Project Based Learning Report

On

SMOKE DETECTION IN HOME AUTOMATION WITH ESP8266 USING IOT

Submitted in the partial fulfillment of the requirements

For the Project based learning in "**Industrial IOT and ML**"in

Electronics & Communication Engineering

By

PRN	NAME
2114110442	Arun Deshmukh
2114110476	Vishal Kumar
2114110490	Utkarsh Kumar Maurya

Under the guidance of Course In-charge

Prof. Arundhati A. Shinde

Department of Electronics & Communication Engineering

Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 411043

Academic Year: 2023-24

Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 411043

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the Project Based Learning report entitled, "Smoke Detection in Home Automation with ESP8266 using IoT" Is work done by

PRN	NAME
2114110442	Arun Deshmukh
2114110476	Vishal Kumar
2114110490	Utkarsh Kumar Maurya

in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in "**Industrial IOT and ML**" of Bachelor of Technology Semester V, in Electronicsand Communication.

Date:

Prof. Arundhati A. Shinde
Course In-charge

Dr. Arundhati A. Shinde
Professor & Head

INDEX

Problem Statement-

Smoke Detection in Home Automation with ESP8266 using IoT

Project Description-

The integration of smoke detection in home automation systems using ESP8266 and IoT (Internet of Things) technology offers an efficient and reliable method to enhance home safety. This project aims to design and implement a system that can detect smoke and trigger appropriate actions to mitigate potential hazards. By leveraging ESP8266 microcontrollers and IoT connectivity, the system can provide real-time alerts and automate responses to ensure the safety of occupants and property.

Project Objectives –

The primary objective of the "Smoke Detection in Home Automation with ESP8266 using IoT" project is to design and implement a comprehensive system that enhances home safety through the integration of smoke detection capabilities into a home automation framework. The project aims to develop a reliable smoke detection mechanism using appropriate sensors and ESP8266 microcontrollers to accurately detect the presence of smoke particles. It seeks to establish connectivity with IoT modules to enable real-time monitoring of the smoke detection system and ensure prompt alerting by activating audible alarms or sending notifications to users' mobile devices in the event of smoke detection. Additionally, the project aims to define logic and rules for automated responses to smoke detection events, such as activating ventilation systems or triggering sprinklers, to mitigate risks and minimize damage. Furthermore, it aims to enable users to remotely monitor the status of the smoke detection system and take manual control if necessary, while also ensuring scalability and integration with other smart home devices for future expansions.

Project Goals –

1.Design and Implement Smoke Detection System: The first step is to design and implement a robust smoke detection system using ESP8266 microcontrollers and appropriate sensors. This system will accurately detect the presence of smoke particles in the air.

2.Enable Real-Time Monitoring and Alerting:The next step involves establishing connectivity with IoT modules to enable real-time monitoring of the smoke detection system. This includes transmitting data to cloud platforms for remote access and promptly alerting occupants via audible alarms or mobile notifications upon detecting smoke.

3.Automate Responses to Smoke Detection: Once smoke is detected, the system will automatically initiate predefined responses to mitigate risks and minimize damage. This may involve activating ventilation systems, triggering sprinklers, or notifying emergency services based on predefined logic and rules.

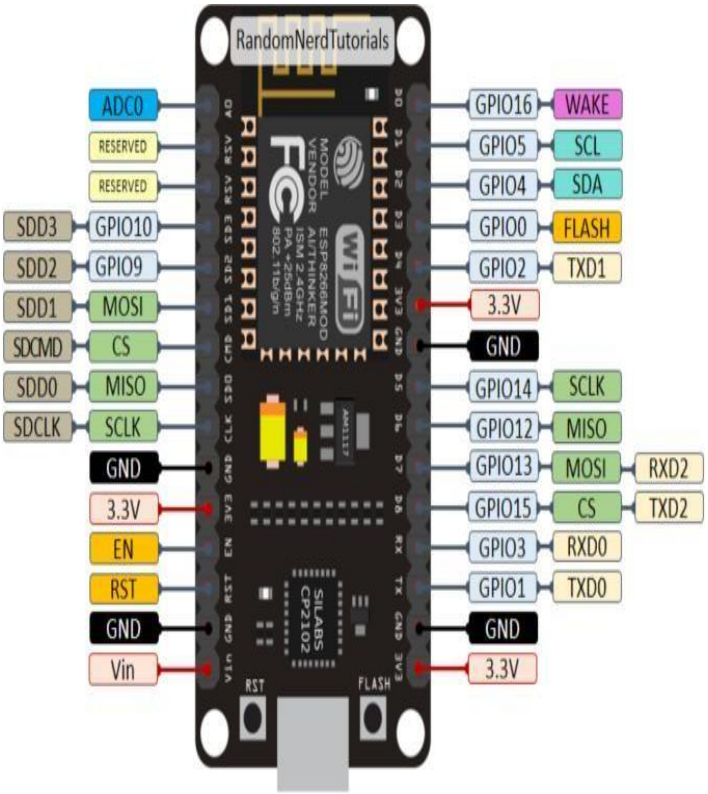
4.Facilitate Remote Control and Ensure Scalability: The final step is to enable users to remotely monitor the system's status and take manual control if necessary, ensuring flexibility and responsiveness even when away from home. Additionally, the system should be designed with modularity and scalability in mind to easily integrate with other smart home devices and accommodate future expansions.

Components Used -

1. ESP8266 Node MCU development board.
2. Breadboard and jumper wires.
3. Power source (USB cable or external power supply).
4. MQ-2 Gas sensor
5. Buzzer

Node MCU ESP8266 –

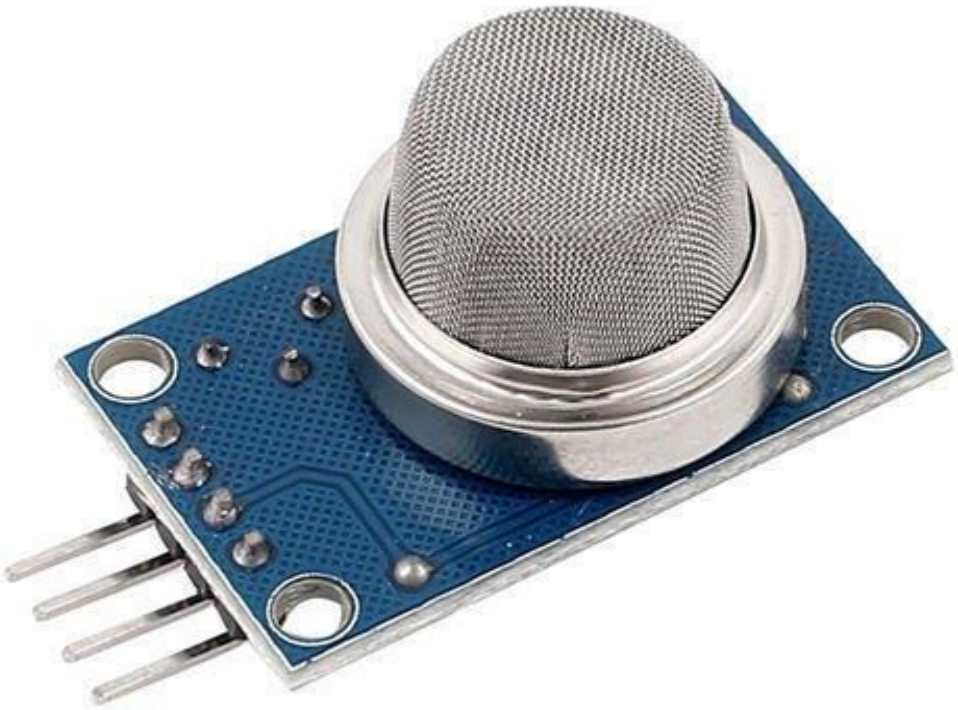
The NodeMCU ESP8266 is a development board integrating the ESP8266 microcontroller unit (MCU), renowned for its built-in Wi-Fi capabilities, making it a staple in IoT projects. Featuring numerous GPIO pins for sensor and actuator interfacing, it supports multiple programming languages like Arduino IDE and Lua scripting, enhancing its versatility. With power options ranging from USB to external sources and flash memory for program storage, it offers convenience in both programming and deployment. Open-source by nature, it provides access to schematics and firmware, fostering an active community of developers. Widely utilized for prototyping IoT applications, home automation, and remote monitoring systems, its affordability, flexibility, and user-friendly design contribute to its popularity and widespread adoptio



MQ-2 Gas sensor

The MQ-2 Smoke LPG Butane Hydrogen Gas Sensor Detector Module is useful for gas leakage detection (home and industry). It is suitable for detecting H2, LPG, CH4, CO, Alcohol, Smoke or Propane. Due to its high sensitivity and fast response time, measurement can be taken as soon as possible.

MQ-2 Gas sensor using gas sensitive material is to be clean air in the lower conductivity of Tin oxide (SnO2). When the sensor when flammable gases are present in the environment in which the conductivity of the sensor with an increasing concentration of combustible gas in the air increases. Use a simple circuit to convert the changes in conductivity and output signal that corresponds to the concentration of the gas



Specification of MQ2 Smoke Sensor

1. **Operating Voltage:**
   - Typically operates at 5V DC.

2. **Detectable Gases:**
   - Methane (CH4)
   - Butane (C4H10)
   - LPG (Liquefied Petroleum Gas)
   - Smoke
   - Other flammable gases

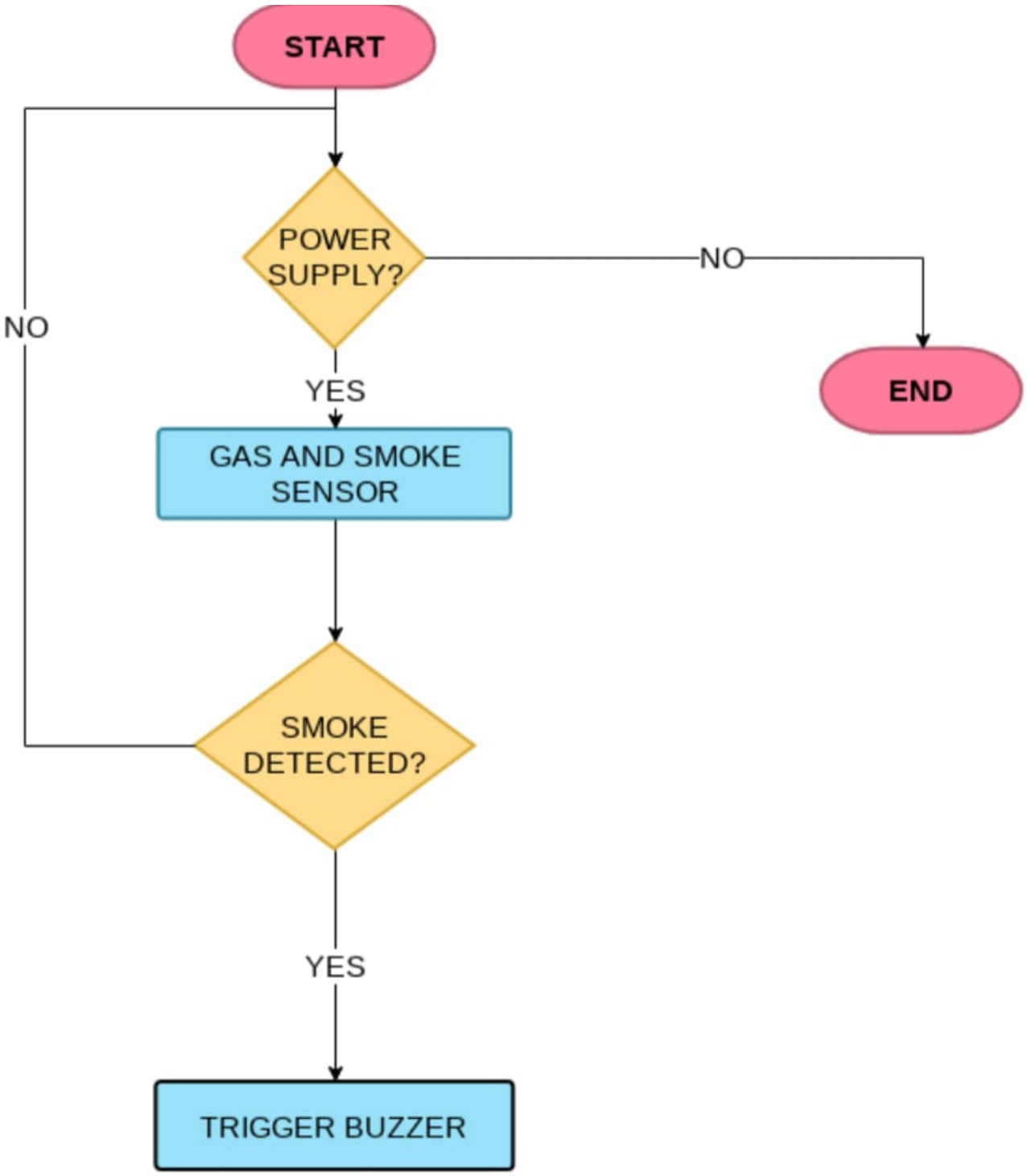
3. **Detection Range:**
   - Concentration range for most gases: 300 to 10,000 ppm (parts per million)
   - Detection range for smoke: 20 to 200 ppm

4. **Output Signal:** Analog output voltage proportional to the gas concentration in the air.

Buzzer

A buzzer is an electromechanical device that converts electrical signals into audible sound waves. It typically consists of a coil of wire (electromagnet) and a flexible diaphragm. When an electrical current passes through the coil, it creates a magnetic field that causes the diaphragm to vibrate, producing sound. There are two main types of buzzers: active and passive. Active buzzers generate sound with an integrated oscillator circuit, while passive buzzers require an external oscillating signal. Buzzers are commonly used in alarm systems, notification devices, and warning systems to provide audible alerts and notifications in various applications.

Flowchart-


START
POWER SUPPLY?
NO
END
NO
YES
GAS AND SMOKE SENSOR
SMOKE DETECTED?
YES
TRIGGER BUZZER

Algorithm:-

Initialization and Setup

1. Include Libraries

- Include necessary libraries for Wi-Fi connectivity (**ESP8266WiFi.h**) and Firebase communication (**Firebase_ESP_Client.h**).

2. Define Network and Firebase Credentials

- Set up constants for Wi-Fi SSID, password, Firebase API key, and database URL.

3. Initialize Pins and Variables

- Define pin assignments (**buzzer** for output, **smokeA0** for analog input).
- Set a threshold value (**sensorThres**) to determine when the sensor detects smoke/gas.

4. Setup Function (setup()):

- Initialize serial communication for debugging (**Serial.begin(9600)**).
- Connect to the specified Wi-Fi network.
- Configure Firebase authentication (**auth**) with the API key and database URL.
- Attempt to sign up with Firebase using the configured credentials.
- Set a callback function (**tokenStatusCallback**) for token generation status.
- Initialize Firebase connection (**Firebase.begin(&config, &auth)**).

Main Loop (loop())

1. Analog Sensor Reading:

- Read analog value from the smoke sensor (**analogRead(smokeA0)**).

2. Data Transmission to Firebase:

- Check if Firebase is ready (**Firebase.ready()**), sign-up was successful (**signupOK**), and a certain time interval has elapsed since the last data transmission (**sendDataPrevMillis**).
- If conditions are met, update the Firebase RTDB with the current sensor value (**Firebase.RTDB.setInt(&fbdo, "fire", value)**).

3. Threshold Check:

- Compare the sensor reading with the predefined threshold (**sensorThres**).
- If the sensor value exceeds the threshold, activate the buzzer (**tone(buzzer, 1000, 200)**) and set the Firebase RTDB node **fire** to **1**.
- Otherwise, turn off the buzzer (**noTone(buzzer)**) and set the Firebase RTDB node **fire** to **0**.

4. Delay:

- Add a short delay (**delay(100)**) to control the loop execution speed.

Firebase Integration

1. Firebase Setup:

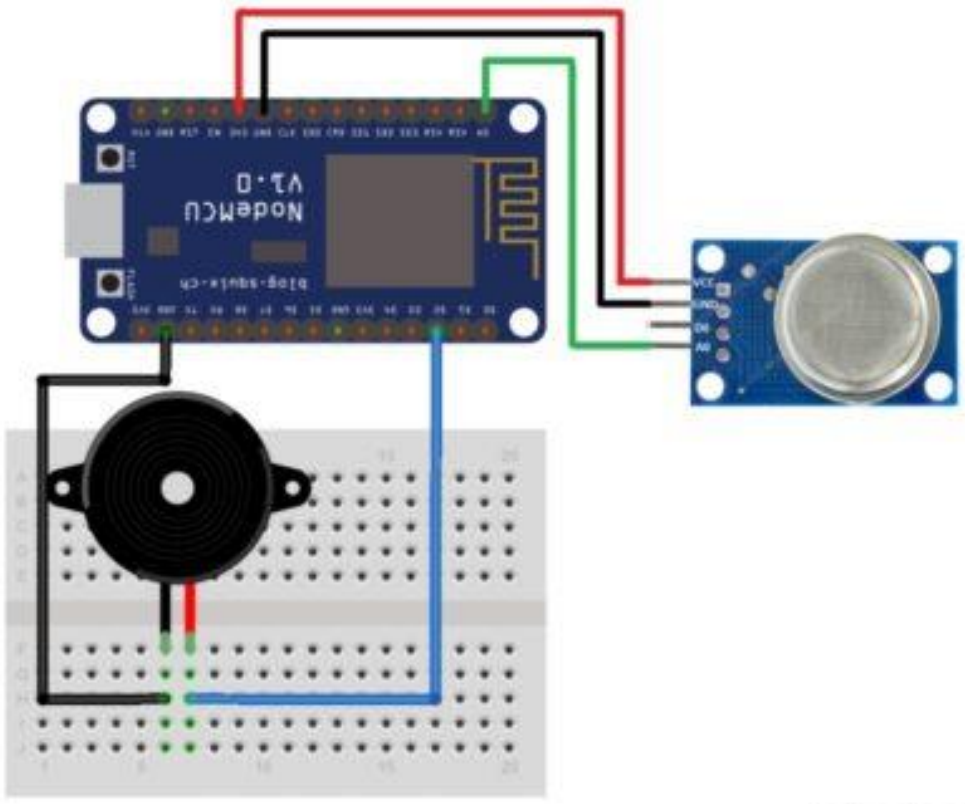
- Initialize Firebase with the provided API key and database URL.
- Use Firebase methods (**Firebase.RTDB.setInt(&fbdo, "fire", value)**) to update data in the RTDB based on sensor readings.

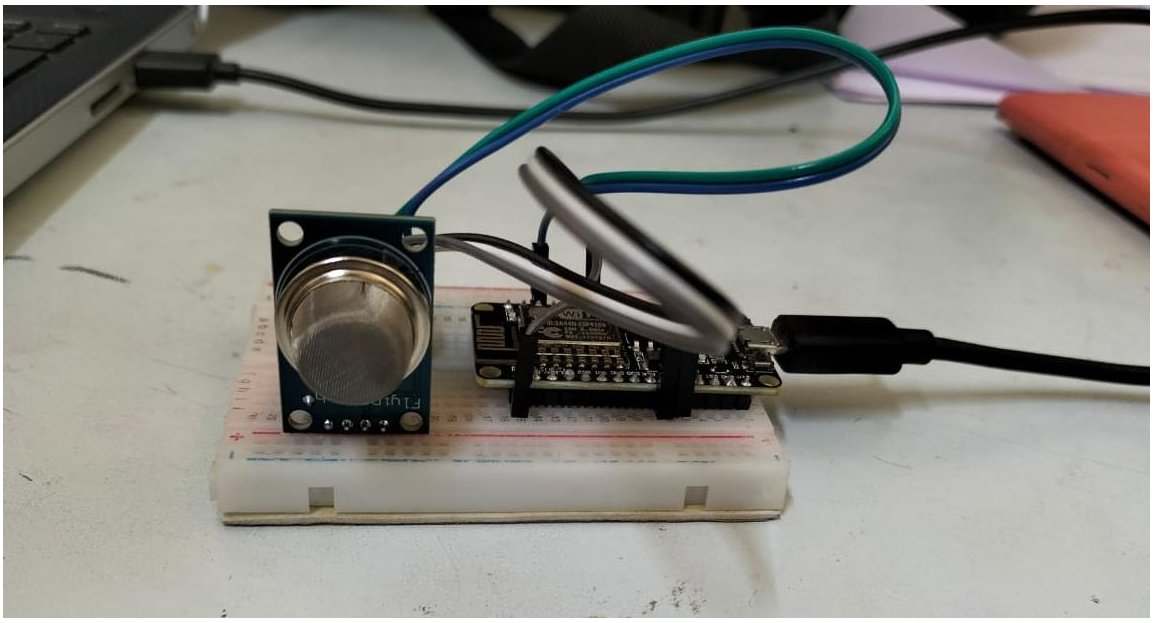
Code for Smoke Detection Using Firebase Console –

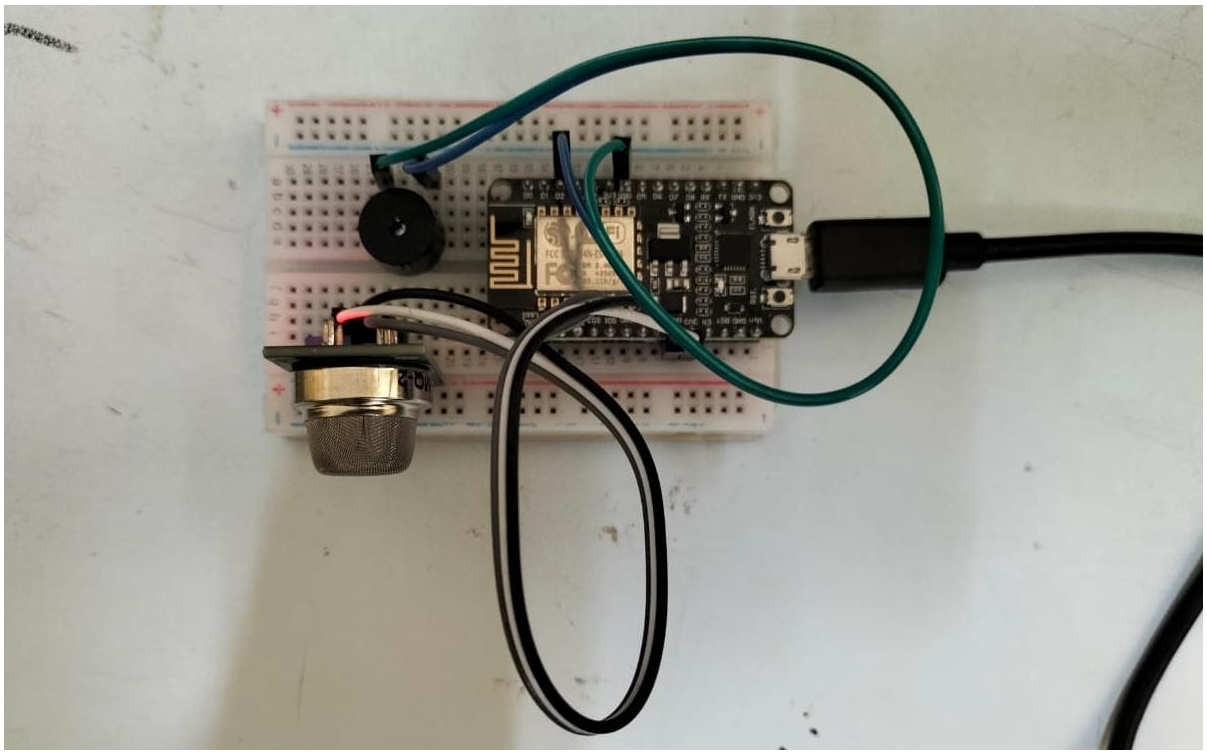
```
#include <ESP8266WiFi.h>
#include <Firebase_ESP_Client.h>

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "HATHWAY_2.4G"
#define WIFI_PASSWORD "H@tH345wAy"

// Insert Firebase project API Key
#define API_KEY "AIzaSyC1DpYF4P6l_JYj4aDHjH7N4aKj64rYEYU"       //API key or
SECRET key

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://smokegas-detector-default-
rtdb.firebaseio.com/"   //url.firebasedatabase.app/

//Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;
bool signupOK = false;
int buzzer = D2;
int smokeA0 = A0;
// Your threshold value. You might need to change it.
int sensorThres = 500;

void setup() {
 pinMode(buzzer, OUTPUT);
 pinMode(smokeA0, INPUT);
 Serial.begin(9600);

WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
```

```
  Serial.println();

  /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;

  /* Sign up */
 if (Firebase.signUp(&config, &auth, "", "")){
    Serial.println("ok");
    signupOK = true;
  }
  else{
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }

  /* Assign the callback function for the long running token generation task */
  config.token_status_callback = tokenStatusCallback;  //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
  Firebase.RTDB.setInt(&fbdo, "fire", 0);
}
void loop() {
 int analogSensor = analogRead(smokeA0);

 Serial.print("Pin A0: ");
 Serial.println(analogSensor);
 delay(1000);

 if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 10 || sendDataPrevMillis
== 0)){
    sendDataPrevMillis = millis();
  }


 // Checks if it has reached the threshold value
 if (analogSensor > sensorThres)
 {
   tone(buzzer, 1000, 200);
   Firebase.RTDB.setInt(&fbdo, "fire", 1);
 }
 else
 {
   noTone(buzzer);
   Firebase.RTDB.setInt(&fbdo, "fire", 0);
 }
 delay(100);
}
```

Circuit Diagram –







Output –

Serial Monitor:-

new_pbl.ino

```
#include <ESP8266WiFi.h>
#include <Firebase_ESP_Client.h>

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "HATHWAY_2.4G"
#define WIFI_PASSWORD "H@tH345wAy"

// Insert Firebase project API Key
#define API_KEY "AIzaSyC1DpYF4P6l_JYj4aDHjH7N4aKj64rYEYU"       //API key or SECRET key

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://smokegas-detector-default-rtdb.firebaseio.com/"   //url.firebasedatabase.app/

//Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
```

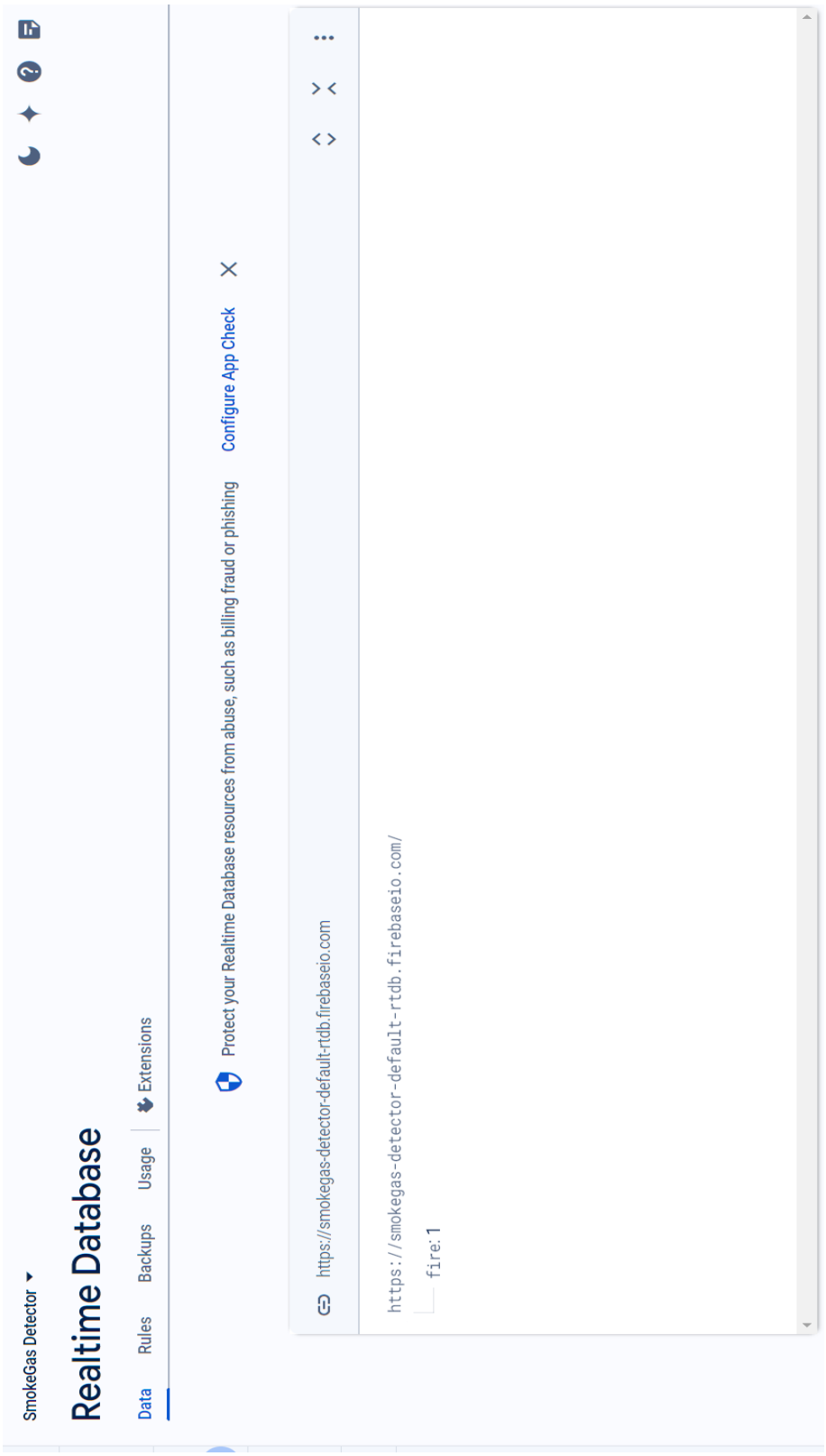
Output Serial Monitor ×

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM3')

```
Pin A0: 729
Pin A0: 727
Pin A0: 726
Pin A0: 723
Pin A0: 721
Pin A0: 720
Pin A0: 717
Pin A0: 716
Pin A0: 713
Pin A0: 712
```

```
..........
Connected with IP: 192.168.1.7

ok
Pin A0: 469
Pin A0: 505
Pin A0: 535
Pin A0: 560
Pin A0: 580
Pin A0: 597
Pin A0: 610
Pin A0: 622
Pin A0: 632
Pin A0: 640
```

Firebase Console :-



Applications –

The project "Smoke Detection in Home Automation with ESP8266 using IoT" involves integrating a smoke/gas sensor with an ESP8266 microcontroller to detect potential hazards such as fire or gas leaks in a home environment. The ESP8266 is connected to the internet via Wi-Fi and leverages IoT (Internet of Things) technology to transmit sensor data to a cloud-based platform—in this case, Google Firebase Realtime Database (RTDB). This setup enables remote monitoring and control of the sensor from anywhere with internet access, providing enhanced safety and automation capabilities for home environments.

1. Early Smoke Detection: The primary application is to detect smoke or gas leaks early, triggering timely alerts to prevent potential disasters like fires or gas-related accidents.

2. Remote Monitoring: Homeowners can remotely monitor the status of the smoke/gas sensor using a smartphone or web interface. Real-time data updates from the sensor are displayed on the dashboard, indicating normal conditions or potential hazards.

3. Automated Alerts: When the sensor detects smoke or exceeds a predefined threshold (indicating a potential hazard), the system automatically sends alerts to the homeowner's smartphone via push notifications or emails. This allows for immediate action, even if the homeowner is away from home.

4. Integration with Home Automation Systems: The smoke detection system can be integrated with existing home automation setups. For example, it could automatically trigger actions such as activating ventilation systems, turning off gas supplies, or alerting emergency services.

5. Enhanced Safety and Peace of Mind: By leveraging IoT technology, homeowners gain enhanced safety and peace of mind, knowing that potential hazards are being monitored continuously and can be addressed promptly, even remotely.

Conclusion –

The integration of a smoke detection system with ESP8266 and IoT technology offers significant improvements in home safety and convenience. By utilizing cloud-based platforms like Google Firebase RTDB, the system enables remote monitoring, timely alerts, and seamless data transmission. This project demonstrates practical IoT applications, showcasing wireless connectivity and sensor integration for enhanced home automation and safety. The system's scalability and adaptability highlight its potential for future expansion and broader smart city applications, ultimately contributing to a safer and more connected living environment.

Key Achievements:

1. Enhanced Safety Monitoring: Real-time detection of smoke or gas enhances home safety.

2. Remote Accessibility and Control: Allows homeowners to monitor and control the system remotely using internet-connected devices.

3. Timely Alerts and Notifications: Sends immediate alerts to homeowners upon detecting potential hazards, facilitating quick response.

4. Integration with Cloud-Based Platforms: Utilizes Google Firebase RTDB for seamless data transmission and storage, enabling remote access and scalability.

5. Contribution to Home Automation: Integrates seamlessly with existing home automation setups, enhancing overall home management.

6. Scalability and Future Expansion: Designed for scalability, allowing for integration of additional sensors or advanced analytics.

Course Outcome –

The successful completion of this project has significantly contributed to the fulfilment of the following course outcomes.

CO-1: Identify the IoT Components and its capabilities

CO-6 : Design an IOT application with ML and Arduino /Raspberry Pi.

References –

- **https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/**
- **https://github.com/topics/mq2-gas-sensor**
- **https://www.researchgate.net/publication/362892981_The_Design_of_Microcontroller_Based_Early_Warning_Fire_Detection_System_for_Home_Monitoring**