FIRE FIGHTING ROBOT

A

Industry Oriented Mini Project Submitted in the Partial fulfilment of the

Academic Requirement

For the Award of the Degree of

Bachelor of Technology

in

Electronics and Communication Engineering

By

E. AMULYA 22AG1A0482

Under the esteemed guidance of

Mr. P. KIRAN KUMAR

ASSISTANT PROFESSOR, ECE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ACE

Engineering College

(NBA ACCREDITED B. TECH COURSES ECE, EEE, CSE, MECH, CIVIL)

(NAAC "A" GRADE)

An Autonomous Institution

Ankushapur (V), Ghatkesar (M), Medchal (Dist)-501 301

2024-2025

ACE
ENGINEERING COLLEGE

An Autonomous Institution

(Approved by AICTE, New Delhi & Affiliated to JNTUH) Accredited with NBA & NAAC 'A' Grade

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that the Industry Oriented Mini Project entitled **"Fire Fighting Robot"** done by

E. AMULYA **22AG1A0482**

Department of Electronics and Communication Engineering, is a record of Bonafide work carried out by them. This Industry Oriented Mini Project is done as partial fulfilment of obtaining Bachelor of Technology degree to be awarded by **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD** during the academic year 2024-2025.

Mr. P. KIRAN KUMAR
Assistant Professor
(Industry Oriented Mini Project Supervisor)

Dr. Y. CHAKRAPANI
Professor & DEAN of ECE

Department Of Electronics & Communication Engg.
ACE Engineering College
An Autonomous Institution
Ankushapur (V), Ghatkesar (M),
M.M. Dist-501 301. T.G.

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

I would like to express our deepest gratitude to **Mr. P. KIRAN KUMAR, Assistant Professor** for his invaluable guidance, constructive criticism, and encouragement during the course of this project.

I am deeply indebted to **Dr. Y. CHAKRAPANI, Professor and DEAN** of the Department for providing the necessary opportunities for the completion of the project.

I am grateful to our principal of our college, for providing an excellent atmosphere, through flexible timing to work.

Thanks to our Project Coordinators **Prof. B. GIRI RAJU** and **Mrs. K. LAKSHMI PRASANNA, Assistant Professor** for being uniformly excellent advisors. They were always open, helpful, and provided strong broad ideas.

I am grateful to **Prof. Y. V. GOPALA KRISHNA MURTHY,** for his moral support and for providing facilities.

I am thankful to the teaching and non-teaching staff members of the Electronics and Communication Engineering department who helped directly or indirectly for successful completion of the project.

Above all, we are very much thankful to the management of **ACE Engineering College** which was established by the high-profiled intellectuals for the cause of Technical Education in the modem era.

I am very grateful to our family and friends for their constant support and encouragement during the project period.

With Regards,

E. AMULYA **22AG1A0482**

TITLE

	PAGE NO'S
ABSTRACT	i
LIST OF FIGURES	ii-iii
CHAPTER 1- INTRODUCTION	**1-2**
1.1 INTRODUCTION OF THE PROJECT	1
1.2 EXSISTING SYSTEM	2
1.3 PROCESSED SYSTEM	2
CHAPTER 2 - COMPONENTS OF THE PROJECT	**3-18**
2.1 HARDWARE COMPONENTS	**3-10**
2.1.1 ARDUINO UNO	3
2.1.2 L293D MOTOR DRIVER	4
2.1.3 IR SENSOR	5
2.1.4 SERVO MOTOR	6
2.1.5 DC MOTORS	7
2.1.6 RELAY MODULE	8
2.1.7 WATER PUMP	9
2.1.8 BATTERY	10
2.2 SOFTWARE COMPONENTS	**11-14**
2.1 ARDUINO	11
2.2 ARDUINO INSTALLATION	12
2.3 ABOUT THE ARDUINO SOFTWARE	13-18

CHAPTER 3 -WORKING OF THE PROJECT 19-17

3.1 FLOW CHART 19

3.2 EXPLANATION OF THE FLOW CHART 20

3.3 BLOCK DIAGRAM 21

3.4 EXPLANATION OF BLOCK DIAGRAM 22

CHAPTER 4 -RESULT **23-25**

CHAPTER 5 -ADVANTAGES, DISADVANTAGES AND APPLICATION **26 – 29**

5.1 ADVANTAGES 27

5.2 DISADVANTAGES 28

5.3 APPLICATIONS 29

CHAPTER 6 - CONCLUTION AND FUTURE SCOPE **30-31**

6.1 CONCLUTION 30

6.2 FUTURE SCOPE 31

REFERENCES **32**

APPENDIX **33-34**

ABSTRACT

A fire fighter's work entails detecting and extinguishing fires. In this rapidly evolving technological age, the world is gradually moving toward automated systems. Firefighters, on the other hand, are often in danger of losing their lives. The majority of the deaths were caused by toxic gases found in the firefighting environment. As a result, in order to resolve these issues, our system was developed a fire-fighting robot. This firefighting robot uses ARDUINO, Fire sensors, etc. When the Robot detects a fire, it gives a message to the ARDUINO. Then ARDUINO sends the signal to the motor driver and thus water is sprayed in the direction of the fire. It assists firefighters in extinguishing the fire. And it will perform its operation where firefighters can't reach. This will save the risk of fire fighters' life and avoid any further damage. The **Fire Fighting Robot** is an autonomous or semi-autonomous robot designed to detect and extinguish fires in environments that may be hazardous to human life. The primary goal of this project is to build a low-cost, efficient robotic system capable of detecting fire and responding by activating a suppression mechanism, such as a water pump or fan. The robot uses **flame sensors** and **infrared (IR) sensors** to detect fire and obstacles, respectively. It is powered by an **Arduino Uno microcontroller,** which processes sensor data and controls the movement and action of the robot.

The firefighting robot is an intelligent, autonomous system designed to detect and extinguish fires, particularly in environments that are hazardous or difficult for humans to access. The project integrates various electronic components, including flame sensors, infrared (IR) sensors, a water pump, motors, and a microcontroller (typically an Arduino), to create an efficient fire response unit. The robot operates on battery power and uses flame sensors to continuously monitor its surroundings for signs of fire. Once a flame is detected, the IR sensors help in determining the exact direction and distance to the source. The robot then navigates towards the fire using motor-driven wheels controlled through a motor driver module (such as L293D), adjusting its path based on real-time feedback from its sensors. Upon reaching a safe and effective range, the onboard water pump is activated to spray water directly onto the flames until the fire is extinguished. Throughout the operation, the robot monitors the fire status; if the fire reignites, it will automatically resume firefighting actions. This system minimizes human risk in dangerous situations and provides continuous fire monitoring in places such as factories, warehouses, and remote locations. The modular design allows for future enhancements such as adding thermal cameras, wireless communication, or integrating with IoT systems for remote control and monitoring. The project not only demonstrates practical implementation of robotics in safety applications but also showcases how embedded systems and sensor integration can contribute to intelligent, autonomous machines capable of performing life-saving tasks.

LIST OF THE FIGURES

Figure No.	Caption	Page No.
FIG 2.1.1.1	Arduino UNO	3
FIG 2.1.2.2	L293D Motor Shield	4
FIG 2.1.3.3	IR Sensor	5
FIG 2.1.4.4	Servo Motor	6
FIG 2.1.5.5	DC Motors	7
FIG 2.1.6.6	Relay Module	8
FIG 2.1.7.7	Water Pump	9
FIG 2.1.8.8	Battery	10
FIG 2.2.1	Arduino IDE Logo	11
FIG 2.2.2	Arduino IDE APP Interface	12
FIG 2.2.3	Writing Sketches	13
FIG 2.2.4	Verify button	13
FIG 2.2.5	Upload button	14
FIG 2.2.6	New file	14
FIG 2.2.7	Open file	14
FIG 2.2.8	Save file	14
FIG 2.2.9	Serial Monitor button	14
FIG 3.1.1	Flow chart Of the Project	19
FIG 3.2.1	Block diagram Of the Project	21
FIG 4.1	Initial state of the Fire Fighting Robot	23

FIG4.2	Robot navigating and avoiding obstacles	23
FIG4.3	Robot detecting fire	24
FIG 4.4	Robot halting operations after extinguishing the fire	25

CHAPTER-I INTRODUCTION

1.1 Introduction of Project:

One of the most important parameters in fire disaster is life, i.e. lives lost in saving someone else life. It is sometimes impossible for fire-fighters personnel to access the site of a fire because of explosive materials, smoke, and high temperatures. A fast response to detect the fire can avoid many disastrous things. From the given statics (Fig. I), it is observed that fire can take place at domestic as well as at industrial level. A normal spark can generate a massive fire breakout. Not only lives of industrial people but also the lives of domestic 's people are at risk because of poor fire management system. But it can be avoided using proper fire controlling methods. For such environments, fire-fighting robot is proposed. In today's generation a lot of robots are proposed and designed to remove the human factor from dangerous and deadly work. The use of robots is becoming very common that safely completes the labour intensive or deadly work for human beings. A Fire Extinguishing Robot is based on IOT Technology. In Fire Extinguishing robot, we intend to build a system that could extinguish a small flame by sensing and moving to the location itself. It will automatically detect the fire with the help of flame sensors. Once it detects the fire breakout location, it navigates itself accordingly to reach the fire source and extinguishes the fire by using built-in fire extinguishing system. For fire detection it is using three flame sensors. First one for the left direction, second one for the forward direction and third one for the right direction. Fire extinguishing system will get activated when fire detection system detects fire. It then reaches the breakout point and water pump will start ejecting the water when it detects fire. The key features of this system are to provide surveillance of fire so that major fire accidents can be prevented and loss of human lives get minimized.

The firefighting robot is an innovative attempt to address this challenge by developing an intelligent mobile platform capable of detecting and extinguishing fires automatically. Combining the power of embedded systems, sensor networks, and mechanical actuation, this robot can autonomously navigate through its environment, identify the presence and location of a fire using flame and infrared sensors, and initiate fire suppression actions by activating an onboard water spraying system. The integration of a microcontroller allows for real-time decision-making and control of various subsystems, including mobility, sensing, and actuation. The use of motor driver circuits enables precise movement and positioning, while the efficient power management system ensures reliable operation. Furthermore, the robot's design emphasizes portability, cost-effectiveness, and adaptability, making it suitable for deployment in a variety of settings. As automation and robotics continue to advance, the development of autonomous firefighting robots represents a significant step toward creating safer living and working environments, reducing reliance on human responders in dangerous situations, and contributing to the broader goal of smart disaster management systems.

1.2 Existing system:

In the existing systems, firefighting tasks are primarily conducted manually or with the help of simple remote-controlled robots. These approaches are often inefficient in hazardous environments where human intervention becomes extremely risky. Some existing firefighting robots require manual operation via remote control, which limits their response time and flexibility during emergencies.
These systems typically use basic flame detection modules and require human oversight to control the movement and activation of extinguishing systems. As a result, their effectiveness in spontaneous or large-scale fire incidents is limited. Additionally, many existing robots are not equipped with proper feedback or intelligence to locate the fire source and act autonomously.

1.3 Processed system:

The firefighting robot begins its operation with the initialization of the power supply, which typically consists of a battery pack (such as 4x AAA batteries or a 12V source). This battery powers the entire system, including the Arduino Uno microcontroller, sensors, motor driver, DC motors, servo motor, relay module, and water pump. Once powered on, the flame sensors start monitoring the surroundings for any sign of fire, while IR (infrared) sensors are used to detect obstacles to prevent collisions as the robot moves. The signals from the sensors are sent to the Arduino Uno, which acts as the brain of the robot. It processes the inputs from the flame sensor to determine the presence and direction of a fire, and uses IR sensor data to navigate safely. Based on this data, the Arduino controls the L293D motor driver, which in tum drives the DC motors to move the robot forward or steer it left or right to approach the fire. A servo motor may be used to rotate either the flame sensor or the water nozzle to better target the flame.

When the robot is close enough to the fire, as indicated by the flame sensor readings, it stops and reconfirms the presence of the flame to avoid false triggering. Once confirmed, the Arduino activates the relay module, which turns on the 12V water pump. The pump then sprays water directly toward the fire to extinguish it. After the fire is extinguished---or if the flame sensor no longer detects fire-the water pump is turned off automatically. The robot then either stops, returns to its original position, or continues scanning for additional fire sources, depending on how the system is programmed. This processed system allows the robot to operate autonomously in detecting and extinguishing small fires, making it a useful tool for safety applications in homes, offices, and industrial environments. As the robot approaches the fire, it uses the flame sensor to continuously check its proximity. Once it is within an effective range, it halts its movement and verifies the fire's presence again to prevent unnecessary activation due to false signals. Upon confirmation, the Arduino triggers the **relay module,** which acts like a switch to activate the **12V water pump.** The water pump then begins spraying water through a nozzle, often assisted by a **servo motor** that may be used to rotate the nozzle or sensor to ensure the water is directed precisely at the flame.

CHAPTER 2 - HARDWARE AND SOFTWARE DESCRIPTION

2.1) HARDWARE COMPONENTS:

2.1.1 ARDUINO UNO:

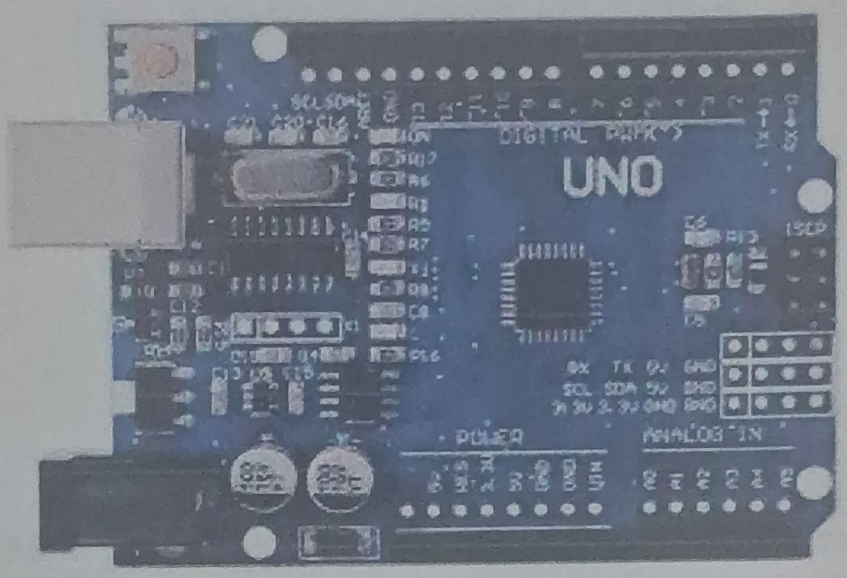

FIG 2.1.1.1 Arduino UNO

The **Arduino Uno** is a microcontroller board based on the **ATmega328P** and serves as the main control unit in the fire-fighting robot. It operates at 5V and features 14 digital input/output pins (including 6 PWM outputs), 6 analog inputs, a 16 MHz clock, USB connection, power jack, and reset button. In this project, the Arduino Uno receives input signals from flame sensors, processes the data through embedded code, and sends output signals to the **L293D motor driver** to control the water pump and DC motors. Its reliability, low power usage, ease of programming, and compact size make it ideal for real-time fire detection and suppression tasks in an autonomous robotic system.

The **Arduino Uno** is a robust, open-source microcontroller board based on the **ATmega328P** microcontroller. It is widely used in embedded systems and loT-based projects due to its simple architecture, reliability, and easy programmability. The Uno operates at **SV** and can be powered through a USB cable or an external 7-12V power source. It features **14 digital input/output pins** (6 of which support PWM), **6 analog input pins, 32 KB of flash memory, 2 KB of SRAM,** and **1 KB of EEPROM.** It runs at a clock speed of **16 MHz,** which is sufficient for real-time operations like fire detection and motor control. In this fire-fighting robot project, the Arduino Uno acts as the **central processing unit,** collecting input from flame sensors and making real-time decisions based programme.

2.1.2 L239D MOTOR DRIVER SHIELD:

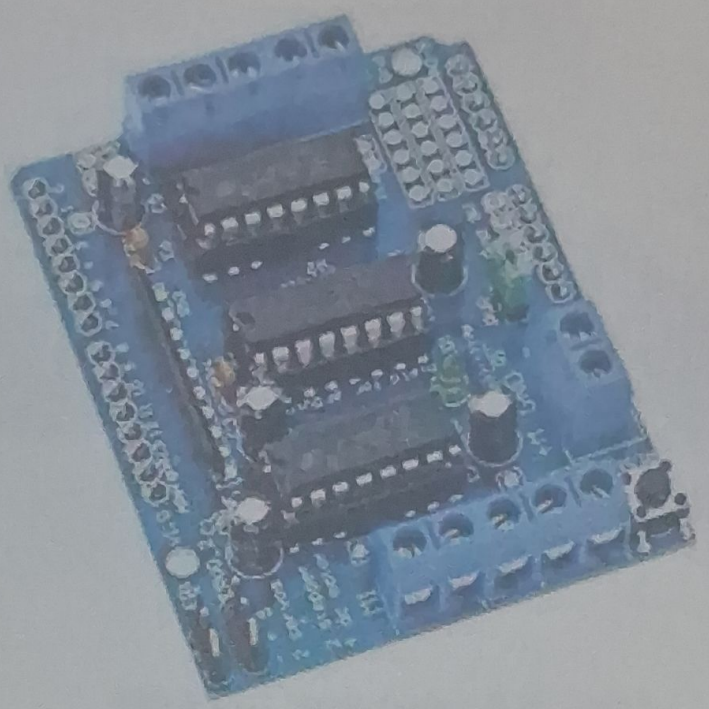

FIG 2.1.2.2 L239D Motor Driver Shield

The L293D Motor Driver Shield is a widely used electronic module that allows microcontrollers, such as Arduino boards, to control the direction and speed of DC motors and stepper motors. Based on the L293D integrated circuit, which is a quadruple high-current half-H driver, this shield can drive two DC motors simultaneously while providing control over both direction and speed. It can handle up to 600 mA of continuous current per channel and voltage up to 36V, making it suitable for a variety of small to medium robotic applications. The shield is designed to sit directly on top of an Arduino board, with stackable headers that allow easy connection without complicated wiring. It typically features terminals for motor connections, jumpers for configuration, and additional pins to control motor enable/disable functions and pulse-width modulation (PWM) for speed control. This plug-and-play nature makes it an ideal choice for hobbyists and students working on projects like line-following robots, obstacle avoiders, and fire fighting robots. By using this shield, the microcontroller can send logic signals to the L293D IC, which then drives the motors with the required voltage and current, isolating the microcontroller from the motor's power demands. Moreover, the L293D includes internal diodes for back EMF protection, ensuring the safety and longevity of both the shield and the connected components. The simplicity, affordability, and reliability of the L293D Motor Driver Shield make it an essential component for robotics and automation projects. The L293D is a dual H-bridge motor driver IC that allows control of two DC motors in both forward and reverse directions. It is specifically designed to handle high current and voltage levels, making it ideal for robotic applications. The IC can control motors operating at voltages from 4.5V to 36V and can deliver a continuous current of up to 600 mA per channel. In this fire-fighting robot project, the L293D acts as an interface between the Arduino Uno and the DC motors and water pump.

2.1.3 SENSORS

Flame Sensor:

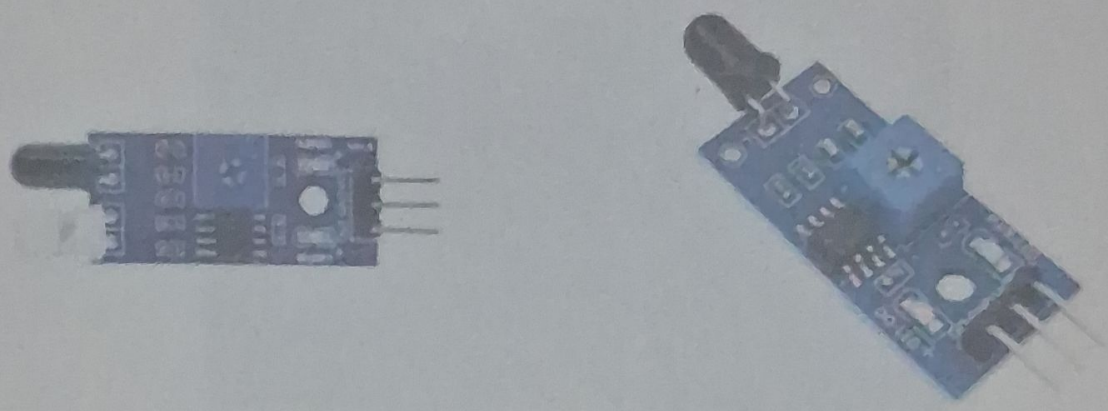

FIG 2.1.3.3 Flame Sensor & IR Sensor

The flame sensor is a crucial component of the fire-fighting robot, responsible for detecting the presence of a flame and triggering a response. It works by sensing the infrared radiation (IR) emitted by fire, typically within the 760 nm to 1100 nm wavelength range. This IR radiation is invisible to the human eye but can be easily picked up by an IR photodiode used in the sensor. The sensor is highly sensitive to light sources that match the flickering pattern and wavelength of an open flame, which makes it ideal for fire detection in safety and robotics applications. Internally, the flame sensor module consists of an IR photodiode, a signal conditioning circuit, and a comparator. The output can be taken as either a digital signal (HIGH or LOW) or an analog voltage that represents the intensity of the detected flame. It also includes a potentiometer to adjust the sensitivity threshold of the flame detection. When a flame is detected, the output pin changes state, which is read by the Arduino Uno. Based on this input, the Arduino triggers the motor driver to activate the water pump.

IR Sensor:

Although not the primary fire detection component, an Infrared (IR) sensor can be used in robotic applications for obstacle detection, line following, or even supplementary fire detection if tuned correctly. An IR sensor typically consists of an IR LED (transmitter) and a photodiode or phototransistor (receiver) that are aligned side-by-side. The IR LED continuously emits infrared light, which is reflected back by nearby objects. The reflected signal is received by the photodiode, and the intensity of the reflection determines how close the object is. In the context of a fire-fighting robot, IR sensors can be used to detect obstacles in the robot's path while it moves toward the fire. This enhances the robot's autonomous navigation capabilities, allowing it to avoid walls or other objects during operation.

2.1.4 SERVO MOTOR:

FIG 2.1.4.4 Servo Motor

A servo motor is a type of rotary actuator that allows precise control of angular position, velocity, and acceleration. It consists of a DC motor, a gear train, a position sensor (usually a potentiometer), and a control circuit. Unlike ordinary DC motors that rotate continuously, servo motors are designed to rotate to a specific angle and hold that position. This makes them highly useful in applications requiring controlled motion, such as robotic arms, camera gimbals, and fire-extinguishing nozzles. The servo motor operates based on a Pulse Width Modulation (PWM) signal. The width of the incoming pulse determines the angle to which the shaft will rotate. Typically, a pulse width of 1 ms rotates the servo to 0°, 1.5 ms sets it to 90°, and 2 ms rotates it to 180°. The control system inside the servo continuously compares the commanded position (from the PWM input) with the actual position (measured by the internal potentiometer), and adjusts the motor rotation accordingly to reduce the error. This closed-loop control system gives the servo its high precision. In the fire-fighting robot, a servo motor can be used to rotate the water spray nozzle or the sensor module, allowing the robot to dynamically adjust the direction of fire extinguishing or scanning. This is particularly useful when the fire is not directly in front of the robot or when multiple fire sources are present. Using a servo for nozzle movement increases the range and flexibility of the robot's extinguishing system without needing to move the entire robot body. Servo motors are generally compact, lightweight, and consume low power, making them ideal for mobile robotic systems.

They usually operate at **SV,** making them directly compatible with the **Arduino Uno.** Some commonly used servo models like **SG90 or MG90S** are made of plastic or metal gears, respectively, depending on the torque requirements of the application. Servo motors come in various sizes and torque ratings, making them versatile for tasks like steering mechanisms in RC cars, controlling robot arms, adjusting camera angles, or opening and closing valves.

2.1.5 DC MOTORS:

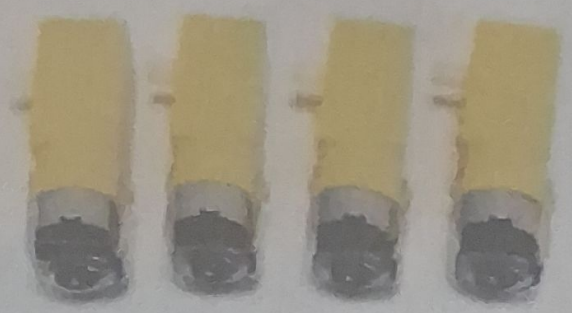

FIG 2.1.5.5 DC Motors

DC motors (Direct Current motors) are electromechanical devices that convert electrical energy into mechanical rotational motion. They are one of the most widely used actuators in robotics due to their simplicity, reliability, and ease of control. A basic DC motor consists of a stator (stationary part) and a rotor or armature (rotating part). When a direct current flows through the motor's windings, it creates a magnetic field that interacts with the field of permanent magnets (or electromagnets in some types), causing the rotor to spin.In the context of your fire-fighting robot, two or more DC gear motors are typically used to control the movement of the wheels. These motors are equipped with gearboxes that reduce the speed and increase the torque, allowing the robot to move steadily even on slightly uneven or inclined surfaces. The direction and speed of each DC motor can be precisely controlled using PWM (Pulse Width Modulation) signals from the Arduino Uno via the L293D motor driver. This enables the robot to go forward, backward, or turn left and right as needed to approach the fire.

DC motors are selected for their high starting torque, compact size, and low cost. They generally operate at voltages between 3V to 12V, depending on the model. Small robotic platforms often use 6V motors that draw low current but are sufficient to move lightweight robots. These motors are also available in different RPM (rotations per minute) ratings such as 100 RPM, 300 RPM, or 500 RPM, depending on the desired speed of the robot. There are different types of DC motors, including brushed and brushless models. Brushed DC motors are simple and cost-effective but require more maintenance due to brush wear, while brushless DC motors offer higher efficiency, reliability, and longer life. Overall, DC motors are a versatile and indispensable component in countless electronic and mechanical systems, from everyday gadgets to advanced robotics.

2.1.6 5V I0ARELAYMODULE:

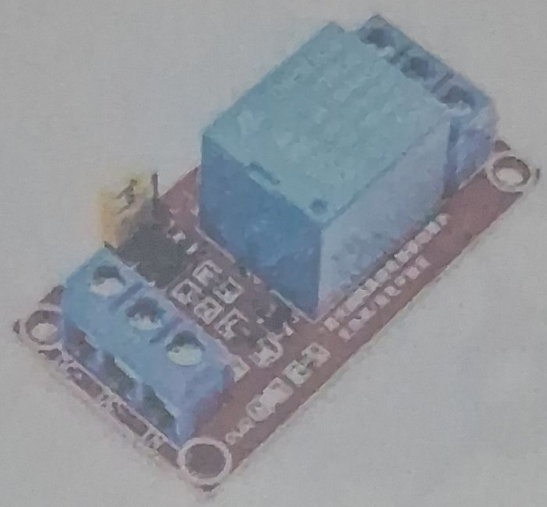

FIG 2.1.6.6 Relay Module

A **relay module** is an essential component used in electronics and embedded systems to control high-voltage or high-current devices using low-voltage signals. It works as an electromechanical switch that is operated by an electrical signal. When a small voltage is applied from a microcontroller like Arduino or Raspberry Pi, it activates an internal coil inside the relay, generating a magnetic field. This magnetic field pulls a metal contact to either connect or disconnect a high-power circuit. Relay modules are widely used to control devices such as fans, bulbs, motors, and pumps in automation projects. They offer electrical isolation between the control circuit and the device being powered, which ensures the safety of sensitive components. Relay modules come in various channel configurations (like I-channel, 2-channel, 4-channel, etc.) depending on how many devices need to be controlled.

Like those controlled by an Arduino, to safely operate high-power devices such as motors, water pumps, alarms, or even household appliances. The relay module acts as a bridge between low-voltage control logic and high-voltage operations, making it a critical safety and control component in embedded systems. At its core, a relay consists of a coil, common terminal (COM), normally open (NO), and normally closed (NC) contacts. When a voltage is applied across the coil (usually 5V for Arduino-compatible modules), it generates a magnetic field that pulls a switch, changing the contact from NC to NO. This switching action allows or interrupts the current flowing to an external device. In a relay module, this mechanism is combined with driver circuitry (like a transistor, flyback diode, and optocoupler) to protect the microcontroller from voltage spikes caused by inductive loads like motor.

2.1.7 WATER PUMP:

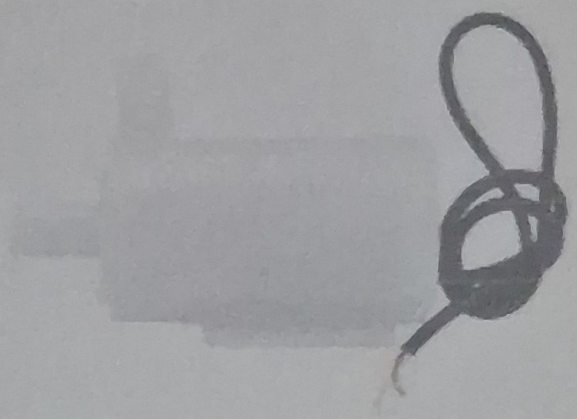

FIG 2.1.7.7 Water Pump

The 12V DC water pump is an essential actuator in the fire-fighting robot, designed to spray water on the fire source upon detection. This type of pump operates using a brush-type DC motor combined with a centrifugal impeller or diaphragm mechanism to push water through an outlet at a moderate pressure. Its working principle is straightforward: when powered, the motor rotates the impeller, creating suction at the inlet, and forces water out through the outlet nozzle. In this project, the 12V water pump is activated through a relay module or motor driver (like L293D) controlled by the Arduino. When the flame sensor detects a fire, the Arduino sends a signal to the relay or driver, turning ON the pump and enabling water to flow. The pump draws water from a small tank mounted on the robot and sprays it in the direction of the fire. These pumps are lightweight, compact, and self-priming, which makes them perfect for portable robots.

Typical 12V pumps used in such applications can handle 0.5 to 1.5 liters per minute and draw 0.5 to 1.5 amps of current depending on the load. Their plastic or ABS housing is usually water-resistant, and the inlet/outlet pipes can be easily connected with silicone tubing. These pumps are robust enough to deliver a consistent spray over short distances, which is sufficient for extinguishing small fires detected by the robot. The use of a 12V pump ensures greater pressure and flow rate than lower-voltage alternatives (like 3V or 6V pumps), making it a more reliable choice for fire suppression. However, it also requires a separate 12V power source or a voltage booster if the main battery pack is rated lower than 12V.

2.1.8 BATTERY:

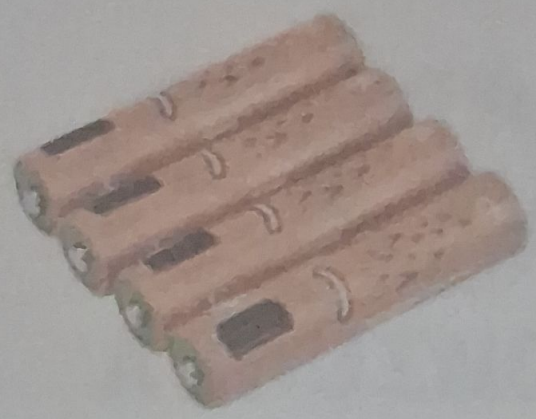

FIG 2.1.8.8 Battery

The AX4444 battery pack is a compact and portable rechargeable power source, commonly used in small robots, RC vehicles, and DIY electronics. Though specific configurations may vary, AX4444 packs are generally Lithium-ion (Li-ion) or Nickel-Metal Hydride (NiMH) based, and rated for 7.4V to 12V with current capacities ranging from 1000mAh to 2200mAh. These batteries are known for their light weight, reusability, and long cycle life. In your fire-fighting robot, the AX4444 battery pack serves as the primary power supply, delivering electricity to the Arduino Uno, motor driver (L293D), flame sensors, and possibly the 12V water pump. These battery packs typically come with built-in protection circuits to prevent overcharging, deep discharge, or short circuits, ensuring safe operation in mobile applications. One of the major advantages of using the AX4444 battery pack is its ability to deliver steady voltage and high discharge rates, which is important when running multiple components like motors and pumps simultaneously. The pack can be recharged using a standard Li-ion charger or a USB charging module, making it user-friendly and efficient. When selecting or wiring the AX4444 battery, care must be taken to ensure voltage matching with connected components. If the pump or motors require 12V and the battery provides 7.4V, a DC-DC boost converter might be used. Otherwise, a 12V version of the AX4444 should be selected to avoid performance issues. The portability of a 4x AAA battery pack allows for a clean and self-contained design without reliance on external power sources. Proper care must be taken when using lithium-ion cells, as they require protection circuits to prevent overcharging, over-discharging, and thermal runaway. The battery pack's performance directly influences the operating time and reliability of the robot, so choosing quality cells and monitoring battery health is crucial for optimal performance. Compact, easy to replace, and widely available, the 4x AAA Li-ion battery pack is an excellent choice for powering lightweight robotics and small embedded systems.

2.2) SOFTWARE REQUIRED:

FIG 2.2.1 Arduino IDE Logo

2.1) ARDUINO:

Arduino is a prototype platform (open-source) based on easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller), and a readymade software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board. The key features are:

•You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).

•Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.

• Additionally, the Arduino IDE uses a simplified version of C+, making it easier to learn to program.

2.2) Arduino Installation:

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on

the Arduino board. In this section, we will learn in easy steps how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1: First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer. In case you use Arduino Nano, you will need an A to Mini-B.

Step 2: Download Arduino IDE Software. You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.

Step 3: Power up your board. The Arduino Uno, Mega, Duemilanove, and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you must make sure that the board is configured to draw power from the USB connection

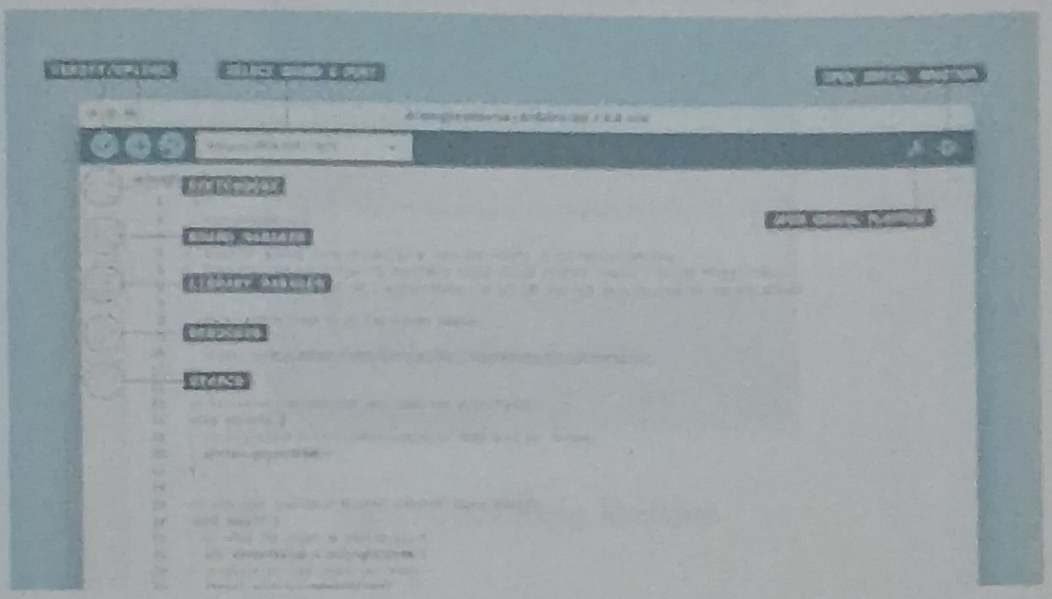

FIG 2.2.2 Arduino IDE APP Interface

The power source is selected with a jumper, a small piece ofplastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. Connect the Arduino board to your computer using the USB cable.The green power LED (labeled PWR) should glow.

Step 4: Launch Arduino IDE. After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

Step 5: Fig 2.1.1. shows an Arduino IDE. Once the software starts, you have two options: Create a new project. Open an existing project example. To create a new project, select File -> New. To open an existing project example, select File -> Example> Basics> Blink.

Step 6: Select your Arduino board. To avoid any error while uploading your program to he board, you must select the correct Arduino board name, which matches the board connected to your computer. Go to Tools - Board and select your board. Here, we have selected Arduithe no Uno board according to our tutorial, but you must select the name matching the board that you are using

2.3) About the Arduino Software:

Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

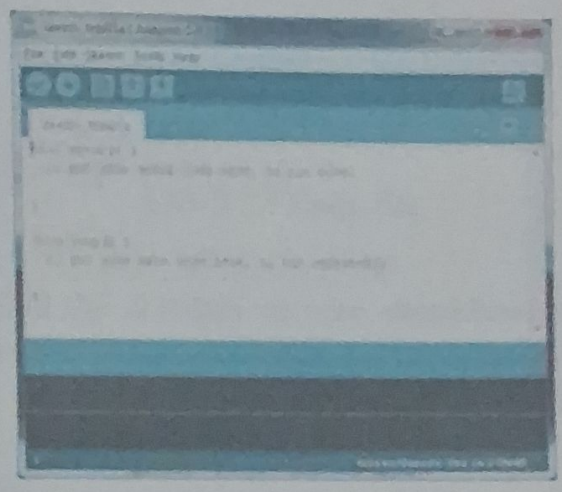

FIG 2.2.3 Writing Sketches

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension. ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), and upload programs, create, open, and save sketches, and open the serial monitor. NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension. pde.

FIG 2.2.4 Verify button

Verify Checks your code for errors compiling it

FIG 2.2.25 Upload button

Upload Compiles your code and uploads it to the configured board. See uploading below for details. Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

FIG 2.2.6 New file

New Creates a new sketch.

FIG 2.2.7 Open file

Open Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content. Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.

FIG 2.2.8 Save file.

Save Saves your sketch.

FIG 2.2.9 Serial Monitor button.

Serial Monitor Opens the serial monitor. Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help.

FILE:

- New Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- Open Allows to load a sketch file browsing through the computer drives and folders.
- Open Recent Provides a short list of the most recent sketches, ready to be opened.
- Sketchbook Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.
- Examples Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.
- Close Closes the instance of the Arduino Software from which it is clicked.
- Save Saves the sketch with the current name. If the file hasn't been named before a name will be provided in a "Save as." window.
- Save as... Allows to save the current sketch with a different name.
- Page Setup It shows the Page Setup window for printing.
- Print Sends the current sketch to the printer according to the settings defined in Page Setup.
- Preferences Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- Quit Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

EDIT:

- Undo/Redo Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- Cut Removes the selected text from the editor and places it into the clipboard.
- Copy Duplicates the selected text in the editor and places it into the clipboard.
- Copy for Forum Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- Copy as HTML Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- Paste Puts the contents of the clipboard at the cursor position, in the editor.
- Select All Selects and highlights the whole content of the editor.
- Comment/Uncomment Puts or removes the// comment marker at the beginning of each selected line.
- Increase/Decrease Indent Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- Find Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- Find Next Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.

• Find Previous Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

SKETCH:

• Verify/Compile Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

• Upload Compiles and loads the binary file onto the configured board through the configured Port.

• Upload Using Programmer This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a Tools -> Burn Bootloader command must be executed.

• Export Compiled Binary Saves a .hex file that may be kept as archive or sent to the board using other tools.

• Show Sketch Folder Opens the current sketch folder.

• Include Library Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from menu item you can access the Library Manager and import new libraries from .zip files.

• Add File... Adds a supplemental file to the sketch (it will be copied from its current location). The file is saved to the data subfolder of the sketch, which is intended for assets such as documentation. The contents of the data folder are not compiled, so they do not become part of the sketch program.

TOOLS:

• Auto Format This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.

• Archive Sketch Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

• Fix Encoding & Reload Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

• Serial Monitor Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

• Board Select the board that you're using. See below for descriptions of the various boards.

• Port This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

• Programmer For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

• Burn Bootloader The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an
Arduino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader).

HELP:

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

• Find in Reference This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

SKETCHBOOK:

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software; it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog. Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the. pde extension. You may still open. pde named files in version 1.0 and later, the software will automatically rename the extension to. ino. Tabs, Multiple Files, and Compilation Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (. cextension), C++ files (.cpp), or header files (.h). Before compiling the sketch, all the normal Arduino code files of the sketch (.ino, .pde) are concatenated into a single file following the order the tabs are shown in. The other file types are left as is.

UPLOADING:

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty. usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty. Usbserial-IBI (for a Duemilanove or earlier USB board), or /dev/ tty. USA19QW1b1P1.1 (for a serial board connected with a Key span USB-to-Serial adapter). On Windows, it's probably COMI or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx , /dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pr Diecimila) that lack auto-reset, you'll need to press the reset button on the board just befor starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

LIBRARIES:

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code. There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third- party library. To write your own library, see this tutorial.

THIRD-PARTY HARDWARE:

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub-directory name or you'll override the built- in Arduino platform.) To uninstall, simply delete its directory. For details on creating packages for third-party hardware, see the Arduino Platform specification.

SERIAL MONITER:

This displays serial sent from the Arduino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

PREFRENCES:

Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

CHAPTER 3 -WORKING OF THE PROJECT

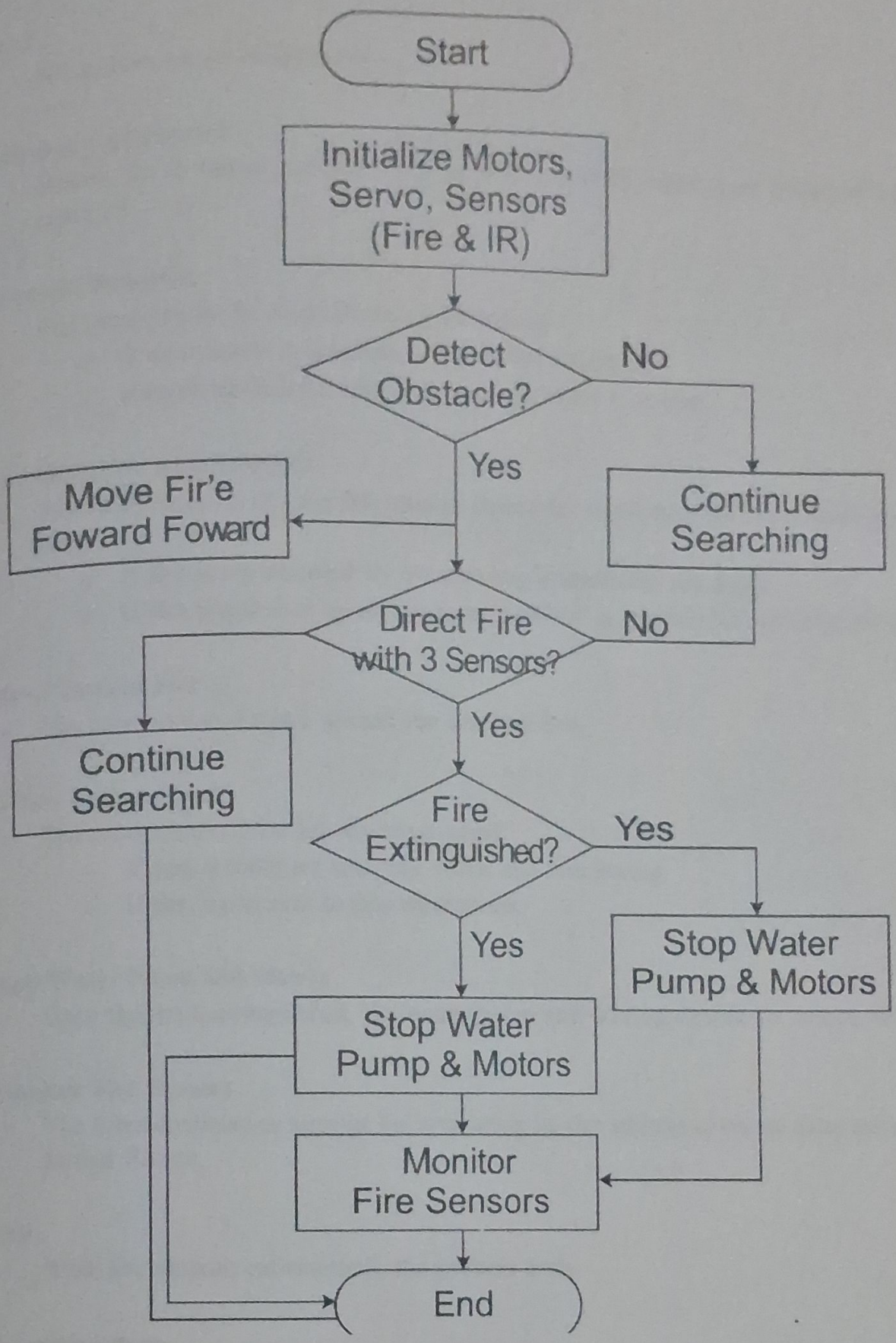

FIG 3.1.1 FLOWCHART OF THE PROJECT

3.2) FLOWCHART DESCRIPTION:

1. Start

- The system begins its operation.

2. Initialize Components

- Motors, Servo motors, and Sensors (Fire sensors and IR sensors) are initialized and activated.

3. Obstacle Detection

- The robot checks for obstacles using IR sensors.
    - If **no obstacle is detected,** it continues searching.
    - If **an obstacle is detected,** it proceeds to the next step.

4. Fire Detection with 3 Sensors

- The robot checks if all three fire sensors detect fire (indicating the exact location of fire).
    - If **fire is not detected by all sensors,** it continues searching.
    - If **fire is detected by all sensors,** it moves to the fire-extinguishing process.

5. Move Toward Fire

- The robot moves forward toward the detected fire.

6. Is Fire Extinguished?

- The robot checks if the fire is extinguished.
    - If **not,** it continues spraying water and monitoring.
    - If **yes,** it proceeds to stop operations.

7. Stop Water Pump and Motors

- Once the fire is extinguished, the water pump and driving motors are turned off.

8. Monitor Fire Sensors

- The robot continues to monitor the area using its fire sensors to ensure there are no further flames.

9. End

- If the fire remains extinguished, the process ends.

Ie) Looping Paths

- If at any point the fire is not detected, or after extinguishing it, the robot continues searching for new fire sources.

3.2.1 BLOCK DIAGRAM:

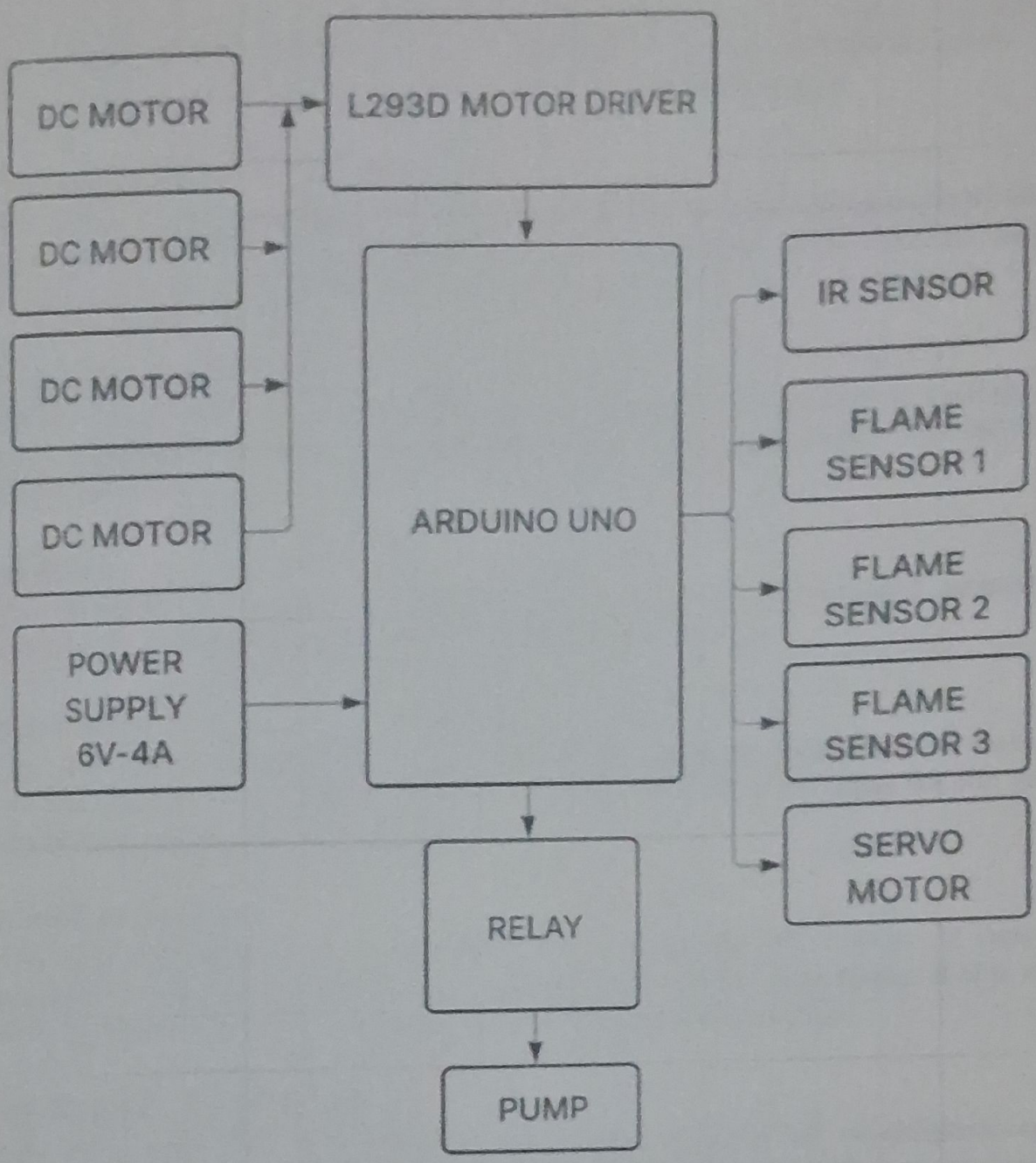

FIG 3.2.1 Block diagram Of the Project

3.2.2 Explanation of Each Block:

1. Power Supply (6V-4A)

This supplies electrical power to the entire system, including the Arduino Uno, motor driver, sensors, relay, and pump. The 6V-4A source provides sufficient current for simultaneous operation of multiple components like DC motors and pumps.

2. Arduino Uno (Central Controller)

The Arduino Uno acts as the brain of the system. It collects input signals from the flame sensors and IR sensor, processes them based on the programmed logic, and sends output signals to control the relay module, motor driver, and servo motor.

3. Flame Sensors (1, 2, and 3)

These sensors detect fire in the environment by sensing infrared radiation. Each sensor covers a specific direction (e.g., left, front, and right), providing the robot with 360° fire detection capability.

4. IR Sensor

The IR sensor is typically used for obstacle detection or environmental awareness, allowing the robot to avoid collisions while moving toward the fire.

5. Servo Motor

The servo motor may be used to rotate the spray nozzle or sensor module, enabling directional movement of either the water jet or sensor head. It helps target the fire more accurately without moving the entire robot.

6. L293D Motor Driver

This dual H-bridge IC is responsible for controlling the DC motors. It receives direction/speed signals from the Arduino and drives the motors accordingly. It acts as a buffer between the low-power Arduino outputs and the high-power motors.

7. DC Motors (4-Wheel Drive)

The robot uses four DC motors to control movement in all directions-forward, backward, left, and right. These motors are connected to the wheels and powered through the motor driver based on commands from the Arduino.

8. Relay Module

The relay module acts as a switch to control the 12V water pump. It allows the Arduino to safely operate the pump by isolating the high-power circuit.

9. Water Pump

The pump is activated by the relay when fire is detected. It draws water from a tank and sprays it through a nozzle toward the fire source until the flame sensors no longer detect h

CHAPTER 4 - RESULT

RESULT:

Implementing a real-time Fire Fighting Robot involves several key steps and observations, outlined as follows:

1) System Starting:

When the system is powered on, it initializes all core components - motors, servo motor, IR sensors, and fire sensors. The robot enters a searching mode, scanning the environment for obstacles and signs of fire. All status updates are internally monitored and executed in real-time by the microcontroller.

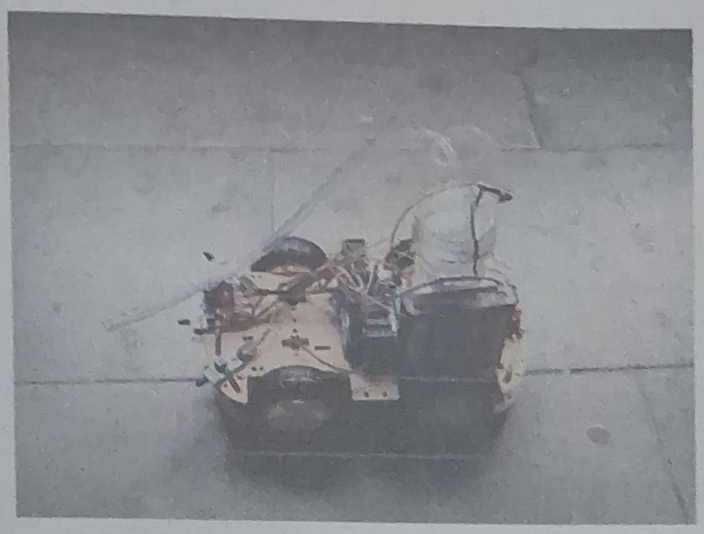

FIG 4.1: Initial state of the Fire Fighting Robot system

2) Obstacle Detection and Movement:

As the robot moves forward, IR sensors detect any obstacles. Upon encountering one, it changes its path and continues navigating the area. If no obstacle is found, the robot keeps advancing and scanning for fire.

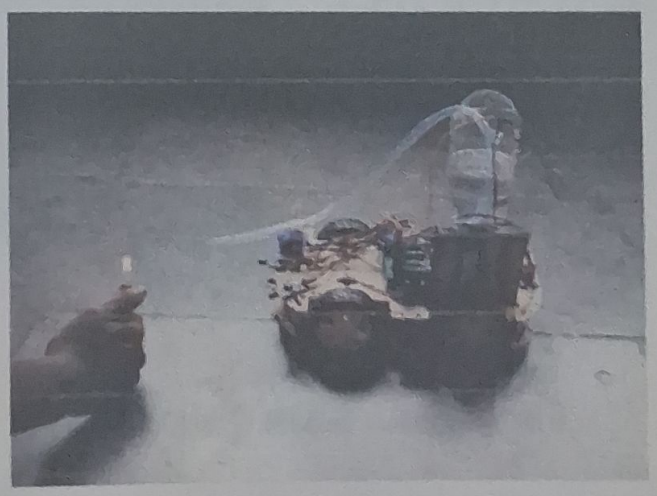

FIG 4.2: Robot navigating and avoiding obstacles

3) Fire Detection and Action:

When the fire sensors (usually 3 in number) detect the presence of fire from all directions, the robot positions itself accurately toward the fire source. It then activates the water pump and moves forward to extinguish the fire.

Robot is detecting fire in the Right direction

Robot is detecting fire in the Centre direction

Robot is detecting fire in the Left direction

FIG 4.3: Robot detecting fire

4) Fire Extinguishing Process:

The robot continues to spray water until the fire is no longer detected. Once the fire is confirmed extinguished by the sensors, the water pump and motors are stopped automatically, and the robot returns to monitoring mode.

FIG 4.4: Robot halting operations after extinguishing the fire

OUTPUT OF THE SYSTEM:

The system produces multiple coordinated outputs to ensure effective firefighting:

1. Fire Detection via Flame Sensors:

Flame sensors detect the presence and direction of fire. Multiple sensors allow accurate triangulation of the fire source.

2. Movement and Obstacle Avoidance:

IR sensors help the robot navigate without colliding with obstacles. This ensures safe and efficient motion even in cluttered environments.

3. Water Pump Activation:

When fire is detected, the pump is activated to spray water through a connected nozzle. This mechanism is fully automated and efficient in targeting small fires.

4. Servo Motor for Directional Adjustment:

The servo motor may be used to adjust the angle of the water nozzle or the camera/sensor system, ensuring accurate aiming.

5. Auto Stop Mechanism:

Once the fire is extinguished, sensors confirm it, and the system stops both water spray and movement, conserving energy and resources.

6. Continuous Fire Monitoring:

Even after the fire is extinguished, the sensors remain active to ensure no reignition occurs. The system can instantly respond if needed.

CHAPTER-5 ADVANTAGES, DISADVANTAGES AND APPLICATION

5.1) ADVANTAGES

The Arduino-based fire-fighting robot offers several key advantages that enhance its functionality, safety, and efficiency in real-world fire suppression scenarios. These benefits make it an effective solution for minimizing fire-related risks, especially in enclosed or hazardous environments.

Human Safety:
The primary advantage of this robot is that it significantly reduces the risk to **human life.** By autonomously detecting and extinguishing fires, it eliminates the need for fire personnel to enter dangerous areas during initial fire outbreaks.

Autonomous Operation:
The system operates without any manual intervention after being powered on. The **Arduino Uno,** combined with flame sensors and logic control, allows the robot to make **real-time decisions** and perform fire extinguishing actions automatically.

Multi-Directional Fire Detection:
With **multiple flame sensors** placed at different angles, the robot can detect fire in various directions. This 360-degree sensing improves response time and ensures that fire is detected and tackled from any orientation.

Low-Cost and Efficient Design:
The entire system is built using **readily available and low-cost components** such as Arduino, L293D motor driver, and DC motors. Despite its affordability, it delivers reliable performance, making it suitable for small-scale industries, homes, and labs.

Compact and Portable:
The robot is small and lightweight, allowing it to **navigate through tight spaces** or under desks and tables where fires might break out. Its portability also makes it easy to deploy in different locations.

Energy Efficient:
It uses **battery power (AX4444 pack or 4xAAA)** and operates on low voltage, making it energy-efficient. Components like the Arduino and sensors consume very little power during continuous monitoring.

5.2) DISADVANTAGES

Limited Fire Extinguishing Capacity

The robot carries only a small onboard water tank, which limits the amount of fire it can suppress. It is not suitable for large-scale fires.

Not Suitable for Outdoor Use

Due to limitations in sensors and mobility, the robot is primarily designed for indoor use. Outdoor environments with wind, uneven surfaces, or obstacles may affect its performance.

Sensor Range Limitationl

Flame sensors have a limited range and may not detect distant or obstructed fires. Their performance can also be affected by ambient light sources.

No Real-Time Communication

In its basic version, the robot does not communicate status or alerts in real-time to a user or control room. This limits its use in monitored safety systems.

Obstacle Navigation Challenges

Without advanced sensors like ultrasonic or LiDAR, the robot may struggle with accurate obstacle detection and avoidance in cluttered spaces.

5.3) APPLICATIONS:

Laboratories, Server Rooms, and Data Centers

These are high-risk environments where electrical equipment operates continuously, increasing the chance of fire due to short circuits or overheating. The robot can autonomously monitor such areas and immediately respond to small fires before they escalate, minimizing downtime and equipment damage.

Homes, Offices, and Classrooms

Small-scale fires caused by candles, faulty wiring, or unattended devices can quickly become dangerous in residential or institutional settings. The fire-fighting robot can be deployed in such spaces for early fire detection and suppression, especially in areas where fire extinguishers may not be easily accessible.

Educational Demonstrations and STEM Projects

This robot serves as a practical learning tool for students in engineering, robotics, and embedded systems fields. It demonstrates real-time integration of sensors, actuators, microcontrollers, and automation logic. It is also ideal for academic projects and technical exhibitions.

Hazardous or Inaccessible Industrial Areas

In factories or warehouses where toxic fumes, combustible materials, or heat are present, sending human responders during a fire can be extremely risky. The robot can be used in these zones to respond quickly, reducing risk to human life while containing the fire.

Research and Development in Fire Safety Robotics

This prototype can be a stepping stone for researchers working on advanced robotic fire-fighting systems. It provides a low-cost platform for experimenting with AI, IoT, or remote-control features that can be added for smarter and more efficient fire management.

CHAPTER- 6 CONCLUSION And FUTURE SCOPE

6.1) CONCLUSION:

The Arduino-Based Fire-Fighting Robot successfully demonstrates the potential of embedded systems and autonomous robotics in critical safety applications. Designed using readily available components such as the Arduino Uno, flame sensors, L293D motor driver, servo motor, and a 12V water pump, the robot effectively detects and extinguishes small-scale fires without human intervention.

The system continuously monitors its surroundings using multiple flame sensors and responds immediately when fire is detected. Upon detection, it processes the input signal through the Arduino, activates the motor driver, and triggers the water pump through a relay to suppress the fire. The robot is also capable of directional movement using DC motors, and optional servo control enhances its precision in aiming at the fire source.

This project not only meets its goal of automating fire detection and suppression but also serves as a cost-effective prototype that can be further developed for broader and more complex applications. Its applications extend from academic learning to real-world fire safety in small environments like labs, homes, and offices.

Though there are certain limitations-such as limited water capacity, obstacle navigation, and sensor range-these can be overcome by future improvements like incorporating temperature sensors, IoT modules, and advanced AI-based navigation.

Overall, this project stands as a successful implementation of real-time response automation, showcasing how technology can be used to enhance human safety and reduce the risks associated with manual fire-fighting operations.

6.2) FUTURE SCOPE:

Integration with Artificial Intelligence (AI) and Machine Learning (ML)

Future robots can use AI/ML to recognize fire patterns, differentiate between smoke, heat, and flame, and make autonomous decisions about how to act in different fire scenarios.

Wireless Communication for Remote Monitoring

Adding Wi-Fi, GSM, or Bluetooth modules would allow the robot to send real-time updates to mobile devices or control centers. This is especially helpful for monitoring fires in inaccessible or hazardous areas.

Advanced Sensors and Thermal Cameras

Replacing basic flame sensors with thermal imaging cameras and gas sensors can improve the accuracy of fire detection, enabling the robot to see heat signatures even through smoke.

Autonomous Navigation using GPS or SLAM

By using Global Positioning System (GPS) or SLAM (Simultaneous Localization and Mapping) technology, the robot can navigate unknown or large environments without human guidance.

Deployment in High-Risk Areas

These robots can be safely deployed in environments like chemical plants, nuclear reactors, forests, tunnels, and military zones, where human entry is dangerous or impossible during a fire.

REFERENCE

1. Arduino Official Documentation
   https://www.arduino.cc

Used for programming and interfacing flame sensors, motor drivers, and controlling components.

2. L293D Motor Driver Datasheet
   https://www.ti.com/lit/ds/symlink/1293d.pdf

Official datasheet explaining pin configuration, voltage ratings, and motor control logic.

3. IR and Flame Sensor Modules - Specifications

Available from component suppliers such as SparkFun, Adafruit, and Robu.in.
Example: https://robu.in/product/ir-sensor-module/

4. Fire Fighting Robot Research Paper

"Design and Implementation of an Autonomous Fire Fighting Robot"
International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181
https://www.ijert.org/research/design-and-implementation-of-an-autonomous-fire-fighting-robot-IJERTV8IS070189.pdf

5. Embedded Systems & Robotics by Raj Kamal

Book reference for understanding embedded control and robotic system design.

6. Fire Safety Guidelines - National Fire Protection Association (NFPA)
   https://www.nfpa.org

To understand basic fire hazards and firefighting protocols.

7. IEEE Papers on Autonomous Robotics

Available on https://ieeexplore.ieee.org

Example search: *Fire Fighting Robots IEEE*

APPENDIX

```
#include <AFMotor.h>
#include <Servo.h>
// Motor objects (1-4)
AF_DCMotor motor 1( 1);
AF_DCMotor  motor2(2);
AF_DCMotor  motor3(3);
AF_DCMotor  motor4(4);
// Servo for water aiming
Servo fireServo;
// Fire sensor pins
const int fireLeftPin = AO;
const int fireRightPin = A1;
const int fireFrontPin = A2;
// IR distance sensor pin (A4)
const int irSensorPin = A4;
// Relay pin for DC pump
const int relayPin= AS;
// Motor speed (65% of 255;:::1;66)
const int motorSpeed = 166;
// Fire and IR thresholds
const int fireThreshold= 500;        // Fire detection threshold
const int minDistanceThreshold = 500; // IR analog value for -3  cm
void setup () {
  Serial.begin(9600);
  motorl .setSpeed(motorSpeed);
  motor2.setSpeed(motorSpeed);
  motor3.setSpeed(motorSpeed);
  motor4.setSpeed(motorSpeed);
  fireServo.attach(9);
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
  Serial.println("System Ready.");
}
void loop () {
  int fireLeft = analogRead(fireLeftPin);
  int fireRight = analogRead(fireRightPin);
  int fireFront = analogRead(fireFrontPin);
  int distance = analogRead(irSensorPin);
  if (fireFront < fireThreshold || fireLeft < fireThreshold || fireRight < fireThreshold) {
    // Fire detected
    if (distance < minDistanceThreshold) {
```

```
Serial.println("liJ Too close to fire. Stopping at-3 cm.");
stopMotors();
} else {
if (fireFront < fireThreshold) {
 Serial.println("(ı) Fire in Front!");
 fireServo.write(90);
 moveForward();
 delay (250);
 stopMotors();
} else if (fireLeft < fireThreshold) {
 Serial.println(" (i) Fire on Left!");
 fireServo.write(150);
 tumLeft();
 delay (350);
 moveForward();
 delay (250);
 stopMotors();
} else if (fireRight < fireThreshold) {
 Serial.println(" (i) Fire on Right!");
 fireServo. write(30);
 tumRight();
 delay(350);
 moveForward();
 delay (250);
 stopMotors();
}
}
// Activate pump
Serial.println(" El Activating pump...");
digitalWrite(relayPin, HIGH);
delay (3000);
digitalWrite(relayPin, LOW);
Serial.println("0 Pump off");
} else {
// No fire detected- use IR to avoid obstacles
Serial.println(", No fire detected. Checking distance...");
if (distance < minDistanceThreshold) {
 Serial.println(" rr4 Obstacle too close. Reversing.");
 moveBackward();
 delay (500);
 stopMotors();
} else {
```

```
  Serial.println("       Area clear. Paused and scanning...");
   // Optional: Add patrol movement here if needed
   stopMotors();
  }
  delay (1000). // Delay to avoid spamming
 }
}
// === Movement functions ===
void moveForward() {
motorl.run (FORWARD);
motor2.run (FORWARD);
motor3.run (FORWARD);
motor4.run (FORWARD);
}
void moveBackward() {
  motorl.run (BACKWARD);
  motor2.run (BACKWARD);
  motor3.run (BACKWARD);
  motor4.run (BACKWARD);
}
void stopMotors() {
  motor I .run (RELEASE);
  motor2.run (RELEASE);
  motor3.run (RELEASE);
  motor4.run (RELEASE);
}
void tumLeft() {
  motorl.run (BACKWARD);
  motor2.run (FORWARD);
  motor3.run (BACKWARD);
  motor4.run (FORWARD);
}
void turnRight() {
  motorl.run (FORWARD);
  motor2.run (BACKWARD);
  motor3.run (FORWARD);
  motor4.run (BACKWARD);
}
```