**IoT-BASED FOREST FIRE DETECTION SYSTEM**

**IOT PROJECT**

**REVIEW**

**SUBMITTED BY,**

**BATCH – 7C**

**PERURI VAMSI KRISHNA PADI HARIKA**

**21331A05E2 21331A05D3**

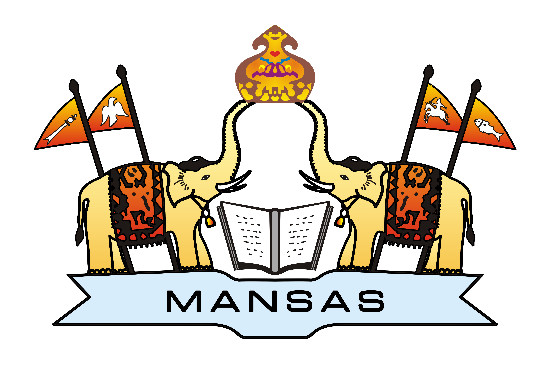
**SABA TANVEER KHAN PATHIVADA GANESH**

**21331A05F0 21331A05D8**

## Under the Supervision of

**Mrs. K. Sobha Rani**

**Professor**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MVGR COLLEGE OF ENGINEERING (AUTONOMOUS)**

**VIZIANAGARAM-535005, AP (INDIA)**

**(Accredited by NBA, NAAC, and Permanently Affiliated to**

**Jawaharlal Nehru Technological University GV)**

**CONTENTS:**

1. Abstract 3
2. Introduction 3

2.1 Objectives 4

1. Problem Statement 4
2. Related Work
   1. IoT-Based Fire Detection 4
   2. Wireless Sensor Networks (WSN) 4
   3. Hybrid Systems 5
3. Proposed Work
   1. Flame Detection 5
   2. Data Processing 5
   3. Immediate Alerts 5
   4. Data Storage 5
   5. Scalability 5
4. Requirements
   1. Hardware Requirements 6
   2. Software Requirements 6
5. Implementation and Methodology
   1. Components Gathering 6
   2. Hardware Connection 6
   3. Software Installation 7
   4. Developing Firmware 7
   5. Flask Application Creation 7
   6. Database Integration 7
   7. Test the System 7
   8. Deploy the System 7
   9. Performance Monitoring 8
6. Result 8
7. References 9
8. **ABSTRACT**

Forest fires pose a significant threat to ecosystems, wildlife, and human life, making early detection crucial for minimizing damage. This project creates a simple system that detects forest fires in real-time and displays the alert on the screen. The system employs flame detectors to monitor fire hazards, with a NodeMCU serving as the central processing unit. When a fire is detected, the NodeMCU via Wi-fi processes the data and communicates with a Flask server, and the detected fire events are stored in a database by integrating MongoDB with a Server for future analysis. The server, in turn, displays an alert notification. This system is affordable, easy to expand, and helps ensure quick action to prevent major fire damage.

1. **INTRODUCTION**

Forest fires are a major environmental concern, causing significant destruction to ecosystems, wildlife, and human lives. In recent years, the frequency and intensity of these fires have increased, primarily due to climate change and human activities. Early detection and timely intervention are critical to controlling the spread of these fires and minimizing their impact. Traditional methods of fire detection, such as satellite imaging or human patrols, often suffer from delayed detection, especially in remote or densely forested areas.

To address this challenge, this project proposes an IoT-based forest fire detection system that can monitor forest areas in real-time and send immediate alerts when a fire is detected. The system uses flame detectors to identify fire hazards and a NodeMCU microcontroller to process the sensor data and communicate with a server. When a fire is detected, the system automatically triggers alerts.

The integration of Internet of Things (IoT) technology makes the system efficient, scalable, and easy to deploy over large areas. By utilizing wireless communication, the system provides a continuous and real-time monitoring solution. This innovative approach aims to reduce the risk and impact of forest fires through early detection and prompt notification, offering a more reliable alternative to traditional fire monitoring methods.

* 1. **Objectives:**
* Early detection of forest fires using sensors.
* Cost-effective and scalable system to monitor large areas.
* Utilize IoT-based technology for processing and communication.

1. **PROBLEM STATEMENT**

Forest fires pose a significant threat to ecosystems and safety, often going undetected until they escalate. Traditional monitoring methods are inadequate in remote areas, highlighting the need for an automated solution. This project aims to develop an IoT-based fire detection system that enables early identification and alerts, improving response times and enhancing forest fire management

1. **RELATED WORK**

Many projects have explored methods to detect forest fires using technology. Traditional approaches, like satellite imaging, can spot fires from above but often take time to send alerts, which can delay response efforts.

* 1. **IoT-Based Fire Detection:**

Recent projects have used Internet of Things (IoT) technology to create real-time fire detection systems. For example, some systems combine flame sensors with microcontrollers like NodeMCU to monitor for fires. When a fire is detected, these systems can send instant alerts to server.

* 1. **Wireless Sensor Networks (WSN):**

Other projects have used networks of sensors to monitor environmental conditions such as temperature and humidity. These networks can provide useful data for fire detection but often require many sensors and can be expensive to set up.

* 1. **Hybrid Systems:**

Some solutions combine IoT and WSNs to improve fire detection accuracy. However, they can be complex and may still face challenges like data overload and reliability in large areas.

1. **PROPOSED WORK**

This system uses flame sensors and a NodeMCU microcontroller to provide real-time monitoring and alerting. The key components of the proposed work include:

* 1. **Flame Detection:**

The system will utilize flame sensors to continuously monitor for signs of fire in forested areas. When a flame is detected, the sensor will send a signal to the NodeMCU.

* 1. **Data Processing:**

The NodeMCU will process the information from the flame sensor and determine if there is a fire. If a fire is confirmed, the NodeMCU will communicate with a flask server.

* 1. **Immediate Alerts:**

The flask server will display alert, ensuring a quick response to the fire.

* 1. **Data Storage:**

The Fire Events which was detected by the sensor is stored in the database for future analysis by using MongoDB. The MongoDB is integrated with the Server.

* 1. **Scalability**

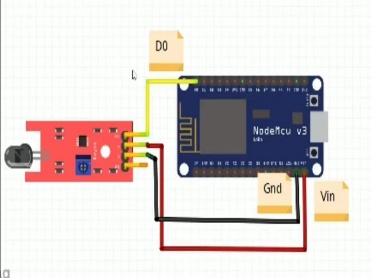
The system can be expanded by adding more sensors and NodeMCUs to cover larger areas, making it a flexible solution for different forest sizes.

1. **REQUIREMENTS**
   1. **Hardware Requirements:**

* Flame Sensor: Sensors that can detect the presence of fire or flames.
* NodeMCU: A microcontroller with Wi-Fi capabilities to process data from the sensors and communicate with the server.
* Jump Wires: Connecting Flame Sensors to NodeMCU
  1. **Software Requirements:**
* Python: Used for server-side code to process fire detection events.
* Arduino IDE: For writing embedded C++ code to program the NodeMCU.
* Flask: Web framework for building the server and displaying alert.
* MongoDB: For Storing Fire events detected by the sensor

1. **IMPLEMENTATION AND METHODOLOGY**
   1. **Components Gathering:**

* Collect all required hardware: fire detection sensor, NodeMCU, power supply, and connecting cables.
  1. **Hardware Connection:**
* Wire the fire detection sensor to the NodeMCU.
* Ensure proper connections according to the sensor's specifications.

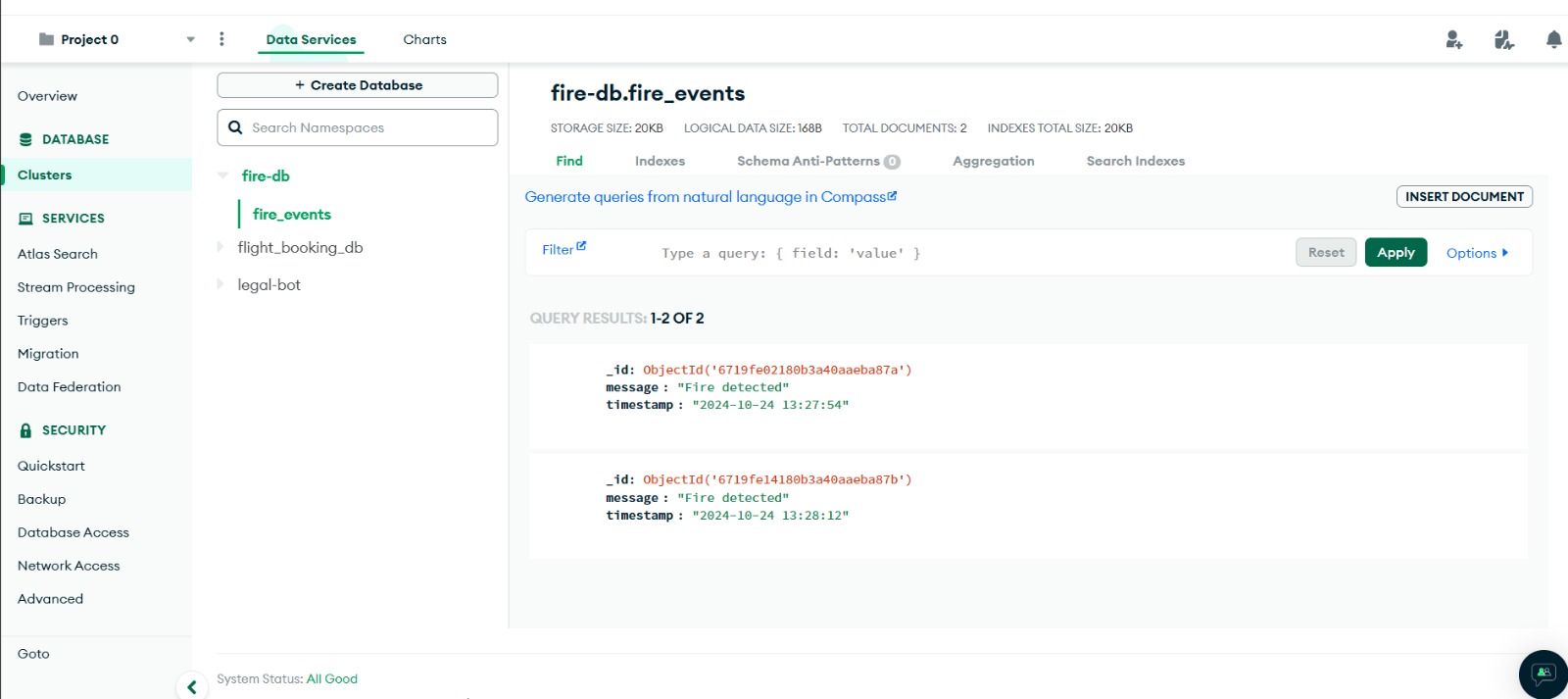


* 1. **Software Installation:**
* Download and install the Arduino IDE for programming the NodeMCU.
* Set up a Flask environment on your server (install Flask required libraries)
  1. **Developing Firmware:**
* Write and upload the NodeMCU code using the Arduino IDE. The code should:
  + Read data from the fire detection sensor.
  + Send alerts to the Flask server when fire conditions are detected.
  1. **Flask Application Creation:**
* Develop the Flask backend to:
  + Receive alerts from the NodeMCU.
  + Process and display alert messages on a web interface.
  1. **Database Integration:**
* Integrate MongoDB to store the detected fire events for future analysis.
  1. **Test the System:**
* Test the setup in a controlled environment.
* Simulate fire conditions to ensure the system detects and sends alerts correctly.
* Adjust sensor calibration as necessary.
  1. **Deploy the System:**
* Install the entire system in the target forested area, ensuring the sensor is mounted securely and powered appropriately.
* Connect the NodeMCU to a reliable Wi-Fi network.
  1. **Performance Monitoring:**
* Regularly check the system for alerts and functionality.
* Perform maintenance and updates as needed to ensure reliability

1. **RESULT**

****

****

****

1. **REFERENCES**

* <https://www.mongodb.com/cloud/atlas/register>
* <https://flask.palletsprojects.com/>
* <https://www.arduino.cc/en/software>