iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70540

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Warehouse Firefighting Rover

Hemalatha G., Abinaya G.P. , Dharneesh B., Rakshitha D., Sivachalapathy V., Uma Maheshwari S.
Department of Mechatronics, PSG Polytechnic College, Peelamedu, Coimbatore

Abstract: ***This paper presents the design and implementation of an autonomous firefighting rover aimed at improving fire response capabilities in hazardous, high-risk, and inaccessible environments, such as warehouses, industrial plants, and remote areas. The rover is engineered to operate independently, leveraging a suite of flame and smoke sensors for early fire detection, alongside a camera module for real-time video monitoring. These sensors work in coordination with a microcontroller-based control system, which governs the rover's movements, fire detection logic, and suppression mechanisms.***

One of the key innovations in this system is its integration with cloud-based wireless communication, enabling seamless remote access and control. Through this IoT-enabled interface, users can monitor the rover's environment in real time, send operational commands, and receive alerts on fire incidents from any location. All system data is stored and managed via the cloud, allowing for post-incident analysis and continuous performance optimization.

By automating fire detection and suppression, the rover significantly reduces the need for human intervention in dangerous situations, thereby minimizing the risk to life. Its ability to navigate complex terrains and deliver a rapid, precise response to fire outbreaks makes it a valuable asset for modern fire safety infrastructure, advancing both efficiency and reliability in emergency management.

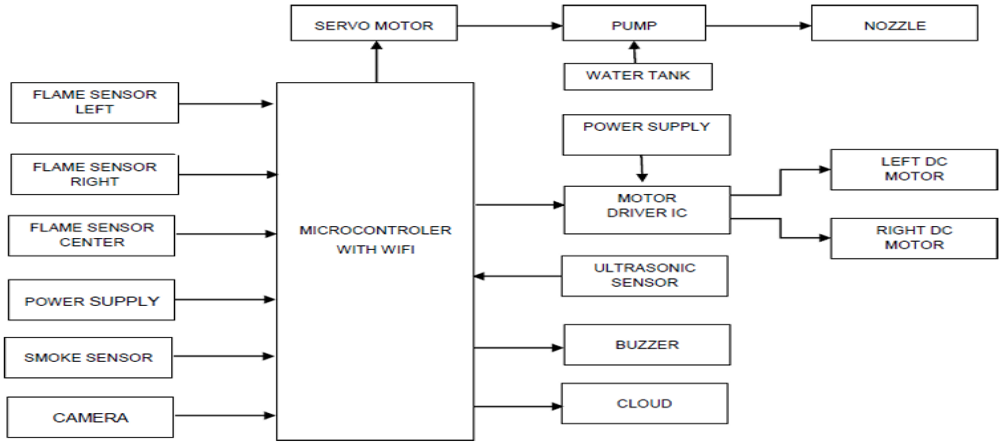
Keywords: IoT, Warehouse, Smoke sensor, Flame sensor, NodeMCU, Automation, Cloud monitoring

I. INTRODUCTION

Fire safety is a major concern in warehouse settings due to the potential for extensive damage to assets, inventory, and human life. Conventional fire protection systems typically utilize basic sensors that trigger alarms upon detecting a fire. While these alarms provide an early alert, extinguishing the fire still requires manual effort, where responders must physically locate and suppress the fire using equipment like water or sand. This process is not only time-intensive but also poses significant danger to personnel. Moreover, traditional systems lack the ability to accurately identify the fire's exact location or to initiate automatic suppression, leading to delays and heightened risk.

This project introduces an advanced fire safety solution that overcomes these shortcomings. By incorporating IoT-enabled sensors and autonomous firefighting robots, the system offers real-time fire detection, precise location identification, and automated response capabilities. This modern approach improves safety, accelerates response times, and reduces risk to both people and property. The subsequent sections outline the design, operation, and potential benefits of this intelligent fire safety system for warehouse environments.

II. BLOCK DIAGRAM



©IJRASET: All Rights are Reserved

III. LITERATURE REVIEW

1) Autonomous Firefighting Robots: A Review of Current Technologies

Author(s): X. Li, J. Wang, & R. Zhang (2020)

This paper reviews the development of autonomous firefighting robots that utilize a combination of sensors, actuators, and AI-based navigation systems. It addresses key challenges such as mobility constraints, the accuracy of fire detection, and real-time decision-making. The authors suggest that employing sensor fusion and cloud-based data processing can enhance performance—strategies that are also applied in our project.

2) Application of IoT in Fire Safety Systems

Author(s): A. Sharma & P. Kumar (2019)

This study examines how Internet of Things (IoT) technology can improve fire detection and response mechanisms. It underscores the benefits of cloud integration, real-time surveillance, and predictive analytics in managing fire hazards. These insights directly influence our project's implementation of cloud-enabled remote monitoring and control features.

3) Sensor-Based Fire Detection Systems for Industrial Environments

Author(s): M. Rodriguez, S. Lee, & T. Chen (2021)

The research compares various fire detection sensors—including flame detectors, smoke sensors, and thermal imaging devices—and concludes that using a combination of sensors greatly enhances detection accuracy. Our Firefighting Rover adopts this multi-sensor approach by integrating both flame and smoke sensors, along with a camera for precise fire localization.

4) Obstacle Avoidance in Mobile Robotics Using Ultrasonic Sensing

Author(s): K. Patel & D. Singh (2018)

This paper explores different techniques for autonomous navigation in complex and cluttered environments, demonstrating the effectiveness of ultrasonic sensors for obstacle detection and avoidance. Our rover incorporates this approach to navigate safely within warehouse and industrial spaces.

5) Water Jet and Foam-Based Fire Suppression Systems in Robotics

Author(s): T. Nakamura & J. Brown (2022)

This study compares fire suppression methods such as water jets, CO_2 extinguishers, and foam-based systems, highlighting the superior precision of servo-controlled nozzles. Reflecting this finding, our design integrates a servo-driven water spraying unit to improve the accuracy and effectiveness of fire suppression.

6) Cloud-Connected Fire Safety Systems for Industrial Applications

Author(s): L. Fernandes & P. Reddy (2023)

The paper discusses the advantages of cloud computing in industrial fire safety, emphasizing features such as real-time alerts, remote system access, and analytics-driven decision-making. In line with this, our Firefighting Rover leverages cloud connectivity to enable efficient monitoring and management across large warehouse facilities.

IV. WORKING PRINCIPLE

The Warehouse Firefighting Rover is an advanced, IoT-enabled autonomous robot designed to detect and extinguish fires in hazardous or hard-to-reach environments such as warehouses, industrial facilities, high-rise buildings, and remote locations. At the heart of the system is the ESP32 microcontroller, which coordinates all operations, while the ESP32-CAM provides real-time video monitoring and image-based fire detection. The robot is equipped with flame and smoke sensors, cameras, and intelligent navigation systems, allowing it to accurately identify fire sources and navigate complex terrains.

Motors controlled via a relay module enable precise movement, and an Agri pump handles the water-based fire suppression. Cloud-based connectivity facilitates remote supervision, real-time alerts, and system updates, allowing operators to control the rover safely from a distance. This minimizes the need for human presence in dangerous environments, thereby reducing risk and improving response time.

By integrating smart sensors, robotics, and IoT technologies, the Firefighting Rover offers a fast, efficient, and reliable solution for fire emergencies, enhancing safety and protecting both lives and property.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

V. HARDWARE USED

1) ESP32-CAM:

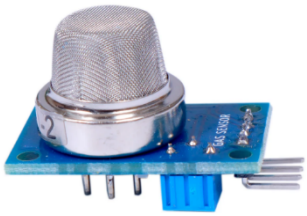
The ESP32-CAM is designed to provide video streaming capabilities and capture real-time images of the warehouse environment.



Powered by an ESP32 dual-core processor, it ensures efficient performance. The camera used is a 2MP OV2640, capable of capturing images at a resolution of up to 1600x1200 pixels. The device supports Wi-Fi connectivity (802.11 b/g/n) and Bluetooth, enabling seamless data transmission. It is equipped with 4MB of Flash memory for storage. The system operates on a 5V DC power supply and includes various GPIO pins for interfacing with other components, offering flexibility for integration into different setups.

2) Smoke Sensor:

Smoke is one of the first indicators of fire, and the sensor helps the Rover identify the presence of smoke, even in the absence of visible flames. The smoke sensor detects changes in the air caused by the presence of smoke particles. When smoke is detected, the sensor sends a signal to the microcontroller, which processes the data and determines if the Rover needs to move toward the source of the smoke. The Rover is set manually and the path is adjusted manually, using its camera to avoid obstacles while navigating towards the fire. This early detection capability is particularly vital in environments where flames may be hidden, such as in rooms filled with dense smoke or in industrial areas where hazardous fumes can obscure visibility. By providing continuous data to the cloud integration system, the smoke sensor enables remote monitoring and control, allowing firefighters to make informed decisions from a safe distance while ensuring the Rover operates effectively in real-time.



3) DC Motor:

In the Warehouse Firefighting Rover, DC motors play a crucial role in enabling mobility and navigation. These motors drive the Rover's wheels, allowing it to move smoothly through both tight indoor spaces and rough outdoor terrain. Controlled by a microcontroller, the motors provide precise movement in all directions—forward, backward, and turning—as required by the Rover. Working together with sensors, the motors help the Rover avoid obstacles by adjusting speed or direction in real time, ensuring safe movement in hazardous environments. Their energy efficiency also supports longer battery life, which is essential for extended operations in areas where recharging is difficult. In short, DC motors are vital to the Rover's ability to move and operate effectively in challenging firefighting scenarios. Additionally, the Rover uses a differential drive system, where two independently controlled DC motors allow for sharp turns and smooth maneuverings. Pulse Width Modulation (PWM) signals are used to control motor speed accurately, enabling smooth acceleration and deceleration. This is especially useful when navigating smoke-filled or unstable areas where sudden movements can be dangerous. The motors are selected based on torque and RPM requirements, ensuring they can handle the Rover's weight and payload, including water tanks or extinguishing equipment. Heat-resistant motor casings are used to ensure performance is not compromised in high-temperature environments. Encoders are sometimes attached to the motors to provide feedback for position tracking and speed monitoring. By integrating these motors with the Rover's onboard control system, the platform achieves reliable, responsive, and intelligent movement essential for real-time firefighting applications.



4) ESP8266 NODE MCU:

ESP8266 Node MCU is an open-source IoT platform that combines a powerful Wi-Fi module with a microcontroller, making it a popular choice for building smart devices and applications. Designed for ease of use, it features enabling rapid prototyping and development. With its built-in Wi-Fi capability, the Node MCU allows developers to connect their projects to the internet seamlessly, facilitating the creation of connected devices that can communicate and be controlled remotely.



The Node MCU board is equipped with the ESP8266 chip, which boasts a 32-bit processor, 80 MHz clock speed, and a variety of GPIO pins (pin configuration of ESP8266 is shown in figure 3.10 for connecting sensors, actuators, and other peripherals. This versatility makes it suitable for a wide range of applications, from home automation and environmental monitoring to smart agriculture and industrial control systems. Programming the Node MCU is straightforward, with support for Arduino IDE, allowing developers to choose their preferred coding environment. Additionally, its compact design and low power consumption make it ideal for battery-operated devices. Overall, the ESP8266 Node MCU is a robust, flexible platform that empowers `creators to build innovative IoT solutions quickly and efficiently.

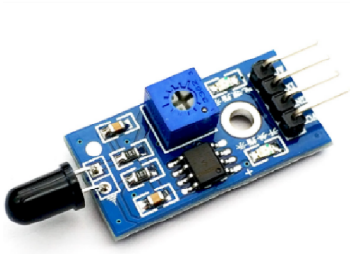
5) Servo Motor:

A key component of the Rover is the servo motor, which is used to adjust the position of mechanical components, such as the water nozzle, based on control signals for precise movement. The servo motor operates within a voltage range of 4.8V to 6V DC, with a torque range of 1.5 to 5 kg/cm, offering a rotation range of either 180 degrees or 90 degrees, depending on the model. Controlled via pulse-width modulation (PWM) signals from the ESP32 controller, it ensures responsive, accurate positioning with variable speed ranging from 0.1 to 0.2 seconds per 60 degrees of rotation. This combination of advanced robotics, sensor technology, and cloud-based control allows the Firefighting Rover to minimize risks to human life, improve response times, and effectively manage fires in diverse environments.



6) Flame Sensor:

The LM393 Flame Sensor detects fire by sensing infrared (IR) radiation in the 760–1100 nm range. It uses a photodiode and an LM393 comparator IC to compare IR levels against a preset threshold (adjustable via potentiometer). When IR from a flame is detected, the digital output (DO) goes LOW (0V), triggering the ESP32 to respond. It includes VCC, GND, and DO pins. A green LED on the sensor glows when fire is detected. This sensor is ideal for locating flame sources in your firefighting robot.



VI. SOFTWARE INTEGRATION

- The Node MCU was programmed to fetch movement commands from an IoT server.
- The ESP32-CAM was programmed to capture images periodically and upload them to a server.
- Error handling and Wi-Fi reconnection mechanisms were implemented to ensure robust operation.

Program:

```
// ROBOT UNIT
int motor = 0;
Servo myServo;
String inData = "";
// Timer interrupt handler for periodic tasks
hw_timer_t * timer = NULL;
volatile bool timerFlag = false;
// Servo control variables
unsigned long lastServoTime = 0;
int servoAngle = 30;
bool servoIncreasing = true;
void IRAM_ATTR onTimer() {
  timerFlag = true;
}
```

```
void setup() {
  Serial.begin(9600);
  delay(10);
  pinMode(MOTOR1, OUTPUT);
  pinMode(MOTOR2, OUTPUT);
  pinMode(MOTOR3, OUTPUT);
  pinMode(MOTOR4, OUTPUT);
  pinMode(PUMP, OUTPUT);
  myServo.attach(4);
  myServo.write(90);
  initWiFi("PROJECT", "withlove", 1);
  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &onTimer, true);
  timerAlarmWrite(timer, 2000000, true);
  timerAlarmEnable(timer);
}
void loop() {
  // Fetch control data from the server
  String url = "/iot-projects/HER24166-ware_house/control/getData.php?data=1";
  String response = requestURL(host, url);
  Serial.println(response);
 // Control motors based on the response
  controlMotors(response);
 // Handle servo motor movement
  handleServoMovement();
  // Reset watchdog timer if necessary
  // esp_task_wdt_reset();
}
void controlMotors(String response) {
  if (response == "N#") {
    Serial.println("FORWARD");
    digitalWrite(MOTOR1, HIGH);
    digitalWrite(MOTOR2, LOW);
    digitalWrite(MOTOR3, HIGH);
    digitalWrite(MOTOR4, LOW);
  }
  else if (response == "S#") {
    Serial.println("BACKWARD");
    digitalWrite(MOTOR1, LOW);
    digitalWrite(MOTOR2, HIGH);
    digitalWrite(MOTOR3, LOW);
    digitalWrite(MOTOR4, HIGH);
  }
  else if (response == "W#") {
    Serial.println("LEFT");
    digitalWrite(MOTOR1, HIGH);
    digitalWrite(MOTOR2, LOW);
    digitalWrite(MOTOR3, LOW);
    digitalWrite(MOTOR4, HIGH);
  }
```

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

```
  else if (response == "E#") {
    Serial.println("RIGHT");
    digitalWrite(MOTOR1, LOW);
    digitalWrite(MOTOR2, HIGH);
    digitalWrite(MOTOR3, HIGH);
    digitalWrite(MOTOR4, LOW);
  }
  else if (response == "C#") {
    Serial.println("STOP");
    digitalWrite(MOTOR1, LOW);
    digitalWrite(MOTOR2, LOW);
    digitalWrite(MOTOR3, LOW);
    digitalWrite(MOTOR4, LOW);
  }
  else if (response == "START#") {
  motor = 1;
  digitalWrite(PUMP, HIGH);
  }
  else if (response == "STOP#") {
  motor = 0;
  myServo.write(90);
  digitalWrite(PUMP, LOW);
  }
}
void handleServoMovement() {
// Check if enough time has passed for the next servo movement
unsigned long currentMillis = millis();
if (currentMillis - lastServoTime >= 100 && motor == 1) {
lastServoTime = currentMillis;  // Update the last time we moved the servo
// Move the servo up or down based on the flag
if (servoIncreasing) {
servoAngle += 5;
if (servoAngle >= 170) {
servoIncreasing = false;
}
} else {
servoAngle -= 5;
if (servoAngle <= 15) {
servoIncreasing = true;
}
}
// Write the new angle to the servo
myServo.write(servoAngle);
Serial.print("Angle: ");
Serial.println(servoAngle);
}
}
// CAMERA UNIT
void setup() {
  Serial.begin(115200);
```

```
Serial.setDebugOutput(true);
Serial.println();
pinMode(LED_PIN, OUTPUT);
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.frame_size = FRAMESIZE_UXGA;
config.pixel_format = PIXFORMAT_JPEG; // for streaming
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.jpeg_quality = 12;
config.fb_count = 1;
if (psramFound()) {
  config.jpeg_quality = 10;
  config.fb_count = 2;
  config.grab_mode = CAMERA_GRAB_LATEST;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.fb_location = CAMERA_FB_IN_DRAM;
}
#if defined(CAMERA_MODEL_ESP_EYE)
pinMode(13, INPUT_PULLUP);
pinMode(14, INPUT_PULLUP);
#endif
// Camera initialization
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
  Serial.printf("Camera init failed with error 0x%x", err);
  return;
}
sensor_t * s = esp_camera_sensor_get();
if (s->id.PID == OV3660_PID) {
  s->set_vflip(s, 1); // flip image vertically
```

```
    s->set_brightness(s, 1); // increase brightness
    s->set_saturation(s, -2); // reduce saturation
   }
   s->set_framesize(s, FRAMESIZE_QVGA); // Reduce frame size for better performance

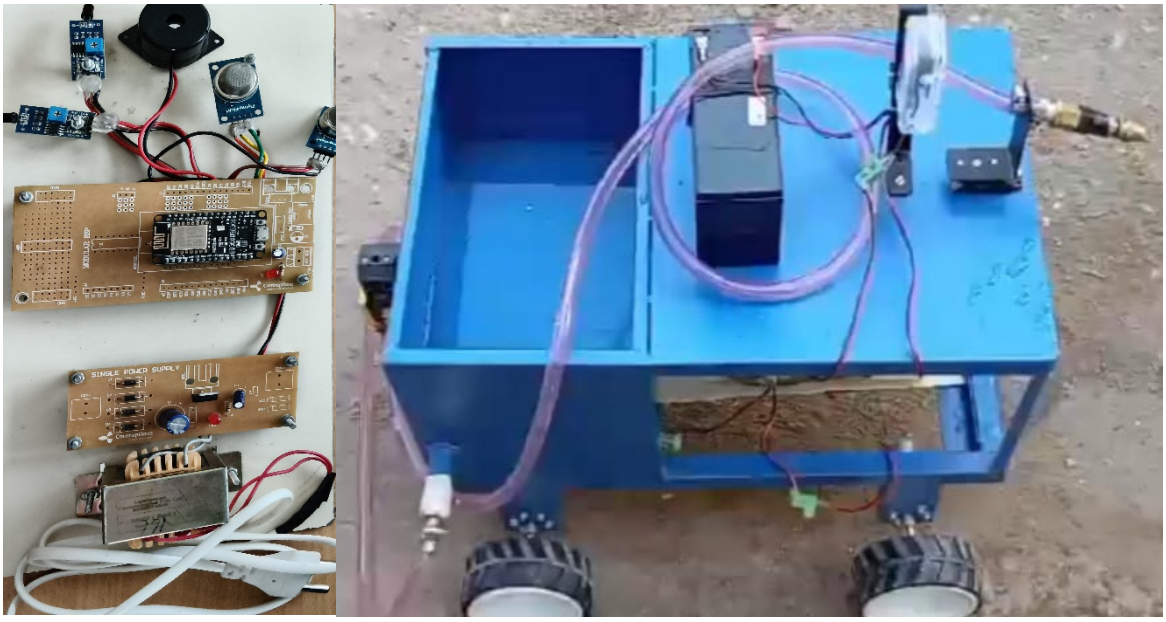
   // WiFi connection
   WiFi.begin(ssid, password);
   WiFi.setSleep(false);
   while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
   }
   Serial.println("");
   Serial.println("WiFi connected");
   digitalWrite(LED_PIN,HIGH);
   delay(100);
   digitalWrite(LED_PIN,LOW);
}
String uploadImage() {
   String getAll;
   String getBody;
  camera_fb_t * fb = esp_camera_fb_get();
   if (!fb) {
    Serial.println("Camera capture failed");
    delay(1000);
    ESP.restart();
   }
   Serial.println("Connecting to server: " + serverName);
  if (client.connect(serverName.c_str(), serverPort)) {
    Serial.println("Connection successful!");
    String head = "--Project\r\nContent-Disposition: form-data; name=\"imageFile\"; filename=\"image.jpg\"\r\nContent-Type:
image/jpeg\r\n\r\n";
    String tail = "\r\n--Project--\r\n";
    uint32_t imageLen = fb->len;
    uint32_t extraLen = head.length() + tail.length();
uint32_t totalLen = imageLen + extraLen;
client.println("POST " + serverPath + " HTTP/1.1");
    client.println("Host: " + serverName);
    client.println("Content-Length: " + String(totalLen));
    client.println("Content-Type: multipart/form-data; boundary=Project");
    client.println();
    client.print(head);
    uint8_t *fbBuf = fb->buf;
    size_t fbLen = fb->len;
    for (size_t n = 0; n < fbLen; n += 1024) {
if (n + 1024 < fbLen) {
    client.write(fbBuf, 1024);
    fbBuf += 1024;
   } else if (fbLen % 1024 > 0) {
   size_t remainder = fbLen % 1024;
```

```
  client.write(fbBuf, remainder);
  }
 }
  client.print(tail);
  esp_camera_fb_return(fb);
   int timoutTimer = 10000;
   long startTimer = millis();
   boolean state = false;
   while ((startTimer + timoutTimer) > millis()) {
   Serial.print(".");
   delay(100);
   while (client.available()) {
   char c = client.read();
   if (c == '\n') {
   if (getAll.length() == 0) { state = true; }
   getAll = "";
   } else if (c != '\r') { getAll += String(c); }
   if (state == true) { getBody += String(c); }
   startTimer = millis();
    }
    if (getBody.length() > 0) { break; }
   }
   Serial.println();
   Serial.println(getBody);
  } else {
   getBody = "Connection to " + serverName + " failed.";
   Serial.println(getBody);
  }
  return getBody;
}
void loop() {
  String response = uploadImage(); // Call the image upload function
  Serial.println("Server Response: " + response);
  delay(200); // Upload image every 10 seconds
}
```

VII.RESULT AND OUTPUT

The Firefighting Rover system undergoes thorough component testing—motors, pump nozzle, battery, microcontrollers, and relays are individually verified for performance and reliability. A web-based interface displays real-time fire status across four warehouse zones. Initially, all zones appear “Safe.” Upon detecting fire, the system flags the affected area as “Not Safe,” prompting immediate response. Arduino IDE and PHP enable efficient microcontroller programming and IoT-based remote monitoring. Real-time control, cloud data management, and autonomous fire suppression make the Rover a powerful solution for high-risk environments, enhancing safety, reducing human involvement, and protecting property during emergencies.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

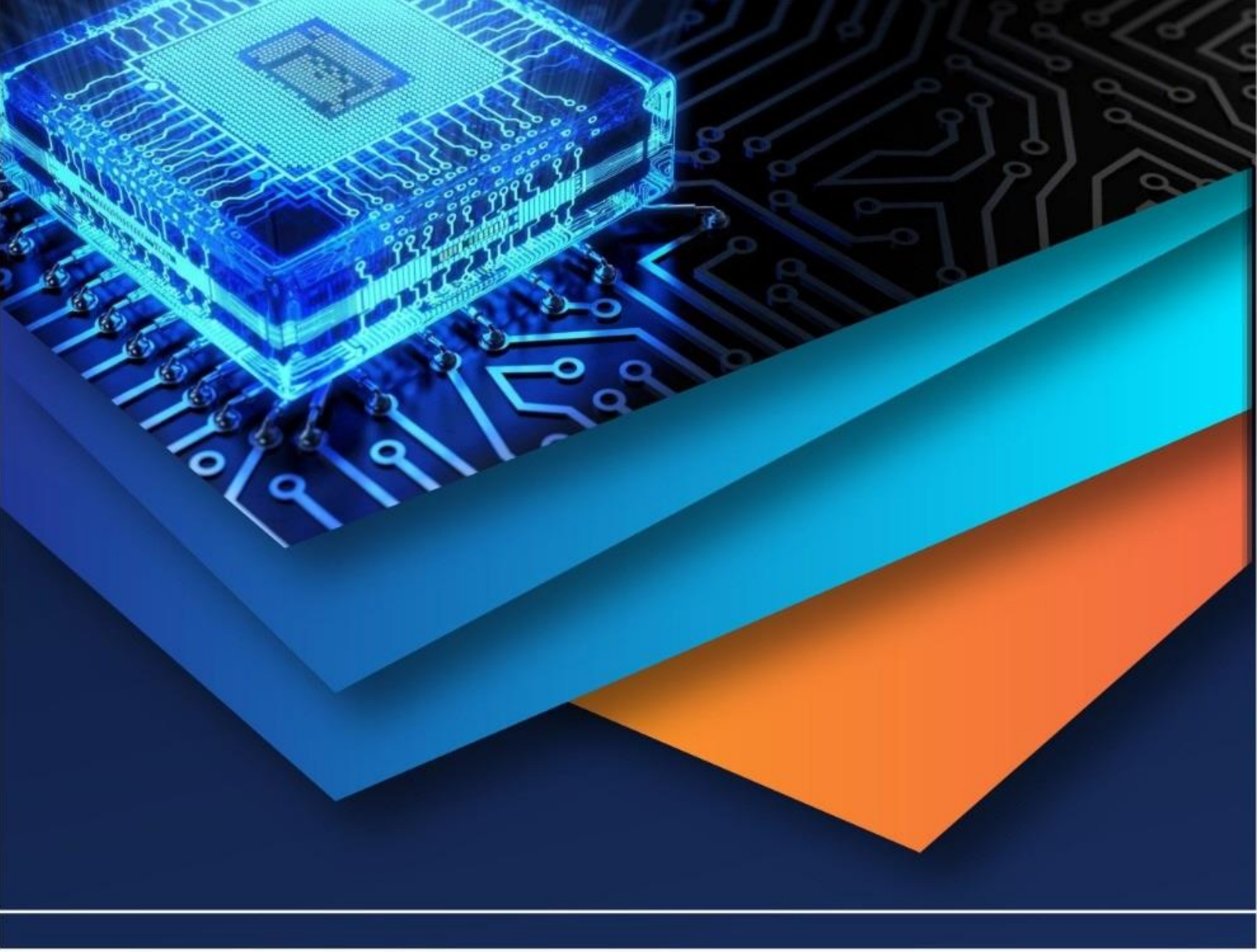


VIII. CONCLUSION

The Firefighting Rover is a major innovation in automated fire response, combining sensors, microcontrollers, motors, a water pump, and cloud connectivity to detect and extinguish fires autonomously. It navigates obstacles, operates in hazardous areas, and minimizes human risk. Real-time monitoring, precise targeting, and cloud-based control ensure rapid and efficient fire suppression. The system is cost-effective, low-maintenance, and adaptable to industrial and remote environments. Future improvements may include AI-based fire type recognition, advanced sensors, swarm coordination, solar power, and 5G communication. These advancements will enhance performance, sustainability, and responsiveness, making the Rover a vital tool in modern fire safety management.

REFERENCES

[1] John J. Craig, Introduction to Robotics: Mechanics and Control, Pearson Education, 2004.
[2] T. Braunl, Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems, Springer, 2006.
[3] Khalil, W., Dombre, E., Modeling, Identification and Control of Robots, Butterworth-Heinemann, 2002.
[4] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, Robotics: Control, Sensing, Vision, and Intelligence, McGraw-Hill, 1987.
[5] IEEE Conference Papers on Fire Detection and Robotic Firefighters, 2017–2022.
[6] Arduino Official Documentation, www.arduino.cc (Accessed 2025).
[7] Research articles from International Journal of Robotics Research on fire-resistant autonomous robots.
[8] Datasheets of Flame Sensor, Smoke Sensor (MQ-2), Ultrasonic Sensor (HC-SR04), and ESP32 WiFi Module.
[9] Technical blogs on IoT and cloud integration for firefighting systems, from sources like Medium and ResearchGate.
[10] YouTube Tutorials and Maker Websites for DIY firefighting robot concepts and real-time navigation syst



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)