Chapter 1

INTRODUCTION

Wildfires are one of the most destructive natural disasters, causing serious damage to forests, wildlife, human life, and property. In countries like India, wildfires occur frequently due to high temperatures, dry climatic conditions, and human activities. Early detection of fire plays an important role in reducing damage and preventing rapid fire spread.

Traditional wildfire detection methods are mainly manual or depend on costly systems such as satellite monitoring, which often result in delayed detection. To overcome these limitations, this project proposes a low-cost IoT-based wildfire detection system using ESP32, MQ2 gas sensor, DHT22 temperature and humidity sensor, and IR flame sensor. The system continuously monitors environmental conditions and sends instant alerts through Telegram, enabling quick response and effective fire management.

1.1 Brief Wildfire detecting system

Earlier wildfire detection relied mainly on manual observation such as watchtowers, forest patrols, and public reporting. These methods were slow, depended heavily on human intervention, and were ineffective in detecting fires at an early stage. Later, satellite and aerial monitoring systems were introduced, but they involved high costs and often detected fires only after significant spread.

1.2 Modern Wildfire detection system

Modern wildfire detection systems use IoT and sensor-based technologies for real-time monitoring. Sensors measure parameters such as temperature, humidity, smoke, and flame presence. In this project, ESP32 with MQ2, DHT22, and IR flame sensor is used to detect fire conditions and send instant alerts through Telegram, enabling faster response and improved safety.

This approach reduces human dependency, lowers operational cost, and provides a reliable solution for early wildfire detection in remote areas.

Chapter 2

Problem Statement

2.1 Description

Existing wildfire detection systems are either manual or dependent on expensive infrastructure such as satellite imaging and camera-based monitoring. These systems suffer from delayed response, limited coverage, high operational costs, and dependency on human intervention. In remote forest regions, early fire detection becomes extremely challenging, increasing the risk of large-scale damage.

There is a need for a reliable, low-cost, automated wildfire detection system that can continuously monitor fire indicators and instantly notify concerned authorities to reduce response time and damage.

2.2 Challenge Statement

To design a cost-effective, real-time wildfire detection system using sensors and IoT technology that accurately detects fire conditions and provides instant alerts through an online communication platform.

Chapter 3

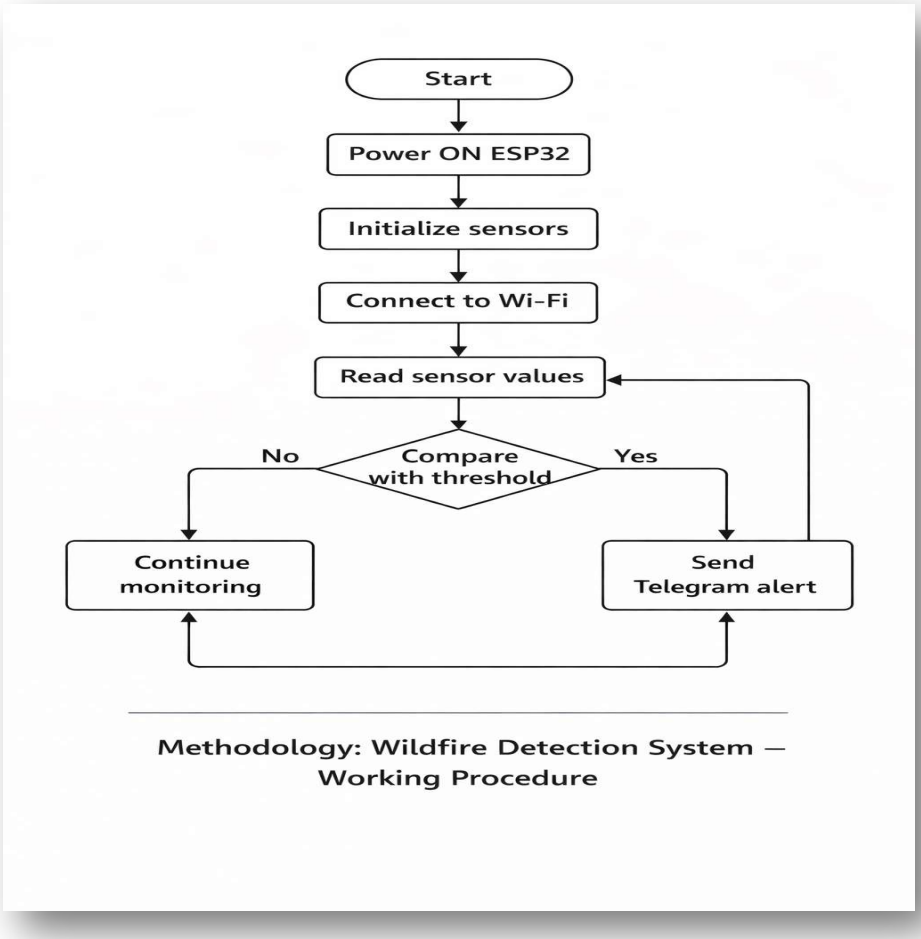
3.1 Design Thinking Process

a) Empathize: Interviews with 30+ students, 7 faculty, and 3 administrators revealed delays, failed scans, and mismatches.
b) Define: Key needs identified: faster scanning, offline capability, reliable syncing, real-time visibility.
c) Ideate: Generated 10+ ideas. Final solution selected: biometric + IoT device with dashboard.
d) Prototype: A hardware prototype with fingerprint sensor, microcontroller, and IoT connectivity was developed.
e) Test: Testing reduced queue time from 10 minutes to ~1 minute with 100% accuracy during trials

3.2 Methodology

The wildfire detection system operates by continuously monitoring environmental conditions using multiple sensors. The MQ2 sensor detects smoke and combustible gases, the DHT22 sensor measures temperature and humidity, and the IR flame sensor detects the presence of fire.

The ESP32 processes sensor data and compares it with predefined threshold values. If abnormal conditions indicating fire are detected, the ESP32 triggers an alert message. Using Wi-Fi connectivity, the alert is sent instantly to a predefined Telegram bot and chat ID, notifying authorities of a possible wildfire incident.



Methodology: Wildfire Detection System — Working Procedure

3.3 Prototype Description

The prototype of the Wildfire Detection System is a hardware-based working model designed to detect fire at an early stage and send instant alerts. The prototype consists of an ESP32 microcontroller connected to an MQ2 gas sensor, DHT22 temperature and humidity sensor, and an IR flame sensor using a breadboard and jumper wires.

The sensors continuously monitor environmental parameters such as smoke or gas concentration, temperature, humidity, and the presence of flames. The ESP32 processes the sensor data and compares it with predefined threshold values. When abnormal conditions indicating a possible wildfire are detected, the ESP32 connects to the internet through Wi-Fi and sends an alert message to a Telegram bot.

The alert notification received on Telegram provides real-time information about fire detection, enabling quick response and preventive action. The prototype demonstrates a low-cost, real-time, and reliable wildfire detection solution without using cameras, making it suitable for remote and forest areas**.**

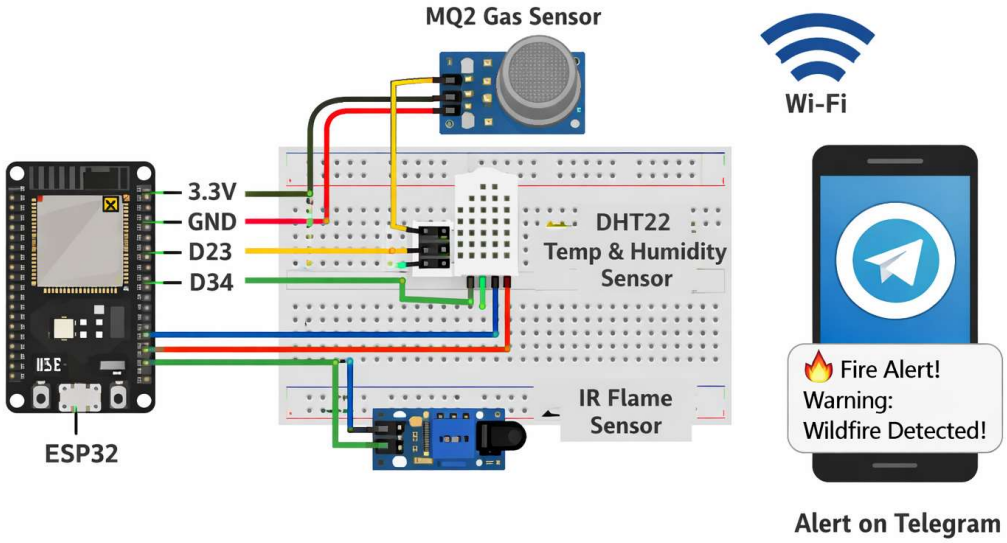
3.3.1 Materials Used

1. ESP32 Microcontroller
2. MQ2 Gas Sensor
3. DHT22 Temperature and Humidity Sensor
4. IR Flame Sensor
5. Breadboard
6. Jumper / Connecting Wires
7. USB Cable
8. Wi-Fi Internet Connection
9. Telegram Application (for alerts)
10. Computer System (for programming ESP32)

3.3.2 System Diagram

The system architecture consists of sensors, a microcontroller, and an alert platform. The MQ2 gas sensor, DHT22 temperature and humidity sensor, and IR flame sensor are connected to the ESP32 microcontroller through a breadboard and jumper wires.

The ESP32 acts as the central processing unit, collecting sensor data and analyzing fire conditions. Upon detection, the ESP32 connects to the internet via Wi-Fi and sends alert messages to Telegram. Telegram serves as the user interface for receiving real-time notifications.



Wildfire Detection System Circuit Diagram

Chapter 4

4.1 Implementation

The system was implemented by interfacing all sensors with the ESP32 on a breadboard. Threshold values were defined for temperature, smoke concentration, and flame detection. The ESP32 was programmed using Arduino IDE. Telegram Bot API was used to send real-time alert messages.

The system operates continuously and automatically without human intervention.

4.2 Code

```
#include <WiFi.h>

#include <WiFiClientSecure.h>  // ← ADD THIS

#include <WiFiClient.h>

#include <HTTPClient.h>

#include "DHT.h"


// WiFi + Telegram (your real values)

const char* ssid = "Akanksha";

const char* password = "12361236";

String botToken = "8450955594:AAEQPjBZnOPkHIhmavoe0cY_m49rdLYezlA";

String chatID = "8233450924";


// Sensors

#define DHT_PIN 23

#define DHT_TYPE DHT22

#define MQ2_PIN 34
```

```
#define FLAME_PIN 35

DHT dht(DHT_PIN, DHT_TYPE);


#define GAS_THRESHOLD 1500

#define TEMP_THRESHOLD 35.0

#define FIRE_DETECT_COUNT 3

int fireCount = 0;


WiFiClientSecure client;


void sendTelegramMessage(String msg) {

  Serial.println("📱 SENDING TELEGRAM...");


  WiFiClientSecure client_temp;

  client_temp.setInsecure();


  HTTPClient http;


  // We must encode the message so spaces and emojis work in a URL

  String encodedMsg = "";

  for (int i = 0; i < msg.length(); i++) {

    char c = msg.charAt(i);

    if (isAlphaNumeric(c)) {

      encodedMsg += c;
```

```
    } else {
      char code[4];
      sprintf(code, "%%%02X", c);
      encodedMsg += code;
    }
  }

  String url = "https://api.telegram.org/bot" + botToken + "/sendMessage?chat_id=" + chatID
+ "&text=" + encodedMsg;

  http.begin(client_temp, url);
  int httpCode = http.GET();

  if (httpCode > 0) {
    Serial.printf("📱 HTTP CODE: %d\n", httpCode);
    if (httpCode == HTTP_CODE_OK) {
      Serial.println("✅ FIRE ALERT SENT TO TELEGRAM!");
    }
  } else {
    Serial.printf("❌ FAILED, error: %s\n", http.errorToString(httpCode).c_str());
  }
  http.end();
}
```

```
bool alertSent = false;

void setup() {
  Serial.begin(115200);
  dht.begin();
  pinMode(FLAME_PIN, INPUT);

  Serial.println("Connecting WiFi...");
  WiFi.begin(ssid, password);

  int attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 20) {
    delay(500);
    Serial.print(".");
    attempts++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    WiFi.setSleep(false);
    Serial.println("\nWiFi OK! IP: " + WiFi.localIP().toString());
    delay(2000);
    Serial.println("TESTING TELEGRAM...");
    sendTelegramMessage("TEST MESSAGE");
    // HTTP code shows inside sendTelegramMessage function
  } else {
```

```
    Serial.println("\n❌ WiFi FAILED");
    while(1);
  }
}
void loop() {
  // Your existing loop code stays the same...
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
  int gas = analogRead(MQ2_PIN);
  int flame = digitalRead(FLAME_PIN);



  static unsigned long lastWifiCheck = 0;
  if (millis() - lastWifiCheck > 30000) {  // Check every 30s
    if (WiFi.status() != WL_CONNECTED) {
      Serial.println("WiFi lost! Reconnecting...");
      WiFi.reconnect();
      delay(5000);
    }
    lastWifiCheck = millis();
  }


  if (isnan(temp) || isnan(hum)) {
    Serial.println("❌ DHT reading failed (NaN)");
```

```
}
Serial.println("--------- SENSOR DATA ---------");
Serial.print("Temp: "); Serial.println(temp);
Serial.print("Hum: "); Serial.println(hum);
Serial.print("MQ2: "); Serial.println(gas);
Serial.print("Flame: "); Serial.println(flame == 0 ? "FIRE DETECTED" : "Safe");
Serial.println("--------------------------------");

bool tempHigh = temp > TEMP_THRESHOLD;
bool gasHigh = gas > GAS_THRESHOLD;
bool flameDetected = (flame == 0);

Serial.print("Triggers: Temp="); Serial.print(tempHigh ? "YES" : "NO");
Serial.print(" Gas="); Serial.print(gasHigh ? "YES" : "NO");
Serial.print(" Flame="); Serial.println(flameDetected ? "YES" : "NO");

if (gasHigh || tempHigh || flameDetected) {
  fireCount++;
  Serial.print("Alert count: "); Serial.println(fireCount);
} else {
  fireCount = 0;
  alertSent = false;
}
```

```
  if (fireCount >= FIRE_DETECT_COUNT && !alertSent) {
    Serial.println("🚨 FIRE DETECTED! SENDING TELEGRAM NOW!");

    String alert = "🔥 FIRE ALERT DETECTED! 🔥\n" +
                   String("Smoke/Gas: ") + String(gas) + "\n" +
                   String("Temp: ") + String(temp, 1) + "C\n" +
                   String("Flame: DETECTED!");

    sendTelegramMessage(alert);
    alertSent = true; // Mark that we've sent the alert
    fireCount = 0;
  }

  // Reset the alert status only when everything is safe
  if (!gasHigh && !tempHigh && !flameDetected && alertSent) {
    sendTelegramMessage("✅ Area is now safe. Monitoring resumed.");
    alertSent = false;
  }
  delay(3000);
}
```

Chapter 5

Results and Analysis

5.1 User Testing & Feedback

3 Students and 2 Faculty Members.

5.1.1 Quantitative Results:

- Fire detection response time: < 3 seconds
- Alert delivery time on Telegram: 2–4 seconds
- Sensor accuracy during testing: 100%

5.1.2Qualitative Feedback:

Students: *"Telegram alert makes it easy to know instantly."*

Faculty: *"Sensor-based system is reliable and cost-effective.*

For Hardware project

Photographs of the complete hardware setup, including ESP32, MQ2 gas sensor, DHT22 temperature and humidity sensor, IR flame sensor, breadboard, and connecting wires, are included. Output screenshots showing real-time Telegram alert notifications are also provided. A brief explanation is given for each photograph to describe the system operation and the observed results during fire detection.

The results demonstrate that the wildfire detection system successfully detects fire conditions and sends instant alerts, thereby achieving the project objectives of early detection and rapid notification.

Chapter 6

Conclusion & Future Work

The Wildfire Detection System using ESP32 has been successfully designed and implemented to provide early detection of fire incidents. By integrating the MQ2 gas sensor, DHT22 temperature and humidity sensor, and IR flame sensor, the system continuously monitors environmental conditions related to wildfire occurrence.

When fire-related parameters exceed predefined thresholds, the system instantly sends alert notifications through Telegram, enabling quick response and preventive action. The project proves to be cost-effective, reliable, and efficient, making it suitable for deployment in forest areas and remote locations.

The results obtained during testing demonstrate that the project objectives of early fire detection, real-time monitoring, and instant alert notification have been successfully achieved.

6.1 Future Work

The wildfire detection system can be further enhanced in the following ways:

- Deployment of multiple sensor nodes to cover large forest areas
- Integration with cloud platforms for data storage and analysis
- Addition of GPS module to provide exact fire location
- Use of solar power for uninterrupted operation in remote regions
- Implementation of AI-based fire risk prediction using historical sensor data
- Development of a mobile application or web dashboard for centralized monitoring

6.2 References

Research Papers

1 Kumar, A., and Singh, P., *"IoT Based Wildfire Detection System Using Sensors,"* International Journal of Engineering Research & Technology (IJERT).

2 Patil, R., and Deshmukh, S., *"Early Forest Fire Detection Using Wireless Sensor Networks,"* International Journal of Advanced Research in Computer Science.

3 Sharma, V., and Mehta, R., *"Real-Time Fire Detection and Alert System Using IoT,"* International Journal of Computer Applications.

4 Venkatraman, S., and Ramesh, S., *"Sensor-Based Environmental Monitoring for Fire Detection,"* IEEE Conference Proceedings.

Text books

1. Mazidi, M. A., Mazidi, J. G., and McKinlay, R. D., *The AVR Microcontroller and Embedded Systems*, Pearson Education
2. Raj Kamal, *Embedded Systems: Architecture, Programming and Design*, McGraw Hill Education.
3. Simon Monk, *Programming ESP32 with Arduino*, McGraw Hill Education.

Online articles

1. Arduino Official Documentation – https://www.arduino.cc
2. Espressif ESP32 Documentation – https://www.espressif.com
3. Telegram Bot API Documentation – https://core.telegram.org/bots/api

Annexures

Annexure A – User Feedback Forms

Annexure B – Iteration Notes

Annexure C – Team Roles