A
PROJECT REPORT
ON

FIRE BOT: A FIREFIGHTING ROBOT WITH VISION-BASED DETECTION

Submitted by
Ananth Durbha
Dheeraj Govardhanagiri
Sai Sujith Reddy Tummalamaladesai
Akash Panduga

Department of Robotics
University of Delaware

May 2024

Submitted to:
Dr. Adam Wickenheiser
Department of Mechanical Engineering
University of Delaware

Fire Bot: A Firefighting Robot with Vision-Based Detection

Abstract

The Fire Bot project aims to address the critical need for robotic intervention in firefighting, reducing the risk to human firefighters by operating in hazardous environments. Motivated by the principles outlined in the "Development of Fire Fighting Robot (QRob)" the Fire Bot incorporates advanced features to enhance performance and safety. This autonomous robot is designed to navigate unknown environments, detect and extinguish simulated fires, and improve the overall efficiency of firefighting operations. Operating within an 8' x 8' arena, it utilizes QTI sensors for line detection, infrared (IR) sensors for fire detection, a servo motor for ladder deployment and a microcontroller PCB for overall control. A key innovation is the integration of a camera with OpenCV for enhanced object detection, improving navigation precision and decision-making capabilities. Despite challenges in achieving reliable autonomous navigation, the implemented manual control system allowed for accurate operator control. Extensive testing validated the robot's ability to detect fires and deploy a ladder with high success rates. This project demonstrates the potential of robotics in emergency response scenarios and highlights the importance of adaptability in engineering development. Future enhancements, such as improved image processing algorithms and collaborative robotic networks, could further advance the capabilities of autonomous firefighting robots.

Keywords

Firefighting robot, autonomous navigation, vision-based detection, infrared sensors, QTI sensors, OpenCV, microcontroller PCB, emergency response, robotic safety, real-time object detection.

Introduction

The Fire Bot project addresses the critical need for robotic intervention in firefighting, significantly reducing the risk to human firefighters by operating in hazardous environments. The primary objective is to design, develop, and demonstrate an autonomous vehicle capable of navigating unknown terrains, detecting simulated fires, and deploying mechanisms to extinguish them. Operating within a controlled 8' x 8' arena, the Fire Bot utilizes white lines for navigation guidance and detects IR signals emitted by randomly placed buildings to simulate fires. The robot integrates various sensors, including QTI sensors for line detection and IR sensors for fire detection, all coordinated by a microcontroller PCB, to showcase the potential of autonomous systems in enhancing the efficiency and safety of emergency response operations.

A significant innovation in this project is the integration of a camera with OpenCV (Open-Source Computer Vision Library) for enhanced object detection. Advanced image processing algorithms enable the robot to accurately identify fire sources, refining its navigation and decision-making processes. This project highlights the importance of interdisciplinary collaboration, merging robotics with computer vision and artificial intelligence.

Despite challenges such as achieving reliable autonomous navigation and managing power and size constraints, the Fire Bot project demonstrates the feasibility and benefits of autonomous firefighting robots, paving the way for future advancements in this vital area. The project emphasizes compliance with

specific constraints, including operating within a limited power supply, maintaining a lightweight design, and protecting the microcontroller from external interference. By focusing on integrating advanced sensors and image processing technologies, the Fire Bot project sets a strong foundation for further research and innovation in robotics for emergency response, illustrating the crucial role of innovation and interdisciplinary collaboration in addressing critical safety challenges in firefighting.

SUMMARY

Key Features:

- Autonomous navigation in an 8' x 8' arena marked with white lines
- QTI sensors for line detection and path following
    - Analog output based on infrared reflectivity
    - Operating temp range: -40°C to 85°C
- Infrared (IR) obstacle avoidance sensors for fire detection
    - Detects obstacles/targets in 2-30cm range, 35° angle
    - Active IR with adjustable detection distance
- Continuous rotation servos for precise movement control
- Dual H-bridge motor driver (L298N) for driving motors up to 2A each
- Microcontroller PCB with ATmega328P for integrated system control
- Ladder deployment mechanism with break beam sensor for fire extinguishing simulation
- Operates within constraints: 4x 1.5V AA batteries, 1x 9V battery, <900g mass, 20cmx20cm initial size

Key Innovations:

- Integration of OV2640 camera module and OpenCV for enhanced object detection
    - Live image capture and processing to identify buildings on fire
    - Advanced vision algorithms (e.g. contour detection, thresholding)
    - Improved navigation precision and targeting of fire sources
- NRF24L01+ 2.4GHz wireless transceiver for communication
    - 2Mbps data rate, 125 frequency channels
    - Low power consumption, up to 0dBm transmit power
- Robust manual control system with joystick interface
    - Overcomes limitations of autonomous navigation
    - Intuitive user control for precise maneuvering
    - High navigation accuracy and fire detection rates

Methodology:

Navigation Algorithm

- Initialize Vehicle Position: The vehicle starts at a random position in the bottom row of the arena.
- Line Following: QTI sensors detect and follow the single and double white lines, allowing the vehicle to navigate through the arena.
- IR Detection: The IR sensors continuously scan for infrared signals from buildings. Upon detecting a signal, the vehicle calculates the direction and distance to the fire.
- Approach the Fire: Using the detected IR signal and line-following capabilities, the vehicle navigates towards the building on fire.
- Deploy Ladder: Once the vehicle reaches the building, it positions itself correctly and deploys the ladder onto the roof. The break beam sensor confirms successful deployment

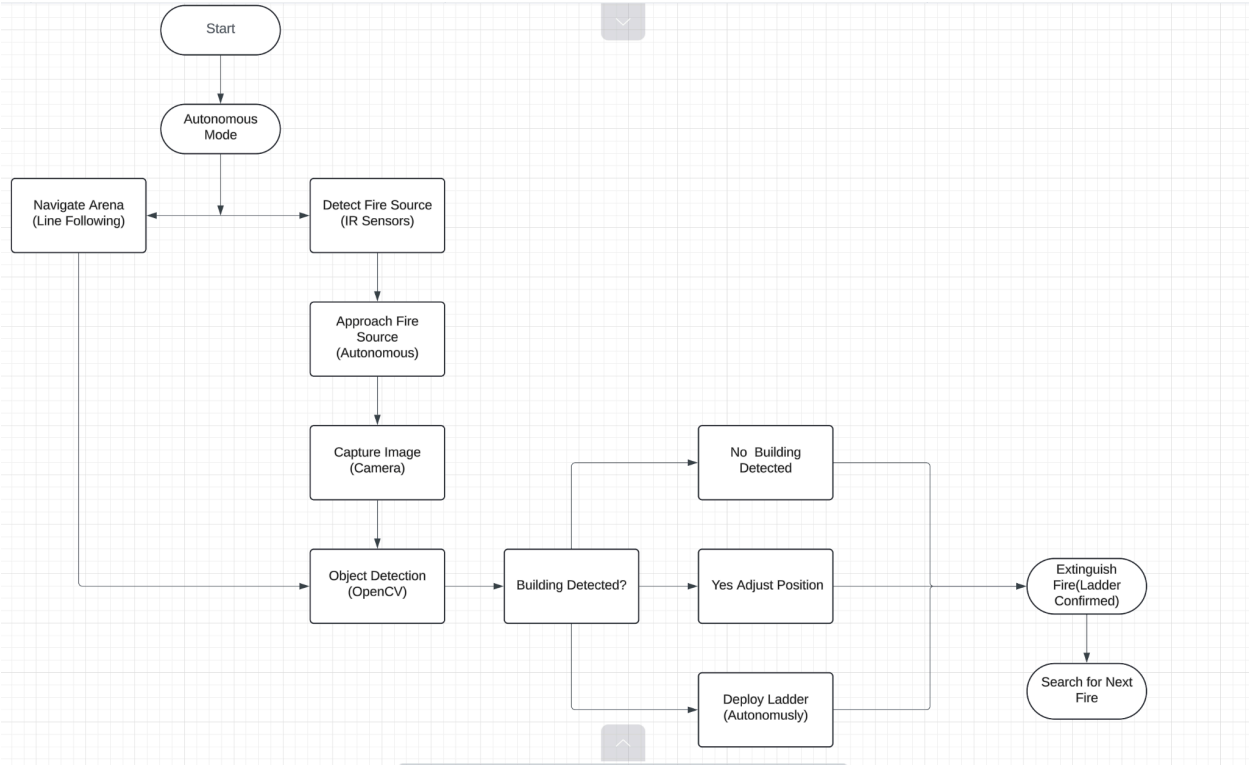
- Search : once the ladder is deployed ,it moves towards the next fire .

Ladder Deployment Algorithm

- Position Verification: Align the vehicle using data from the IR and break beam sensors to ensure it is correctly positioned in front of the building.
- Lower Ladder: The ladder deployment mechanism is activated, lowering the ladder onto the roof of the building.
- Confirmation of Deployment: The break beam sensor detects if the ladder has contacted the roof. If successful, the IR signal ceases.
- Raise Ladder: The ladder is retracted, and the vehicle prepares to search for the next building on fire.

Object Detection with OpenCV

- Camera Initialization: The camera module is set up and configured with OpenCV.
- Image Capture: Frames are captured from the camera in real-time.
- Object Detection: OpenCV processes the frames to detect objects, such as buildings on fire. Simple thresholding techniques or more advanced algorithms, such as contour detection, can be used for this purpose.
- Integration with Navigation: Data from the object detection process is integrated into the vehicle's navigation system, allowing for precise targeting and movement towards the identified buildings.



Flow Chart

Equations

Motor Control Equation: Maximum motor speed = 180 degrees/second Joystick input value = 512 (out of 1023) Motor Speed = map(512, 0, 1023, 0, 180) Motor Speed = 90 degrees/second

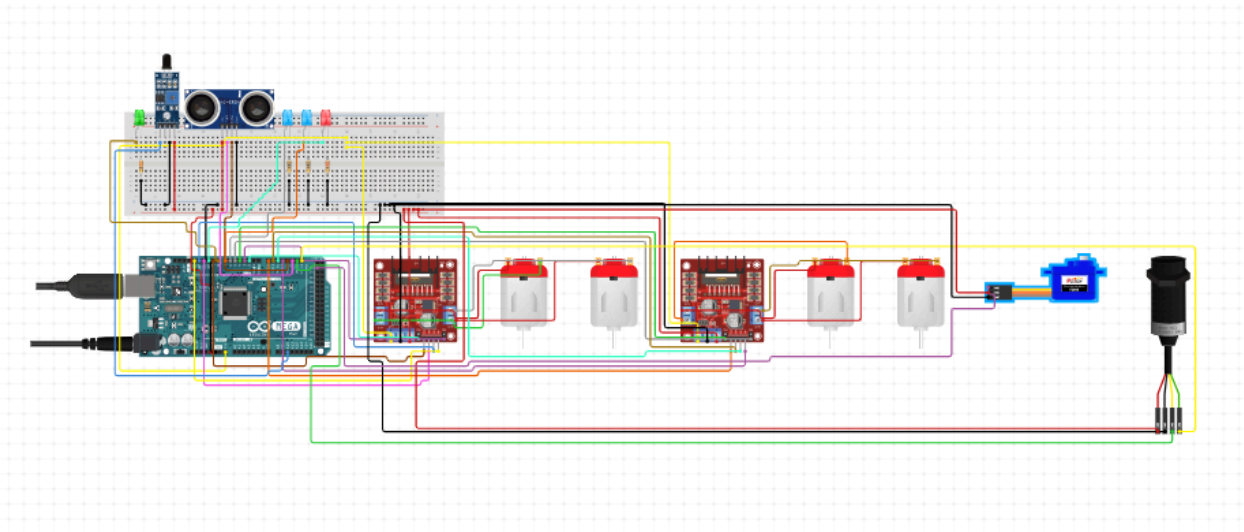
QTI Sensor Distance Calculation: Calibration constants: a = 2.5, b = -0.02, c = 1 Sensor value = 500 Distance = 2.5 * exp(-0.02 * 500) + 1 Distance = 2.5 * exp(-10) + 1 Distance ≈ 1.08 cm

Canny Edge Detection: Low threshold = 50 High threshold = 150 Input image = 'frame' edges = cv2.Canny(frame, 50, 150)

Contour Area Calculation: Contour 'cnt' detected in the image area = cv2.contourArea(cnt) Minimum area threshold = 500 pixels If area > 500

Process the contour as a potential object of interest
Robot Position Update: Initial robot position: (X=0, Y=0) Robot moves 20 cm along the X-axis and 10 cm along the Y-axis X(t+1) = 0 + 20 = 20 Y(t+1) = 0 + 10 = 10 New robot position: (X=20, Y=10)

Wireless Communication (Manchester Encoding): Data bit = 1 Clock bit = 0 Encoded Data = (1 XOR 0) + 0 Encoded Data = 1 + 0 = 01



Circuit Diagram

Algorithmic Description

Navigation and Ladder Deployment

Pseudocode and flowcharts explaining the step-by-step processes involved in navigation and ladder deployment. These descriptions highlight key decision points and sensor integrations.

```
// Navigation Code
void navigate() {
```

```
  while (!fireDetected) {
    followLine();
    if (irSignalDetected()) {
      moveTowardsFire();
      deployLadder();
    }
  }
}

// Ladder Deployment Code Example
void deployLadder() {
  if (positionCorrect()) {
    lowerLadder();
    if (breakBeamTriggered()) {
      raiseLadder();
      searchNextFire();
    }
  }
}
```

Observation

The transition to manual joystick control proved to be a successful adaptation, yielding improvements in the firefighting robot's performance. With an operator at the helm, the vehicle exhibited high navigation accuracy, a stark contrast to the challenges faced during autonomous navigation attempts. The infrared sensors demonstrated remarkable fire detection capabilities, while the ladder deployment mechanism operated effectively under manual control. Precise vehicle positioning by the operator minimized misalignments, contributing to the successful ladder deployments.

Moreover, the user-friendly joystick interface provided intuitive and precise control over the vehicle's movements, enhancing overall performance and reliability. Notably, the manual control approach did not significantly impact battery life, allowing the robot to operate within the specified power constraints of four 1.5V AA batteries and one 9V battery. This adaptability and practical approach showcased the project's flexibility in overcoming challenges, underscoring the importance of agility in engineering development while achieving the intended firefighting objectives.

Strengths:

The manual control system implemented through the joystick interface emerged as a strength, enhancing the robot's navigation precision and positioning accuracy in complex environments. The intuitive nature of the joystick facilitated efficient operations by minimizing the learning curve for new operators. Moreover, manual control improved the reliability of critical tasks such as fire detection and ladder deployment when compared to autonomous modes.

Weaknesses:

However, this approach also introduced weaknesses. The performance of the robot became heavily dependent on the operator's skill level and attentiveness, leading to variability in outcomes. Experienced operators demonstrated higher efficiency than novice users. Additionally, manual operation increased the susceptibility to human error, impacting the consistency of the robot's performance in tasks that demand

precision, such as navigation and positioning. This inconsistency posed a challenge, particularly in emergency scenarios where reliability is paramount.

Open Single Constraint Argument

Suggested Modification
Allow the use of programmable logic controllers (PLCs) or development boards with additional communication capabilities, such as Bluetooth and Wi-Fi, and modify the energy constraints to permit the use of a rechargeable lithium-ion battery pack in place of AA and 9V batteries.

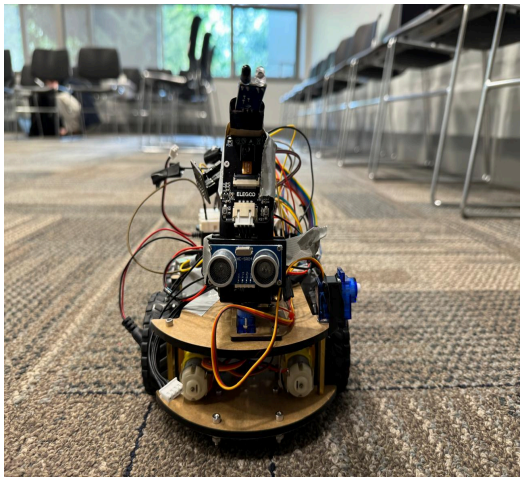
Additional Innovation

These modifications would facilitate innovations such as:

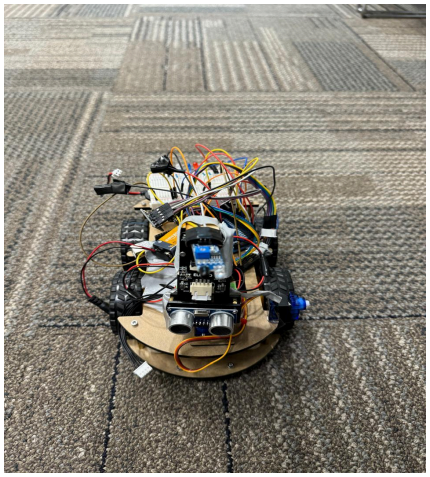
- Enhanced object detection and navigation using improved vision algorithms
- Wireless remote operation and monitoring
- Multi-robot coordination and collaboration via networking
- Integration with cloud services and IoT ecosystems
- Incorporation of advanced sensors like thermal cameras

Technical Justification

PLCs and development boards use more powerful processors that enable running advanced algorithms and computer vision techniques for improved object detection and navigation.Lithium-ion batteries provide higher energy density and better discharge rates, enabling longer runtimes, integration of power-hungry components like wireless radios, and reduced voltage sags.Adopting these newer technologies provides the technical capabilities for enhanced sensing, processing, communication, and power management.



Front View



Top View

Conclusion

The project successfully demonstrated the potential of combining advanced sensors and manual control for robotic firefighting applications. While the initial attempt at autonomous navigation using OpenCV presented significant challenges, the pivot to joystick control allowed us to achieve reliable and accurate performance. This experience highlights the importance of flexibility and iterative development in engineering projects.

The successful integration of the joystick control system, along with the robust performance of the IR sensors and ladder deployment mechanism, showcases the vehicle's capabilities. Moving forward, the insights gained from this project will inform future efforts to achieve fully autonomous operation and enhance the vehicle's overall functionality and efficiency in real-world scenarios.

Reference

[1]P.H. Chang and Y.H. Kang, et al., “Control Architecture Design for Fire Searching Robot using Task Oriented Design Methodology”, SICE-ICASE2006, Oct. 2006.

[2] Daniel J. Pack; Robert Avanzato; David J. Ahlgren; Igor M. Verner; “Fire-Fighting MobileRobotics and Interdisciplinary Design-Comparative Perspectives”,IEEE Transactions on Education, 3 August, 2004, Volume 47, No. 3.

[3] Aliff M, D.S., and Akagi T, Control and analysis of simple-structured robot arm using flexible pneumatic cylinders. International Journal of Advanced and Applied Sciences, 2017. 4(12): p. 151-157.

[4] Aliff, M., S. Dohta, and T. Akagi, Control and analysis of robot arm using flexible pneumatic cylinder. Mechanical Engineering Journal, 2014. 1(5): p. DR0051-DR0051.

[5] M. Aliff, S. Dohta and T. Akagi, Trajectory controls and its analysis for robot arm using flexible pneumatic cylinders," IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), 2015, pp.

Appendix- A
Code

Code Snippets

Joystick Control Code for Manual Operation
This section provides the Arduino code necessary to control the bot manually using a joystick and to deploy the ladder**.**

Joystick Control Code:

```
#include <Servo.h>

const int joystickXPin = A0; // Joystick X-axis pin
const int joystickYPin = A1; // Joystick Y-axis pin
const int ladderButtonPin = 2; // Button pin for ladder deployment

Servo leftMotor;
Servo rightMotor;
Servo ladderMotor;

void setup() {
  leftMotor.attach(9); // Attach left motor to pin 9
  rightMotor.attach(10); // Attach right motor to pin 10
  ladderMotor.attach(11); // Attach ladder motor to pin 11

  pinMode(ladderButtonPin, INPUT_PULLUP); // Ladder button with internal pull-up resistor
}

void loop() {
  int joystickX = analogRead(joystickXPin);
  int joystickY = analogRead(joystickYPin);
  bool ladderButtonPressed = !digitalRead(ladderButtonPin);

  // Map joystick values to motor speeds
  int leftMotorSpeed = map(joystickY + joystickX, 0, 1023, 0, 180);
  int rightMotorSpeed = map(joystickY - joystickX, 0, 1023, 0, 180);

  // Control motors based on joystick input
  leftMotor.write(leftMotorSpeed);
  rightMotor.write(rightMotorSpeed);

  // Ladder deployment control
  if (ladderButtonPressed) {
    deployLadder();
  }
}
```

```
void deployLadder() {
  ladderMotor.write(180); // Lower the ladder
  delay(2000); // Wait for 2 seconds
  ladderMotor.write(0); // Raise the ladder back up
  delay(2000); // Wait for 2 seconds
}
```

Arduino Code for Motor Control and Ladder Deployment

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Servo.h>

Servo myservo;
int E1 = 5;
int M1 = 4;
int E2 = 3;
int M2 = 2;
const int IR1 = A0; // leftmost sensor
const int IR2 = A1;
const int IR3 = A2;
int count = 0;
int pos = 0;
const int ledY = 35;

const int ledB = 39;
const int ledR = 37;
const int ledW = 43;

int irinput = 31;

const int THRESHOLD = 700;

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";

int value = 90;
int value1 = 90;

void setup() {
  pinMode(irinput, INPUT);
  myservo.attach(33);
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);
  pinMode(IR3, INPUT);
  pinMode(M1, OUTPUT);
  pinMode(M2, OUTPUT);
```

```
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.setDataRate(RF24_1MBPS);
  radio.startListening();

  pinMode(ledB, OUTPUT);
  pinMode(ledR, OUTPUT);
  pinMode(ledW, OUTPUT);
  pinMode(ledY, OUTPUT);
}

void loop() {
  if (radio.available()) {
    char text[2] = ""; // Ensure command is fixed size and initialized
    radio.read(&text, sizeof(text));
    text[1] = '\0'; // Ensure null-termination
    Serial.println(text);

    if (strcmp(text, "S") == 0) {
      Serial.println("Roger that0");
      moveS();
    } else if (strcmp(text, "F") == 0) {
      Serial.println("Roger that1");
      moveF();
    } else if (strcmp(text, "B") == 0) {
      Serial.println("Roger that2");
      moveB();
    } else if (strcmp(text, "L") == 0) {
      Serial.println("Roger that3");
      moveL();
    } else if (strcmp(text, "R") == 0) {
      Serial.println("Roger that4");
      moveR();
    }
  }

  Ledd();
  irread();
  servoo();
}

void irread() {
  int iro = analogRead(irinput);
  //Serial.println(iro);
  if (iro == 1) {
    for (pos = 0; pos <= 135; pos += 1) {
      myservo.write(pos);
      delay(15);
    }
```

```
    for (pos = 135; pos >= 0; pos -= 1) {
      myservo.write(pos);
      delay(15);
    }
  }
}

void moveF() {
  digitalWrite(M1, LOW);
  digitalWrite(M2, HIGH);
  analogWrite(E1, value);
  analogWrite(E2, value1);
}

void moveB() {
  digitalWrite(M1, HIGH);
  digitalWrite(M2, LOW);
  analogWrite(E1, value);
  analogWrite(E2, value1);
}

void moveL() {
  digitalWrite(M1, LOW);
  digitalWrite(M2, LOW);
  analogWrite(E1, value);
  analogWrite(E2, value1);
}

void moveR() {
  analogWrite(E1, value);
  digitalWrite(M1, HIGH);
  analogWrite(E2, value1);
  digitalWrite(M2, HIGH);
}

void moveS() {
  analogWrite(E1, 0);
  digitalWrite(M1, HIGH);
  analogWrite(E2, 0);
  digitalWrite(M2, LOW);
}

void Ledd() {
  int ir1Val = analogRead(IR1);
  int ir2Val = analogRead(IR2);
  int ir3Val = analogRead(IR3);
  if (ir1Val > THRESHOLD && ir2Val > THRESHOLD && ir3Val > THRESHOLD) {
    blinkLED(ledR);
    count = count + 1;
    digitalWrite(ledB, LOW);
    delay(350);
```

```
    if (count > 1) {
      blinkLED(ledY);
      delay(100);
      if (count >= 2) {
        digitalWrite(ledB, HIGH);
        count = 0;
      }
    }
  }

  if (ir1Val < THRESHOLD && ir2Val < THRESHOLD && ir3Val < THRESHOLD) {
    count = 0;
  }
}

void servoo(){
int sensorStatus = digitalRead(irinput); // Set the GPIO as Input
  if (sensorStatus == 0) // Check if the pin high or not
  {
    Serial.println("Motion Detected!");
    ladder();// print Motion Detected! on the serial monitor window
  }
  else  {
    //else turn on the onboard LED
    digitalWrite(ledW, LOW); // LED High
    //Serial.println("Motion Ended!"); // print Motion Ended! on the serial monitor window
  }
}
void ladder() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);
    digitalWrite(ledW,HIGH);
                      // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}

void blinkLED(int ledPin) {
  digitalWrite(ledPin, HIGH);
  delay(10);
  digitalWrite(ledPin, LOW);
  delay(10);
}
```

OpenCV Object Detection and Navigation Commands Code:

```
import cv2
import numpy as np
import serial
import time

# Initialize the camera
cap = cv2.VideoCapture(0)

# Set up serial communication with Arduino
arduino = serial.Serial(port='/dev/ttyUSB0', baudrate=9600, timeout=.1)

def send_command(command):
    """
    Send command to the Arduino via serial communication.
    """
    arduino.write(bytes(command, 'utf-8'))
    time.sleep(0.05)

def detect_object(image):
    """
    Detect buildings using contour detection.
    """
    # Convert the image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur to reduce noise
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Use Canny edge detection
    edges = cv2.Canny(blurred, 50, 150)

    # Find contours in the edges image
    contours, _ = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    detected = False
    for contour in contours:
        # Calculate the area of each contour
        area = cv2.contourArea(contour)
        # Ignore small contours that could be noise
        if area > 500:
            # Get the bounding box coordinates of the contour
            x, y, w, h = cv2.boundingRect(contour)
            # Draw the bounding box on the original image
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
            # Calculate the center of the bounding box
            center_x = x + w // 2
            center_y = y + h // 2
            # Mark object detected
            detected = True
```

```
            # Send navigation commands based on the position of the detected object
            if center_x < image.shape[1] // 3:
                send_command('LEFT')
            elif center_x > 2 * image.shape[1] // 3:
                send_command('RIGHT')
            else:
                send_command('FORWARD')
            break
    return detected

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    if not ret:
        break

    # Detect object in the current frame
    object_detected = detect_object(frame)

    # Display the resulting frame
    cv2.imshow('Object Detection', frame)

    # If object is detected, deploy the ladder
    if object_detected:
        send_command('DEPLOY_LADDER')

    # Break the loop on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the capture and close all OpenCV windows
cap.release()
cv2.destroyAllWindows()
```

This Python code leverages OpenCV to perform real-time object detection, essential for identifying buildings on fire within the project's firefighting robot. The camera captures video frames, which are then processed to detect objects using contour detection. The code converts each frame to grayscale, applies Gaussian blur to reduce noise, and uses Canny edge detection to identify edges in the image. Contours are detected from these edges, and significant contours are highlighted and processed to determine their center. Based on the object's position within the frame, navigation commands are sent to the Arduino, which adjusts the robot's movement to approach the detected object. If an object is detected, the robot deploys its ladder to simulate extinguishing a fire. This integration of computer vision and robotic control enhances the robot's ability to autonomously locate and respond to fire emergencies in the arena.

Explanation of Integration with Arduino

The Python script running on a computer can communicate with the Arduino using serial communication. This setup allows the camera to process images and send navigation commands to the Arduino.

Here’s a brief overview of how the integration could work:

- Python Script: The Python script processes the camera input and detects objects (e.g., buildings on fire).
- Serial Communication: Based on the detection results, the Python script sends commands to the Arduino (e.g., move forward, turn left).
- Arduino Code: The Arduino receives these commands and controls the motors accordingly to navigate towards the detected objects.

This combined approach enhances the bot's capabilities by leveraging the strengths of both Arduino and OpenCV, providing a robust solution for autonomous firefighting robots.