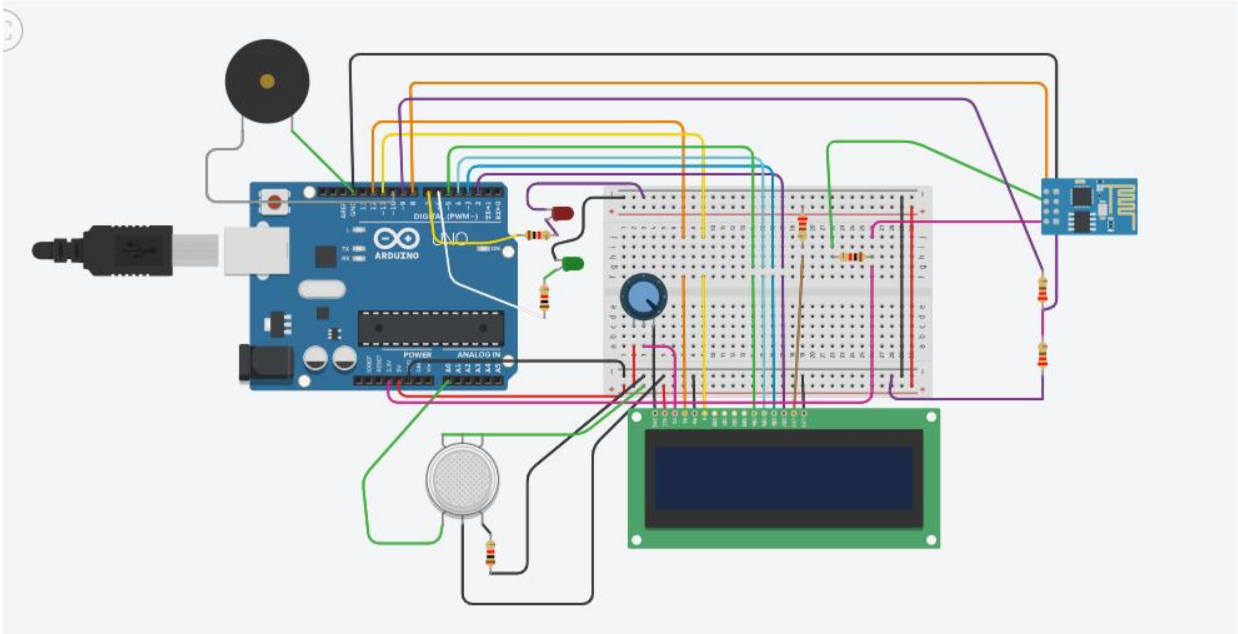
SOFTWARE DESCRIPTION:

We used **Tinker Cad,** an online platform to design circuits for embedded systems. The components like **ESP8266** Wi-Fi module, 16X2 **LCD** screen, **LED'S** , **MQTT** Gas sensor and a buzzer to aware the users about the smoke is used. Apart from the construction of the circuit, the code is written in **Arduino uno** to train the circuit about the functionalities. We have used the **ThingSpeak** cloud platform to display the data and if the data is more than the specified level then, **Twilio** is called and a message is sent to the user and the user is aware of it. Also the LED'Sand alarms start buzzing and hence the system works properly.

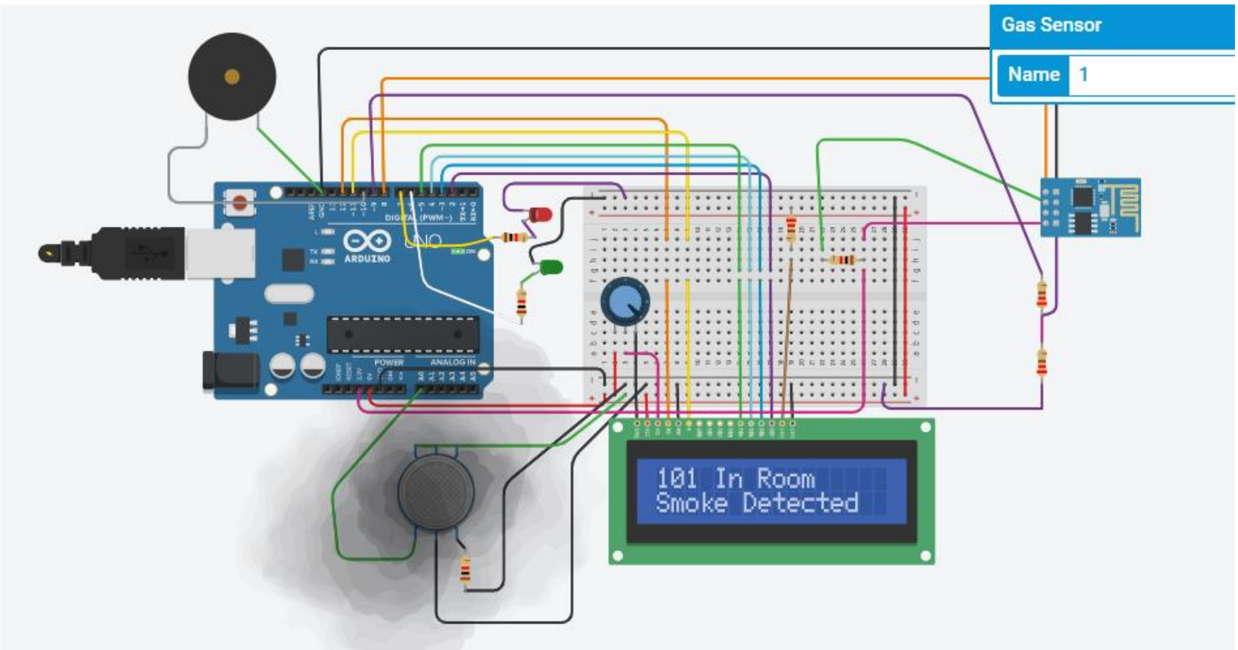
SAMPLE CODE

```
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // LCD Connections
SoftwareSerial SerCommESP8266(8,9); // RX, TX  connect 8 to TX of ESP, connect 9 to RX of ESP

int smokeVal=0;
int smoke_sensor_pin=A0; // MQ2 Gas Sensor
int red_led_pin=7;  // Smoke indication
int green_led_pin=6; // No Smoke indication
int buzzer_pin = 10; // Buzzer

String apiKey = "OOTDUPXSJD56COGZ";  // Write API key

void setup()
{
  pinMode(red_led_pin, OUTPUT);
  pinMode(green_led_pin, OUTPUT);
  pinMode(buzzer_pin, OUTPUT);
  pinMode(smoke_sensor_pin, INPUT);
  Serial.begin(9600); //  serial data transmission at Baudrate of 9600
  SerCommESP8266.begin(9600);   // Initialize the serial communication baud rate

  lcd.begin(16, 2);  // to intialize LCD
  lcd.setCursor(0,0);
  lcd.print("    CYBER");
  lcd.setCursor(0,1);
  lcd.print("      PHYSICAL       ");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("   SYSTEM");
  lcd.setCursor(0,1);
  lcd.print("    REVIEW2");
  delay(1000);
  SerCommESP8266.println("AT");  // Start ESP8266 Module
  delay(1000);
  SerCommESP8266.println("AT+GMR"); // To view version info for ESP-01 output: 00160901 and ESP-12 output: 0018000902-AI03
  delay(1000);
  SerCommESP8266.println("AT+CWMODE=3"); // To determine WiFi mode
  delay(1000);
  SerCommESP8266.println("AT+RST"); // To restart the module
  delay(1000);
  SerCommESP8266.println("AT+CIPMUX=1"); // Enable multiple connections 0: Single connection  1: Multiple connections (MAX 4)

  delay(1000);
  String cmd="AT+CWJAP=\"SSID NAME\",\"SSID PASSWORD\""; // connect to Wi-Fi
  SerCommESP8266.println(cmd);
  delay(1000);
  SerCommESP8266.println("AT+CIFSR"); // Return or get the local IP address
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("    WIFI");
  lcd.setCursor(0,1);
  lcd.print("  CONNECTED");
}
void loop()
{
  delay(1000);
  smokeVal = map(analogRead(A0),10,350,0,100);
  Serial.println();
  lcd.clear();
  lcd.setCursor (0, 0);
  lcd.print (smokeVal);
  lcd.print (" In Room");
  lcd.setCursor (0,1);
  if (smokeVal>30)
   {
    lcd.print("Smoke Detected");
    Serial.print("Smoke Detected");
    digitalWrite(red_led_pin, HIGH);
    digitalWrite(green_led_pin, LOW);
    tone(buzzer_pin, 1000, 200);
   }
  else
   {
    lcd.print("Safe");
    Serial.print("Safe");
    digitalWrite(red_led_pin, LOW);
    digitalWrite(green_led_pin, HIGH);
    noTone(buzzer_pin);
   }
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("  SENDING DATA");
  lcd.setCursor(0,1);
  lcd.print(" TO THINGSPEAK");
  SetupESP8266_HA(); // For ThingSpeak Data Transfer
  delay(1000);
}
void SetupESP8266_HA()
{
   // TCP connection AT+CIPSTART=4,"TCP","184.106.153.149",80
    String cmd = "\nAT+CIPSTART=4,\"TCP\",\"";  // Establish TCP connection
    cmd += "184.106.153.149"; // api.thingspeak.com
    cmd += "\",80"; // Port Number
    SerCommESP8266.println(cmd);
    Serial.println(cmd);
    if(SerCommESP8266.find("Error"))
    {
      Serial.println("AT+CIPSTART error");
      return;
    }
  String getStr = "GET /update?api_key=";  // API key
  getStr += apiKey;
  getStr +="&field1=";  // Field variable as Smoke
  getStr +=String(smokeVal);
  getStr += "\r\n\r\n";
  // send data length
  cmd = "AT+CIPSEND="; // Send data AT+CIPSEND=id,length
  cmd += String(getStr.length());
  SerCommESP8266.println(cmd);
  Serial.println(cmd);
  delay(1000);
  SerCommESP8266.print(getStr);
  Serial.println(getStr);
  // thingspeak needs max 16 sec delay between updates
  delay(10000);
}
```

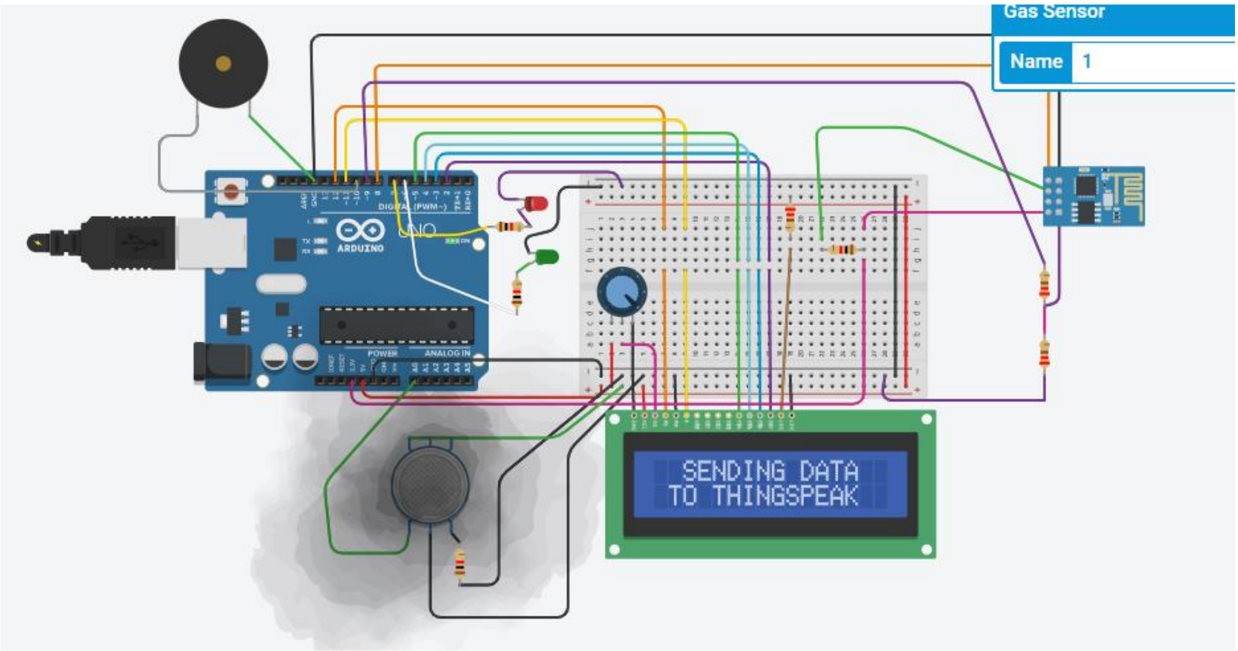
RESULT:



Circuit



The value of smoke is detected and displayed in LCD



The data is being sent to Thing Speak.

CONCLUSION:

Hence, our project is working well and covers all the modules regarding the smoke alarm and works in the favour of the users.